UCUENCA

Universidad de Cuenca

Facultad de Ingeniería

Carrera de Ingeniería de Sistemas

Desarrollo de un proceso de creación y despliegue de microservicios a partir de requerimientos funcionales

Trabajo de titulación previo a la obtención del título de Ingeniero de Sistemas.

Autores:
Jhoan Sebastian Arias Barros
Pamela Aracely Suquisupa Nacipucha

Director:
Victor Hugo Saquicela Galarza
ORCID: 0000-0002-2438-9220

Codirector:
María Fernanda Granda Juca
ORCID: 0000-0002-5125-8234

Cuenca - Ecuador
2023-17-07

Resumen

El paulatino avance de las arquitecturas de software basadas en microservicios y la creciente relevancia de la ingeniería de requerimientos en el desarrollo de aplicaciones y sistemas web han generado la necesidad de automatizar el proceso de creación y despliegue de microservicios. En este sentido, la combinación de técnicas de Procesamiento de Lenguaje Natural (PLN) con análisis de grafos y detección de comunidades se presenta como un enfoque poderoso para abordar desafíos en diversos campos, incluyendo la ingeniería de software y el diseño de arquitecturas basadas en microservicios. Asimismo, en los últimos tiempos, los modelos de lenguaje de inteligencia artificial, como ChatGPT, han experimentado una notable popularidad y atención en diferentes ámbitos. Su capacidad para mejorar y generar texto existente ha despertado un gran interés entre investigadores, empresas y usuarios en general. Teniendo en cuenta este contexto, el siguiente trabajo de titulación propone un proceso para la identificación y despliegue automatizado de microservicios, aprovechando las ventajas que ofrecen las técnicas de procesamiento de lenguaje natural y análisis de grafos. Se espera que este enfoque contribuya significativamente a mejorar y automatizar el desarrollo de aplicaciones basadas en microservicios.

Palabras clave: ingeniería de sistemas, microservicios, software, chatgpt

Abstract

With the gradual advancement of microservices-based software architectures and the increasing relevance of requirements engineering in the development of applications and web systems, there is a need for methods or processes that automate the creation and deployment of microservices. The combination of Natural Language Processing (NLP) techniques with graph analysis and community detection offers a powerful approach to address challenges in various fields. In the realm of software engineering, this combination has been explored to enhance the design and deployment process of microservices-based architectures. Additionally, in recent times, language models of artificial intelligence, such as ChatGPT, have gained great popularity and attention in various domains. The ability to improve and generate existing text has sparked interest among researchers, companies, and users in general. In this context, the following thesis work proposes a process for the identification and deployment of microservices, leveraging the advantages of natural language processing techniques and graph analysis. It is expected that this approach will contribute to improving and automating the development of microservices-based applications.

Keywords: systems engineering, microservices, software, chatgpt

Índice de contenidos

Índice de figuras

Índice de tablas

Agradecimientos

Queremos expresar nuestro profundo agradecimiento a todas las personas e instituciones que contribuyeron de manera significativa a la realización de este trabajo de titulación. Agradecemos a nuestro director de tesis Víctor Saquicela y a nuestra codirectora María Fernanda Granda, por su guía experta y apoyo a lo largo de este proyecto. Sus conocimientos y orientación fueron fundamentales para su éxito.

No podemos dejar de agradecer a nuestros amigos y familiares por su apoyo incondicional y palabras de aliento durante todo el proceso.

¡Gracias!

Aracely y Sebastian.

Dedicatoria

A mis padres Paola y Fernando que han sido la motivación y me han brindado la fuerza para mejorar y lograr alcanzar mis metas.

A mis abuelos Hugo y Lucia que con su cariño y sabiduría me han sabido guiar todo el camino recorrido hasta hoy.

Y a mis amigos Ismael y Aracely que han sido la alegría y el apoyo incondicional en los momentos difíciles a lo largo de la carrera.

Sebastian Arias B.

Dedicatoria

Dedico este trabajo en primer lugar a Dios, que me ha dado la fortaleza para seguir adelante y nunca rendirme.

A mis abuelitos Rosario y Salustino, que son la parte fundamental de mi vida, que han sido un apoyo incondicional y la razón por la cual me esfuerzo cada día.

A mi familia, que son el ejemplo de esfuerzo y dedicación, a ellos agradezco sus valiosas enseñanzas y el siempre apoyarme en los momentos mas difíciles de este camino.

A Sebastian, mi compañero de tesis, que ha sido mi amigo a lo largo de la carrera y ha hecho todo este trayecto mas ameno y divertido.

A mis mascotas, por su lealtad incondicional y por brindarme su compañía.

Aracely Suquisupa.

1. Introducción

En los últimos años, las arquitecturas orientadas a servicios han evolucionado hacia los microservicios, que representan una innovación dentro del desarrollo de software, eliminando el carácter monolítico del pasado y llevando al uso de nuevas estrategias de división de software que permiten proporcionar características de alta calidad [1]. Estas estrategias se relacionan con la ingeniería de requerimientos, por lo que se busca formas, modelos, o procesos que permitan la implementación de los requerimientos funcionales definidos en un sistema de información para lograr un software completo usando microservicios [2].

En cuanto a requerimientos funcionales, estos juegan un papel crucial e importante en el desarrollo de software, debido a que son los que definen el comportamiento y la funcionalidad que se espera de un sistema de software, debido a que especifican las acciones que el sistema debe realizar, las entradas que se deben aceptar y las salidas que debe retornar en respuesta a las entradas. Los requerimientos funcionales usualmente están descritos en lenguaje natural, es por esto que, se usan técnicas de procesamiento de lenguaje natural (PLN) para su análisis de manera automática [3], permitiendo el estudio del texto para determinar sustantivos, verbos y entidades que con su debido tratamiento según el campo de investigación sirven para la determinación de patrones, conexiones, etc. Con el gran avance del procesamiento de lenguaje natural, hoy en día es posible que las máquinas entiendan y procesen el lenguaje humano de manera más efectiva. Sin embargo, antes de aplicar las técnicas de PLN al texto, este debe ser preprocesado para mejorar su calidad y obtener información significativa. El preprocesamiento implica varios pasos, incluida la tokenización, la derivación, lematización, la eliminación de palabras vacías y el reconocimiento de entidades, entre otros.

Un área donde convergen las técnicas de PLN y el análisis de datos es en el uso de grafos para representar datos textuales. Los grafos constituyen un poderoso marco para modelar y analizar relaciones entre entidades textuales como palabras, oraciones o documentos. Al construir una representación gráfica de datos textuales, se puede percibir las relaciones semánticas y sintácticas de varios elementos. El análisis de grafos proporciona información valiosa sobre la estructura subyacente, es decir, la forma en que los componentes principales están interconectados y organizados para dar lugar a la entidad y la organización de los datos textuales. Una

técnica popular de análisis de grafos es la detección de comunidades, cuyo objetivo es identificar grupos o comunidades de nodos conectados en un grafo. Los algoritmos de detección de comunidades utilizan interconexiones entre nodos para identificar grupos que tienen un mayor grado de similitud o relación.

Con este antecedente, Tyszberowicz [4] propone un proceso para obtener microservicios a través de la descomposición de requerimientos funcionales, dicho proceso utiliza herramientas de visualización para realizar la clusterización entre los requerimientos y llegar a los microservicios. Sin embargo, este proceso aún necesita la intervención humana para identificar una división óptima de los microservicios. Por lo tanto en este trabajo de titulación se propone utilizar como base el proceso propuesto por Tyszberowicz, con la adición de técnicas PLN con métodos de detección de comunidades en grafos para extraer información significativa de textos, particularmente de un conjunto de requerimientos expresados formalmente en casos de uso. Se propone un proceso semiautomático de preprocesamiento de datos textuales para la posterior creación de una matriz de adyacencia que permita la construcción de grafos y la aplicación de algoritmos de detección de comunidades para encontrar posibles microservicios que puedan ser desplegados y que de esta forma el trabajo de los desarrolladores se vea simplificado, con respecto al análisis de requerimientos y la definición de microservicios iniciales.

1.1. Objetivos

Para el desarrollo de este trabajo de titulación se ha definido un objetivo general y tres objetivos específicos, los mismos que se explican a continuación:

1.1.1. Objetivo General

Proponer un proceso utilizando técnicas manuales, semiautomáticas o automáticas que permita la creación de microservicios a partir de requerimientos funcionales.

1.1.2. Objetivo Específicos

El presente trabajo tiene los siguientes objetivos específicos:

1. Plantear el proceso para la creación de microservicios a partir de requerimientos funcionales.
2. Desplegar los microservicios obtenidos a través del proceso propuesto.

3. Evaluar el proceso mediante la opinión de expertos.

1.2. Metodología

La metodología utilizada para el desarrollo del proceso propuesto es una adaptación de la metodología "Transferencia de Tecnología Basada Empíricamente" [5] (Empirically-Based Technology Transfer), propuesta por Gorschek et al. [6], en la figura 1.1 se presentan los pasos que la componen. Es importante mencionar que hay fases adicionales que no han sido consideradas en este trabajo de titulación. Se ha escogido esta metodología puesto que está estrechamente relacionada con la discusión sobre la mejora del proceso de software, el enfoque principal de la metodología está en el uso de diferentes métodos para crear una solución a un problema y llevarlo a la aplicación.

1. **Problema:** El problema abordado en esta investigación es que no existe un proceso semiautomático en el cual se identifique posibles microservicios analizando requerimientos funcionales.

2. **Formulación del problema:**

   - ¿Existe un proceso establecido para transformar los requerimientos funcionales de manera automatizada a posibles microservicios ?
   - ¿Se podría definir un proceso en el cual mediante el procesamiento de lenguaje natural de requerimientos funcionales se pueda identificar posibles microservicios?

3. **Estudio del estado del arte:** En esta fase se realizó un exhaustivo estudio del estado del arte para conocer las investigaciones y metodologías existentes relacionadas con la identificación de microservicios, el procesamiento del lenguaje natural y el análisis sintáctico aplicado a requerimientos funcionales. Además, se revisaron y analizaron libros, artículos científicos y otros recursos relevantes para comprender las tendencias y enfoques actuales en el campo.

4. **Solución Candidata:** La solución propuesta en este estudio se basa en la integración de técnicas de PLN, análisis sintáctico y algoritmos de detección de comunidades. Por lo cual, se ha desarrollado un proceso que emplea el PLN para analizar un conjunto de requerimientos funcionales. A continuación, se generan grafos y se identifican comunidades, estas comunidades permiten identificar posibles microservicios.

5. **Evaluación:** En esta fase se utilizó una encuesta (***Ver Anexo B***) dirigida a profesionales de software con experiencia en microservicios con el fin de validar y determinar cuál de los conjuntos de microservicios identificados es el más acertado según su criterio.

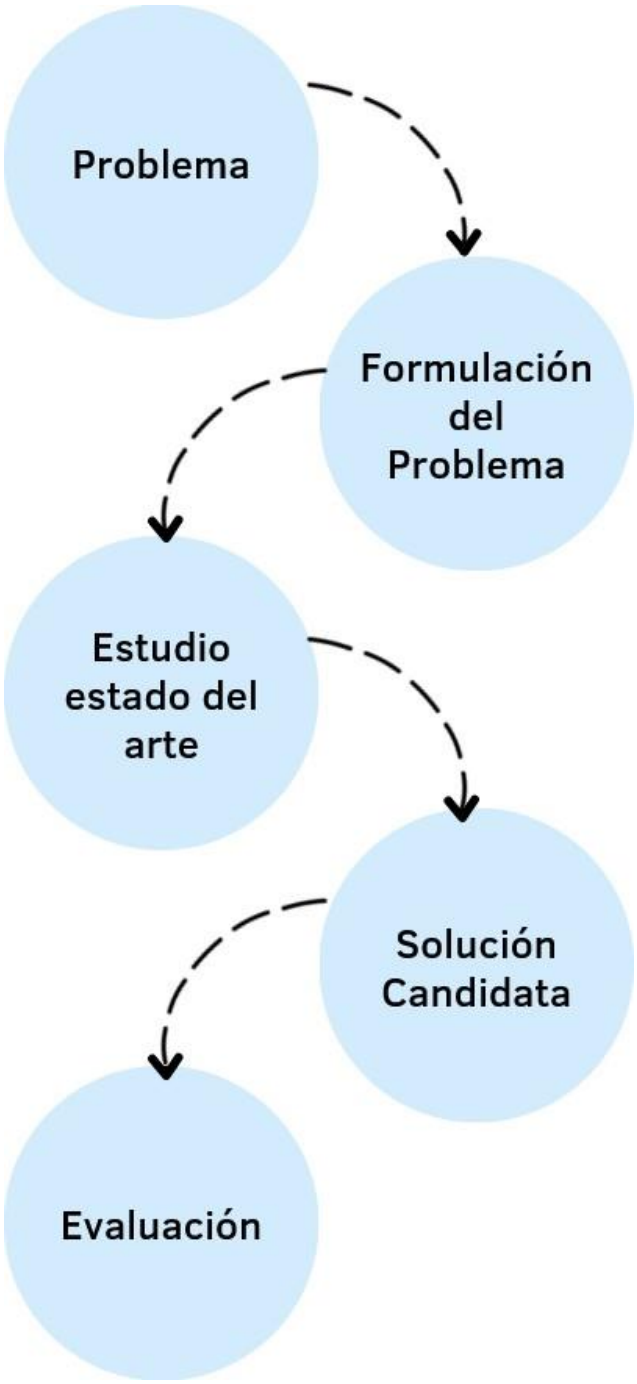


Fig 1.1: Adaptación de la Metodología Transferencia de Tecnología Basada Empíricamente.

2. Marco Teórico y Trabajos Relacionados

En este capítulo se detallará la base teórica para abordar el tema de investigación, consta de dos partes, siendo la primera el marco teórico, donde se definirán conceptos para el entendimiento del tema a tratar y la segunda los trabajos relacionados para conocer el estado actual de la investigación con respecto al tema de este trabajo de titulación.

2.1. Marco Conceptual

A continuación se explican conceptos importantes y relevantes que facilitan y esclarecen el entendimiento del proceso desarrollado.

2.1.1. Ingeniería de requerimientos

La ingeniería de requerimientos es un proceso fundamental para el éxito de cualquier proyecto de software, dado que establece las bases necesarias para el diseño, construcción e implementación del software. Según Phillip A. Laplante y Mohamad H. Kassab [7] en su libro "Requirements Engineering for Software and Systems", la ingeniería de requerimientos se considera un proceso sistemático y disciplinado que implica trabajar con las partes interesadas, tanto internas como externas, para comprender sus necesidades, expectativas y requerimientos. Una vez identificados, estos requerimientos se analizan minuciosamente para garantizar su completitud, consistencia, verificabilidad y factibilidad. Además, los requerimientos son documentados y verificados con el fin de asegurar su comprensión y aceptación por parte de todos los involucrados [8]. Es importante destacar que el propósito fundamental de la ingeniería de requerimientos es asegurar que los requerimientos del software y del sistema sean claros, precisos y completos, lo que permite a los equipos de desarrollo crear productos (software) que satisfagan las necesidades tanto de los clientes como de los usuarios finales. Por otra parte, la gestión de requerimientos también desempeña un papel crucial en el desarrollo de un proyecto, esta gestión implica definir prioridades, establecer trazabilidad, objetivos y alcances, y adaptarlos a las limitaciones del proyecto. Al abordar estos aspectos desde las fases iniciales del proceso de desarrollo de software, se logra una reducción de costos al identificar y

solucionar problemas de manera temprana.

La ingeniería por otra parte también involucra la clasificación de los requerimientos en funcionales como no funcionales, los cuales se explicarán a continuación:

1. **Requerimientos Funcionales:** Los requerimientos funcionales se refieren a las características o atributos que determinan el comportamiento que una función del sistema debe cumplir [9]. Estos requerimientos se obtienen a través de las necesidades y expectativas de los usuarios, así como de los objetivos del negocio. Según lo definido en el estándar IEEE Std 830-1998 [10], los requerimientos funcionales se centran en las acciones fundamentales que el software debe realizar al aceptar y procesar estas necesidades, estos requerimientos se enfocan en lo que el sistema debe hacer, en lugar de cómo hacerlo.

2. **Requerimientos no Funcionales:** Los requerimientos no funcionales son aquellos que se enfocan en las propiedades emergentes del software, es decir, aquellas que no están relacionadas directamente con las funciones específicas del software. Además, pueden describir una propiedad o característica que el sistema deberá mostrar o una restricción que debe cumplir. Por otro lado, implican también detallar interfaces externas entre el sistema y el mundo exterior. Estos incluyen conexiones a otros sistemas de software, componentes de hardware y usuarios, así como interfaces de comunicación. Por último, las restricciones de diseño e implementación imponen restricciones a las opciones disponibles para el desarrollador durante la construcción del producto [11].

En este trabajo de titulación se usarán únicamente los requerimientos funcionales para demostrar la validez del enfoque propuesto.

2.1.2. Sistemas monolíticos y microservicios.

En los últimos años, las arquitecturas orientadas a servicios han evolucionado hacia los microservicios, que representan una innovación dentro del desarrollo de software, eliminando el carácter monolítico del pasado y llevando al uso de nuevas estrategias de división de software que le permiten proporcionar características de alta calidad [1], a continuación se proporciona más información sobre estos tipos de sistemas.

1. **Sistemas monolíticos**

   Los sistemas monolíticos son arquitecturas de software en el que diferentes componentes como la lógica de negocios, almacenamiento de datos, etc., se combinan en un solo programa desde una sola plataforma [12], estos componentes están estrechamente

integrados y se ejecutan como uno solo. En este tipo de sistemas toda la aplicación está diseñada para ser una sola unidad, esto significa que todo el código y la funcionalidad de la aplicación están contenidos en el mismo código y los cambios en una parte del sistema pueden afectar a otras partes, lo cual es perjudicial al momento de realizar una actualización y mantenimiento, sin embargo, al no necesitar sincronización debido a que toda la información se encuentra en una sola base de datos, la comunicación se realiza a través de mecanismos intraproceso, por lo tanto no sufre los problemas típicos de la comunicación entre procesos (IPC) [13].

2. **Sistemas basados en microservicios**

   Los sistemas basados en microservicios (microservices) son una arquitectura de software que se enfoca en la construcción de aplicaciones como un conjunto de servicios independientes y pequeños, cada uno con su propia lógica de negocio e intercambiando información mediante mecanismos de comunicación ligeros y bien definidos. Estos servicios, que son pequeñas aplicaciones, se pueden desarrollar, desplegar, escalar y mantener de forma independiente.

   La arquitectura de microservicios descompone una aplicación en una colección de servicios autónomos, fácilmente desplegables, altamente cohesivos y de bajo acoplamiento, que pueden ser implementados, desplegados, escalar y actualizarse de forma independiente. Además, esta arquitectura se basa en el principio de responsabilidad única, donde cada microservicio se enfoca en una tarea o función específica [13]. La ventaja de utilizar microservicios es que permite una mayor flexibilidad y agilidad en el desarrollo, puesto que cada microservicio se puede desarrollar y desplegar de manera independiente, también, es escalable y fácil de mantener, puesto que los microservicios están altamente acoplados y tienen límites de responsabilidad definidos.

   En cuanto a las tecnologías que se utilizan para implementar esta arquitectura, destacan herramientas de contenedores como Docker [14] y Kubernetes [15], las cuales facilitan la implementación y gestión de los microservicios en entornos de producción.

2.1.3. Procesamiento de Lenguaje Natural (PLN)

El procesamiento del lenguaje natural (PLN) comenzó como una disciplina en la intersección de la inteligencia artificial y la lingüística [16] para posteriormente ser considerado como una rama de la inteligencia artificial que se centra en analizar y generar lenguaje humano mediante métodos y algoritmos computacionales [17]. Asimismo, PLN se basa en varios campos muy

diferentes por lo que requiere que los investigadores y desarrolladores de PLN de hoy amplíen significativamente su base de conocimientos [16].

Existe una gran variedad de aplicaciones basadas en PLN, desde asistentes virtuales como Siri [18] o Alexa [19], hasta sistemas de traducción automática y análisis de sentimientos en redes sociales. Adicionalmente, PLN juega un papel fundamental en el desarrollo de chatbots, sistemas de recomendaciones y análisis de datos. Para poder generar resultados confiables PLN utiliza técnicas como el análisis morfológico, la segmentación de palabras, el análisis sintáctico y semántico, la identificación de entidades y el aprendizaje automático, todo esto con el fin de entender el significado del lenguaje humano y generar respuestas coherentes [20]. Según Khurana [17] "se puede clasificar en dos partes el PLN, es decir, comprensión del lenguaje natural y generación del lenguaje natural, que desarrolla la tarea de comprender y generar el texto".

En los siguientes apartados, se abordarán dos conceptos fundamentales en PLN: la lematización y la tokenización.

- **Tokenización**, es un proceso esencial en el procesamiento del lenguaje natural, que implica dividir un texto u oración en unidades más pequeñas llamadas tokens (Ver figura 2.1). Estos tokens, que pueden ser palabras individuales o secuencias de caracteres, se utilizan ampliamente en algoritmos de procesamiento del lenguaje natural para identificar y clasificar palabras, así como para detectar patrones y estructuras en un texto [21]. Además, la tokenización desempeña un papel fundamental en el análisis sintáctico y semántico, considerando que ayuda a identificar y clasificar las palabras según su función gramatical [22].

Fig 2.1: Tokenización

- **Lematización**, en procesamiento del lenguaje natural, es el proceso de encontrar la forma base o raíz de una palabra, lo cual permite representar un conjunto de palabras que comparten la misma raíz con un único lema. Además, la lematización es útil para reducir el tamaño del vocabulario y simplificar la búsqueda y comparación de palabras en análisis de texto como se puede observar en la Figura 2.2. La lematización es comúnmente utilizada en tareas como la búsqueda de información, el análisis sintáctico y la traducción

automática, también reduce las formas flexionadas y derivadas de una palabra a una forma base común, empleando el análisis morfológico de palabras [23].

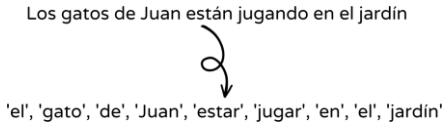


Fig 2.2: Lematización

2.1.4. Algoritmos de comparación sintáctica.

En el procesamiento del lenguaje natural, existen varios algoritmos para la comparación sintáctica o palabras que se utilizan para identificar similitudes o diferencias entre ellas. Dos de los más comunes son el algoritmo de Jaccard y SequenceMatcher.

- **El algoritmo de Jaccard:** Es un algoritmo utilizado para medir la similitud entre conjuntos y se puede aplicar a conjuntos de palabras o tokens en un texto. El algoritmo de Jaccard se basa en la comparación de la intersección y la unión de dos conjuntos para calcular su coeficiente de similitud [24].
- **SequenceMatcher:** Es un algoritmo utilizado para comparar dos secuencias de caracteres, como palabras o tokens. SequenceMatcher compara las secuencias de caracteres mediante la identificación de la secuencia más larga de caracteres coincidentes, y calcula una puntuación de similitud basada en el número de caracteres coincidentes y no coincidentes en las secuencias [25].

2.1.5. Matriz de Adyacencia

Una matriz de adyacencia es una estructura de datos que sirve para representar grafos en forma bidimensional. Dicha matriz representa el vínculo entre los nodos de un grafo, donde las filas y las columnas corresponden a los nodos, y los elementos de la matriz indican si hay una conexión entre los nodos correspondientes. En una matriz de adyacencia, el valor en la posición (i, j), los índices ”i” y ”j” representan las filas y columnas de la matriz e indica si hay una conexión entre los nodos i y j. Dependiendo del tipo de grafo que se esté modelando, los valores pueden ser binarios (1 ó 0), números enteros que representan el peso de la conexión, o incluso valores no numéricos que representan el tipo de relación entre los nodos.

Según Goodrich y Tamassia en su libro ”Estructuras de Datos y Algoritmos en Java ” [26], las matrices de adyacencia son útiles para ciertos tipos de algoritmos que requieren la identificación rápida de conexiones entre nodos, como la búsqueda en profundidad y la búsqueda en anchura.

2.1.6. Grafos

En el campo de las ciencias de la computación y las matemáticas, un grafo es una estructura matemática que permite visualizar un conjunto de datos y las relaciones que existen entre ellos. Los grafos están conformados por vértices o nodos y aristas, donde cada arista conecta los vértices existentes, tanto vértices y aristas pueden obtener valores denominados como pesos o costos.

De acuerdo con Ferozuddin Riaz, los grafos se encuentran entre los modelos más ubicuos, es decir, tienen capacidad de comunicación a través de redes interconectadas [27]. Específicamente en la informática se pueden utilizar para modelar relaciones y dinámicas de procesos en sistemas informáticos, físicos, biológicos y sociales. Existen cuatro formas de representar un grafo en un sistema informático: la representación de lista de incidencia, la representación de matriz de incidencia, la representación de lista de adyacencia y la representación de adyacencia. Los grafos se dividen en dos grandes grupos los cuales son: grafos dirigidos y grafos no dirigidos, en este trabajo se utilizan grafos no dirigidos, los cuales se explican a continuación.

1. **Grafo Dirigido** también conocido como dígrafo es aquel en el que cada arista del grafo tiene una dirección representada por una flecha que apunta al vértice o nodo a otro, este tipo de grafos pueden representar relaciones asimétricas donde un vértice está conectado a otro vértice en una dirección específica. Los grafos dirigidos se utilizan en muchas aplicaciones como: el modelado de redes sociales, análisis de flujos de tráfico y el diseño de algoritmos informáticos.

2. **Grafo No dirigido** es aquel en el cual las aristas no tienen ninguna dirección asociada, simplemente líneas que conectan los vértices, es decir cada arista representa una relación bidireccional entre dos vértices o nodos. Los grafos no dirigidos se utilizan en aplicaciones como: el análisis de redes sociales, procesamiento de imágenes y los sistemas de recomendación.

2.1.7. Clusterización

Clustering es una técnica de minería de datos y aprendizaje automático que consiste en agrupar datos similares en clústeres en función de su similitud o proximidad. Esta agrupación de datos tiene un objetivo específico que es la detección de patrones o estructuras significativas en un conjunto de datos, en algunos casos la clusterización de datos es solo el inicio para otros objetivos. Los grupos de objetos que comparten características comunes, juegan un papel importante en la forma en que las personas analizan y describen el mundo, los seres humanos son hábiles para dividir objetos en grupos (clustering) y asignar objetos a estos grupos (clasificación), además, en el contexto de la comprensión de datos, los grupos son clases potenciales y el análisis de grupos es el estudio de técnicas para encontrar automáticamente clases [28].

Matemáticamente la clusterización se la puede representar como un problema de optimización, donde el objetivo es minimizar la distancia entre los objetos dentro de un grupo mientras se maximiza la distancia entre los objetos en diferentes grupos. Se pueden utilizar varias técnicas y algoritmos matemáticos para realizar este trabajo como: k-means que es de los algoritmos más populares en la comunidad de investigación [29], agrupación jerárquica y agrupación basada en la densidad, siendo la última técnica la que se usa en este trabajo.

2.1.8. Detección de comunidades

Hoy en día empresas como Facebook utiliza teoría de grafos para representar y analizar su red social, siendo la red social un grafo, donde los nodos representan usuarios y las aristas representan conexiones entre usuarios, con este ejemplo se puede decir que la mayoría de las redes (grafos) de interés muestran una estructura comunitaria, es decir, sus vértices están organizados en grupos, llamados comunidades, conglomerados o módulos [30].
El objetivo de la detección de comunidades en grafos es identificar los módulos y posiblemente su organización jerárquica, utilizando únicamente la información codificada en la topología de grafos [31]. La identificación de comunidades puede ofrecer información sobre cómo está organizado el grafo, permite la visualización de regiones que tienen algún grado de autonomía dentro del grafo, también ayuda a clasificar los vértices, según su rol con respecto a las comunidades a las que pertenecen [30].
La problemática de la agrupación en grafos, en realidad no está bien definida, sin embargo, los elementos principales del problema se dice que son los conceptos tanto de ”comunidad” como de ”partición”, debido a que no están establecidos con rigor y requieren cierto grado de

arbitrariedad y/o sentido común [31]. Existen varios algoritmos para la detección de comunidades, sin embargo en este trabajo se usará el algoritmo de Louvain, que acorde a [32] es el algoritmo más efectivo para detección de comunidades.

2.1.9. Algoritmo de Louvain

Dado un grafo $G = (V, E)$, donde V son los vértices y E son las aristas, el objetivo del problema de detección de la comunidad es identificar una partición de vértices en comunidades (clústeres) de modo que los vértices relacionados se asignen a la misma comunidad y los vértices dispares o no relacionados se asignen a comunidades diferentes [33]. El algoritmo de Louvain es un método popular para la detección de comunidades en redes o grafos. Fue propuesto por Vincent D. Blondel, Jean-Loup Guillaume y Renaud Lambiotte en su artículo ”Fast unfolding of communities in large networks” [34] publicado en la revista Journal of Statistical Mechanics: Theory and Experiment en 2008.

El algoritmo de Louvain se basa en la optimización modular, que busca maximizar la modularidad de la red al agrupar los nodos en comunidades, un valor de modularidad cercano a 1 indica una alta modularidad, por lo tanto existen comunidades bien definidas en el grafo, caso contrario un valor cercano a 0 o negativo indica una modularidad baja, lo que significa que las comunidades no están bien definidas o no existen en el grafo. La modularidad es una medida de la estructura modular de una red, es decir, la densidad de conexiones dentro de los grupos en comparación con las conexiones entre grupos, en otras palabras, la modularidad se utiliza para medir cuánto los nodos en un grafo están agrupados o conectados de manera más densa dentro de sus respectivas comunidades en comparación con las conexiones entre comunidades. Por ejemplo, una red social representada por un grafo, donde cada nodo son personas y las aristas representan amistades entre ellas, se utiliza la modularidad para detectar grupos de personas que tienen una alta densidad de conexiones entre sí, es decir, aquellos que son más amigos entre ellos en comparación con personas fuera de ese grupo. La modularidad se calcula comparando la cantidad real de aristas dentro de las comunidades en el grafo con la cantidad esperada de aristas en una red aleatoria equivalente, si la cantidad real de aristas es mucho mayor de lo esperado, se considera que la modularidad es alta y que existen comunidades bien definidas en el grafo. A continuación se explica la fórmula para el cálculo de la modularidad (Q) [35].

$$Q = \frac{1}{2m} \sum_{i,j} \left(e_{ij} - \frac{a_i \cdot a_j}{2m} \right)$$

Tal que:

- m es el número total de aristas en el grafo.
- e_{ij} es el número de aristas entre los nodos i y j.
- a_i es el grado del nodo i, es decir, el número de aristas que conectan al nodo i.
- a_j es el grado del nodo j.

Cada término:

- $\frac{1}{2m}$: normaliza la fórmula dividiendo por el doble del número total de aristas en el grafo, así se asegura que el valor de modularidad se encuentre en el rango de −1 a 1.
- e_{ij}: es el número de aristas que conectan los nodos i y j, si i y j están en la misma comunidad, entonces e_{ij} será mayor que cero; de lo contrario, será cero.
- $\frac{a_i \cdot a_j}{2m}$: Este término representa la fracción esperada de aristas entre los nodos i y j en un grafo aleatorio equivalente, multiplica los grados de los nodos i y j y lo divide por el doble del número total de aristas en el grafo.

En la figura 2.3 se puede observar una ejemplificación de modularidad [36], para tener un concepto más claro de este concepto.

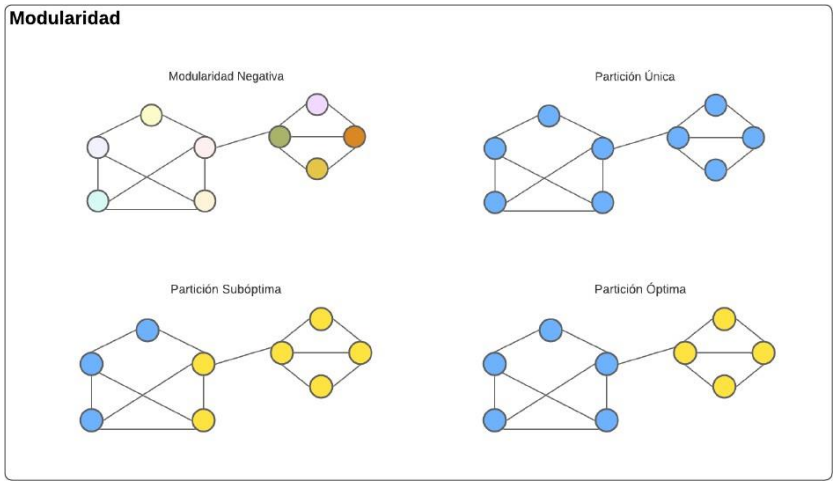


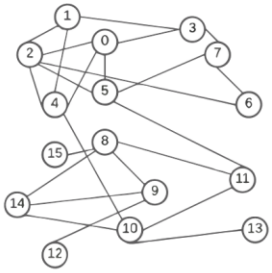
Fig 2.3: Ejemplificación de Modularidad

El algoritmo de Louvain utiliza un enfoque heurístico y de dos pasos para mejorar la modularidad de la red. En el primer paso del algoritmo, se asigna inicialmente cada nodo a su propia comunidad individual. Luego, se examina la ganancia de modularidad que se obtendría

al mover un nodo a una comunidad vecina. Este proceso se repite iterativamente para cada nodo hasta que ya no se obtiene ninguna ganancia significativa en modularidad.

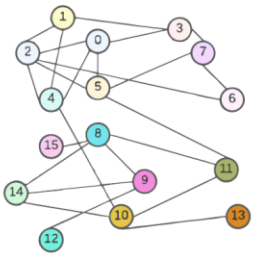
En el segundo paso del algoritmo, se crea un nuevo grafo donde los nodos representan las comunidades encontradas en el paso anterior. Las aristas del nuevo grafo se calculan agregando los pesos de las aristas internas de cada comunidad original. Este nuevo grafo se utiliza como entrada para repetir el primer paso y encontrar nuevas comunidades en una escala mayor. El algoritmo de Louvain es conocido por su eficiencia y su capacidad para detectar comunidades en grandes redes y ha sido ampliamente utilizado en diversos campos, incluyendo ciencias sociales, biología, informática y análisis de redes. Este proceso se ejemplifica en la figura 2.4 [34].

Grafo Inicial



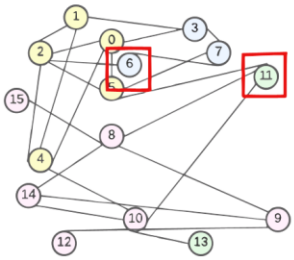
Paso 1

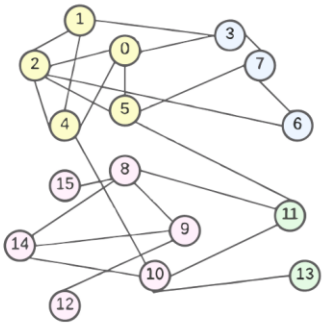
Cada nodo es asignado a su propia comunidad



Paso 2

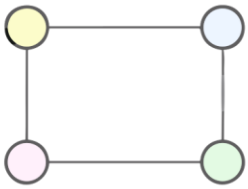
Se itera sobre cada nodo y se considera moverlo a una comunidad vecina de manera que se maximice la modularidad, que es calculada para cada movimiento posible. Para ejemplificación se se observa el nodo 11 en una comunidad en la que no se obtiene la mayor modularidad al igual que el nodo 6, sin embargo en la segunda imagen cada nodo está en la comunidad en la que la modularidad es maxima.





Paso 3

Se repite el Paso 2 hasta que no se logre ninguna mejora en la modularidad.
En la imagen se muestra la nueva red únicamente con las comunidades encontradas, en este caso cuatro comunidades.



Nota* este es un ejemplo simplificado, en escenarios del mundo real las redes pueden ser mucho más grandes y complejas.

Fig 2.4: Ejemplificación del Algoritmo de Louvain

2.2. Trabajos Relacionados

El desarrollo de software usando arquitectura de microservicios tiene como objetivo superar las deficiencias de los estilos de arquitectura monolítica, donde la interfaz de usuario, la lógica y bases de datos se encuentran en una sola aplicación, por lo cual un sistema monolítico puede ser difícil de entender y mantener. Dos aspectos que se ven favorecidos por el uso de microservicios son la descomposición funcional de una aplicación y la descentralización. Sin embargo uno de los grandes desafíos en el diseño de la arquitectura de microservicios es encontrar una partición adecuada del sistema en microservicios [37]. Por lo tanto, el diseño de microservicios generalmente se realiza de manera intuitiva, según la experiencia de los diseñadores del sistema [4].

En el artículo "Identifying Microservices Using Functional Decomposition" [4], los autores proponen un enfoque para identificar microservicios en la fase inicial de diseño que consiste en identificar las relaciones entre las operaciones requeridas del sistema y variables de estado de las operaciones, para luego visualizar las relaciones entre las operaciones del sistema y las variables de estado, por lo que se reconoce grupos de relaciones densas. Este proceso se realiza usando requerimientos de CoCoME [38], que pasan por un proceso de descomposición para luego mostrar cada microservicio en un grafo y de manera manual agrupan los nodos para obtener el microservicio.

Por otro lado, la ingeniería de requerimientos y el análisis de los mismos, es una parte esencial para cualquier proyecto de desarrollo de software, la especificación de requerimientos de software son expresados en lenguaje natural por lo que pueden ser generalmente informales, imprecisos y ambiguos, y su análisis se convierte en una tarea compleja. Aunque los métodos para el análisis automático de los requerimientos de software se han estudiado anteriormente, muchos de ellos tienen limitaciones y aún faltan investigaciones efectivas en esta área. Por lo tanto, Wang [3], propone un nuevo enfoque para extraer automáticamente los requerimientos funcionales con el fin de facilitar el análisis de requerimientos, este enfoque combina técnicas de aprendizaje automático, procesamiento del lenguaje natural y análisis sintáctico. En la primera etapa, se recopilan documentos relacionados con sistemas de comercio electrónico y sistemas de fabricantes de automóviles, a partir de estos documentos, se extraen oraciones que contienen requerimientos funcionales, la siguiente etapa se basa en el procesamiento del lenguaje natural (PLN) y consta de procesos como tokenización, lematización, etiquetado de partes del discurso (POS tagging) y análisis de dependencias. Posteriormente, se realiza la extracción de una lista adecuada de características a nivel sintáctico, la cual juega un papel

crucial para que los algoritmos de aprendizaje automático descubran patrones realmente relevantes en los datos, este enfoque ha sido rigurosamente evaluado y validado, lo que respalda la efectividad de los métodos propuestos, aunque presenta algunas limitaciones, logra capturar información detallada sobre requerimientos funcionales y los experimentos demuestran un buen rendimiento, siendo independiente del dominio y robusto hasta cierto punto [3].

El autor Vidya en su artículo "Conceptual modeling of natural language functional requirements" [39] propone otro enfoque para el modelado conceptual de los requerimientos funcionales expresados en lenguaje natural, se presenta un marco de trabajo que utiliza técnicas de análisis léxico y sintáctico para capturar de manera precisa y comprensible los requerimientos de un sistema de software. La fase de análisis de requerimientos depende en gran medida de las opiniones personales y es de naturaleza subjetiva, en la mayoría de casos las partes interesadas tienen problemas para comprender los requerimientos presentados en forma de texto sin la ayuda de un analista, pero es de gran importancia que las partes interesadas puedan comprender los requerimientos presentados por el analista de requerimientos y comprender el impacto de los requerimientos de manera uniforme y eficiente. Este enfoque tiene como objetivo mejorar la precisión y la comprensión en la recopilación de los requerimientos funcionales, lo que puede conducir a un mejor diseño y desarrollo de sistemas de software [39].

Para el preprocesamiento de los requerimientos se usan también técnicas de procesamiento de lenguaje natural, funciones como tokenización de palabras, derivación, etc. Además se usa una librería de Stanford Parser, la cual simplifica la mayoría de las actividades típicas de preprocesamiento. Por lo tanto en el preprocesamiento se limita a realizar tokenización de oraciones para identificar los límites de las oraciones. Posteriormente, se extraen las características sintácticas de cada oración y el texto en general. Para cada frase los sustantivos se derivan, es decir, se convierten a su forma base, estos resultados se combinan para obtener un conjunto final de palabras clasificadas en sus partes del discurso, finalmente se extraen las ocurrencias gramaticales que puedan existir y se crea el modelo conceptual, es decir, en este artículo se realiza un análisis sintáctico de los requerimientos establecidos explícitamente para la creación de modelos conceptuales satisfactorios, que pueden usarse dentro de la fase de requerimientos [39].

Por otra parte, el uso de técnicas de inteligencia artificial en los últimos meses ha tenido un aumento debido a su gran popularidad y está siendo utilizado tanto por empresas como por consumidores para una gran variedad de tareas, en su mayoría textuales [40]. ChatGPT ha demostrado habilidades impresionantes en muchas áreas de PLN, así como habilidades emergentes como la generación de código y la generación multimodal.

El uso de ChatGPT para el desarrollo de software también se ha visto involucrado en el último tiempo, puesto que se percibe como una herramienta prometedora. Investigaciones previas han destacado su capacidad para mejorar la calidad del código, facilitar la refactorización, agilizar la obtención de requerimientos y asistir en el diseño de software [41]. Estos estudios han proporcionado ejemplos concretos de cómo los patrones de instrucciones (prompt) en la interacción con ChatGPT han llevado a resultados más precisos y coherentes en tareas relacionadas con el desarrollo de software [42]. Estos trabajos respaldan la idea de que el uso de ChatGPT puede ser una adición valiosa en el campo del desarrollo de software, abriendo nuevas oportunidades para la mejora de procesos y la eficiencia en la creación de software de calidad [41].

White [42] propone utilizar prompts para mejorar la obtención de requerimientos de software mediante la interacción con grandes modelos de lenguaje (LLMs [43]), como ChatGPT. El uso de patrones de instrucción específicas, denominadas también como ”prompt patterns”, que guían al modelo durante las interacciones para poder realizar varias tareas entre ellas el poder obtener requerimientos de manera efectiva. Estos prompts incluyen especificar el formato de entrada de los requerimientos, solicitar ejemplos o casos de uso específicos y pedir explicaciones adicionales cuando sea necesario. El estudio demuestra cómo la aplicación de patrones de prompts puede ayudar a obtener requerimientos de software más precisos y completos al interactuar con ChatGPT. Los usuarios pueden hacer preguntas específicas sobre los requerimientos del software en desarrollo y obtener respuestas relevantes y útiles. Además, se presentan ejemplos concretos de cómo aplicar estos patrones en la práctica, estos patrones forman un catálogo amplio que busca automatizar tareas de ingeniería de software, incluyendo un patrón específico para obtener requerimientos en diferentes contextos y dominios. Es importante iterar y recibir retroalimentación continua para ajustar y mejorar los patrones a medida que se utilizan en la interacción con el modelo.

Basado en la revisión de literatura realizada, se concluye que no existe un proceso semiautomático para la identificación de posibles microservicios usando el análisis de requerimientos funcionales a través del uso de técnicas de procesamiento del lenguajes natural, grafos y detección de comunidades.

3. Proceso de creación y despliegue de microservicios a partir de requerimientos funcionales

En este capítulo, se presenta de manera detallada y formalizada el proceso propuesto para la creación y despliegue de microservicios a partir de requerimientos funcionales, se describen las fases y sub-fases del proceso, brindando una guía clara y estructurada. Además, se incluyen ejemplos cortos que ilustran cada fase del proceso, facilitando la comprensión del mismo.

El proceso inicia a partir de nueve requerimientos formalizados en casos de uso, definidos en CoCoME, a través de un caso de estudio expuesto por Rausch [38], en el que se muestra un sistema comercial para una cadena de supermercados y presenta procesos comunes como venta, pedido y recepción de productos, informes de inventario, actualización de precios, intercambio de productos entre tiendas (en stock bajo), etc. Este sistema se usa ampliamente como un caso de estudio común para el modelado y la evaluación de la arquitectura de software.
En la figura 3.1 se observa el proceso propuesto y a continuación se describe detalladamente cada una de las fases:

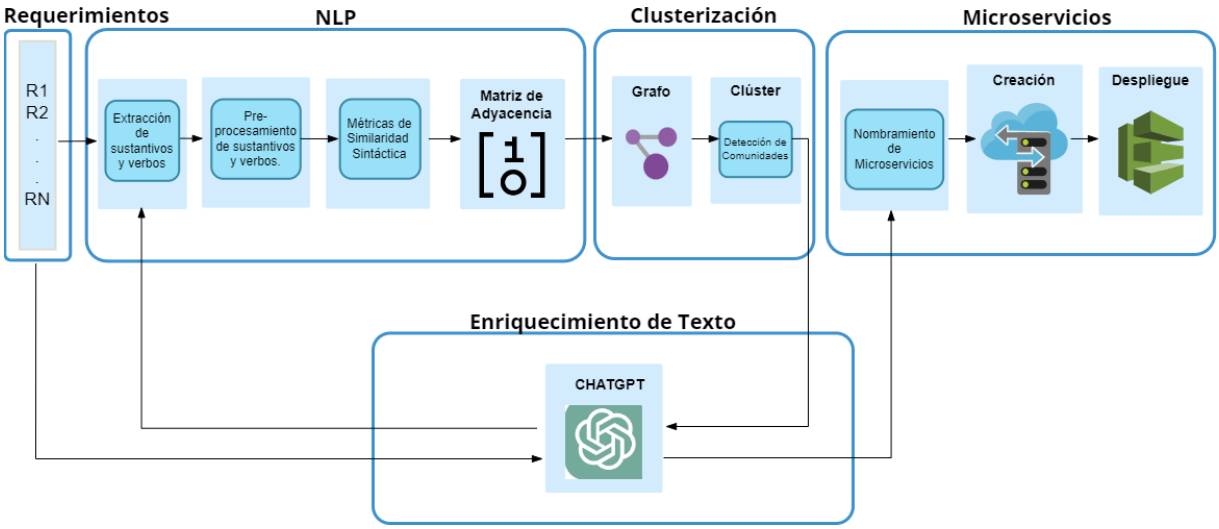


Fig 3.1: Proceso de creación y despliegue de microservicios a partir de requerimientos funcionales

3.1. Recolección de Requerimientos

Los requerimientos de CoCoME [38] se formalizan a través de casos de uso, cuyas descripciones y nombres se almacenan en una hoja de cálculo para su posterior uso. Todos los requerimientos se encuentran en el **Anexo A** en la tabla A.1.
Los requerimientos se definen formalmente en la ecuación 3.1

$$R = \{r_1, r_2, ..., r_n\} \tag{3.1}$$

donde R es el conjunto de requerimientos y donde cada elemento está representado desde r_i hasta r_n donde el intervalo de i es: $1 \leq i \leq n; \forall i \in N$ y representa un conjunto finito de n requerimientos de R, y cada r_i contiene un conjunto de palabras. Para ilustrar la validez del proceso, se ejemplificara con tres de los nueve requerimientos propuestos, los cuales se pueden observar en la tabla 3.1.

3.2. Procesamiento de Lenguaje Natural

En el proceso de creación de microservicios, el uso del Procesamiento del Lenguaje Natural (PLN) permite extraer información relevante de los requerimientos funcionales, incluyendo sustantivos y verbos mediante el uso de técnicas y algoritmos de PLN. La identificación de palabras clave y su relación puede ser analizada para mejorar la comprensión de los requer-

r_i	Descripción
r_1	At the Cash Desk the products a Customer wants to buy are detected and the payment, either by credit card or cash, is performed.
r_2	If some conditions are fulfilled a Cash Desk automatically switches into an express mode. The Cashier is able to switch back into normal mode by pressing a button at his Cash Desk. To indicate the mode the Light Display shows different colors.
r_3	The Trading System provide the opportunity to order product items.

Tabla 3.1: Requerimientos de

CoCoMe.(*R*)imientos, por lo cual se siguieron las siguientes sub-

fases.

3.2.1. Extracción de sustantivos y verbos

La extracción de sustantivos y verbos de un requerimiento es crucial, puesto que estos elementos son necesarios para comprender y expresar la semántica y la estructura organizativa del requerimiento. Por un lado, los sustantivos representan las entidades sobre las cuales se definen las funcionalidades del sistema mientras que los verbos describen las acciones o comportamientos que se esperan por parte del sistema. Es a través del análisis de las relaciones entre sustantivos y verbos, identificándolos y extrayéndolos, lo que facilita la comprensión de la situación y el desarrollo de soluciones adecuadas.

La aplicación de técnicas de procesamiento de lenguaje natural permite la extracción de los sustantivos y verbos necesarios de cada requerimiento, se representa de la siguiente manera:

$$PLN(R) = S, S_o \tag{3.2}$$

$$PLN\ (R) = V, V_O \tag{3.3}$$

Donde $S = \{s_1, s_2, \quad s_n\}$, cada *si* representa un elemento de S y $1 \leq i \leq n; \forall n \in \mathrm{N}$ representa el conjunto finito de n sustantivos y $V = \{v_1, v_2, \quad v_n\}$, cada *vi* representa un elemento de V y $1 \leq j \leq n; \forall n \in \mathrm{N}\}$ representa el conjunto finito de n verbos de V , obtenidos a partir de los requerimientos de R.

La tecnología utilizada para el procesamiento del lenguaje natural (*ecuaciones 3.2 y 3.3*) es el modelo pre entrenado ”en_core_web_sm” [44] de Spacy [45], una de las librerías más sólidas en el campo del PLN. Este modelo es conocido por su capacidad para aplicar diversas técnicas de PLN, como el análisis morfológico, la lematización, el análisis sintáctico, la identificación de entidades nombradas y la clasificación de dependencias. Puesto que, al emplear estas técnicas, el modelo es capaz de analizar, comprender y extraer información de los requerimientos

expresados en inglés.

El proceso se ejemplifica en la figura 3.2 donde se ilustra los sustantivos extraídos, mientras que la figura 3.3 muestra los verbos identificados en los requerimientos, basándose en una muestra de tres requerimientos.

```
['CashDesk',
 'Product',
 'Customer',
 'Payment',
 'CreditCard',
 'Cash',
 'Condition',
 'CashDesk',
 'ExpressMode',
 'Cashier',
 'NormalMode',
 'Button',
 'CashDesk',
 'Mode',
 'LightDisplay',
 'DifferentColor',
 'TradingSystem',
 'Opportunity',
 'ProductItem']

Total Sustantivos: 19
```

Fig 3.2: Sustantivos Identificados (*S*)

```
['want',
 'buy',
 'detect',
 'detect',
 'perform',
 'perform',
 'fulfil',
 'fulfil',
 'switch',
 'switch',
 'press',
 'indicate',
 'show',
 'provide',
 'order']

Total Verbos: 15
```

Fig 3.3: Verbos Identificados (*V*)

Spacy [45] también permite obtener información como el lema, el nombre del sustantivo y la raíz del sustantivo que se utilizan para establecer la relación con los verbos. De esta manera, se obtiene el conjunto representado en la ecuación 3.4 y se visualiza de manera gráfica en la figura 3.4.

$$S_O = \{(a_1, b_1, s_1), (a_2, b_2, s_2), \ldots, (a_n, b_n, s_n)\} \quad (3.4)$$

```
(nombre='the cash desk', lema='cashdesk', raiz='at')
(nombre='the products', lema='product', raiz='detect')
(nombre='a customer', lema='customer', raiz='want')
(nombre='the payment', lema='payment', raiz='detect')
(nombre='credit card', lema='creditcard', raiz='by')
(nombre='cash', lema='cash', raiz='card')
(nombre='some conditions', lema='condition', raiz='fulfil')
(nombre='a cash desk', lema='cashdesk', raiz='switch')
(nombre='an express mode', lema='expressmode', raiz='into')
(nombre='the cashier', lema='cashier', raiz='be')
(nombre='normal mode', lema='normalmode', raiz='into')
(nombre='a button', lema='button', raiz='press')
(nombre='his cash desk', lema='cashdesk', raiz='at')
(nombre='the mode', lema='mode', raiz='show')
(nombre='the light display', lema='lightdisplay', raiz='show')
(nombre='different colors', lema='differentcolor', raiz='show')
(nombre='the trading system', lema='tradingsystem', raiz='provide')
(nombre='the opportunity', lema='opportunity', raiz='provide')
(nombre='product items', lema='productitem', raiz='order')

Total Sustantivos: 19
```

Fig 3.4: Sustantivos con nombre, lema y raíz (S_O)

Donde cada elemento de S_O se representa en la ecuación 3.5.

$$s_o i = \{(a_i, b_i, s_i) | a_i \in \text{Nombre}, \ b_i \in \text{Raíz}, \ s \in S\} \quad (3.5)$$

Donde a_i representa el nombre del sustantivo, b_i representa la raíz del sustantivo y s_i repre- senta el lema del sustantivo, donde s_i son elementos que forman parte del conjunto S. Además, los conjuntos Nombre, Raíz y S contienen todos los nombres, lemas y raíces posibles respec- tivamente, estos conjuntos se obtienen a partir de cada requerimiento en R.

Por otro lado, de los verbos se obtiene información como el lema y el verbo (verbo o verbo + auxiliar), la cual se obtiene el conjunto representado en la ecuación 3.6 y se visualiza de manera gráfica en la figura 3.5.

$$V_O = \{(c_1, v_1,), (c_2, v_2), \ldots, (c_n, v_n)\} \quad (3.6)$$

```
(nombre='wants', lema='want')
(nombre='buy', lema='buy')
(nombre='are detected', lema='detect')
(nombre='detected', lema='detect')
(nombre='is performed', lema='perform')
(nombre='performed', lema='perform')
(nombre='are fulfilled', lema='fulfil')
(nombre='fulfilled', lema='fulfil')
(nombre='switches', lema='switch')
(nombre='switch', lema='switch')
(nombre='pressing', lema='press')
(nombre='indicate', lema='indicate')
(nombre='shows', lema='show')
(nombre='provide', lema='provide')
(nombre='order', lema='order')

Total Verbos: 15
```

Fig 3.5: Verbos con nombre y lema (V_O)

Donde cada elemento de V_O se representa en la ecuación 3.7.

$$v_{oi} = \{(c_i, v_i) | c_i \in \text{Nombre}, \ v \in V\} \tag{3.7}$$

Donde c_i representa el nombre del verbo y v_i es el lema correspondiente, donde v_i pertenece al conjunto V. A su vez, los conjuntos Nombre y V contienen todos los posibles nombres y lemas que se obtienen a partir de cada requerimiento en R

3.2.2. Pre-procesamiento de Sustantivos y Verbos

La siguiente fase es el preprocesamiento de sustantivos y verbos, la cual es una fase crítica para el proceso propuesto. Esta fase involucra eliminar sustantivos y verbos duplicados y eliminar palabras sin significado, como: artículos, pronombres y preposiciones [46]. El propósito de esta fase es mejorar la calidad y precisión de los sustantivos y verbos, permitiendo así una comprensión más profunda y de calidad de los requerimientos funcionales.

Para poder llevar a cabo la limpieza de los verbos y sustantivos se utilizaron en primer lugar la lista de todos los signos de puntuación [47] dados por Python y en segundo lugar la librería NLTK [48] que provee palabras sin significado [46] conocidas comúnmente como ”stopwords”. La eliminación de palabras sin significado reduce la complejidad de los sustantivos y verbos obtenidos, facilitando su análisis.

Para la eliminación de duplicados, se puede utilizar el método ”set” [49] de Python. Al convertir la lista en un conjunto con este método, se eliminan automáticamente todos los elementos duplicados y se obtiene una lista única y sin redundancias que facilita el análisis de verbos y sustantivos para su procesamiento posterior.

Por lo tanto, el uso del método ”set” es una solución eficiente y rápida para manejar grandes volúmenes de datos.

Posteriormente, se convierten en singular todos los sustantivos y verbos tanto de S como V.

$$(set \circ NLTK)(S) = set(NLTK(S)) = S_2 \tag{3.8}$$

El resultado final es un nuevo conjunto S_2 que contiene los elementos únicos y limpios de sustantivos del conjunto S.

$$(set \circ NLTK)(V) = set(NLTK(V)) = V_2 \tag{3.9}$$

El resultado final es un nuevo conjunto V_2 que contiene los elementos únicos y limpios de verbos del conjunto V.

Al aplicar estas dos técnicas, se puede observar en la figura 3.6 la presencia de sustantivos y su frecuencia de aparición en el conjunto S_O. En la figura 3.7 se muestra una representación de los verbos presentes en el conjunto V_O y la cantidad de veces que se repiten. Además, en ambas figuras se incluye el número de verbos y sustantivos eliminados de sus respectivos conjuntos.

Sustantivo	# Repeticiones
CashDesk	3

Total sustantivos eliminados: 2

Fig 3.6: Sustantivos Repetidos en (S)

Verbo	# Repeticiones
detect	2
perform	2
fulfil	2
switch	2

Total verbos eliminados: 4

Fig 3.7: Verbos Repetidos (V)

En la figura 3.8, se aprecia una reducción de dos sustantivos del conjunto *S* y se obtiene el conjunto S_2, al igual que en la figura 3.9, donde se observa una disminución significativa de cuatro verbos del conjunto *V* y se obtiene el conjunto V_2.

```
['cash',
 'customer',
 'expressmode',
 'condition',
 'productitem',
 'mode',
 'payment',
 'cashier',
 'opportunity',
 'lightdisplay',
 'button',
 'differentcolor',
 'cashdesk',
 'product',
 'creditcard',
 'tradingsystem',
 'normalmode']

Total Sutantivos: 17
```

Fig 3.8: Nuevo conjunto de Sustantivos (S_2)

```
['buy',
 'indicate',
 'order',
 'press',
 'provide',
 'fulfil',
 'show',
 'perform',
 'switch',
 'want',
 'detect']

Total Verbos: 11
```

Fig 3.9: Nuevo conjunto de Verbos (S_2)

3.2.3. Métricas de Similaridad Sintáctica

La utilización de métricas de similaridad es crucial para evaluar la similitud sintáctica entre conjuntos de sustantivos y verbos. Estas métricas permiten determinar qué términos deben ser reemplazados en base a la comparación de elementos dentro de cada conjunto. Por lo tanto, se aplican los algoritmos de *Jaccard* [24] y *SequenceMatcher* [25] para realizar una comparación sintáctica de los conjuntos de sustantivos (S_2) y verbos (V_2) obtenidos en la fase anterior. A través de estos algoritmos, cada elemento de cada conjunto se compara con todos los demás

elementos del mismo conjunto S_2 con S_2 y V_2 con V_2 respectivamente. Posteriormente, se calcula el promedio de los resultados obtenidos por ambos algoritmos, lo que genera una métrica de similitud que varía entre 0 y 1. En base en esta métrica de similitud se procede a reemplazar los verbos y sustantivos similares en los conjuntos correspondientes.

$$Similar(S_2,\ S_2) = S_3 \tag{3.10}$$

$$Similar(V_2,\ V_2) = V_3 \tag{3.11}$$

El resultado de aplicar dichas métricas de similaridad son los conjuntos S_3 y V_3 respectivamente. En la figura 3.10 se observan los sustantivos que fueron eliminados del conjunto S_2 debido a su similitud con otros sustantivos presentes en el conjunto. Es relevante mencionar que no fue posible eliminar verbos similares, ya que al emplear únicamente tres requerimientos se logra una notable reducción en la información obtenida, no obstante, al ejecutar todos los requerimientos, se observa un aumento en la sustitución de sustantivos y verbos debido a la mayor cantidad de información presente.

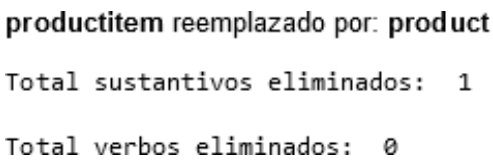


Fig 3.10: Sustantivos Reemplazados

Los resultados de esta fase se pueden visualizar en las Figuras 3.11 y 3.12, donde se muestran los conjuntos de verbos y sustantivos respectivamente, después de aplicar las métricas de similitud sintáctica previamente mencionadas.

```
['lightdisplay',
 'customer',
 'mode',
 'button',
 'opportunity',
 'cashier',
 'differentcolor',
 'expressmode',
 'tradingsystem',
 'normalmode',
 'product',
 'payment',
 'creditcard',
 'cash',
 'cashdesk',
 'condition']

Total Sustantivos: 16
```

Fig 3.11: Nuevo conjunto de Sustantivos (S_3)

```
['perform',
 'detect',
 'fulfil',
 'want',
 'order',
 'indicate',
 'provide',
 'buy',
 'show',
 'switch',
 'press']

Total Verbos: 11
```

Fig 3.12: Nuevo conjunto de Verbos (V_3)

3.2.4. Matriz de Adyacencia

Continuando con el proceso, se procede a generar la matriz de adyacencia con el fin de representar las relaciones de conexión entre los elementos de los conjuntos S_3 y V_3 en un grafo de palabras. Para ello, se sigue un proceso que se basa en la información obtenida de los conjuntos S_3 y V_3.

Este proceso implica identificar los verbos y sustantivos relevantes para la matriz de adyacencia, para lograr esto, se debe recorrer los conjuntos V_3 y S_3. Primero, se recorre cada elemento de V_3 para que cada verbo obtenido actúe como una fila de la matriz. Luego, se recorren los elementos de S_3, los cuales actuarán como columnas de la matriz. En otras palabras, por cada verbo en V_3, se recorre cada uno de los sustantivos en S_3, este proceso se muestra visualmente en la figura 3.13.

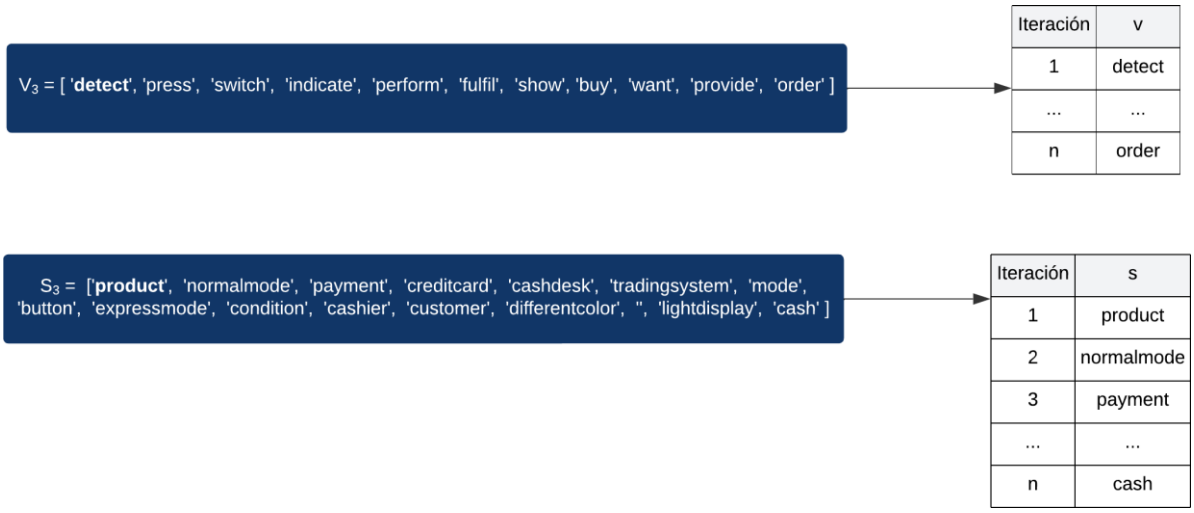


Fig 3.13: Recorrido de V_3 y S_3

Durante el recorrido de cada verbo en V_3 y cada sustantivo en S_3, se realiza una comparación con cada elemento del conjunto S_O. Es importante destacar que cada elemento de S_O se compara individualmente con cada sustantivo de S_3 y cada verbo de V_3, respectivamente. Para esta comparación, se considera el lema del sustantivo y la raíz del verbo, dicha comparación nos permite identificar posibles conexiones, la formalización de este proceso se muestra en laecuación 3.12.

$$S_4 = s.lema \mid s_3i \cap s.\text{lema} \wedge v \cap s.\text{root} \wedge s \in S_O \wedge v \in V_3 \wedge s_3i \in S_3 \qquad (3.12)$$

El conjunto S_4 se obtiene al filtrar S_O y retener aquellos sustantivos cuyas raíces son verbos en V_3 y cuyos lemas pertenecen a S_3, como se puede observar en la figura 3.14.

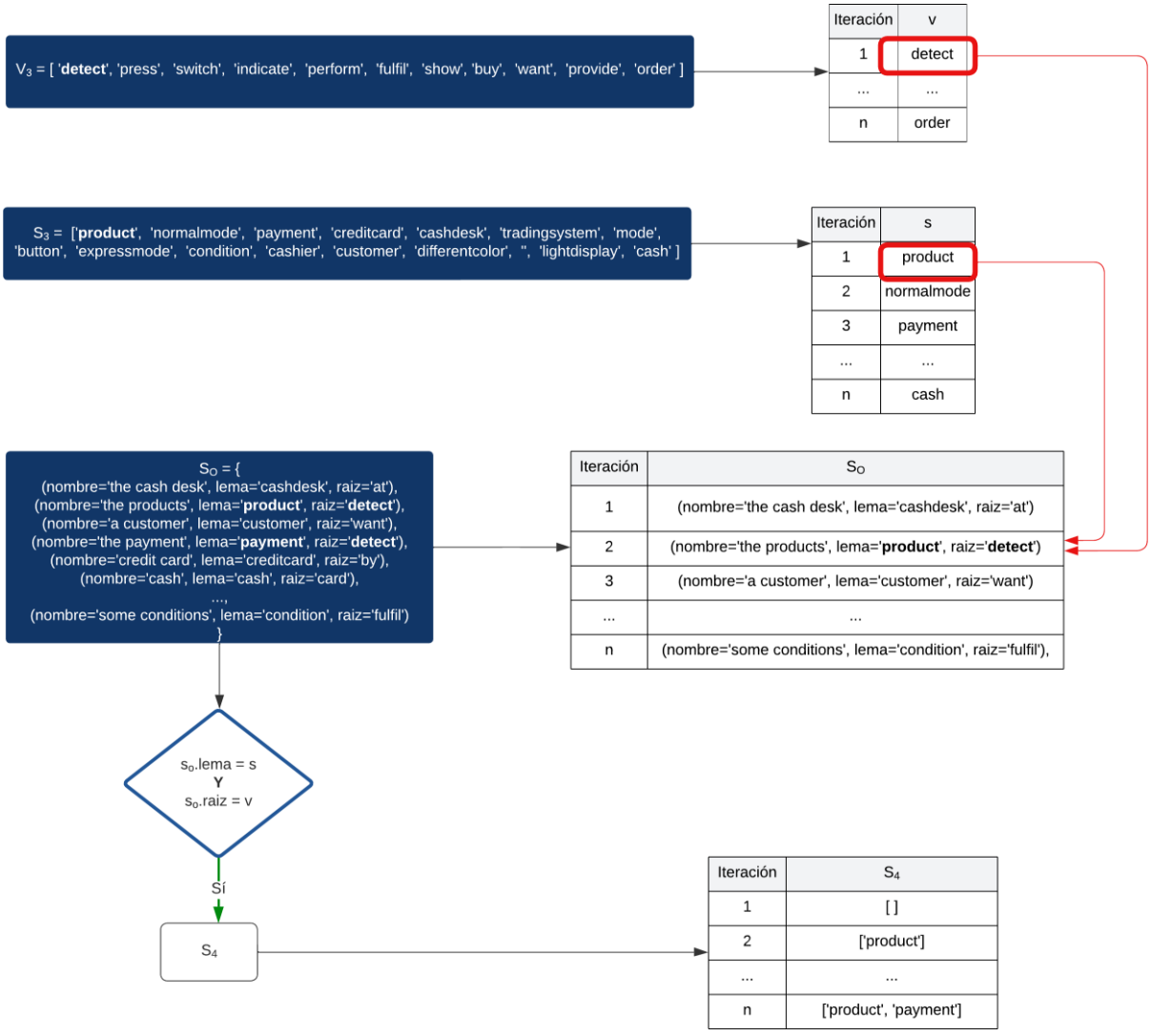


Fig 3.14: Ejemplo de S_4

Es importante recalcar que en el último paso presentando en la Figura 3.14, los sustantivos y verbos que se comparan con cada lema y raíz deben ser exactamente iguales y no similares, debido a que en la sección 3.2.3 se llevó a cabo una sustitución previa de todos los verbos y sustantivos similares.

Si el tamaño de S_4 es distinto de cero, indica que existe una conexión entre un elemento s_{i3} del conjunto S_3 y un verbo v en el conjunto V_3. Por lo tanto la matriz de adyacencia se representacomo:

$$M_0[i,j] = \begin{cases} 1 & \text{si } len(S_4) > \\ 0 & \text{si } len(S_4) \leq 0 \end{cases} \quad (3.13)$$

Donde si la longitud de S_4 es mayor que cero, se asigna el valor de 1 en la formación de lamatriz, caso contrario se asigna el valor de 0.

Mediante las fases previamente descritas, que incluyen la extracción de sustantivos y verbos, la eliminación de duplicados y la aplicación de métricas de similitud sintáctica, se logró obtener una representación más refinada y estructurada de los requerimientos funcionales. Con base en esta información procesada, y la definición anteriormente mencionada, en la figura 3.15, se presenta la matriz de adyacencia $M[i, j]$.

	button	cash	cash desk	cashier	condition	credit card	customer	differen tcolor	express mode	light display	mode	normal mode	opportunity	payment	product	trading system
detect	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
provide	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
show	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0

Fig 3.15: Matriz de Adyacencia

3.3. Clusterización

En esta fase, se llevará a cabo un proceso de agrupación de datos basado en sus características similares, como se ha descrito en capítulos anteriores. Para lograr esto, se empleará el algoritmo de Louvain, conocido por su capacidad para identificar comunidades o grupos dentro de conjuntos de datos.

3.3.1. Generación del Grafo

Una vez obtenida la matriz de adyacencia M se representa en un grafo los valores de i y j que contiene entidades y relaciones que han sido procesadas en fases anteriores, esta representación gráfica permite un análisis sencillo de las relaciones entre las entidades, así como la identificación de cualquier patrón o grupo que pueda estar presente en los datos. Formalmente esto se expresa como:

$$N = \{1, 2, \ldots, n\} \tag{3.14}$$

$$E = \{(i, j) \in N \times N \mid M[i, j] \neq 0\} \tag{3.15}$$

$$G = (N, E) \tag{3.16}$$

Donde N es el conjunto de nodos y n es el número total de nodos del grafo G. La ecuación 3.15 referente a las aristas, utiliza la condición de que la arista (i,j) existe si y sólo si el valor de M[i,j] es diferente de cero. La ecuación 3.16 define el grafo G como una tupla de su conjunto de

nodos N y su conjunto de aristas E. En la figura 3.16 se muestra el resultado de este proceso a partir de la matriz de adyacencia obtenida anteriormente.

Fig 3.16: Generación del grafo

Una vez obtenido el grafo G a partir de la matriz M[i,j], se procede a verificar la existencia de nodos aislados. Si se detectan nodos que no tienen conexión con ningún otro nodo del grafo, se eliminan del mismo para simplificar la estructura y evitar posibles errores en el análisis posterior. La eliminación de nodos aislados también puede ayudar a simplificar el gráfico y facilitar su interpretación.

$$Eliminar\_Aislados(G) = G' \tag{3.17}$$

El resultado de este proceso se muestra a continuación en la figura 3.17

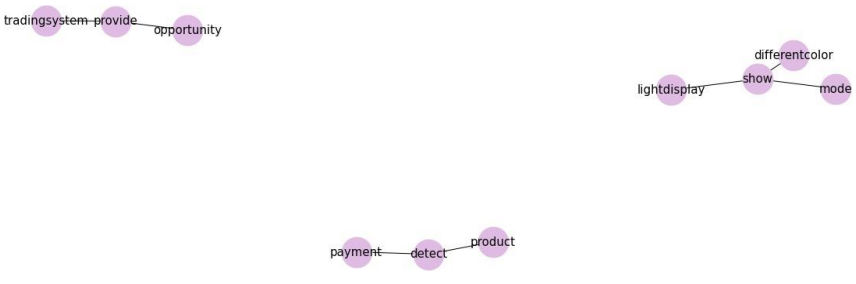


Fig 3.17: Grafo sin nodos aislados

Para la representación del grafo se usó la librería Networkx de Python, que es una herramienta sumamente útil para el manejo de grafos y permite trabajar con estos de forma dinámica y fácil de entender. Como se puede observar en la figura 3.17 existen únicamente tres subgrafos que

no se encuentran conectados entre sí, por lo que a simple vista se pueden visualizar tres comunidades, cabe recalcar que este es un caso aislado debido a la cantidad de requerimientos usados para la ejemplificación.

3.3.2. Detección de Comunidades

En esta fase se utiliza el algoritmo de *Louvain* para detectar comunidades en función de la densidad de conexiones presentes en el grafo. Esta fase es crucial para identificar grupos significativos de entidades y relaciones dentro del conjunto de datos. El grafo utilizado en este caso es el que se representa en la ecuación 3.17.

$$Louvain(G') = C \tag{3.18}$$

Donde $C = \{C_1, C_2, \ldots C_n | 1 \leq i \leq n; \forall i \in N\}$ representa el conjunto finito de i Comunidades. Donde cada comunidad C_n es a su vez un conjunto de nodos del grafo G', es decir:

$$Ci = \{n_{i,1}, n_{i,2}, \ldots, n_{i,j}\} \tag{3.19}$$

Donde $n_{i,j}$ representa el nodo j-ésimo de la comunidad i-ésima.

Para el análisis del grafo, se usaron varias librerías de python para comprobar que el proceso sea válido, lo que aumenta la precisión y la confiabilidad de la detección de comunidades. Las librerías utilizadas fueron: CDlib [50], yFiles [51] y Networkx [52], que brindan herramientas para el manejo de grafos. A continuación en la figura 3.18 se muestra el diccionario obtenido con el nombre del nodo como ”clave” y la comunidad como ”valor”.

Palabra	# Comunidad
show	0
lightdisplay	0
mode	0
differentcolor	0
product	1
detect	1
payment	1
opportunity	2
tradingsystem	2
provide	2

Fig 3.18: Diccionario de Comunidades

La representación gráfica del diccionario de comunidades se muestra en la figura 3.19. Se puede observar el resultado de la aplicación del algoritmo de Louvain donde cada color representa una comunidad.

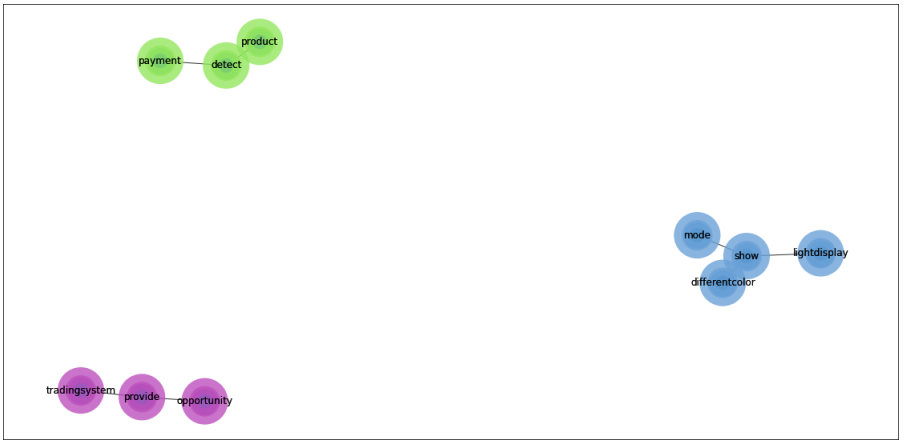


Fig 3.19: Detección de Comunidades

Cabe recalcar que en el caso del ejemplo presentado, se usan únicamente tres requerimientos sin ningún tipo de enriquecimiento, por lo cual el grafo que se obtiene en la figura 3.17 sin nodos aislados es reducido y a simple vista se pueden diferenciar tres comunidades claramente definidas sin necesidad de agregar un algoritmo de detección de comunidades, sin embargo, como se muestra en el Anexo A, en las tablas A.1 y A.2, donde existen más requerimientos, por lo tanto más información, el grafo inicial no es una red simple, entonces es necesario aplicar el algoritmo de Louvain para detección de comunidades.

3.4. Microservicios

A continuación se usarán las comunidades detectadas en el grafo para construir los posibles microservicios, dándoles un nombre que se ajuste a su funcionalidad.

3.4.1. Nombramiento de Microservicios

Una vez obtenido cada valor de $C_1, ..., C_n$, cada elemento representa un posible microservicio, por lo tanto, los nombres de estos nodos se ingresan en ChatGPT para obtener un nombre genérico para cada microservicio.

$$ChatGpt(C) = Micro \tag{3.20}$$

Donde $Micro = \{Micro_1, Micro_2, \ldots Micro_n | 1 \leq i \leq n; \forall i \in N\}$ representa el conjunto finito de n Microservicios. Para esto se utiliza la librería **openia** [53] que es provista por ChatGPT para código abierto, en la cual se generó un prompt donde se especifican los requerimientos a manera de contexto y el diccionario con las comunidades, así, los nombres sugeridos por esta inteligencia artificial son los se muestran en la figura 3.20:

```
1. Payment Microservice
2. Light Display Microservice
3. Trading System Microservice
```

Fig 3.20: Nombramiento de Microservicios

Se debe resaltar que para obtener los resultados más precisos posibles al momento de armar el prompt para utilizar en ChatGPT se pasa no solamente la lista de requerimientos iniciales sino también la lista de comunidades obtenidas en el proceso, por lo tanto, de esta forma ChatGPT tiene el contexto necesario para crear una respuesta acertada de acuerdo a cada comunidad encontrada.

3.4.2. Creación de Microservicios

Habiendo obtenido el nombre de cada microservicio, la siguiente fase es la creación de cada microservicio, esto implica toda la parte relacionada con la programación y escritura de código, para esto se usó SpringBoot [54], que es una herramienta reconocida para desarrollar microservicios, proporcionando un conjunto de herramientas que simplifican el desarrollo, la im-

plementación y la administración de aplicaciones basadas en microservicios. Es importante tener en cuenta que se puede analizar las comunidades y el grafo en general para poder identificar posibles métodos del microservicio, pero esto queda a consideración del desarrollador. En las figuras 3.21, 3.22, y 3.23 se muestra la creación y la implementación del código de los microservicios detectados en el ejemplo propuesto, las figuras ilustran el proceso de desarrollo y la estructura del código de los microservicios identificados.

Fig 3.21: ”Payment Microservice”

Fig 3.22: ”Light Display Microservice”

Fig 3.23: ”Trading System Microservice”

3.4.3. Despliegue de Microservicios

Como última fase del proceso propuesto, se realizó el despliegue de los microservicios identificados, esto implica la implementación y configuración de cada microservicio, asegurando su disponibilidad y escalabilidad. El despliegue exitoso de los microservicios permite poner en marcha el sistema basado en microservicios, brindando una arquitectura flexible y modular que cumple con los requerimientos funcionales identificados en fases anteriores del proceso. A continuación se muestra la formalización de esta fase.

$$Desplegar(M) \tag{3.21}$$

En las figuras 3.24, 3.25 y 3.26 se muestra el despliegue de los microservicios, en un entorno local de desarrollo, estas figuras ilustran la implementación de un método *GET* genérico. Cabe destacar que los detalles específicos de los endpoints y su configuración dependerán del criterio de desarrollo y del progreso del proyecto.

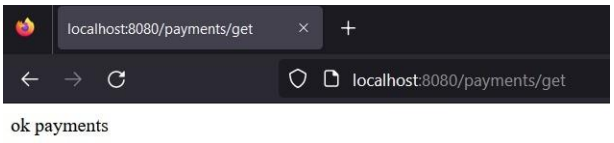


Fig 3.24: Despliegue de "Payment Microservice"

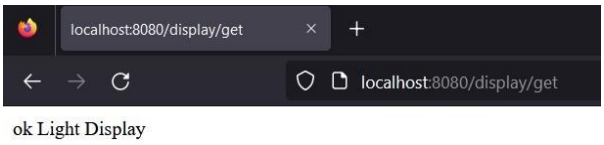


Fig 3.25: Despliegue de "Light Display Microservice"

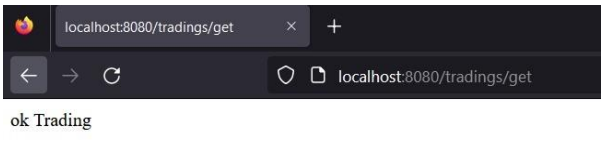


Fig 3.26: Despliegue de "Trading System Microservice"

3.5. Enriquecimiento de Texto

Un desafío importante que se presenta con los requerimientos es el hecho de que el texto siempre omite información relevante. Para el ser humano deducir esta información y recuperar el sentido no requiere de esfuerzo, sin embargo para las máquinas el procesamiento de lenguaje natural no es tarea fácil. [55]. Es por esta razón, que los requerimientos iniciales fueron procesados de manera que brinden más información para lograr un mejor resultado.

3.5.1. ChatGPT

ChatGPT ofrece diversas posibilidades para enriquecer el texto, como la corrección de errores gramaticales, la sugerencia de sinónimos o alternativas léxicas, la generación de resúmenes y la ampliación de conceptos y textos, entre otros. Además, puede ser programado para responder preguntas específicas sobre un tema en particular, lo que resulta especialmente útil en esta fase de investigación.

En la figura 3.27 se puede observar cómo el uso de ChatGPT, aplicando el ”prompt” que se puede visualizar en el ***Anexo C***, enriquece cada uno de los requerimientos iniciales propuestos en CoCoMe [38]. La lista completa de requerimientos se encuentra en el **Anexo A**, en la tabla A.2. Formalmente, esto se describe de la siguiente manera:

$$ChatGPT\ (R) = R' \tag{3.22}$$

Una vez enriquecidos los requerimientos(figura 3.27), se procede a ejecutar nuevamente el proceso descrito anteriormente y aplicar todos las fases propuestos, con el objetivo de comparar los resultados obtenidos en ambas ejecuciones.

```
['At the Cash Desk, customers can purchase products using either a credit card or cash. The products a customer wants to b
uy are detected using a barcode scanner or similar device. The customer then pays for their items using the credit card or
cash. The payment is processed and the customer is given a receipt for their purchase. The cashier can also provide any ad
ditional information or assistance the customer may need.',
 'The cash desk automatically switches into express mode when certain conditions are met. This mode allows the cashier to
process payments quickly, as the cash desk is optimized for speed. The cashier can switch back to normal mode by pressing
a button at the cash desk. The light display on the cash desk will indicate the current mode, usually with different color
s for express and normal modes.',
 'The Trading System provides an opportunity to order product items from a variety of different suppliers. This system all
ows businesses to compare prices from different suppliers and select the best deal for their needs. It also allows busines
ses to track their orders and receive notifications when new products are available. The system also provides tools for tr
acking inventory and managing customer relationships. Additionally, the system allows businesses to search for products an
d compare prices from different suppliers. This helps businesses save time and money by finding the best deals for their n
eeds.']
```

Fig 3.27: Enriquecimiento con ChatGPT.

3.6. Pseudocódigo

En el algoritmo 1 se proporciona una representación lógica del proceso generado para expresar el flujo, fases y operaciones que se siguieron para llegar a la solución del problema planteado.

Algorithm 1: Proceso general para obtener microservicios mediante requerimientos funcionales

```
Input: R ;                                  /* R representa la lista de requerimientos */
Result: Micros ;                  /* M representa la lista de microservicios recomendados */
S, SO ← pln_nouns(R);
V,  VO  ←
pln_verbs(R); S2 ←
set(nlkt(S));
V2 ← set(nlkt(V ))
S3  ←
similar(S2); V3 ←
similar(V2);
M [i, j] ← [];
for lemav en V3 do
    listav ← [];
    for verbo en VO do
        if verbo.lema == lemav then
            listav.agregar(verbo.lema)
        end
    end
    fila ← [];
    for lemas en S3 do
        contador ← 0;
        lista_s ← [];
        for sustantivo en SO do
            if (sustantivo.lema == lemas and sustantivo.raiz en listav) then
                listas.agregar(sustantivo.lema);
            end
        end
        contador ← longitud(lista_s);
        fila.agregar(contador);
    end
    total_conexiones ← suma(fila);
    if total_conexiones > 1 then
        M [i, j].agregar(fila);
    end
end
G = (M [i, j]) ;          /* G representa el grafo generado a partir de la Matriz M [i, j] */
G' = Eliminar_Aislados(G);
C = Louvain(G') ;/* C representa la lista de comunidades obtenidas a partir de G' */
Micros = ChatGPT (C);
```

4. Resultados y Evaluación

En este capítulo, se presentan los resultados obtenidos y la evaluación realizada sobre el proceso propuesto en este trabajo de titulación, tanto la comparación y análisis de resultados como la evaluación son fundamentales para corroborar la eficacia, eficiencia y viabilidad del proceso propuesto.

4.1. Resultados

La tabla 4.1 muestra la comparación entre la aplicación del proceso para dos conjuntos diferentes de requerimientos, siendo el primero los requerimientos iniciales de CoCoMe y el segundo conjunto enriquecido con ChatGPT, en términos de conteo de sustantivos y verbos iniciales, sustantivos y verbos procesados y el número de comunidades.

En los requerimientos iniciales de CoCoMe, se identificaron, mediante PLN, 94 sustantivos y 57 verbos, mientras que en los requerimientos enriquecidos con ChatGPT se lograron identificar 215 sustantivos y 143 verbos. Estos representan los sustantivos y verbos que aparecen al inicio del proceso, en cuanto a los sustantivos y verbos procesados después de aplicar eliminación de duplicados y métricas de similaridad sintáctica, en los requerimientos iniciales se redujo el número a 53 sustantivos y 38 verbos, y en los enriquecidos se redujo a 99 sustantivos y 64 verbos.

Además, ambos sistemas presentan un recuento de comunidades obtenidos a partir de los sustantivos y verbos procesados. En los requerimientos iniciales de CoCoMe se encontraron 7 comunidades, mientras que en los requerimientos enriquecidos con ChatGPT se encontraron 9. El número de comunidades se puede observar que depende también del total de verbos y sustantivos que se han obtenido a partir de cada uno de los conjuntos de requerimientos. Estas comunidades representan grupos de palabras relacionadas a los requerimientos analizados.

	CoCoMe	ChatGPT
Sustantivos	94	215
Verbos	57	143
Sustantivos Procesados	53	99
Verbos Procesados	38	64
Nro. Comunidades	7	9

Tabla 4.1: Comparación Comunidades CoCoMe y ChatGPT.

En la tabla 4.2 se muestran los microservicios obtenidos a partir de los resultados obtenidos en la tabla 4.1. En la primera columna, se presentan los microservicios identificados utilizando el proceso presentado en el Capítulo 3 que involucra Procesamiento del Lenguaje Natural (PLN) sin utilizar el enriquecimiento del texto con ChatGPT. Mientras que en la segunda columna, se muestran los microservicios obtenidos al aplicar el proceso mencionado pero enriqueciendo el texto de los requerimientos con ChatGPT. Esto implica que el análisis de los requerimientos se realiza con la ayuda del modelo de lenguaje de ChatGPT para mejorar la comprensión y precisión en la identificación de los microservicios.

En la tercera columna, se presentan los microservicios identificados en el artículo científico "Identifying Microservices Using Functional Decomposition" por Tyszberowicz [4], estos microservicios se obtienen mediante el enfoque específico propuesto en el artículo mencionado.

CoCoMe	Enriquecidos con ChatGPT	Tyszberowicz [4]
1. CashDeskService	1. CashDeskService	1. ReportService
2. LightDisplayService	2. TradingSystemService	2. StockOrderService
3. TradingSystemService	3. InventoryService	3. SaleService
4. InventoryService	4. DeliveryTimeService	4. ProductListService
5. DeliveryService	5. PaymentService	
6. PriceChangeService	6. StoreStaffService	
7. QueryService	7. LightDisplayService	
	8. QueryService	
	9. UserInterfaceService	

Tabla 4.2: Comparación MicroServicios CoCoMe, ChatGPT y Tyszberowicz.

4.2. Evaluación

En esta sección, se presenta la evaluación del proceso propuesto de creación y despliegue de microservicios a partir de requerimientos funcionales. Se diseñó y aplicó una encuesta (***Ver Anexo B***) para obtener la opinión y los comentarios de expertos y profesionales con experiencia en el desarrollo de microservicios. La participación de estos profesionales es funda-

mental para evaluar la viabilidad y relevancia práctica del proceso propuesto. Los resultados obtenidos, junto con el análisis y las conclusiones, servirán para respaldar el proceso propuesto para este trabajo de titulación y ofrecer una visión práctica en el campo del desarrollo de microservicios. La información recopilada en esta evaluación enriquecerá el conocimiento y promoverá la adopción efectiva de microservicios en entornos empresariales y de desarrollo. En la figura 4.1 se muestran las respuestas obtenidas de las personas encuestadas en relación a sus profesiones o cargos actuales, esta figura permite visualizar de manera clara y concisa la distribución de los perfiles profesionales que participaron en la encuesta. Al analizar esta figura, se observa que la evaluación del proceso propuesto contó con la participación de un total de 18 profesionales provenientes de diferentes áreas y con diversos niveles de experiencia en el desarrollo de microservicios. La distribución de cargos o profesiones entre los encuestados fue la siguiente: 13 desarrolladores de software, 4 ingenieros DevOps y 3 tech leads, esta diversidad de perfiles profesionales enriquece los resultados y proporciona una visión más clara y sólida del impacto potencial del proceso propuesto en distintos contextos laborales. Al contar con la representación de estos distintos roles, se obtiene una perspectiva más completa de las opiniones y experiencias de profesionales en el ámbito del desarrollo de microservicios.

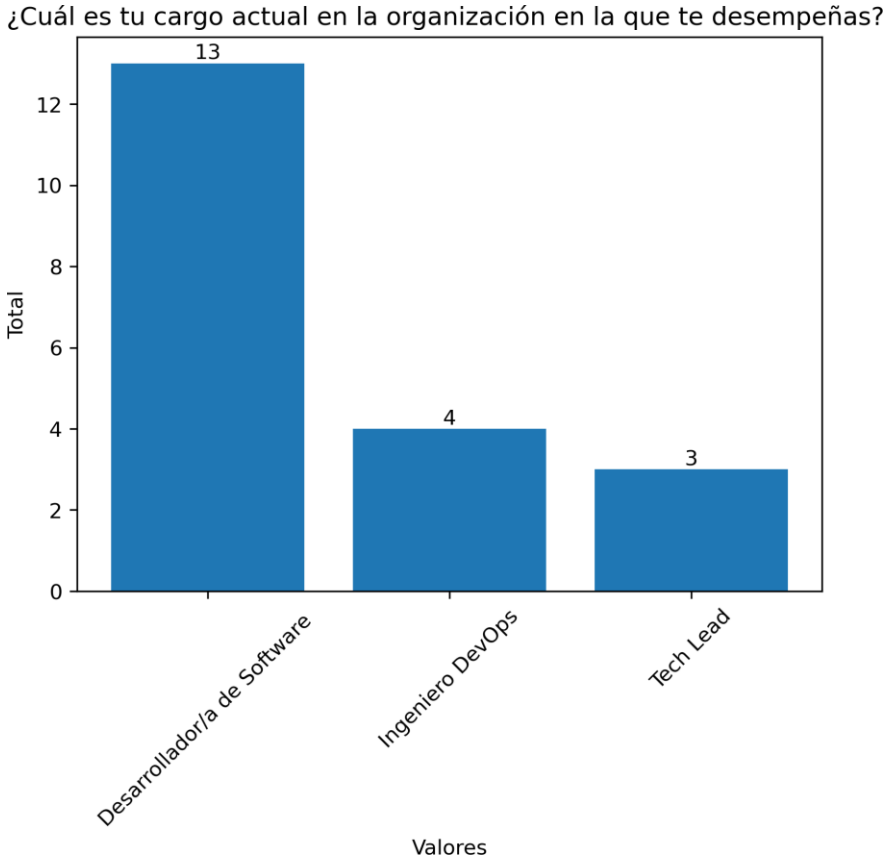


Fig 4.1: Puestos/Cargos Ocupados.

En la figura 4.2 se muestra el porcentaje de los grupos de microservicios elegidos por los encuestados. Se observa que el Grupo A representa el 45% de las respuestas, seguido por el Grupo B con un 40%, y el Grupo C con un 15%. Es importante destacar que los Grupos A y B son los obtenidos a través del proceso propuesto sumando así un 85% de preferencia por parte de los profesionales, mientras que el Grupo C es el enfoque por Tyszberowicz. [4].

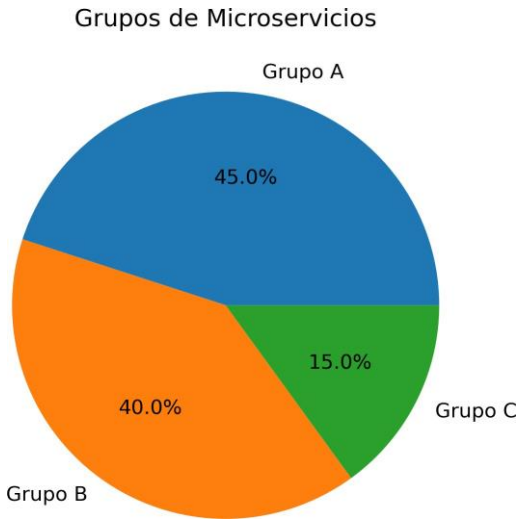


Fig 4.2: Grupo de Microservicios.

Es relevante mencionar que a los profesionales encuestados se les presentaron tres grupos de microservicios para que realizaran su elección, sin que se les revelara qué grupo fue generado mediante el proceso propuesto y cuál fue obtenido mediante el enfoque de Tyszberowicz [4]. Además, se les proporcionó la lista de requerimientos que se puede consultar en la tabla A.1 del Anexo A, con el objetivo de asegurar la objetividad en su elección del grupo de microservicios preferido.

Estos resultados reflejan la preferencia de los encuestados en relación a los grupos de microservicios presentados. El hecho de que el Grupo A obtenga el mayor porcentaje indica que el enfoque propuesto en este trabajo de tesis ha sido bien valorado y considerado como una opción viable para el desarrollo de microservicios. El Grupo B, enriquecido con ChatGPT, también ha recibido una proporción significativa de respuestas, lo que sugiere que la incorporación de técnicas de inteligencia artificial puede proporcionar mejoras y ventajas adicionales al proceso propuesto. Por otro lado, el Grupo C, basado en el paper base, ha obtenido un porcentaje más bajo, lo que indica que los encuestados consideran que el enfoque propuesto en

este trabajo de tesis y el enriquecido con ChatGPT son más adecuados para sus necesidades. Estos hallazgos respaldan la validez y relevancia del proceso propuesto, así como la importancia de la incorporación de tecnologías avanzadas como la inteligencia artificial para mejorar y optimizar el desarrollo de microservicios.

5. Conclusiones y Trabajos Futuros

En este capítulo se presenta una recopilación de los resultados más relevantes en torno a la investigación y proceso realizado, teniendo como objetivo presentar una visión general de lo que se ha logrado.

5.1. Conclusiones

- Los resultados muestran que se ha logrado desarrollar un proceso válido para la identificación de microservicios a partir de requerimientos funcionales, este proceso se caracteriza por ofrecer fases claras y estructuradas que guían a los desarrolladores en la identificación de los microservicios necesarios para cumplir con dichos requerimientos funcionales. Además, se destaca que este proceso proporciona una visión clara, detallada y práctica de cada fase, lo cual facilita el entendimiento de los resultados obtenidos y contribuye a una implementación exitosa.
- Uno de los objetivos fundamentales de este proyecto de titulación fue lograr el despliegue exitoso de los microservicios obtenidos a través del proceso propuesto. Este logro fue posible gracias a la meticulosa atención dedicada a cada una de las fases del proceso y a la valiosa contribución de las comunidades de grafos de palabras. A lo largo de estas fases, se obtuvo un entendimiento detallado de los requerimientos funcionales, lo que permitió desarrollar métodos específicos para cada uno de los microservicios finales propuestos. En conjunto, el proceso propuesto demostró ser una guía sólida y efectiva para el despliegue de microservicios, asegurando la coherencia en el resultado final.
- Con la finalidad de evaluar el proceso planteado, los resultados obtenidos tras la ejecución del proceso fueron evaluados por profesionales de manera que se pueda obtener un criterio objetivo y más claro con respecto a la eficacia y eficiencia del proceso, estos resultados fueron favorables ya que el 45% de los encuestados votaron por el resultado obtenido en la primera ejecución del proceso, en el cual únicamente se trabaja con los requerimientos de CoCoMe sin ningún tipo de enriquecimiento, mientras que en segundo lugar con el 40% de los votos se encuentra el segundo grupo de microservicios aplicando

el enriquecimiento de texto con ChatGPT, por lo tanto habiendo obtenido el 85% de la votación total a favor del proceso propuesto, se puede decir que, el proceso funciona y es una buena opción para detectar posibles microservicios partiendo de requerimientos funcionales.

5.2. Trabajos Futuros

A continuación, se presentan diversas líneas de trabajo futuro que se vislumbraron durante la ejecución de esta investigación:

- Se puede explorar la aplicación de técnicas de generación de lenguaje natural utilizando ChatGPT para generar prompts específicos que guíen el desarrollo de métodos o funciones asociadas a cada microservicio identificado.
- Es posible profundizar la utilización de ChatGPT para obtener métodos, entidades y nomenclatura potencial en las rutas de los microservicios detectados mediante el proceso propuesto. Esta exploración busca aprovechar el poder del procesamiento del lenguaje natural para mejorar la calidad y la comprensión de los microservicios generados.
- Evaluar la efectividad y adaptabilidad del proceso propuesto en proyectos de mayor escala y complejidad, abarcando distintos dominios de negocio.
- Investigar la integración de herramientas y tecnologías emergentes para mejorar aún más la eficiencia y automatización del proceso de creación y despliegue de microservicios.
- Realización de estudios comparativos con otros enfoques y metodologías similares en el campo, para analizar las ventajas y desventajas del proceso propuesto y su relevancia en el contexto actual del desarrollo de microservicios.

Referencias

[1] K. Chavez, P. Cedillo, M. Espinoza, and V. Saquicela, “A systematic literature review on composition of microservices through the use of semantic annotations: Solutions and techniques,” in *2019 International Conference on Information Systems and Computer Science (INCISCOS)*, Quito, Ecuador: IEEE, Nov. 2019, pp. 311–318, ISBN: 978-1-72815-581-4. DOI: 10.1109/INCISCOS49368.2019.00056. [Online]. Available: https://ieeexplore.ieee.org/document/9052275/ (visited on 05/26/2023).

[2] V. Saquicela, G. Campoverde, J. Avila, and M. E. Fajardo, “Building microservices for scalability and availability: Step by step, from beginning to end,” in *New Perspectives in Software Engineering*, J. Mejia, M. Muñoz, Á. Rocha, and Y. Quiñonez, Eds., vol. 1297, Series Title: Advances in Intelligent Systems and Computing, Cham: Springer International Publishing, 2021, pp. 169–184, ISBN: 978-3-030-63328-8 978-3-030-63329-5. DOI: 10.1007/978-3-030-63329-5_12. [Online]. Available: http://link.springer.com/10.1007/ 978-3-030-63329-5_12 (visited on 05/26/2023).

[3] Y. Wang and J. Zhang, “Experiment on automatic functional requirements analysis with the EFRF’s semantic cases,” en, in *2016 International Conference on Progress in Informatics and Computing (PIC)*, Shanghai, China: IEEE, Dec. 2016, pp. 636–642, ISBN: 978-1-5090-3484-0. DOI: 10.1109/PIC.2016.7949577. [Online]. Available: http://ieeexplore.ieee.org/document/7949577/ (visited on 05/22/2023).

[4] S. Tyszberowicz, R. Heinrich, B. Liu, and Z. Liu, “Identifying microservices using functional decomposition,” in *Dependable Software Engineering. Theories, Tools, and Applications*, X. Feng, M. Müller-Olm, and Z. Yang, Eds., vol. 10998, Series Title: Lecture Notes in Computer Science, Cham: Springer International Publishing, 2018, pp. 50–65, ISBN: 978-3-319-99932-6 978-3-319-99933-3. DOI: 10 .1007 /978 - 3 - 319 - 99933 - 3 _4. [Online]. Available: http://link.springer.com/10.1007/978-3-319-99933-3_4 (visited on 05/25/2023).

[5] C. Wohlin, *Experimentation in software engineering*, en, Nouv. ed. Heidelberg: Springer, 2012, ISBN: 978-3-642-29044-2.

[6] T. Gorschek, P. Garre, S. B. M. Larsson, and C. Wohlin, “Industry evaluation of the Requirements Abstraction Model,” en, *Requirements Engineering*, vol. 12, no. 3, pp. 163–190, Jul. 2007, ISSN: 0947-3602, 1432-010X. DOI: 10.1007/s00766-007-0047-z. [Online]. Available: http://link.springer.com/10.1007/s00766-007-0047-z (visited on 06/22/2023).

[7] P. A. Laplante and M. H. Kassab, *Requirements Engineering for Software and Systems*, en. CRC Press, Jun. 2022, ISBN: 978-1-00-059379-2.

[8] I. Sommerville, *Software engineering*, en, 9th ed. Boston: Pearson, 2011, OCLC: ocn462909026, ISBN: 978-0-13-703515-1 978-0-13-705346-9.

[9] K. Pohl and C. Rupp, *Requirements engineering fundamentals*, en, Second edition. Santa Barbara, CA: Rocky Nook, 2015, ISBN: 978-1-937538-77-4.

[10] "IEEE Recommended Practice for Software Requirements Specifications," IEEE, Tech. Rep., ISBN: 9780738104485. DOI: 10.1109/IEEESTD.1998.88286. [Online]. Available: http://ieeexplore.ieee.org/document/720574/ (visited on 05/09/2023).

[11] K. E. Wiegers and J. Beatty, *Software requirements*, en, Third edition. Redmond, Washington: Microsoft Press, s division of Microsoft Corporation, 2013, OCLC: ocn850176256, ISBN: 978-0-7356-7966-5.

[12] K. Gos and W. Zabierowski, "The comparison of microservice and monolithic architecture," in *2020 IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH)*, Lviv, Ukraine: IEEE, Apr. 2020, pp. 150–153, ISBN: 978-1-72817-179-1 978-1-72817-180-7. DOI: 10.1109/MEMSTECH49584.2020.9109514. [Online]. Available: https://ieeexplore.ieee.org/document/9109514/ (visited on 04/28/2023).

[13] G. Blinowski, A. Ojdowska, and A. Przybylek, "Monolithic vs. microservice architecture: A performance and scalability evaluation," *IEEE Access*, vol. 10, pp. 20 357–20 374, 2022, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2022.3152803. [Online]. Available: https://ieeexplore.ieee.org/document/9717259/ (visited on 04/28/2023).

[14] *Docker Docs: How to build, share, and run applications*, en, May 2023. [Online]. Available: https://docs.docker.com/ (visited on 05/29/2023).

[15] *Documentación de Kubernetes*, es. [Online]. Available: https://kubernetes.io/es/docs/home/ (visited on 05/29/2023).

[16] P. M. Nadkarni, L. Ohno-Machado, and W. W. Chapman, "Natural language processing: An introduction," en, *Journal of the American Medical Informatics Association*, vol. 18, no. 5, pp. 544–551, Sep. 2011, ISSN: 1067-5027, 1527-974X. DOI: 10.1136/amiajnl-2011-000464. [Online]. Available: https://academic.oup.com/jamia/article-lookup/doi/10.1136/amiajnl-2011-000464 (visited on 05/10/2023).

[17] D. Khurana, A. Koli, K. Khatter, and S. Singh, "Natural language processing: State of the art, current trends and challenges," en, *Multimedia Tools and Applications*, vol. 82, no. 3, pp. 3713–3744, Jan. 2023, ISSN: 1380-7501, 1573-7721. DOI: 10.1007/s11042-022-13428-4. [Online]. Available: https://link.springer.com/10.1007/s11042-022-13428-4 (visited on 05/09/2023).

[18] "Siri," Apple (España). (), [Online]. Available: https://www.apple.com/es/siri/ (visited on 06/16/2023).

[19] "Amazon alexa official site: What is alexa?" Amazon (Alexa). (), [Online]. Available: https://developer.amazon.com/es-ES/alexa.html (visited on 06/16/2023).

[20] E. D. Liddy, "Natural Language Processing," en, (visited on 05/09/2023).

[21] S. Choo and W. Kim, "A study on the evaluation of tokenizer performance in natural language processing," *Applied Artificial Intelligence*, vol. 37, no. 1, p. 2 175 112, Dec. 2023, Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/08839514.2023.2175112, ISSN: 0883-9514. DOI: 10.1080/08839514.2023.2175112. [Online]. Available: https://doi.org/10.1080/08839514.2023.2175112 (visited on 05/26/2023).

[22] C. University, *Tokenization*, Apr. 2009. [Online]. Available: https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html (visited on 05/26/2023).

[23] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press, 2008, ISBN: 978-0-521-86571-5. [Online]. Available: http://nlp.stanford.edu/IR-book/information-retrieval-book.html.

[24] S. Bag, S. K. Kumar, and M. K. Tiwari, "An efficient recommendation generation using relevant Jaccard similarity," en, *Information Sciences*, vol. 483, pp. 53–64, May 2019, ISSN: 00200255. DOI: 10.1016/j.ins.2019.01.023. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0020025519300325 (visited on 05/16/2023).

[25] N. Jaiswal, *SequenceMatcher in Python*, en, Jul. 2021. [Online]. Available: https://towardsdatascience.com/sequencematcher-in-python-6b1e6f3915fc (visited on 05/16/2023).

[26] M. T. Goodrich, R. Tamassia, and M. H. Goldwasser, *Data structures and algorithms in Java*, en, Sixth edition. Hoboken, NJ: Wiley, 2014, OCLC: ocn847126203, ISBN: 978-1-118-77133-4.

[27] F. Riaz and K. M. Ali, "Applications of graph theory in computer science," in *2011 Third International Conference on Computational Intelligence, Communication Systems and Networks*, Bali, Indonesia: IEEE, Jul. 2011, pp. 142–145, ISBN: 978-1-4577-0975-3. DOI:

10.1109/CICSyN.2011.40. [Online]. Available: http://ieeexplore.ieee.org/document/6005872/ (visited on 05/26/2023).

[28] Jiawei Han, Micheline Kamber, and Jian Pei, "Data mining: Concepts and techniques," in *Data Mining: Concepts and Techniques*, 3rd ed., Morgan Kaufmann Publishers.

[29] M. Ahmed, R. Seraj, and S. M. S. Islam, "The k-means Algorithm: A Comprehensive Survey and Performance Evaluation," en, *Electronics*, vol. 9, no. 8, p. 1295, Aug. 2020, ISSN: 2079-9292. DOI: 10.3390/electronics9081295. [Online]. Available: https://www.mdpi.com/2079-9292/9/8/1295 (visited on 05/29/2023).

[30] S. Fortunato and D. Hric, "Community detection in networks: A user guide," *Physics Reports*, vol. 659, pp. 1–44, Nov. 2016, ISSN: 03701573. DOI: 10.1016/j.physrep.2016.09.002. arXiv: 1608.00163[physics]. [Online]. Available: http://arxiv.org/abs/1608.00163 (visited on 04/30/2023).

[31] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3, pp. 75–174, 2010, ISSN: 0370-1573. DOI: https://doi.org/10.1016/j.physrep.2009.11.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0370157309002841.

[32] J. Zhang, J. Fei, X. Song, and J. Feng, "An improved louvain algorithm for community detection," *Mathematical Problems in Engineering*, vol. 2021, X. Tian, Ed., pp. 1–14, Nov. 23, 2021, ISSN: 1563-5147, 1024-123X. DOI: 10 . 1155 / 2021 / 1485592. [Online]. Available: https://www.hindawi.com/journals/mpe/2021/1485592/ (visited on 05/08/2023).

[33] S. Ghosh, M. Halappanavar, A. Tumeo, *et al.*, "Distributed louvain algorithm for graph community detection," in *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, Vancouver, BC: IEEE, May 2018, pp. 885–895, ISBN: 978-1-5386-4368-6. DOI: 10.1109/IPDPS.2018.00098. [Online]. Available: https://ieeexplore.ieee.org/document/8425242/ (visited on 05/20/2023).

[34] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, P10008, Oct. 9, 2008, ISSN: 1742-5468. DOI: 10.1088/1742- 5468/2008/10/P10008. arXiv: 0803.0476[cond- mat, physics:physics]. [Online]. Available: http://arxiv.org/abs/0803.0476 (visited on 06/08/2023).

[35] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Physical Review E*, vol. 70, no. 6, p. 066 111, Dec. 6, 2004, ISSN: 1539-3755,

1550-2376. DOI: 10.1103/PhysRevE.70.066111. arXiv: cond-mat/0408187. [Online]. Available: http://arxiv.org/abs/cond-mat/0408187 (visited on 06/16/2023).

[36] "Detección de comunidades en grafos y redes con python." (), [Online]. Available: https://www.cienciadedatos.net/documentos/pygml02-detecion-comunidades-grafos-redes-python.html (visited on 06/08/2023).

[37] D. Namiot and M. Sneps-Sneppe, "On micro-services architecture," vol. 2, no. 9, 2014.

[38] A. Rausch, R. Reussner, R. Mirandola, *et al.*, Eds., *The Common Component Modeling Example: Comparing Software Component Models* (Lecture Notes in Computer Science), en. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, vol. 5153, ISBN: 978-3-540-85288-9 978-3-540-85289-6. DOI: 10.1007/978-3-540-85289-6. [Online]. Available: http://link.springer.com/10.1007/978-3-540-85289-6 (visited on 04/30/2023).

[39] V. B. R. Vidya Sagar and S. Abirami, "Conceptual modeling of natural language functional requirements," *Journal of Systems and Software*, vol. 88, pp. 25–41, Feb. 2014, ISSN: 01641212. DOI: 10.1016/j.jss.2013.08.036. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0164121213002379 (visited on 05/23/2023).

[40] M. Fraiwan and N. Khasawneh, "A Review of ChatGPT Applications in Education, Marketing, Software Engineering, and Healthcare: Benefits, Drawbacks, and Research Directions," en,

[41] Y. Bang, S. Cahyawijaya, N. Lee, *et al.*, *A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity*, en, arXiv:2302.04023 [cs], Feb. 2023. [Online]. Available: http://arxiv.org/abs/2302.04023 (visited on 05/21/2023).

[42] J. White, S. Hays, Q. Fu, J. Spencer-Smith, and D. C. Schmidt, *ChatGPT Prompt Patterns for Improving Code Quality, Refactoring, Requirements Elicitation, and Software Design*, en, arXiv:2303.07839 [cs], Mar. 2023. [Online]. Available: http://arxiv.org/abs/2303.07839 (visited on 05/21/2023).

[43] *NVIDIA Large language models (LLMs)*, es-la. [Online]. Available: https://www.nvidia.com/es-la/deep-learning-ai/solutions/large-language-models/ (visited on 06/22/2023).

[44] *English · spaCy Models Documentation*, en. [Online]. Available: https://spacy.io/models/en (visited on 05/30/2023).

[45] *spaCy · Industrial-strength Natural Language Processing in Python*, en. [Online]. Available: https://spacy.io/ (visited on 04/30/2023).

[46] N. Hardeniya, J. Perkins, D. Chopra, N. Joshi, and I. Mathur, *Natural Language Processing: Python and NLTK*, en. Packt Publishing Ltd, Nov. 2016, Google-Books-ID: 0J_cDgAAQBAJ, ISBN: 978-1-78728-784-6.

[47] *String — Common string operations*. [Online]. Available: https://docs.python.org/3/library/string.html (visited on 04/30/2023).

[48] *NLTK :: Natural Language Toolkit*. [Online]. Available: https://www.nltk.org/ (visited on 04/30/2023).

[49] *Built-in Types — Python 3.8.16 documentation*. [Online]. Available: https://docs.python.org/3.8/library/stdtypes.html#set-types-set-frozenset (visited on 04/30/2023).

[50] G. Rossetti, L. Milli, and R. Cazabet, "CDLIB: A python library to extract, compare and evaluate communities from complex networks," *Applied Network Science*, vol. 4, no. 1, p. 52, Dec. 2019, ISSN: 2364-8228. DOI: 10.1007/s41109-019-0165-9. [Online]. Available: https://appliednetsci.springeropen.com/articles/10.1007/s41109-019-0165-9 (visited on 06/16/2023).

[51] y. company the diagramming the diagramming. "yFiles product details," yWorks, the diagramming experts. (), [Online]. Available: https://www.yworks.com/yfiles (visited on 06/16/2023).

[52] "NetworkX — NetworkX documentation." (), [Online]. Available: https://networkx.org/ (visited on 06/16/2023).

[53] *OpenAI API*, en. [Online]. Available: https://platform.openai.com (visited on 05/21/2023).

[54] "Spring boot," Spring Boot. (), [Online]. Available: https://spring.io/projects/spring-boot (visited on 06/23/2023).

[55] A. Penas and E. Hovy, "Semantic enrichment of text with background knowledge,"

Anexos

A. Tablas de Requerimientos

A.1. CoCoMe

Tabla A.1: Descripción de Requerimientos propuestos por CoCoMe.

Nro.	Name	Description
UC 1	Process Sale	At the Cash Desk the products a Customer wants to buy are detected and the payment, either by credit card or cash, is performed.
UC 2	Manage Express Checkout	If some conditions are fulfilled a Cash Desk automatically switches into an express mode. The Cashier is able to switch back into normal mode by pressing a button at his Cash Desk. To indicate the mode the Light Display shows different colors.
UC 3	Order Products	The Trading System provide the opportunity to order product items.
UC 4	Receive Ordered Products	Ordered products which arrive at the Store have to be checked for correctness and inventoried.
UC 5	Show Stock Reports	The opportunity to generate stock-related reports is provided by the Trading System.
UC 6	Show Delivery Reports	The Trading System provides the opportunity to calculate the mean times a delivery from each supplier to a considered enterprise takes.
UC 7	Change Price	The System provides the opportunity to change the sales price for a product.

Nro.	Name	Description
UC 8	Product Exchange (on Low Stock) among Stores	If a store runs out of a certain product (or a set of products; "required good"), it is possible to start a query to check whether those products are available at other Stores of the Enterprise ("providing Stores"). Therefore the Enterprise Server and the Store Servers need to synchronize their data on demand (one scheduled update per day or per hour is not sufficient). After a successful query the critical product can be shipped from one to other Stores. But it has to be decided (using heuristics to compute the future selling frequency), whether the transportation is meaningful. For example, if the product is propably sold out at all Stores within the same day, a transportation does not make sense. Expressed in a more technical way one Store Server is able to start a query at the Enterprise Server. The Enterprise Server in turn starts a query for products available at other Stores. As the Enterprise Server does not have the current global data for Stores at any time (due to a write caching latency at the Store Servers) the Enterprise Server has to trigger all Store Servers to push their local data to the Enterprise Server.
UC 8 ext.	Remove Incoming Status	For moved products an amount marked as incoming remains at the Inventory of the Store receiving the products. An extension allows to change that incoming mark via a user interface at the Store Client if the moved products arrive at a Store.

A.2. Requerimientos enriquecidos con ChatGPT

Tabla A.2: Descripción de Requerimientos enriquecidos con ChatGPT.

Name	Description
Process Sale	”At the Cash Desk, customers can purchase products using either a credit card or cash. The products a customer wants to buy are detected using a barcode scanner or similar device. The customer then pays for their items using the credit card or cash. The payment is processed and the customer is given a receipt for their purchase. The cashier can also provide any additional information or assistance the customer may need.”
Manage Express Checkout	”The cash desk automatically switches into express mode when certain conditions are met. This mode allows the cashier to process payments quickly, as the cash desk is optimized for speed. The cashier can switch back to normal mode by pressing a button at the cash desk. The light display on the cash desk will indicate the current mode, usually with different colors for express and normal modes.”
Order Products	”The Trading System provides an opportunity to order product items from a variety of different suppliers. This system allows businesses to compare prices from different suppliers and select the best deal for their needs. It also allows businesses to track their orders and receive notifications when new products are available. The system also provides tools for tracking inventory and managing customer relationships. Additionally, the system allows businesses to search for products and compare prices from different suppliers. This helps businesses save time and money by finding the best deals for their needs.”
Receive Ordered Products	”When ordered products arrive at the store, they must be inspected to ensure that they are correct and in good condition. The store staff must then take an inventory of the products to ensure that all items have been received and to track the quantity of each item. This inventory should be updated regularly to reflect any changes in product quantity. The store staff should also take the time to check the expiration dates of any perishable items, as these items must be discarded if they are past their expiration date.”

Name	Description
Show Stock Reports	”The Trading System provides users with the opportunity to generate stock-related reports. These reports can provide valuable insights into the performance of stocks and the overall market. They can also be used to identify trends and make informed decisions about investments. The reports are generated using data from the stock exchange, and can include information such as the daily price and volume of stocks, the number of shares traded, the current market capitalization, and more. Additionally, the reports can be customized to focus on specific stocks or sectors, allowing users to gain a better understanding of the market.”
Show Delivery Reports	The Trading System provides the opportunity to calculate the mean times a delivery from each supplier to a considered enterprise takes. This system allows the enterprise to compare the delivery times of different suppliers and make informed decisions on which suppliers to use in the future. This system also allows the enterprise to track the delivery times of each supplier and make changes as needed to ensure the best delivery times possible. The system also allows the enterprise to identify trends in the delivery times of each supplier and make changes in the supply chain accordingly.
Change Price	”The System provides the opportunity to change the sales price for a product. This is a useful tool for businesses that need to adjust prices according to market conditions or customer demand. With the System, businesses can quickly and easily adjust prices without having to manually enter new prices into their system. This feature allows businesses to remain competitive and maximize their profits. Additionally, the System can be used to track sales trends and make informed decisions about pricing.”

Name	Description
Product Exchange (on Low Stock) among Stores	”If a store runs out of a certain product, it is possible to start a query to check whether those products are available at other stores of the enterprise. The enterprise server and the store servers need to synchronize their data on demand, as one scheduled update per day or per hour is not sufficient. After a successful query, the critical product can be shipped from one to other stores, but it has to be decided using heuristics to compute the future selling frequency, whether the transportation is meaningful. The store server is able to start a query at the enterprise server, which in turn starts a query for products available at other stores. The enterprise server does not have the current global data for stores at any time due to a write caching latency at the store servers, so it has to trigger all store servers to push their local data to the enterprise server.”
Remove Incoming Status	”This extension allows the store client to change the incoming mark of the moved products that have arrived at the store. This means that the store client can adjust the amount of incoming products in the inventory, which is helpful for keeping track of stock levels and ensuring that the store has the correct amount of products. The user interface makes this process quick and easy, allowing the store to quickly adjust the incoming mark and keep their inventory up to date.”

B. Encuesta Microservicios

Esta encuesta tiene como objetivo validar y evaluar los resultados obtenidos en nuestro estudio. Trata sobre el desarrollo de un **proceso de creación y despliegue de microservicios a partir de requerimientos funcionales**, y forma parte de nuestra tesis de grado. Tu participación es crucial para obtener información valiosa que nos permitirá comprender mejor la eficacia y viabilidad de nuestro proceso de creación y despliegue de microservicios. A través de esta encuesta, queremos conocer tu perspectiva y opiniones como profesional. Agradecemos de antemano tu tiempo y colaboración al responder a esta encuesta. Tus aportes serán de gran ayuda para mejorar y validar nuestro proceso, así como contribuir al conocimiento en el campo de los microservicios.
Si tienes alguna pregunta o inquietud, no dudes en ponerte en contacto con nosotros.
¡Muchas gracias por tu participación!

1. **Por favor, indícanos tu nombre completo (nombres y apellidos):**
2. **¿Has trabajado o estás trabajando actualmente con microservicios en tu organización?**
   - Sí
   - No
3. En el contexto de tu organización, actividades actuales y conocimiento previo, nos gustaría conocer tus preferencias en cuanto a microservicios. Los microservicios son componentes de software independientes y autónomos que se pueden utilizar para construir y desplegar aplicaciones de manera modular.

   A continuación, te presentamos una serie de requerimientos funcionales que ejemplifican el comportamiento de un sistema comercial. Te solicitamos que analices los siguientes requerimientos:

   (a) At the Cash Desk the products a Customer wants to buy are detected and the payment, either by credit card or cash, is performed.

(b) If some conditions are fulfilled a Cash Desk automatically switches into an express mode. The Cashier is able to switch back into normal mode by pressing a button at his Cash Desk. To indicate the mode the Light Display shows different colors.

(c) The Trading System provide the opportunity to order product items.

(d) Ordered products which arrive at the Store have to be checked for correctness and inventoried.

(e) The opportunity to generate stock-related reports is provided by the Trading System.

(f) The Trading System provides the opportunity to calculate the mean times a delivery from each supplier to a considered enterprise takes.

(g) The System provides the opportunity to change the sales price for a product.

(h) If a store runs out of a certain product (or a set of products; "required good"), it is possible to start a query to check whether those products are available at other Stores of the Enterprise ("providing Stores"). Therefore the Enterprise Server and the Store Servers need to synchronize their data on demand (one scheduled update per day or per hour is not sufficient). After a successful query the critical product can be shipped from one to other Stores. But it has to be decided (using heuristics to compute the future selling frequency), whether the transportation is meaningful. For example, if the product is propably sold out at all Stores within the same day, a transportation does not make sense. Expressed in a more technical way one Store Server is able to start a query at the Enterprise Server. The Enterprise Server in turn starts a query for products available at other Stores. As the Enterprise Server does not have the current global data for Stores at any time (due to a write caching latency at the Store Servers) the Enterprise Server has to trigger all Store Servers to push their local data to the Enterprise Server.

(i) For moved products an amount marked as incoming remains at the Inventory of the Store receiving the products. An extension allows to change that incoming mark via a user interface at the Store Client if the moved products arrive at a Store.

En función de tus preferencias y criterio, elige uno de los siguientes grupos de microservicios que consideres más adecuado para los requerimientos presentados:

- Grupo A:

1. CashDeskService
2. LightDisplayService
3. TradingSystemService
4. InventoryService
5. DeliveryService
6. PriceChangeService
7. QueryService

- Grupo B:

1. CashDeskService
2. TradingSystemService
3. InventoryService
4. DeliveryTimeService
5. PaymentService
6. StoreStaffService
7. LightDisplayService
8. QueryService
9. UserInterfaceService

- Grupo C:

1. ReportService
2. StockOrderService
3. SaleService
4. ProductListService

4. **Si deseas agregar algún comentario adicional o tienes alguna observación que consideres relevante para nuestro estudio, por favor, indícalo a continuación.**

C. Prompt utilizado en ChatGPT

I need only one possible name for the microservices, I have these communities of words %s give me only one name for each community, a community represents a microservice, so I need %s names in total, also for context, these are the requirements %s with which I get the communities for each community?”

D. Caso Práctico del Proceso de Identificación de Microservicios: Requerimientos de CoCoMe

D.1. Recolección de Requerimientos

Conjunto $R = \{r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8, r_9\}$:

Tabla D.1: Descripción del conjunto *R*

r_i	**Descripción**
r_1	At the Cash Desk the products a Customer wants to buy are detected and the payment, either by credit card or cash, is performed.
r_2	If some conditions are fulfilled a Cash Desk automatically switches into an express mode. The Cashier is able to switch back into normal mode by pressing a button at his Cash Desk. To indicate the mode the Light Display shows different colors.
r_3	The Trading System provide the opportunity to order product items.
r_4	Ordered products which arrive at the Store have to be checked for correctness and inventoried.
r_5	The opportunity to generate stock-related reports is provided by the Trading System.
r_6	The Trading System provides the opportunity to calculate the mean times a delivery from each supplier to a considered enterprise takes.
r_7	The System provides the opportunity to change the sales price for a product.

r_i	Descripción
r_8	If a store runs out of a certain product (or a set of products; "required good"), it is possible to start a query to check whether those products are available at other Stores of the Enterprise ("providing Stores"). Therefore the Enterprise Server and the Store Servers need to synchronize their data on demand (one scheduled update per day or per hour is not sufficient). After a successful query the critical product can be shipped from one to other Stores. But it has to be decided (using heuristics to compute the future selling frequency), whether the transportation is meaningful. For example, if the product is propably sold out at all Stores within the same day, a transportation does not make sense. Expressed in a more technical way one Store Server is able to start a query at the Enterprise Server. The Enterprise Server in turn starts a query for products available at other Stores. As the Enterprise Server does not have the current global data for Stores at any time (due to a write caching latency at the Store Servers) the Enterprise Server has to trigger all Store Servers to push their local data to the Enterprise Server.
r_9	For moved products an amount marked as incoming remains at the Inventory of the Store receiving the products. An extension allows to change that incoming mark via a user interface at the Store Client if the moved products arrive at a Store.

D.2. PLN

D.2.1. Extracción de sustantivos y verbos

Sustantivos S = { 'CashDesk', 'Product', 'Customer', 'Payment', 'CreditCard', 'Cash', 'Condition', 'CashDesk', 'ExpressMode', 'Cashier', 'NormalMode', 'Button', 'CashDesk', 'Mode', 'Light-Display', 'DifferentColor', 'TradingSystem', 'Opportunity', 'ProductItem', 'OrderedProduct', 'Store', 'Correctness', 'Opportunity', 'Stock-RelateReport', 'TradingSystem', 'TradingSystem', 'Opportunity', 'MeanTime', 'Delivery', 'Supplier', 'ConsiderEnterpriseTake', 'System', 'Opportunity', 'SalePrice', 'Product', 'Store', 'CertainProduct', 'Set', 'Product', 'Good', 'Query', 'Product', 'Stores', 'Enterprise', 'Store', 'EnterpriseServer', 'StoreServers', 'Datum', 'Demand', 'OneScheduleUpdate', 'Day', 'Hour', 'SuccessfulQuery', 'CriticalProduct', 'Stores', 'Heuristic', 'FutureSellingFrequency', 'Transportation', 'Example', 'Product', 'Stores', 'Day', 'Transportation', 'Sense', 'TechnicalWay',

'OneStoreServer', 'Query', 'EnterpriseServer', 'EnterpriseServer', 'Turn', 'Query', 'Product', 'Stores', 'EnterpriseServer', 'CurrentGlobalDatum', 'Stores', 'Time', 'WriteCacheLatency', 'StoreServers', 'EnterpriseServer', 'StoreServers', 'LocalDatum', 'EnterpriseServer', 'MoveProduct', 'Amount', 'Inventory', 'Store', 'Product', 'Extension', 'IncomingMark', 'UserInterface', 'StoreClient', 'Move-Product', 'Store' }

S_O = { (nombre='the cash desk', lema='cashdesk', raiz='at'), (nombre='the products', lema='product', raiz='detect'), (nombre='a customer', lema='customer', raiz='want'), (nombre='the payment', lema='payment', raiz='detect'), (nombre='credit card', lema='creditcard', raiz='by'), (nombre='cash', lema='cash', raiz='card'), (nombre='some conditions', lema='condition', raiz='fulfil'), (nombre='a cash desk', lema='cashdesk', raiz='switch'), (nombre='an express mode', lema='expressmode', raiz='into'), (nombre='the cashier', lema='cashier', raiz='be'), (nombre='normal mode', lema='normalmode', raiz='into'), (nombre='a button', lema='button', raiz='press'), (nombre='his cash desk', lema='cashdesk', raiz='at'), (nombre='the mode', lema='mode', raiz='show'), (nombre='the light display', lema='lightdisplay', raiz='show'), (nombre='different colors', lema='differentcolor', raiz='show'), (nombre='the trading system', lema='tradingsystem', raiz='provide'), (nombre='the opportunity', lema='opportunity', raiz='provide'), (nombre='product items', lema='productitem', raiz='order'), (nombre='ordered products', lema='orderedproduct', raiz='have'), (nombre='the store', lema='store', raiz='at'), (nombre='correctness', lema='correctness', raiz='for'), (nombre='the opportunity', lema='opportunity', raiz='provide'), (nombre='stock-related reports', lema='stock-relatereport', raiz='generate'), (nombre='the trading system', lema='tradingsystem', raiz='by'), (nombre='the trading system', lema='tradingsystem' raiz='provide'), (nombre='the opportunity', lema='opportunity', raiz='provide'), (nombre='the mean times', lema='meantime', raiz='calculate'), (nombre='a delivery', lema='delivery', raiz='calculate'), (nombre='each supplier', lema='supplier', raiz='from'), (nombre='a considered enterprise takes', lema='considerenterprisetake', raiz='to'), (nombre='the system', lema='system', raiz='provide'), (nombre='the opportunity', lema='opportunity', raiz='provide'), (nombre='the sales price', lema='saleprice', raiz='change'), (nombre='a product', lema='product', raiz='for'), (nombre='a store', lema='store', raiz='run'), (nombre='a certain product', lema='certainproduct', raiz='of'), (nombre='a set', lema='set', raiz='product'), (nombre='products', lema='product', raiz='of'), (nombre='good', lema='good', raiz='require'), (nombre='a query', lema='query', raiz='start'), (nombre='those products', lema='product', raiz='be'), (nombre='other stores', lema='stores', raiz='at'), (nombre='the enterprise', lema='enterprise', raiz='of'), (nombre='stores', lema='store', raiz='provide'), (nombre='the enterprise server', lema='enterpriseser raiz='need'), (nombre='the store servers', lema='storeservers', raiz='Server'), (nombre='their data', lema='datum', raiz='synchronize'), (nombre='demand', lema='demand', raiz='on'), (nombre='one scheduled update', lema='onescheduleupdate', raiz='be'), (nombre='day', lema='day',

raiz='per'), (nombre='hour', lema='hour', raiz='per'), (nombre='a successful query', lema='successfulquery', raiz='after'), (nombre='the critical product', lema='criticalproduct', raiz='ship'), (nombre='other stores', lema='stores', raiz='to'), (nombre='heuristics', lema='heuristic', raiz='use'), (nombre='the future selling frequency', lema='futuresellingfrequency', raiz='compute'), (nombre='the transportation', lema='transportation', raiz='be'), (nombre='example', lema='example', raiz='for'), (nombre='the product', lema='product', raiz='sell'), (nombre='all stores', lema='stores', raiz='at'), (nombre='the same day', lema='day', raiz='within'), (nombre='a transportation', lema='transportation', raiz='make'), (nombre='sense', lema='sense', raiz='make'), (nombre='a more technical way', lema='technicalway', raiz='in'), (nombre='one store server', lema='onestoreserver', raiz='be'), (nombre='a query', lema='query', raiz='start'), (nombre='the enterprise server', lema='enterpriseserver', raiz='at'), (nombre='the enterprise server', lema='enterpriseserver', raiz='start'), (nombre='turn', lema='turn', raiz='in'), (nombre='a query', lema='query', raiz='start'), (nombre='products', lema='product', raiz='for'), (nombre='other stores', lema='stores', raiz='at'), (nombre='the enterprise server', lema='enterpriseserver', raiz='have'), (nombre='the current global data', lema='currentglobaldatum', raiz='have'), (nombre='stores', lema='stores', raiz='for'), (nombre='any time', lema='time', raiz='at'), (nombre='a write caching latency', lema='writecachelatency', raiz='due'), (nombre='the store servers', lema='storeservers', raiz='at'), (nombre='the enterprise server', lema='enterpriseserver', raiz='have'), (nombre='all store servers', lema='storeservers', raiz='trigger'), (nombre='their local data', lema='localdatum', raiz='push'), (nombre='the enterprise server', lema='enterpriseserver', raiz='to'), (nombre='moved products', lema='moveproduct', raiz='for'), (nombre='an amount', lema='amount', raiz='remain'), (nombre='the inventory', lema='inventory', raiz='at'), (nombre='the store', lema='store', raiz='of'), (nombre='the products', lema='product', raiz='receive'), (nombre='an extension', lema='extension', raiz='allow'), (nombre='that incoming mark', lema='incomingmark', raiz='change'), (nombre='a user interface', lema='userinterface', raiz='via'), (nombre='the store client', lema='storeclient', raiz='at'), (nombre='the moved products', lema='moveproduct', raiz='arrive'), (nombre='a store', lema='store', raiz='at') }

Verbos

V = { 'want', 'buy', 'detect', 'detect', 'perform', 'perform', 'fulfil', 'fulfil', 'switch', 'switch', 'press', 'indicate', 'show', 'provide', 'order', 'arrive', 'check', 'check', 'generate', 'relate', 'provide', 'provide', 'provide', 'calculate', 'consider', 'provide', 'change', 'run', 'require', 'start', 'check', 'provide', 'need', 'synchronize', 'schedule', 'ship', 'ship', 'decide', 'decide', 'use', 'compute', 'sell', 'make', 'express', 'start', 'start', 'cache', 'trigger', 'push', 'move', 'mark', 'remain', 'receive', 'allow', 'change', 'move', 'arrive' }

V_O = { (nombre='wants', lema='want'), (nombre='buy', lema='buy'), (nombre='are detected', lema='detect'), (nombre='detected', lema='detect'), (nombre='is performed', lema='perform'), (nombre='performed', lema='perform'), (nombre='are fulfilled', lema='fulfil'), (nombre='fulfilled', lema='fulfil'), (nombre='switches', lema='switch'), (nombre='switch', lema='switch'), (nombre='pressing', lema='press'), (nombre='indicate', lema='indicate'), (nombre='shows', lema='show'), (nombre='provide', lema='provide'), (nombre='order', lema='order'), (nombre='arrive', lema='arrive'), (nombre='be checked', lema='check'), (nombre='checked', lema='check'), (nombre='generate', lema='generate'), (nombre='related', lema='relate'), (nombre='is provided', lema='provide'), (nombre='provided', lema='provide'), (nombre='provides', lema='provide'), (nombre='calculate', lema='calculate'), (nombre='considered', lema='consider'), (nombre='provides', lema='provide'), (nombre='change', lema='change'), (nombre='runs', lema='run'), (nombre='required', lema='require'), (nombre='start', lema='start'), (nombre='check', lema='check'), (nombre='providing', lema='provide'), (nombre='need', lema='need'), (nombre='synchronize', lema='synchronize'), (nombre='scheduled', lema='schedule'), (nombre='be shipped', lema='ship'), (nombre='shipped', lema='ship'), (nombre='be decided', lema='decide'), (nombre='decided', lema='decide'), (nombre='using', lema='use'), (nombre='compute', lema='compute'), (nombre='sold', lema='sell'), (nombre='make', lema='make'), (nombre='expressed', lema='express'), (nombre='start', lema='start'), (nombre='starts', lema='start'), (nombre='caching', lema='cache'), (nombre='trigger', lema='trigger'), (nombre='push', lema='push'), (nombre='moved', lema='move'), (nombre='marked', lema='mark'), (nombre='remains', lema='remain'), (nombre='receiving', lema='receive'), (nombre='allows', lema='allow'), (nombre='change', lema='change'), (nombre='moved', lema='move'), (nombre='arrive', lema='arrive') }

D.2.2. Pre-procesamiento de Sustantivos y Verbos

Sustantivos

S_2 = { 'supplier', 'userinterface', 'correctness', 'payment', 'set', 'datum', 'sense', 'lightdisplay', 'successfulquery', 'store', 'day', 'button', 'productitem', 'oneschedule update', 'cashdesk', 'differentcolor', 'writecachelatency', 'product', 'creditcard', 'hour', 'transportation', 'mode', 'consider-enterprisetake', 'normalmode', 'expressmode', 'enterprise', 'cashier', 'cash', 'currentglobaldatum', 'onestoreserver', 'moveproduct', 'certainproduct', 'opportunity', 'storeservers', 'amount', 'criticalproduct', 'extension', 'storeclient', 'query', 'saleprice', 'turn', 'condition', 'stock-relatereport', 'time', 'customer', 'localdatum', 'example', 'futuresellingfrequency', 'enterpriseserver', 'heuristic', 'tradingsystem', 'demand', 'system', 'orderedproduct', 'good', 'technicalway', 'incoming-mark', 'delivery', 'stores', 'inventory', 'meantime' }

Verbos

V_2 = { 'sell', 'receive', 'order', 'trigger', 'decide', 'use', 'start', 'mark', 'generate', 'press', 'show', 'consider', 'synchronize', 'switch', 'change', 'ship', 'compute', 'express', 'perform', 'schedule', 'run', 'make', 'push', 'detect', 'move', 'check', 'require', 'cache', 'allow', 'remain', 'relate', 'need', 'indicate', 'fulfil', 'provide', 'arrive', 'calculate', 'buy', 'want' }

D.2.3. Métricas de Similaridad Sintáctica

Sustantivos

S_3 = { 'supplier', 'userinterface', 'correctness', 'payment', 'datum', 'sense', 'set', 'lightdisplay', 'successfulquery', 'store', 'day', 'button', 'onescheduleupdate', 'cashdesk', 'differentcolor', 'write-cachelatency', 'product', 'creditcard', 'hour', 'transportation', 'mode', 'considerenterprisetake', 'normalmode', 'enterprise', 'expressmode', 'cashier', 'cash', 'currentglobaldatum', 'certainprod-uct', 'opportunity', 'amount', 'extension', 'storeclient', 'query', 'saleprice', 'turn', 'condition', 'stock-relatereport', 'time', 'customer', 'localdatum', 'example', 'futuresellingfrequency', 'heuristic', 'de-mand', 'tradingsystem', 'good', 'system', 'incomingmark', 'technicalway', 'delivery', 'inventory', 'meantime' }

Verbos

V_3 = { 'sell', 'receive', 'order', 'trigger', 'decide', 'use', 'start', 'mark', 'generate', 'press', 'show', 'consider', 'synchronize', 'switch', 'change', 'compute', 'ship', 'perform', 'schedule', 'run', 'make', 'push', 'detect', 'move', 'cache', 'check', 'require', 'allow', 'remain', 'relate', 'need', 'indicate', 'fulfil', 'arrive', 'calculate', 'provide', 'buy', 'want' }

D.3. Clusterización

D.3.1. Generación del Grafo

Nodos

N = { 'amount', 'button', 'cash', 'cashdesk', 'cashier', 'certainproduct', 'condition', 'consid-erenterprisetake', 'correctness', 'creditcard', 'currentglobaldatum', 'customer', 'datum', 'day', 'delivery', 'demand', 'differentcolor', 'enterprise', 'example', 'expressmode', 'extension', 'future-sellingfrequency', 'good', 'heuristic', 'hour', 'incomingmark', 'inventory', 'lightdisplay', 'localda-tum', 'meantime', 'mode', 'normalmode', 'onescheduleupdate', 'opportunity', 'payment', 'prod-uct', 'query', 'saleprice', 'sense', 'set', 'stock-relatereport', 'store', 'storeclient', 'successfulquery',

'supplier', 'system', 'technicalway', 'time', 'tradingsystem', 'transportation', 'turn', 'userinterface', 'writecachelatency', 'calculate', 'change', 'detect', 'make', 'provide', 'show', 'start' }

Aristas

E = { ('delivery', 'calculate'), ('differentcolor', 'show'), ('enterprise', 'start'), ('incomingmark', 'change'), ('lightdisplay', 'show'), ('meantime', 'calculate'), ('mode', 'show'), ('opportunity', 'provide'), ('payment', 'detect'), ('product', 'detect'), ('query', 'start'), ('saleprice', 'change'), ('sense', 'make'), ('store', 'provide'), ('system', 'provide'), ('tradingsystem', 'provide'), ('transportation', 'make') }

Eliminar Nodos Aislados

Nodos

N' = { 'delivery', 'differentcolor', 'enterprise', 'incomingmark', 'lightdisplay', 'meantime', 'mode', 'opportunity', 'payment', 'product', 'query', 'saleprice', 'sense', 'store', 'system', 'tradingsystem', 'transportation', 'calculate', 'change', 'detect', 'make', 'provide', 'show', 'start' }

Aristas

E' = { ('delivery', 'calculate'), ('differentcolor', 'show'), ('enterprise', 'start'), ('incomingmark', 'change'), ('lightdisplay', 'show'), ('meantime', 'calculate'), ('mode', 'show'), ('opportunity', 'provide'), ('payment', 'detect'), ('product', 'detect'), ('query', 'start'), ('saleprice', 'change'), ('sense', 'make'), ('store', 'provide'), ('system', 'provide'), ('tradingsystem', 'provide'), ('transportation', 'make') }

D.4. Detección de Comunidades

C = { 'provide': 0, 'opportunity': 0, 'tradingsystem': 0, 'system': 0, 'store': 0, 'show': 1, 'mode': 1, 'lightdisplay': 1, 'differentcolor': 1, 'delivery': 2, 'calculate': 2, 'meantime': 2, 'detect': 3, 'product': 3, 'payment': 3, 'incomingmark': 4, 'saleprice': 4, 'change': 4, 'sense': 5, 'transportation': 5, 'make': 5, 'enterprise': 6, 'query': 6, 'start': 6 }

D.5. Microservicios

D.5.1. Nombramiento de Microservicios

Micros = { 'CashDeskService', 'LightDisplayService', 'TradingSystemService', 'InventoryService', 'DeliveryService', 'PriceChangeService', 'QueryService' }

E. Caso Práctico del Proceso de Identificación de Microservicios: Requerimientos enriquecidos con ChatGpt

E.1. Recolección de Requerimientos

Conjunto $R = \{r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8, r_9\}$:

Tabla E.1: Descripción del conjunto *R*

r_i	**Descripción**
r_1	At the Cash Desk, customers can purchase products using either a credit card or cash. The products a customer wants to buy are detected using a barcode scanner or similar device. The customer then pays for their items using the credit card or cash. The payment is processed and the customer is given a receipt for their purchase. The cashier can also provide any additional information or assistance the customer may need.
r_2	The cash desk automatically switches into express mode when certain conditions are met. This mode allows the cashier to process payments quickly, as the cash desk is optimized for speed. The cashier can switch back to normal mode by pressing a button at the cash desk. The light display on the cash desk will indicate the current mode, usually with different colors for express and normal modes.
r_3	The Trading System provides an opportunity to order product items from a variety of different suppliers. This system allows businesses to compare prices from different suppliers and select the best deal for their needs. It also allows businesses to track their orders and receive notifications when new products are available. The system also provides tools for tracking inventory and managing customer relationships. Additionally, the system allows businesses to search for products and compare prices from different suppliers. This helps businesses save time and money by finding the best deals for their needs.

r_i	**Descripción**
r_4	When ordered products arrive at the store, they must be inspected to ensure that they are correct and in good condition. The store staff must then take an inventory of the products to ensure that all items have been received and to track the quantity of each item. This inventory should be updated regularly to reflect any changes in product quantity. The store staff should also take the time to check the expiration dates of any perishable items, as these items must be discarded if they are past their expiration date.
r_5	The Trading System provides users with the opportunity to generate stock-related reports. These reports can provide valuable insights into the performance of stocks and the overall market. They can also be used to identify trends and make informed decisions about investments. The reports are generated using data from the stock exchange, and can include information such as the daily price and volume of stocks, the number of shares traded, the current market capitalization, and more. Additionally, the reports can be customized to focus on specific stocks or sectors, allowing users to gain a better understanding of the market.
r_6	The Trading System provides the opportunity to calculate the mean times a delivery from each supplier to a considered enterprise takes. This system allows the enterprise to compare the delivery times of different suppliers and make informed decisions on which suppliers to use in the future. This system also allows the enterprise to track the delivery times of each supplier and make changes as needed to ensure the best delivery times possible. The system also allows the enterprise to identify trends in the delivery times of each supplier and make changes in the supply chain accordingly.
r_7	The System provides the opportunity to change the sales price for a product. This is a useful tool for businesses that need to adjust prices according to market conditions or customer demand. With the System, businesses can quickly and easily adjust prices without having to manually enter new prices into their system. This feature allows businesses to remain competitive and maximize their profits. Additionally, the System can be used to track sales trends and make informed decisions about pricing.

r_i	Descripción
r_8	If a store runs out of a certain product, it is possible to start a query to check whether those products are available at other stores of the enterprise. The enterprise server and the store servers need to synchronize their data on demand, as one scheduled update per day or per hour is not sufficient. After a successful query, the critical product can be shipped from one to other stores, but it has to be decided using heuristics to compute the future selling frequency, whether the transportation is meaningful. The store server is able to start a query at the enterprise server, which in turn starts a query for products available at other stores. The enterprise server does not have the current global data for stores at any time due to a write caching latency at the store servers, so it has to trigger all store servers to push their local data to the enterprise server.
r_9	Allows the store client to change the incoming mark of the moved products that have arrived at the store. This means that the store client can adjust the amount of incoming products in the inventory, which is helpful for keeping track of stock levels and ensuring that the store has the correct amount of products. The user interface makes this process quick and easy, allowing the store to quickly adjust the incoming mark and keep their inventory up to date.

E.2. PLN

E.2.1. Extracción de sustantivos y verbos

Sustantivos S = { 'CashDesk', 'Customer', 'Product', 'EitherCreditCard', 'Cash', 'Product', 'Customer', 'BarcodeScanner', 'SimilarDevice', 'Customer', 'Item', 'CreditCard', 'Cash', 'Payment', 'Customer', 'Receipt', 'Purchase', 'Cashier', 'AdditionalInformation', 'Assistance', 'Customer', 'CashDesk', 'ExpressMode', 'CertainCondition', 'Mode', 'Cashier', 'Payment', 'CashDesk', 'Speed', 'Cashier', 'NormalMode', 'Button', 'CashDesk', 'LightDisplay', 'CashDesk', 'CurrentMode', 'DifferentColor', 'ExpressNormalMode', 'TradingSystem', 'Opportunity', 'ProductItem', 'Variety', 'DifferentSupplier', 'System', 'Business', 'Price', 'DifferentSupplier', 'GoodDeal', 'Need', 'Business', 'Order', 'Notification', 'NewProduct', 'System', 'Tool', 'Inventory', 'CustomerRelationship', 'System', 'Business', 'Product', 'Price', 'DifferentSupplier', 'Business', 'Time', 'Money', 'GoodDeal', 'Need', 'Product', 'Store', 'GoodCondition', 'StoreStaff', 'Inventory', 'Product', 'Item',

'Quantity', 'Item', 'Inventory', 'Change', 'ProductQuantity', 'StoreStaff', 'Time', 'ExpirationDate', 'PerishableItem', 'Item', 'ExpirationDate', 'TradingSystem', 'User', 'Opportunity', 'Stock-RelateReport', 'Report', 'ValuableInsight', 'Performance', 'Stock', 'OverallMarket', 'Trend', 'InformedDecision', 'Investment', 'Report', 'Datum', 'StockExchange', 'Information', 'DailyPrice', 'Volume', 'Stock', 'Number', 'Share', 'CurrentMarketCapitalization', 'Report', 'SpecificStock', 'Sector', 'User', 'WellUnderstanding', 'Market', 'TradingSystem', 'Opportunity', 'MeanTime', 'Delivery', 'Supplier', 'ConsiderEnterpriseTake', 'System', 'Enterprise', 'DeliveryTime', 'DifferentSupplier', 'InformedDecision', 'Supplier', 'Future', 'System', 'Enterprise', 'DeliveryTime', 'Supplier', 'Change', 'GoodDeliveryTime', 'System', 'Enterprise', 'Trend', 'DeliveryTime', 'Supplier', 'Change', 'SupplyChain', 'System', 'Opportunity', 'SalePrice', 'Product', 'UsefulTool', 'Business', 'Price', 'MarketCondition', 'CustomerDemand', 'System', 'Business', 'Price', 'NewPrice', 'System', 'Feature', 'Business', 'Profit', 'System', 'SaleTrend', 'InformedDecision', 'Pricing', 'Store', 'CertainProduct', 'Query', 'Product', 'Store', 'Enterprise', 'EnterpriseServer', 'StoreServer', 'Datum', 'Demand', 'OneScheduleUpdate', 'Day', 'Hour', 'SuccessfulQuery', 'CriticalProduct', 'Store', 'Heuristic', 'FutureSellingFrequency', 'Transportation', 'StoreServer', 'Query', 'EnterpriseServer', 'Turn', 'Query', 'Product', 'Store', 'EnterpriseServer', 'CurrentGlobalDatum', 'Store', 'Time', 'WriteCacheLatency', 'StoreServer', 'StoreServer', 'LocalDatum', 'EnterpriseServer', 'Extension', 'StoreClient', 'IncomingMark', 'MoveProduct', 'Store', 'StoreClient', 'Amount', 'IncomingProduct', 'Inventory', 'Track', 'StockLevel', 'Store', 'CorrectAmount', 'Product', 'UserInterface', 'Process', 'Store', 'IncomingMark', 'Inventory', 'Date' }

S_O = { (nombre='the cash desk', lema='cashdesk', raiz='at'), (nombre='customers', lema='customer', raiz='purchase'), (nombre='products', lema='product', raiz='purchase'), (nombre='either a credit card', lema='eithercreditcard', raiz='use'), (nombre='cash', lema='cash', raiz='card'), (nombre='the products', lema='product', raiz='detect'), (nombre='a customer', lema='customer', raiz='want'), (nombre='a barcode scanner', lema='barcodescanner', raiz='use'), (nombre='similar device', lema='similardevice', raiz='scanner'), (nombre='the customer', lema='customer', raiz='pay'), (nombre='their items', lema='item', raiz='for'), (nombre='the credit card', lema='creditcard', raiz='use'), (nombre='cash', lema='cash', raiz='card'), (nombre='the payment', lema='payment', raiz='process'),
(nombre='the customer', lema='customer', raiz='give'), (nombre='a receipt', lema='receipt', raiz='give'), (nombre='their purchase', lema='purchase', raiz='for'), (nombre='the cashier', lema='cashier', raiz='provide'), (nombre='any additional information', lema='additionalinformation', raiz='provide'), (nombre='assistance', lema='assistance', raiz='information'), (nombre='the customer', lema='customer', raiz='need'), (nombre='the cash desk', lema='cashdesk', raiz='switch'), (nombre='express mode', lema='expressmode', raiz='into'), (nombre='certain conditions', lema='certaincondition', raiz='meet'),

(nombre='this mode', lema='mode', raiz='allow'), (nombre='the cashier', lema='cashier', raiz='process'), (nombre='payments', lema='payment', raiz='process'), (nombre='the cash desk', lema='cashdesk', raiz='optimize'), (nombre='speed', lema='speed', raiz='for'), (nombre='the cashier', lema='cashier', raiz='switch'), (nombre='normal mode', lema='normalmode', raiz='to'), (nombre='a button', lema='button', raiz='press'), (nombre='the cash desk', lema='cashdesk', raiz='at'), (nombre='the light display', lema='lightdisplay', raiz='indicate'), (nombre='the cash desk', lema='cashdesk', raiz='on'), (nombre='the current mode', lema='currentmode', raiz='indicate'), (nombre='different colors', lema='differentcolor', raiz='with'), (nombre='express and normal modes', lema='expressnormalmode', raiz='for'), (nombre='the trading system', lema='tradingsystem', raiz='provide'), (nombre='an opportunity', lema='opportunity', raiz='provide'), (nombre='product items', lema='productitem', raiz='order'), (nombre='a variety', lema='variety', raiz='from'), (nombre='different suppliers', lema='differentsupplier', raiz='of'), (nombre='this system', lema='system', raiz='allow'), (nombre='businesses', lema='business', raiz='compare'), (nombre='prices', lema='price', raiz='compare'), (nombre='different suppliers', lema='differentsupplier', raiz='from'), (nombre='the best deal', lema='gooddeal', raiz='select'), (nombre='their needs', lema='need', raiz='for'), (nombre='businesses', lema='business', raiz='track'), (nombre='their orders', lema='order', raiz='track'), (nombre='notifications', lema='notification', raiz='receive'), (nombre='new products', lema='newproduct', raiz='be'), (nombre='the system', lema='system', raiz='provide'), (nombre='tools', lema='tool', raiz='provide'), (nombre='inventory', lema='inventory', raiz='track'), (nombre='customer relationships', lema='customerrelationship', raiz='manage'), (nombre='the system', lema='system', raiz='allow'), (nombre='businesses', lema='business', raiz='search'), (nombre='products', lema='product', raiz='for'), (nombre='prices', lema='price', raiz='compare'), (nombre='different suppliers', lema='differentsupplier', raiz='from'), (nombre='businesses', lema='business', raiz='save'), (nombre='time', lema='time', raiz='save'), (nombre='money', lema='money', raiz='time'), (nombre='the best deals', lema='gooddeal', raiz='find'), (nombre='their needs', lema='need', raiz='for'), (nombre='products', lema='product', raiz='order'), (nombre='the store', lema='store', raiz='at'), (nombre='good condition', lema='goodcondition', raiz='in'), (nombre='the store staff', lema='storestaff', raiz='take'), (nombre='an inventory', lema='inventory', raiz='take'), (nombre='the products', lema='product', raiz='of'), (nombre='all items', lema='item', raiz='receive'), (nombre='the quantity', lema='quantity', raiz='track'), (nombre='each item', lema='item', raiz='of'), (nombre='this inventory', lema='inventory', raiz='update'), (nombre='any changes', lema='change', raiz='reflect'), (nombre='product quantity', lema='productquantity', raiz='in'), (nombre='the store staff', lema='storestaff', raiz='take'), (nombre='the time', lema='time', raiz='take'), (nombre='the expiration dates', lema='expirationdate', raiz='check'), (nombre='any perishable items', lema='perishableitem', raiz='of'), (nombre='these items', lema='item', raiz='discard'), (nombre='their expiration date',

lema='expirationdate', raiz='past'), (nombre='the trading system', lema='tradingsystem', raiz='provide'), (nombre='users', lema='user', raiz='provide'), (nombre='the opportunity', lema='opportunity', raiz='with'), (nombre='stock-related reports', lema='stock-relatereport', raiz='generate'), (nombre='these reports', lema='report', raiz='provide'), (nombre='valuable insights', lema='valuableinsight', raiz='provide'), (nombre='the performance', lema='performance', raiz='into'), (nombre='stocks', lema='stock', raiz='of'), (nombre='the overall market', lema='overallmarket', raiz='stock'), (nombre='trends', lema='trend', raiz='identify'), (nombre='informed decisions', lema='informeddecision', raiz='make'), (nombre='investments', lema='investment', raiz='about'), (nombre='the reports', lema='report', raiz='generate'), (nombre='data', lema='datum', raiz='use'), (nombre='the stock exchange', lema='stockexchange', raiz='from'), (nombre='information', lema='information', raiz='include'), (nombre='the daily price', lema='dailyprice', raiz='as'), (nombre='volume', lema='volume', raiz='price'), (nombre='stocks', lema='stock', raiz='of'), (nombre='the number', lema='number', raiz='include'), (nombre='shares', lema='share', raiz='of'), (nombre='the current market capitalization', lema='currentmarketca raiz='number'), (nombre='the reports', lema='report', raiz='customize'), (nombre='specific stocks', lema='specificstock', raiz='on'), (nombre='sectors', lema='sector', raiz='stock'), (nombre='users', lema='user', raiz='gain'), (nombre='a better understanding', lema='wellunderstanding', raiz='gain'), (nombre='the market', lema='market', raiz='of'), (nombre='the trading system', lema='tradingsystem', raiz='provide'), (nombre='the opportunity', lema='opportunity', raiz='provide'), (nombre='the mean times', lema='meantime', raiz='calculate'), (nombre='a delivery', lema='delivery', raiz='calculate'), (nombre='each supplier', lema='supplier', raiz='from'), (nombre='a considered enterprise takes', lema='considerenterprisetake', raiz='to'), (nombre='this system', lema='system', raiz='allow'), (nombre='the enterprise', lema='enterprise', raiz='compare'), (nombre='the delivery times', lema='deliverytime' raiz='compare'), (nombre='different suppliers', lema='differentsupplier', raiz='of'), (nombre='informed decisions', lema='informeddecision', raiz='make'), (nombre='suppliers', lema='supplier', raiz='use'), (nombre='the future', lema='future', raiz='in'), (nombre='this system', lema='system', raiz='allow'), (nombre='the enterprise', lema='enterprise', raiz='track'), (nombre='the delivery times', lema='deliverytime', raiz='track'), (nombre='each supplier', lema='supplier', raiz='of'), (nombre='changes', lema='change', raiz='make'), (nombre='the best delivery times', lema='gooddeliverytime', raiz='ensure'), (nombre='the system', lema='system', raiz='allow'), (nombre='the enterprise', lema='enterprise', raiz='identify'), (nombre='trends', lema='trend', raiz='identify'), (nombre='the delivery times', lema='deliverytime', raiz='in'), (nombre='each supplier', lema='supplier', raiz='of'), (nombre='changes', lema='change', raiz='make'), (nombre='the supply chain', lema='supplychain', raiz='in'), (nombre='the system', lema='system', raiz='provide'), (nombre='the opportunity', lema='opportunity', raiz='provide'), (nombre='the sales price', lema='saleprice', raiz='change'), (nombre='a product', lema='product',

raiz='for'), (nombre='a useful tool', lema='usefultool', raiz='be'), (nombre='businesses', lema='business', raiz='for'), (nombre='prices', lema='price', raiz='adjust'), (nombre='market conditions', lema='marketcondition', raiz='to'), (nombre='customer demand', lema='customerdemand', raiz='condition'), (nombre='the system', lema='system', raiz='with'), (nombre='businesses', lema='business', raiz='adjust'), (nombre='prices', lema='price', raiz='adjust'), (nombre='new prices', lema='newprice', raiz='enter'), (nombre='their system', lema='system', raiz='into'), (nombre='this feature', lema='feature', raiz='allow'), (nombre='businesses', lema='business', raiz='remain'), (nombre='their profits', lema='profit', raiz='maximize'), (nombre='the system', lema='system', raiz='use'), (nombre='sales trends', lema='saletrend', raiz='track'), (nombre='informed decisions', lema='informeddecision', raiz='make'), (nombre='pricing', lema='pricing', raiz='about'), (nombre='a store', lema='store', raiz='run'), (nombre='a certain product', lema='certainproduct', raiz='of'), (nombre='a query', lema='query', raiz='start'), (nombre='those products', lema='product', raiz='be'), (nombre='other stores', lema='store', raiz='at'), (nombre='the enterprise', lema='enterprise', raiz='of'), (nombre='the enterprise server', lema='enterpriseserver raiz='need'), (nombre='the store servers', lema='storeserver', raiz='server'), (nombre='their data', lema='datum', raiz='synchronize'), (nombre='demand', lema='demand', raiz='on'), (nombre='one scheduled update', lema='onescheduleupdate', raiz='be'), (nombre='day', lema='day', raiz='per'), (nombre='hour', lema='hour', raiz='per'), (nombre='a successful query', lema='successfulquery', raiz='after'), (nombre='the critical product', lema='criticalproduct', raiz='ship'), (nombre='other stores', lema='store', raiz='to'), (nombre='heuristics', lema='heuristic', raiz='use'), (nombre='the future selling frequency', lema='futuresellingfrequency', raiz='compute'), (nombre='the transportation', lema='transportation', raiz='be'), (nombre='the store server', lema='storeserver', raiz='be'), (nombre='a query', lema='query', raiz='start'), (nombre='the enterprise server', lema='enterpriseserver', raiz='at'), (nombre='turn', lema='turn', raiz='in'), (nombre='a query', lema='query', raiz='start'), (nombre='products', lema='product', raiz='for'), (nombre='other stores', lema='store', raiz='at'), (nombre='the enterprise server', lema='enterpriseserver', raiz='have'), (nombre='the current global data', lema='currentglobaldatum', raiz='have'), (nombre='stores', lema='store', raiz='for'), (nombre='any time', lema='time', raiz='at'), (nombre='a write caching latency', lema='writecachelatency', raiz='due'), (nombre='the store servers', lema='storeserver', raiz='at'), (nombre='all store servers', lema='storeserver', raiz='trigger'), (nombre='their local data', lema='localdatum', raiz='push'), (nombre='the enterprise server', lema='enterpriseserver', raiz='to'), (nombre='this extension', lema='extension', raiz='allow'), (nombre='the store client', lema='storeclient', raiz='change'), (nombre='the incoming mark', lema='incomingmark', raiz='change'), (nombre='the moved products', lema='moveproduct', raiz='of'), (nombre='the store', lema='store', raiz='at'), (nombre='the store client', lema='storeclient', raiz='adjust'), (nombre='the amount', lema='amount', raiz='adjust'),

(nombre='incoming products', lema='incomingproduct', raiz='of'), (nombre='the inventory', lema='inventory', raiz='in'), (nombre='track', lema='track', raiz='keep'), (nombre='stock levels', lema='stocklevel', raiz='of'), (nombre='the store', lema='store', raiz='have'), (nombre='the correct amount', lema='correctamount', raiz='have'), (nombre='products', lema='product', raiz='of'), (nombre='the user interface', lema='userinterface', raiz='make'), (nombre='this process', lema='process', raiz='quick'), (nombre='the store', lema='store', raiz='adjust'), (nombre='the incoming mark', lema='incomingmark', raiz='adjust'), (nombre='their inventory', lema='inventory', raiz='keep'), (nombre='date', lema='date', raiz='to')}

Verbos

V = { 'purchase', 'purchase', 'use', 'want', 'buy', 'detect', 'detect', 'use', 'pay', 'use', 'process', 'process', 'give', 'give', 'provide', 'mayneed', 'need', 'switch', 'meet', 'meet', 'allow', 'process', 'optimize', 'optimize', 'switch', 'switch', 'press', 'indicate', 'indicate', 'provide', 'order', 'allow', 'compare', 'select', 'allow', 'track', 'receive', 'provide', 'track', 'manage', 'allow', 'search', 'compare', 'help', 'save', 'find', 'order', 'arrive', 'inspect', 'inspect', 'ensure', 'take', 'ensure', 'receive', 'receive', 'track', 'update', 'update', 'reflect', 'take', 'check', 'discard', 'discard', 'provide', 'generate', 'relate', 'provide', 'provide', 'use', 'use', 'identify', 'make', 'generate', 'generate', 'use', 'include', 'include', 'trade', 'customize', 'customize', 'focus', 'allow', 'gain', 'provide', 'calculate', 'consider', 'allow', 'compare', 'make', 'use', 'allow', 'track', 'make', 'need', 'ensure', 'allow', 'identify', 'make', 'provide', 'change', 'need', 'adjust', 'accord', 'adjust', 'enter', 'allow', 'remain', 'maximize', 'use', 'use', 'track', 'make', 'run', 'start', 'check', 'need', 'synchronize', 'schedule', 'ship', 'ship', 'decide', 'decide', 'use', 'compute', 'start', 'start', 'cache', 'trigger', 'push', 'allow', 'change', 'move', 'arrive', 'arrive', 'mean', 'adjust', 'adjust', 'keep', 'ensure', 'make', 'allow', 'adjust', 'keep' }

V0 = { (nombre='can purchase', lema='purchase'), (nombre='purchase', lema='purchase'), (nombre='using', lema='use'), (nombre='wants', lema='want'), (nombre='buy', lema='buy'), (nombre='are detected', lema='detect'), (nombre='detected', lema='detect'), (nombre='using', lema='use'), (nombre='pays', lema='pay'), (nombre='using', lema='use'), (nombre='is processed', lema='process'), (nombre='processed', lema='process'), (nombre='is given', lema='give'), (nombre='given', lema='give'), (nombre='provide', lema='provide'), (nombre='may need', lema='mayneed'), (nombre='need', lema='need'), (nombre='switches', lema='switch'), (nombre='are met', lema='meet'), (nombre='met', lema='meet'), (nombre='allows', lema='allow'), (nombre='process', lema='process'), (nombre='is optimized', lema='optimize'), (nombre='optimized', lema='optimize'), (nombre='can switch', lema='switch'), (nombre='switch', lema='switch'), (nombre='pressing', lema='press'), (nombre='will indicate', lema='indicate'), (nombre='indicate', lema='indicate'), (nombre='provides', lema='provide'), (nom-

bre='order', lema='order'), (nombre='allows', lema='allow'), (nombre='compare', lema='compare'), (nombre='select', lema='select'), (nombre='allows', lema='allow'), (nombre='track', lema='track'), (nombre='receive', lema='receive'), (nombre='provides', lema='provide'), (nombre='tracking', lema='track'), (nombre='managing', lema='manage'), (nombre='allows', lema='allow'), (nombre='search', lema='search'), (nombre='compare', lema='compare'), (nombre='helps', lema='help'), (nombre='save', lema='save'), (nombre='finding', lema='find'), (nombre='ordered', lema='order'), (nombre='arrive', lema='arrive'), (nombre='be inspected', lema='inspect'), (nombre='inspected', lema='inspect'), (nombre='ensure', lema='ensure'), (nombre='take', lema='take'), (nombre='ensure', lema='ensure'), (nombre='been received', lema='receive'), (nombre='received', lema='receive'), (nombre='track', lema='track'), (nombre='be updated', lema='update'), (nombre='updated', lema='update'), (nombre='reflect', lema='reflect'), (nombre='take', lema='take'), (nombre='check', lema='check'), (nombre='be discarded', lema='discard'), (nombre='discarded', lema='discard'), (nombre='provides', lema='provide'), (nombre='generate', lema='generate'), (nombre='related', lema='relate'), (nombre='can provide', lema='provide'), (nombre='provide', lema='provide'), (nombre='be used', lema='use'), (nombre='used', lema='use'), (nombre='identify', lema='identify'), (nombre='make', lema='make'), (nombre='are generated', lema='generate'), (nombre='generated', lema='generate'), (nombre='using', lema='use'), (nombre='can include', lema='include'), (nombre='include', lema='include'), (nombre='traded', lema='trade'), (nombre='be customized', lema='customize'), (nombre='customized', lema='customize'), (nombre='focus', lema='focus'), (nombre='allowing', lema='allow'), (nombre='gain', lema='gain'), (nombre='provides', lema='provide'), (nombre='calculate', lema='calculate'), (nombre='considered', lema='consider'), (nombre='allows', lema='allow'), (nombre='compare', lema='compare'), (nombre='make', lema='make'), (nombre='use', lema='use'), (nombre='allows', lema='allow'), (nombre='track', lema='track'), (nombre='make', lema='make'), (nombre='needed', lema='need'), (nombre='ensure', lema='ensure'), (nombre='allows', lema='allow'), (nombre='identify', lema='identify'), (nombre='make', lema='make'), (nombre='provides', lema='provide'), (nombre='change', lema='change'), (nombre='need', lema='need'), (nombre='adjust', lema='adjust'), (nombre='according', lema='accord'), (nombre='adjust', lema='adjust'), (nombre='enter', lema='enter'), (nombre='allows', lema='allow'), (nombre='remain', lema='remain'), (nombre='maximize', lema='maximize'), (nombre='be used', lema='use'), (nombre='used', lema='use'), (nombre='track', lema='track'), (nombre='make', lema='make'), (nombre='runs', lema='run'), (nombre='start', lema='start'), (nombre='check', lema='check'), (nombre='need', lema='need'), (nombre='synchronize', lema='synchronize'), (nombre='scheduled', lema='schedule'), (nombre='be shipped', lema='ship'), (nombre='shipped', lema='ship'), (nombre='be decided', lema='decide'), (nombre='decided', lema='decide'), (nombre='using', lema='use'), (nombre='compute', lema='compute'), (nombre='start', lema='start'),

(nombre='starts', lema='start'), (nombre='caching', lema='cache'), (nombre='trigger', lema='trigger'), (nombre='push', lema='push'), (nombre='allows', lema='allow'), (nombre='change', lema='change'), (nombre='moved', lema='move'), (nombre='have arrived', lema='arrive'), (nombre='arrived', lema='arrive'), (nombre='means', lema='mean'), (nombre='can adjust', lema='adjust'), (nombre='adjust', lema='adjust'), (nombre='keeping', lema='keep'), (nombre='ensuring', lema='ensure'), (nombre='makes', lema='make'), (nombre='allowing', lema='allow'), (nombre='adjust', lema='adjust'), (nombre='keep', lema='keep') }

E.2.2. Pre-procesamiento de Sustantivos y Verbos

Sustantivos

S_2 = { 'quantity', 'stockexchange', 'differentcolor', 'lightdisplay', 'storeclient', 'normalmode', 'deliverytime', 'order', 'enterprise', 'customerdemand', 'inventory', 'productquantity', 'price', 'profit', 'future', 'dailyprice', 'writecachelatency', 'button', 'user', 'customerrelationship', 'tool', 'marketcondition', 'criticalproduct', 'datum', 'saletrend', 'turn', 'userinterface', 'report', 'storeserver', 'date', 'cashdesk', 'customer', 'assistance', 'moveproduct', 'certaincondition', 'query', 'business', 'variety', 'incomingmark', 'overallmarket', 'demand', 'need', 'item', 'heuristic', 'creditcard', 'stock', 'eithercreditcard', 'perishableitem', 'amount', 'stocklevel', 'day', 'additionalinformation', 'speed', 'number', 'cashier', 'meantime', 'opportunity', 'change', 'expressmode', 'payment', 'hour', 'storestaff', 'incomingproduct', 'product', 'mode', 'performance', 'similardevice', 'barcodescanner', 'purchase', 'productitem', 'system', 'delivery', 'futuresellingfrequency', 'process', 'valuableinsight', 'currentmode', 'pricing', 'feature', 'newproduct', 'trend', 'share', 'stock-relatereport', 'expressnormalmode', 'extension', 'supplier', 'gooddeal', 'investment', 'saleprice', 'currentmarketcapitalization', 'receipt', 'notification', 'usefultool', 'certainproduct', 'newprice', 'sector', 'wellunderstanding', 'successfulquery', 'store', 'localdatum', 'expirationdate', 'gooddeliverytime', 'cash', 'tradingsystem', 'goodcondition', 'currentglobaldatum', 'enterpriseserver', 'correctamount', 'differentsupplier', 'volume', 'market', 'supplychain', 'onescheduleupdate', 'time', 'track', 'information', 'transportation', 'money', 'specificstock', 'considerenterprisetake', 'informeddecision' }

Verbos

V_2 = { 'check', 'ensure', 'calculate', 'change', 'mean', 'indicate', 'keep', 'save', 'meet', 'run', 'help', 'compare', 'maximize', 'accord', 'identify', 'remain', 'order', 'purchase', 'switch', 'want', 'find', 'take', 'discard', 'adjust', 'cache', 'need', 'generate', 'use', 'provide', 'buy', 'update', 'process', 'enter', 'press', 'start', 'trigger', 'reflect', 'inspect', 'trade', 'allow', 'arrive', 'synchronize', 'search', 'consider', 'manage', 'gain', 'ship', 'compute', 'include', 'decide', 'move', 'detect', 'pay',

'select', 'relate', 'schedule', 'track', 'push', 'mayneed', 'make', 'receive', 'give', 'optimize', 'focus', 'customize' }

E.2.3. Métricas de Similaridad Sintáctica

Sustantivos

S_3 = { 'quantity', 'stockexchange', 'differentcolor', 'lightdisplay', 'storeclient', 'normalmode', 'order', 'enterprise', 'inventory', 'productquantity', 'price', 'informeddecision', 'future', 'writecachelatency', 'button', 'user', 'customerrelationship', 'tool', 'datum', 'saletrend', 'turn', 'userinterface', 'report', 'date', 'cashdesk', 'customer', 'assistance', 'query', 'business', 'variety', 'incomingmark', 'overallmarket', 'demand', 'need', 'item', 'heuristic', 'creditcard', 'stock', 'perishableitem', 'amount', 'stocklevel', 'day', 'speed', 'number', 'cashier', 'meantime', 'opportunity', 'change', 'hour', 'payment', 'incomingproduct', 'storestaff', 'product', 'mode', 'performance', 'similardevice', 'barcodescanner', 'purchase', 'system', 'delivery', 'futuresellingfrequency', 'process', 'valuableinsight', 'currentmode', 'pricing', 'trend', 'share', 'stock-relatereport', 'extension', 'supplier', 'gooddeal', 'investment', 'saleprice', 'currentmarketcapitalization', 'receipt', 'notification', 'usefultool', 'wellunderstanding', 'successfulquery', 'store', 'localdatum', 'expirationdate', 'cash', 'tradingsystem', 'goodcondition', 'currentglobaldatum', 'correctamount', 'differentsupplier', 'market', 'onescheduleupdate', 'supplychain', 'volume', 'track', 'information', 'transportation', 'money', 'specificstock', 'considerenterprisetake', 'profit' }

Verbos

V_3 = { 'change', 'calculate', 'check', 'ensure', 'keep', 'indicate', 'mean', 'save', 'meet', 'run', 'help', 'compare', 'maximize', 'accord', 'identify', 'remain', 'order', 'purchase', 'find', 'adjust', 'discard', 'switch', 'take', 'want', 'cache', 'need', 'generate', 'use', 'provide', 'buy', 'update', 'enter', 'press', 'start', 'trigger', 'reflect', 'inspect', 'trade', 'arrive', 'allow', 'synchronize', 'search', 'consider', 'manage', 'gain', 'ship', 'compute', 'include', 'decide', 'move', 'detect', 'pay', 'select', 'relate', 'schedule', 'track', 'push', 'make', 'mayneed', 'receive', 'give', 'optimize', 'focus', 'customize' }

E.3. Clusterización

E.3.1. Generación del Grafo

Nodos

N = { 'amount', 'assistance', 'barcodescanner', 'business', 'button', 'cash', 'cashdesk', 'cashier', 'change', 'considerenterprisetake', 'correctamount', 'creditcard', 'currentglobaldatum', 'current-marketcapitalization', 'currentmode', 'customer', 'customerrelationship', 'date', 'datum', 'day', 'delivery', 'demand', 'differentcolor', 'differentsupplier', 'enterprise', 'expirationdate', 'extension', 'future', 'futuresellingfrequency', 'goodcondition', 'gooddeal', 'heuristic', 'hour', 'incomingmark', 'incomingproduct', 'information', 'informeddecision', 'inventory', 'investment', 'item', 'lightdis-play', 'localdatum', 'market', 'meantime', 'mode', 'money', 'need', 'normalmode', 'notification', 'number', 'onescheduleupdate', 'opportunity', 'order', 'overallmarket', 'payment', 'performance', 'perishableitem', 'price', 'pricing', 'process', 'product', 'productquantity', 'profit', 'purchase', 'quan-tity', 'query', 'receipt', 'report', 'saleprice', 'saletrend', 'share', 'similardevice', 'specificstock', 'speed', 'stock', 'stock-relatereport', 'stockexchange', 'stocklevel', 'store', 'storeclient', 'storestaff', 'successfulquery', 'supplier', 'supplychain', 'system', 'tool', 'track', 'tradingsystem', 'transporta-tion', 'trend', 'turn', 'usefultool', 'user', 'userinterface', 'valuableinsight', 'variety', 'volume', 'wellun-derstanding', 'writecachelatency', 'adjust', 'allow', 'calculate', 'compare', 'gain', 'generate', 'stock-relatereport', 'give', 'identify', 'include', 'indicate', 'keep', 'make', 'press', 'provide', 'receive', 'save', 'start', 'switch', 'take', 'use' }

Aristas

E = { ('amount', 'adjust'), ('barcodescanner', 'use'), ('business', 'adjust'), ('business', 'com-pare'), ('business', 'save'), ('business', 'track'), ('button', 'press'), ('cashdesk', 'switch'), ('cashier', 'press'), ('cashier', 'provide'), ('cashier', 'switch'), ('change', 'incomingmark'), ('change', 'saleprice'), ('change', 'storeclient'), ('change', 'make'), ('creditcard', 'use'), ('currentmode', 'indicate'), ('cus-tomer', 'give'), ('customer', 'need'), ('customer', 'purchase'), ('datum', 'use'), ('delivery', 'calcu-late'), ('delivery', 'compare'), ('delivery', 'track'), ('enterprise', 'compare'), ('enterprise', 'iden-tify'), ('enterprise', 'need'), ('enterprise', 'track'), ('extension', 'allow'), ('future', 'allow'), ('heuris-tic', 'use'), ('incomingmark', 'adjust'), ('information', 'include'), ('information', 'provide'), ('in-formeddecision', 'make'), ('inventory', 'keep'), ('inventory', 'take'), ('inventory', 'track'), ('item', 'receive'), ('item', 'save'), ('item', 'take'), ('lightdisplay', 'indicate'), ('meantime', 'calculate'), ('mode', 'allow'), ('notification', 'receive'), ('number', 'include'), ('opportunity', 'provide'), ('order', 'prod-uct'), ('order', 'track'), ('payment', 'press'), ('price', 'adjust'), ('price', 'compare'), ('product', 'pur-chase'), ('quantity', 'track'), ('query', 'start'), ('receipt', 'give'), ('report', 'generate'), ('report', 'pro-vide'), ('saletrend', 'track'), ('store', 'adjust'), ('storeclient', 'adjust'), ('storestaff', 'take'), ('sup-plier', 'use'), ('system', 'allow'), ('system', 'provide'), ('system', 'use'), ('tool', 'provide'), ('track', 'keep'), ('tradingsystem', 'provide'), ('trend', 'identify'), ('user', 'gain'), ('user', 'provide'), ('userin-terface', 'make'), ('valuableinsight', 'provide'), ('wellunderstanding', 'gain'), ('generate', 'stock-

relatereport') }

Eliminar Nodos Aislados

Nodos

N' = { 'amount', 'barcodescanner', 'business', 'button', 'cashdesk', 'cashier', 'change', 'creditcard', 'currentmode', 'customer', 'datum', 'delivery', 'enterprise', 'extension', 'future', 'heuristic', 'incomingmark', 'information', 'informeddecision', 'inventory', 'item', 'lightdisplay', 'meantime', 'mode', 'need', 'notification', 'number', 'opportunity', 'order', 'payment', 'price', 'product', 'purchase', 'quantity', 'query', 'receipt', 'report', 'saleprice', 'saletrend', 'store', 'storeclient', 'storestaff', 'supplier', 'system', 'tool', 'track', 'tradingsystem', 'trend', 'user', 'userinterface', 'valuableinsight', 'wellunderstanding', 'adjust', 'allow', 'calculate', 'compare', 'gain', 'generate', 'stockrelatereport', 'give', 'identify', 'include', 'indicate', 'keep', 'make', 'press', 'provide', 'receive', 'save', 'start', 'switch', 'take', 'use' }

Aristas

E' = { ('amount', 'adjust'), ('barcodescanner', 'use'), ('business', 'adjust'), ('business', 'compare'), ('business', 'save'), ('business', 'track'), ('button', 'press'), ('cashdesk', 'switch'), ('cashier', 'press'), ('cashier', 'provide'), ('cashier', 'switch'), ('change', 'incomingmark'), ('change', 'saleprice'), ('change', 'storeclient'), ('change', 'make'), ('creditcard', 'use'), ('currentmode', 'indicate'), ('customer', 'give'), ('customer', 'need'), ('customer', 'purchase'), ('datum', 'use'), ('delivery', 'calculate'), ('delivery', 'compare'), ('delivery', 'track'), ('enterprise', 'compare'), ('enterprise', 'identify'), ('enterprise', 'need'), ('enterprise', 'track'), ('extension', 'allow'), ('future', 'allow'), ('heuristic', 'use'), ('incomingmark', 'adjust'), ('information', 'include'), ('information', 'provide'), ('informeddecision', 'make'), ('inventory', 'keep'), ('inventory', 'take'), ('inventory', 'track'), ('item', 'receive'), ('item', 'save'), ('item', 'take'), ('lightdisplay', 'indicate'), ('meantime', 'calculate'), ('mode', 'allow'), ('notification', 'receive'), ('number', 'include'), ('opportunity', 'provide'), ('order', 'product'), ('order', 'track'), ('payment', 'press'), ('price', 'adjust'), ('price', 'compare'), ('product', 'purchase'), ('quantity', 'track'), ('query', 'start'), ('receipt', 'give'), ('report', 'generate'), ('report', 'provide'), ('saletrend', 'track'), ('store', 'adjust'), ('storeclient', 'adjust'), ('storestaff', 'take'), ('supplier', 'use'), ('system', 'allow'), ('system', 'provide'), ('system', 'use'), ('tool', 'provide'), ('track', 'keep'), ('tradingsystem', 'provide'), ('trend', 'identify'), ('user', 'gain'), ('user', 'provide'), ('userinterface', 'make'), ('valuableinsight', 'provide'), ('wellunderstanding', 'gain'), ('generate', 'stockrelatereport') }

E.4. Detección de Comunidades

C = { ’opportunity’: 0, ’report’: 0, ’stockrelatereport’: 0, ’user’: 0, ’information’: 0, ’generate’: 0, ’number’: 0, ’provide’: 0, ’tool’: 0, ’tradingsystem’: 0, ’gain’: 0, ’include’: 0, ’wellunderstanding’: 0, ’valuableinsight’: 0, ’incomingmark’: 1, ’userinterface’: 1, ’change’: 1, ’adjust’: 1, ’make’: 1, ’amount’: 1, ’price’: 1, ’storeclient’: 1, ’compare’: 1, ’informeddecision’: 1, ’store’: 1, ’business’: 1, ’saleprice’: 1, ’heuristic’: 2, ’creditcard’: 2, ’allow’: 2, ’datum’: 2, ’extension’: 2, ’supplier’: 2, ’mode’: 2, ’system’: 2, ’use’: 2, ’future’: 2, ’barcodescanner’: 2, ’receipt’: 3, ’order’: 3, ’purchase’: 3, ’enterprise’: 3, ’need’: 3, ’product’: 3, ’trend’: 3, ’give’: 3, ’customer’: 3, ’identify’: 3, ’saletrend’: 4, ’meantime’: 4, ’quantity’: 4, ’calculate’: 4, ’track’: 4, ’keep’: 4, ’inventory’: 4, ’delivery’: 4, ’cashier’: 5, ’switch’: 5, ’button’: 5, ’payment’: 5, ’press’: 5, ’cashdesk’: 5, ’receive’: 6, ’item’: 6, ’notification’: 6, ’take’: 6, ’storestaff’: 6, ’save’: 6, ’currentmode’: 7, ’indicate’: 7, ’lightdisplay’: 7, ’query’: 8, ’start’: 8 }

E.5. Microservicios

E.5.1. Nombramiento de Microservicios

Micros = { CashDeskService’, ’TradingSystemService’, ’InventoryService’, ’SaleTrendService’, ’CashierService’, ’StoreStaffService’, ’LightDisplayService’, ’QueryService’, ’IncomingMarkService’ }