Pontificia Universidad JAVERIANA Cali | Facultad de Ingeniería y Ciencias

Acta de Correcciones al Proyecto de Grado
Ingeniería de Sistemas y Computación

Fecha: 01 de septiembre 2021

Autores: Yan Carlos Certuche Grueso

Nombre del Proyecto de Grado: Buscador sobre casos industriales de aplicación de Ingeniería de Líneas de Productos

Director:

Como indica el artículo 2.27 de las Directrices de Trabajo de Grado, he verificado que los estudiantes indicados arriba han implementado todas las correcciones que los Jurados del Proyecto de Grado definieron que se efectuaran, como consta en el Acta de Calificación correspondiente.

M.sC Luisa Rincón

Nota de Aceptación

Aprobado por el Comité de Trabajo de Grado en cumplimiento de los requisitos exigidos por la Pontificia Universidad Javeriana para optar el título de Ingeniero de Sistemas y computación.

Dr. HERNÁN CAMILO ROCHA NIÑO
Decano de la Facultad de Ingeniería.

Dr. GERARDO MAURICIO SARRIA
Director Carrera Ingeniería de Sistemas y Computación.

MsC. LUISA FERNANDA RINCÓN PÉREZ
Director(a) Trabajo.

MsB. JUAN PABLO GARCIA
Jurado 1

MsC. JUAN CARLOS MARTINEZ
Jurado 2

Pontificia Universidad Javeriana Cali
Facultad de Ingeniería.
Ingeniería de Sistemas y Computación.
Proyecto de Grado.

Buscador sobre casos industriales de aplicación de Ingeniería de Líneas de Productos

Yan Carlos Certuche Grueso

Directora: MSc. Luisa Fernanda Rincón Pérez

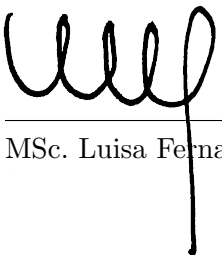
1 de septiembre de 2021

Santiago de Cali, 1 de septiembre de 2021.

Señores
Pontificia Universidad Javeriana Cali.
Dr. Gerardo M. Sarria
Director Carrera de Ingeniería de Sistemas y Computación.
Cali.

Cordial Saludo.

Por medio de la presente hago constar que en mi calidad de director de trabajo de grado he revisado el proyecto titulado "Buscador sobre casos industriales de aplicación de Ingeniería de Líneas de Productos" realizado por el estudiante de Ingeniería de Sistemas y Computación Yan Carlos Certuche Grueso (cod: 8919527), el cual se encuentra terminado y considero que cumple con los requisitos para ser sustentado.
Atentamente,



MSc. Luisa Fernanda Rincón Pérez

Santiago de Cali, 1 de septiembre de 2021.

Señores
Pontificia Universidad Javeriana Cali.
Dr. Gerardo M. Sarria
Director Carrera de Ingeniería de Sistemas y Computación.
Cali.

Cordial Saludo.

Me permito presentar a su consideración el proyecto de grado titulado "Buscador sobre casos industriales de aplicación de Ingeniería de Líneas de Productos" con el fin de cumplir con los requisitos exigidos por la Universidad y para que sea sometido a revisión del jurado y cumpla su aprobación, para conseguir posteriormente el título de Ingeniero de Sistemas y Computación.

Atentamente,

Yan Carlos Certuche G

Yan Carlos Certuche Grueso
Código: 8919527

Agradecimientos

Un agradecimiento especial a la directora de este proyecto, Luisa Fernanda Rincón, por su valioso acompañamiento, tiempo, mentoría y lecciones aprendidas tanto académicas como profesionales. A mis padres, Heraldo Certuche y Lucero Grueso, por enseñarme la virtud del servicio, el respeto, la humildad y el trabajo duro. Por sus incansables esfuerzos por ayudarme a ser mejor y ser mi apoyo incondicional. A mis hermanos, Yulieth, Daniel y Balentina, por compartir sus vidas conmigo, llenarme de amor y experiencias de vida. Por su apoyo, amistad y complicidad. A mis tías, Lilia y Juliana, y a mi primo Andrés Collazos, por sus sabios consejos y palabras de aliento. A Andrés Navia, por su apoyo absoluto en cada momento de mi carrera profesional. A mi pueblo, Crucero la Estrella - Cajibío, a mis compañeros de carrera, amigos y al grupo representativo Matamba Javeriana Cali por convertirse en mi segunda familia. Gracias a Dios por las personas que pone en mi camino. Este proyecto es por y para ellos.

Resumen

La Ingeniería de Líneas de Productos, un paradigma para desarrollar familias de productos, está enfocado en optimizar los recursos. Este modelo tiene el objetivo disminuir costos y aumentar la productividad. Un ejemplo es la industria automovilística, donde su implementación ha sido un éxito. De igual manera, se ha optado por implementar en otras industrias, como la del Software. Sin embargo, aquellas empresas que buscan adoptar el nuevo paradigma, no tienen a disposición la información de otras empresas que han logrado realizar la adopción de las Líneas de Productos.

No obstante, la labor de consultar información pertinente sobre las experiencias documentadas en Ingeniería de Líneas de Productos de otras empresas no es sencilla, puesto que no hay una herramienta que facilite este proceso. Este proyecto presenta el diseño, desarrollo e implementación de un prototipo web llamado Buscador sobre Casos Industriales de Aplicación de Ingeniería de Líneas de productos. Esta herramienta hace uso de una ontología desarrollada en el dominio del conocimiento de las experiencias documentadas en Ingeniería de Líneas de Productos. El objetivo es mitigar la brecha entre la información y el entidad interesada.

Palabras Clave: Líneas de Productos, Buscador, Visualización de Datos, Ingeniería de Líneas de productos, Experiencias Industriales, Ontologías, Desarrollo de Software, Desarrollo Web.

Abstract

Product Line Engineering, a paradigm for developing product families, it is focused on optimizing resources. This model aims to reduce costs and increase productivity. An example is the automotive industry, where its implementation has been a success. Similarly, it has been implemented in other industries, such as the software industry. However, those companies seeking to adopt the new paradigm do not have at their disposal the information of other companies that have succeeded in adopting Product Lines.

However, the task of consulting relevant information on documented experiences in Product Line Engineering from other companies is not easy, since there is no tool that facilitates this process. This project presents the design, development and implementation of a web prototype called Product Line Engineering Industrial Application Cases Search Engine. This tool makes use of an ontology developed in the knowledge domain of documented experiences in Product Line Engineering. The objective is to mitigate the gap between the information and the interested entity.

Keywords: Product Lines, Search Engine, Data Visualization, Product Line Engineering, Industry Experiences, Ontologies, Software Development, Web Development.

Índice general

Índice de figuras

Índice de cuadros

Introducción

La Ingeniería de Líneas de Productos consiste en el diseño eficiente de un conjunto de productos relacionados y surge a partir de las Líneas de Productos. Se conoce como una línea de productos a un conjunto de productos que se direccionan a un mismo segmento de mercado, comparten elementos comunes aunque también tienen características que los distinguen entre sí, permitiendo cierto grado de variabilidad y reuso. Es por esto, que dicho paradigma ofrece múltiples beneficios para una empresa. Estas ventajas están relacionadas a la productividad, disminución de costos e incremento de calidad de los productos.

Pensar en la adopción de una nueva herramienta dentro de una compañía requiere un proceso de investigación, que puede nutrirse en gran medida de la literatura resultante de las experiencias de otras empresas que han adoptado o han realizado un proceso de acercamiento a dicho paradigma. Por tal razón, los documentos sobre las experiencias industriales en Ingeniería de Líneas de Productos son un referente importante para aquellas empresas interesadas en integrar el paradigma de Líneas de Productos.

Aunque la información actualmente es abundante y está disponible, su utilidad práctica es limitada. Además, dicha información se encuentra distribuida en fuentes diversas de información y tiene diferentes niveles de abstracción. Por ejemplo, sólo después de un estudio riguroso de la literatura los interesados en el tema podrían saber por ejemplo: ¿cuáles son las ventajas o desventajas de las herramientas utilizadas en el proceso de migración?, ¿cuáles productos han sido migrados a Líneas de Productos?, ¿qué resultados se han obtenido en el proceso de migración? o según las experiencias industriales ¿cuáles empresas pueden ser un punto de referencia para orientar su búsqueda?

Este proyecto acerca la Ingeniería de Lineas de Productos a las empresas o personas interesadas por medio de un prototipo de Buscador que usa una ontología con el fin de organizar la información y facilitar el proceso de búsqueda sobre experiencias industriales en Ingeniería de Líneas de Productos. Esta herramienta brinda resultados de texto y usa gráficos como forma de visualización que les permita tener datos adicionales relacionados a la búsqueda, y así, apoyar el proceso informativo.

Este documento contiene la información del diseño, implementación y pruebas de la ontología y del prototipo del buscador.

Capítulo 1

Descripción del Problema

1.1. Planteamiento del Problema

La Ingeniería de Líneas de Productos (PLE), es un paradigma de desarrollo para elaborar líneas de productos. Este paradigma ofrece beneficios como la reutilización, la disminución de errores, tiempos y costos (Pohl et al., 2005). De hecho, empresas como Nokia, Philiphs, Hewlett Packard, entre muchas otras, han adoptado exitosamente el paradigma de Ingeniería de Líneas de Productos, obteniendo beneficios como la disminución de costos y tiempos de desarrollo y el incremento de la calidad de sus productos (Linden van der et al., 2007).

Cuando las empresas están en proceso de decidir si adoptar o no un paradigma, el análisis de ejemplos de aplicaciones reales y experiencias similares les sirve como fuente de información para apoyar su decisión, pues las empresas tienden a incorporar innovaciones ya probadas por otras entidades (Rogers, 1983). Si bien, en los primeros años de la Ingeniería de Líneas de Productos una limitación era la escasez de trabajos industriales que reportaran casos de éxito en su adopción en las empresas, con el pasar de los años cada vez son más las experiencias industriales que reportan su proceso de adopción de la Ingeniería de Líneas de Productos (Birk, 2002; Linden van der et al., 2007; Breivold et al., 2008; Krsek et al., 2008; Knauber et al., 2000; Kircher et al., 2006; Quilty, Cinnéide, 2011; Hamza et al., 2010; Falvo et al., 2014; Fajar et al., 2010; Dordowsky, Hipp, 2009; Dillon et al., 2014; Ardis, Green, 1998). Una motivación recurrente que justifica la publicación de estos trabajos es que proveen información que puede servir de apoyo para otros interesados en adoptar la Ingeniería de Líneas de Productos.

Sin embargo, aunque la información está disponible su utilidad práctica es escasa. La información es abundante, heterogénea, se encuentra distribuida en fuentes diversas de información y tiene diferentes niveles de abstracción. Así, sólo después de un estudio riguroso de la literatura los interesados en el tema podrían ampliar su conocimiento en adopción de Líneas de Productos.

Con el ánimo de capitalizar el conocimiento publicado sobre experiencias industriales a fin de que sirvan de apoyo e inspiración para aquellos interesados en adoptar la Ingeniería de Líneas de Productos, esta investigación se interesa en proponer un prototipo de buscador de casos de aplicación de Ingeniería de Líneas de Productos que agrupe, sintetice y la presente de forma atractiva para el usuario la información que reportan las experiencias industriales. De esta manera, interesados podrán capitalizar las experiencias de otros que han adoptado la Ingeniería de Líneas de Productos.

1.1.1. Formulación

¿Cómo identificar, extraer, ordenar, recuperar y sintetizar información publicada sobre casos industriales de aplicación de la ingeniería de líneas de productos?

1.1.2. Sistematización

- ¿Cómo analizar, organizar y sintetizar la información disponible de los casos industriales?
- ¿Qué estrategia se debe utilizar para modelar la información recolectada sobre los casos industriales?
- ¿Qué técnicas usar para recuperar la información ?
- ¿Cuáles técnicas de visualización de la información aplicar para la presentación de los datos?.

1.2. Objetivos

1.2.1. Objetivo General

Implementar un buscador sobre casos de aplicación de Ingeniería de Líneas de Productos, que sintetice y presente la información de forma atractiva para el usuario.

1.2.2. Objetivos Específicos

1. Recolectar, analizar y sintetizar información sobre casos industriales de aplicación de Ingeniería de líneas de productos.
2. Diseñar una ontología que organice la información reportada en los casos industriales y sirva como fuente de consulta
3. Desarrollar un prototipo de buscador con características semánticas para recuperar información sobre los casos industriales
4. Aplicar técnicas de visualización de la información para implementar la solución y presentar los resultados de manera atractiva para el usuario.

1.3. Justificación

Debido a la creciente literatura en Ingeniería de Líneas de Productos, sus diversas fuentes, niveles de abstracción y/o heterogeneidad, hace que este proyecto se enfoque en proponer una herramienta que permita capitalizar el conocimiento, más específicamente de las experiencias industriales. Esta prototipo podría servir para facilitar el proceso de familiarización con la ingeniería de línea de productos permitiendo a las empresas interesadas obtener rápidamente información relacionada con experiencias ya probadas por otras organizaciones en el proceso de adopción de este paradigma.

1.4. Delimitaciones y Alcances

- **Información**: Las experiencias industriales que se incluyen en el buscador corresponden a un subconjunto de aquellas publicadas en la conferencia de Ingeniería de Líneas de Productos SPLC hasta el año 2020.

- **Buscador**: A partir de una revisión preliminar y exploratoria de un subconjuntos de casos industriales el buscador cuenta con las siguientes funcionalidades:

  - Búsquedas exploratorias: Esta consulta está relacionada al retorno de individuos de una clase seleccionada. Por ejemplo, ventajas, desventajas o retos sobre aplicación en general de Ingeniería de Líneas de Productos.
  - Búsquedas sencillas : Esta consulta está relacionada al retorno de individuos relacionados entre dos clases. Por ejemplo, las ventajas reportadas por una compañía en el proceso de adopción.
  - El buscador se ofrecerá como una plataforma web

  En cuanto a los requisitos no funcionales se tendrán en cuenta los siguientes elementos:

  - Exploración de la web semántica: En este proyecto se explorará la web semántica para dar respuesta a las consultas de los usuarios.

- **Visualización de información**: El buscador muestra en pantalla las gráficas correspondientes a las información relevante de las experiencia industriales reportadas.

- **Entregables**: En la etapa final del desarrollo de este proyecto se presenta lo siguiente:

  - Ontología que organice la información recopilada a partir de los casos industriales. Esta será la fuente de información para dar respuesta a las búsquedas llevadas a cabo en el buscador
  - El prototipo funcional del buscador de casos industriales de Ingeniería Líneas de productos.

- Trabajo de grado con la información pertinente del desarrollo del prototipo

Capítulo 2

Marco Conceptual

2.1. Marco de Referencia

2.1.1. Áreas Temáticas

En este proyecto se reconocen las siguientes definiciones propias de la disciplina como punto de partida de la contextualización.

- Software and its engineering - Software creation and management - Software development techniques - Reusability - Software product lines
- Information systems - World Wide Web - Web data description languages - Semantic web description languages - Web Ontology Language (OWL)
- Human-centered - computing Visualization - Visualization application domains - Information visualization

2.1.2. Marco Teórico

2.1.2.1. Ingeniería de Líneas de Productos

La Ingeniería de Líneas de Productos, o por sus siglas en inglés PLE, es una disciplina perteneciente a la Ingeniería de Sistemas y está enfocada en diseñar un portafolio de productos relacionados de manera eficiente. PLE tiene el objetivo de aprovechar al máximo y de forma continua las similitudes de los productos, respetando y gestionando sus diferencias (Pohl et al., 2005).
Adoptar la Ingeniería de Líneas de Productos exitosamente trae beneficios como los tiempos de entrega del producto(*Time to market*), costos en ingeniería, tamaño en el portafolio de productos, reducción en la tasa de defectos y la calidad en los productos (Clements, Northrop, 2001).

2.1.2.2. Líneas de Productos

Las Líneas de Productos o PL por sus siglas en inglés, están referidas a un conjunto de productos que poseen características comunes que al ser gestionadas cumplen con una necesidad de un segmento de mercado específico, permitiendo cierto grado controlado de variabilidad y reuso (Cal, O'Neill, 2014).
Por ejemplo, el modelo básico de una Línea de Productos en el área de software (Figura 2.1) consta de 4 partes :

- La entrada (activos de software): Es una colección de partes de software, como, requisitos, diseño, componentes, casos de prueba, entre otros, que se configuran y componen de una manera ya establecida para producir los productos de la línea.
- El control (modelos de decisión y decisiones de productos): Descripción de los aspectos variables y opcionales de los productos de dicha línea, donde cada una de estas es definida por un conjunto de decisiones.
- El proceso de producción: Se establecen los pasos para componer y configurar productos a partir de la entrada utilizando el conjuntos de decisiones para determinar cuales activos de entrada usar y también configurar la variación de los mismos.
- La salida (productos de software): Conjunto de todos los productos que se pueden producir por la línea de productos.

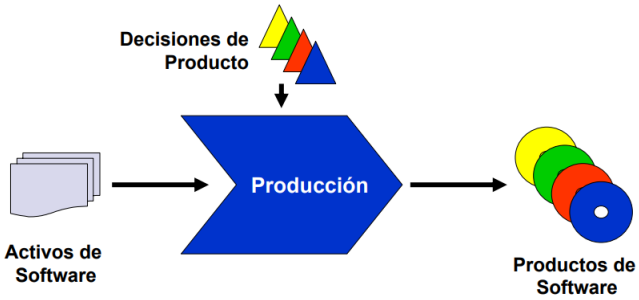


Figura 2.1: Modelo básico de una línea de Productos de Software. Tomado de (Krueger, 2006) traducido por (Montilva, 2006).

Este proceso trae consigo beneficios en aspectos como la entrega de productos de software de manera más rápida, económica y con una mejor calidad.

2.1.2.3. Buscadores web

Los buscadores web son aquellas herramientas encargadas de ayudar a un usuario a encontrar cierto tipo de información. Estas búsquedas dependen de los parámetros de entrada y buscan sobre miles de páginas que están disponibles en internet. Esta herramientas responde a miles de consultas por día sometido a un gran uso (Raval, Kumar, 2012). Algunos ejemplos de buscadores web son Google, Yahoo, Bing, entre otros.

Usualmente los buscadores trabajan bajo tres componentes: Recorrer todo el contenido y el código que compone una página web con el fin de analizarlo incluyendo saltos correspondientes a

través de una web (Crawling). Posteriormente, guardan un índice de las palabras que se encuentran y donde las encuentran (Indexing). Por último, permiten a los usarios buscar las palabras o combinaciones de palabras que se encuentren en ese índice (Searching) (Raval, Kumar, 2012).

De esta manera, es posible intuir los principales componentes dentro de los buscadores. Dichas partes son la interfaz de usuario, un robot o spider (para la técnica de Crawling e Indexamiento), un algoritmo de búsqueda y por último una base de datos. La Figura 2.2 presenta la arquitectura de referencia de un motor de búsqueda.

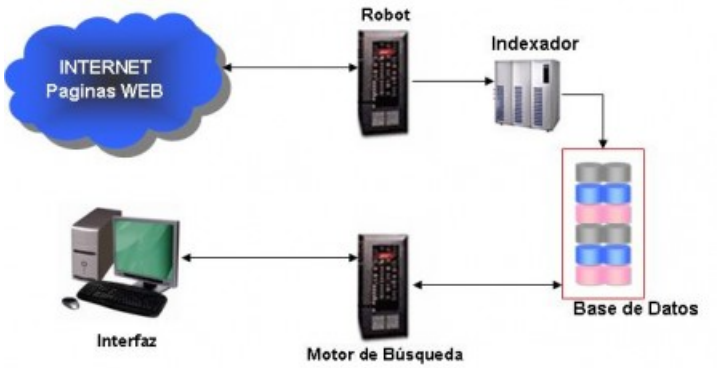


Figura 2.2: Arquitectura de un motor de búsqueda. Tomado de (Raval, Kumar, 2012)

2.1.2.4. Web semántica

La web semántica tiene el objetivo de introducir descripciones explícitas sobre el significado de los recursos web y permitir que los agentes usados en los buscadores tengan un mayor nivel de comprensión de la web (Castells, 2003). La web semántica está compuesta por seis capas como se muestra en la Figura 2.3. Las tres capas inferiores con el sentido de crear bases para representar la información de forma semánticamente accesible y las siguientes con el fin de que los agentes puedan comprender dicha información (Peis et al., 2007).

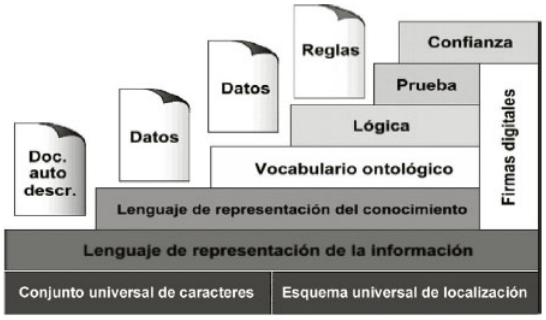


Figura 2.3: Niveles de la web semántica. Tomado de (Peis et al., 2007)

2.1.2.5. Ontologías

El termino ontología tiene su origen en la filosofía y se trata del estudio del ser. Sin embargo desde el punto de vista de ciencias de la computación, una ontología se define como un conjunto de representaciones con las cuales se puede modelar un dominio de conocimiento. Dichas representaciones contienen clases o categorías, atributos o propiedades y relaciones. (Lim et al., 2013).

Las ontologías son uno de los componentes básicos de la web semántica y para investigadores en el campo de Inteligencia Artificial, la ontología es un documento (en lenguaje OWL) que posee relaciones entre términos y conceptos de manera formal. El tipo mas común de ontología posee una taxonomía y un conjunto de reglas de inferencia (Lim et al., 2013). La Figura 2.4 muestra un ejemplo gráfico y sencillo de una ontología del dominio de la Lingüística.

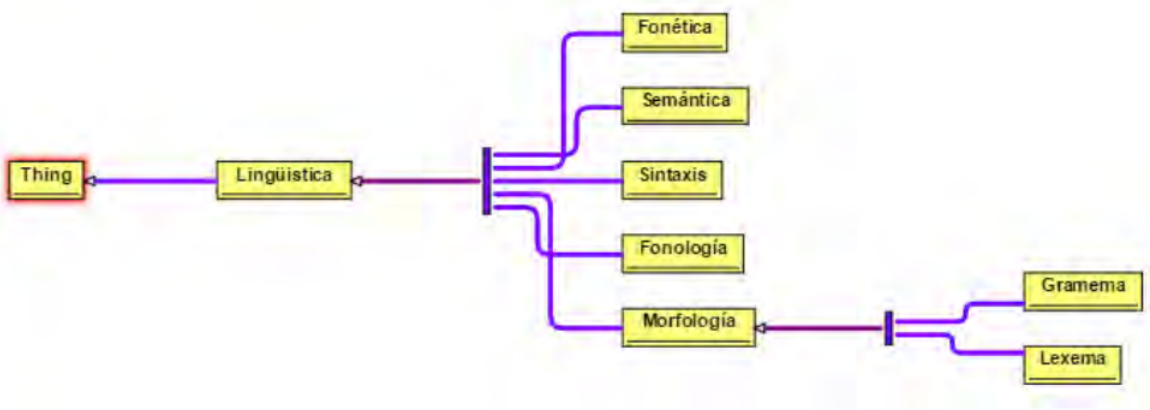


Figura 2.4: Ejemplo de Ontología del dominio de la lingüística. Tomado de (Malpartida Valverde, 2017)

2.1.3. Portafolio de Tecnologías

En el desarrollo de este proyecto se identificaron tres áreas de interés y las tecnologías asociadas a cada área que apoyaron el proceso de desarrollo. Dichas áreas son: Recuperación de Información, Implementación de la herramienta y Creación—Consulta de la ontología. A continuación se describe cada una:

2.1.3.1. Recuperación de información

- **Framework para la Descripción de Recursos - RDF:** *Resource Description Framework (RDF)* es un lenguaje para especificar metadatos que permite la operabilidad entre aplicaciones que intercambian información. RDF se basa en la identificación de recursos web usando Uniform Resource Identifiers (URI's) para describirlos en términos de valores y atributos simples. Una descripción RDF es un conjunto de cláusulas simples, declaraciones, enunciados o tripletas. Las tripleta constan de: sujeto, predicado y objeto. Estas tripletas se pueden representar mediante grafos dirigidos, donde los predicados son arcos y los sujetos y objetos son nodos (McBride, 2004). La Figura 2.5 muestra un ejemplo de una sentencia RDF.

- **Lenguaje de Ontologías Web - OWL:** *Ontology Web Language (OWL)* es un Lenguaje de marcado para la exposición de ontologías en la World Wide Web (WWW), con el objetivo de hacer más sencillo un modelo de marcado que se construye sobre RDF y está codificado en XML. OWL es una extensión de RDF que permite tener más vocabulario y una sintáxis más fuerte (Antoniou, Van Harmelen, 2004). Por esta razón, OWL se puede utilizar para articular el significado de términos léxicos y definir las relaciones entre ellos.

- **SPARQL:** Es un acrónimo recursivo del inglés SPARQL Protocol and RDF Query Language. Se trata de un lenguaje estandarizado para la consulta de grafos RDF, normalizado por el RDF Data Access Working Group (DAWG) del World Wide Web Consortium (W3C). Es una tecnología clave en el desarrollo de la web semántica que se constituyó como recomendación oficial del W3C el 15 de enero de 2008 [1].

- **Jena:** Apache Jena es un framework de trabajo de código abierto de la Web Semántica para Java. Proporciona una API para extraer datos y escribir en grafos RDF. Los grafos se representan como un "modelo" abstracto. Un modelo puede ser obtenido con datos de archivos, bases de datos, URLs o una combinación de estos. Un modelo también puede ser consultado a través de SPARQL 1.1. Jena es similar a RDF4J (anteriormente OpenRDF Sesame); aunque, a diferencia de RDF4J, Jena proporciona soporte para OWL (Web Ontology Language). El marco de trabajo tiene varios razonadores internos y el razonador Pellet (un razonador de código abierto Java OWL-DL) puede ser configurado para trabajar en Jena. Jena soporta la serialización de gráficos RDF a bases de datos relacionales, RDF/XML, Turtle y Notation 3 [2].

Figura 2.5: Ejemplo Sentencia RDF. Tomado de (Raval, Kumar, 2012)

2.1.3.2. Implementación

- **React.js — Front-end:** React es una librería desarrollada inicialmente por Facebook. Es software libre y a partir de su liberación acapara una creciente comunidad de desarrolladores y entusiastas. Su creación se realizó en base a unas necesidades concretas, derivadas del desarrollo de la web de la popular red social. Además de facilitar el desarrollo ágil de componentes de interfaces de usuario, el requisito principal con el que nació React era ofrecer un elevado rendimiento, mayor que otras alternativas existentes en el mercado [3].

- **Material UI:** Material UI es una herramienta de diseño para aplicaciones que ayuda a desarrolladores a través de la disposición de recursos de diseño, plantillas y componentes.

[1] https://www.w3.org/TR/rdf-sparql-query/Overview.html
[2] https://jena.apache.org/
[3] https://es.reactjs.org/

Además, se encuentra disponible para todos los entornos de trabajo basados en javascript enfocados en la elaboración de interfaces de usuario [4].

- **Boostrap:** Boostrap es una de las herramientas de diseño y personalización de sitios web open source más populares para el desarrollo front-end. Esta posee componentes pre-construidos, responsivos y plantillas para JavaScript que disminuyen los tiempos de desarrollo y son compatibles con dispositivos móviles [5].

- **Chart.js:** Chart.js es una librería open source de JavaScript que proporciona una serie de gráficos para las visualización de información. Cada gráfico se puede animar, personalizar y es responsivo guardando estándares de visualización. Además posee un alto rendimiento en todos los navegadores [6].

- **Java — Back-end:** Java es un lenguaje de programación y una plataforma informática que fue comercializada por primera vez en 1995 por Sun Microsystems. Java es rápido, seguro y fiable. A partir de 2012, Java es uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos diez millones de usuarios reportados [7].

- **Rest:** Representational State Transfer traducido como Transferencia de Estado Representacional, es un conjunto de reglas de arquitectura web creada por Roy Fielding para la creación de interfaces de programación de aplicaciones (API), que conecta varios sistemas basados en protocolos HTTP. Dicha comunicación sirve para obtener y generar datos en formatos específicos como XML y JSON [8].

2.1.3.3. Creación de ontología

- **PROTÉGÉ:** Es un editor de código abierto usado para construir Ontologías y un marco general para representar el conocimiento. Está escrito en Java que es un lenguaje de programación orientado a objetos. Se usa para construir aplicaciones para la Web Semántica, para hacer una descripción semántica de la información. Sus archivos se hacen en el lenguaje OWL [9].

- **MAXQDA:** Es un software utilizado para el análisis cualitativo de datos. La primera versión se creó en 1989 y el software fue diseñado para investigaciones cualitativas, cuantitativas y para métodos mixtos (Silver, Lewins, 2014). En la actualidad, MAXQDA es uno de los programas para análisis cualitativo de información más utilizados gracias a su sistema de codificación y herramientas para el cruce y análisis de datos (Gil, Arana, 2010).

[4] https://material-ui.com/
[5] https://getbootstrap.com/
[6] https://www.chartjs.org/
[7] https://www.java.com/es/about/
[8] https://www.redhat.com/es/topics/api/what-is-a-rest-api
[9] https://protege.stanford.edu/about.php

2.1.4. Trabajos Relacionados

Son diversos los autores quienes han aplicado el uso de ontologías para la recuperación de información, entre ellos se encuentran :

- Ordoñez, Cobos (2010) proponen una herramienta de búsqueda que por medio del uso de una ontología de dominio general encontrada en la base de datos léxica WordNet, refina las consultas para lograr que los resultados entregados por otros motores de búsqueda (Google, Yahoo, Bing) sean más cercanos a los que realmente están buscando los usuarios. Además con el uso del perfil de usuario se categorizan aquellas resultados que el cliente considera relevantes por el contenido, marcando dichas páginas para que en futuras búsquedas estén entre los primeros resultados.

- Chauhan et al. (2013) diseñan una base para un sistema basado en una ontología. La ontología está definida en el dominio del deporte. Este sistema realiza las consultas de forma directa o expandida construyendo sentencias SPARQL para la recuperación de las tripletas RDF en la base de conocimiento. Por último, el sistema clasifica por relevancia y muestra los elementos recuperados. Además presenta en detallado esquema de prueba para verificar la efectividad de los resultados entregados por consulta realizada.

Más relacionados con la Ingeniería de Líneas de Productos, se encuentran trabajos orientados a la definición de modelos de variabilidad y características, basados en ontologías. Por ejemplo, Ferreira et al. (2009) proponen un marco conceptual para una casa de desarrollo de software, que tiene una variabilidad muy amplia y por medio del desarrollo de software basado en modelos, crean un capa ontológica que valida artefactos de acuerdo al dominio y así poder promover la reutilización de componentes en diversos dominios. Asimismo, Mohan, Ramesh (2003) desarrollan una ontología para gestión de la variabilidad que tiene como objetivo caracterizar elementos del dominio necesarios y así gestionar la variación en familias de productos.

Por otro lado, algunos están enfocados en la forma de los documentos. Vignando et al. (2020) proponen una herramienta que por medio del uso de una ontología formaliza la información sobre experimentos en Ingeniería de Líneas de Productos de Software. El proceso se hace sobre documentos de revistas como la *Association for Computing Machinery* (ACM), *Institute of Electrical and Electronics Engineers*(IEEE), *Scopus* y *Springer*, donde se extraen características como el tipo de plantilla del reporte, el diseño experimental, los elementos experimentales reportados, entre otros. Esta herramienta tiene como objetivo brindar información de por ejemplo tipos de plantillas utilizadas en Ingeniería de Líneas de Productos

Ahora, en el marco del uso de las experiencias industriales, Foguem et al. (2008) proponen una estructura para formalizar y capitalizar el proceso de retroalimentación de una experiencia, la cual tiene el objetivo de transformar la información obtenida de la experiencia ‘para la clarificación del

conocimiento explícito y del saber hacer, cuyo objetivo es obtener descripciones de las lecciones aprendidas que sean significativas, correctas y aplicables'. Desde otro punto de vista, Rabiser et al. (2018) muestra que las experiencias industriales en líneas de productos de software van en consonancia con la investigación académica en el área, lo que se traduce en información clave de la aplicación particular del paradigma en la industria.

Si bien, en la literatura se ha propuesto el uso de ontologías con diversos fines, hasta donde se conoce, a la fecha, no se ha desarrollado ontologías para la recuperación de información de la aplicación de casos industriales de Ingeniería de Líneas de Productos. Los trabajos relacionados han estado orientados en apoyar la definición de modelos centrados en algún campo de la Ingeniería de Líneas de productos y forma de documentos, pero no desde el punto de vista de la capitalización del conocimiento a raíz de una experiencia industrial.

Capítulo 3

Diseño e Implementación

En este capítulo se especifica el análisis, diseño y desarrollo del buscador.

La sección de análisis y diseño contiene el levantamiento de requerimientos (Sección 3.1.1) y la definición de la arquitectura (Sección 3.1.2).

La Siguiente sección, presenta la implementación en detalle cada módulo desarrollado. Estos módulos son: Base del conocimiento (Sección 3.2.1), Módulo de búsqueda (Sección 3.2.2), Módulo de consulta (Sección 3.2.3) y Módulo de visualización (Sección 3.2.4).

3.1. Análisis y Diseño

3.1.1. Requerimientos

Siguiendo el análisis y la documentación sobre el desarrollo de software y con base al planteamiento del problema, donde el principal *stakeholder* es la directora del proyecto Luisa Fernanda Rincón, se identifican las siguientes funcionalidades principales:

- Consultas Exploratorias: Esta consulta está relacionada al retorno de individuos de una clase seleccionada. Por ejemplo, ventajas, desventajas o retos sobre aplicación en general de Ingeniería de Líneas de Productos.
- Consultas Sencillas: Esta consulta está relacionada al retorno de individuos relacionados entre dos clases. Por ejemplo, las ventajas reportadas por una compañía en el proceso de adopción.
- Presentación de gráficos: El buscador debe presentar gráficos que resuman información relevante de las experiencias industriales.

Para dar cumplimiento a dichas funcionalidades se crean nueve requisitos funcionales y tres requisitos no funcionales. Para cada requisito funcional se especificaron fichas detalladas como la que muestra el Cuadro 3.1. La totalidad de las fichas de especificación se encuentran disponibles en la sección de anexos (Sección A.1). El Cuadro 3.2 expone los requisitos no funcionales.

Especificación de Requisito	
Código Requisito	RF-001
Nombre	Listar Clases
Prioridad	Alta
Entrada	Modelo OWL
Salida	Lista de clases
Fuente Datos	Ontología
Destino Datos	Front-end
Restricciones	La salida debe estar en formato JSON
Descripción	Desarrollar un servicio REST tipo GET para consultar las clases disponibles en la ontología y retornarlas en formato JSON.
Proceso	Cargar modelo OWL al back-end y realizar la búsqueda de las clases sobre ese modelo.
Efecto Colateral	–

Cuadro 3.1: Especificación de Requisitos Funcionales. Elaboración propia.

Código	Nombre	Descripción
RNF-001	Web semántica	El buscador debe explorar la incorporación de recursos de la web semántica desarrollados en el dominio de las experiencias industriales.
RNF-002	Congruencia	El buscador debe tener gráficos acordes a los tipos de datos recuperados.
RNF-003	Tiempo de respuestas	El buscador debe tener tiempos de respuesta no superiores a los 5 segundos.

Cuadro 3.2: Requisitos No Funcionales. Elaboración propia.

3.1.2. Arquitectura

El prototipo de buscador propuesto está compuesto por cuatro partes según sus responsabilidades. Estas partes son: el tratamiento de la consulta, el proceso de búsqueda, la base del conocimiento y para terminar, la visualización de los resultados. Esta sección presenta tres diagramas del modelo de vistas de arquitectura 4+1 que articulan los módulos de la solución propuesta. La Figura 3.1 presenta el diagrama de módulos del sistema. La Figura 3.2 muestra el diagrama de despliegue de la aplicación. Finalmente, la Figura 3.3 muestra un diagrama de secuencia de una de las principales funcionalidades del buscador.

Vista de despliegue

Es de notar que la Figura 3.1 se construye teniendo como fuente principal el flujo de la consulta ingresada por el usuario. Se contempla que a través del flujo se transforma la consulta, se realiza la búsqueda haciendo uso de la base del conocimiento y por último se entrega un pre resultado que se transforma una última vez para entregarse al usuario.

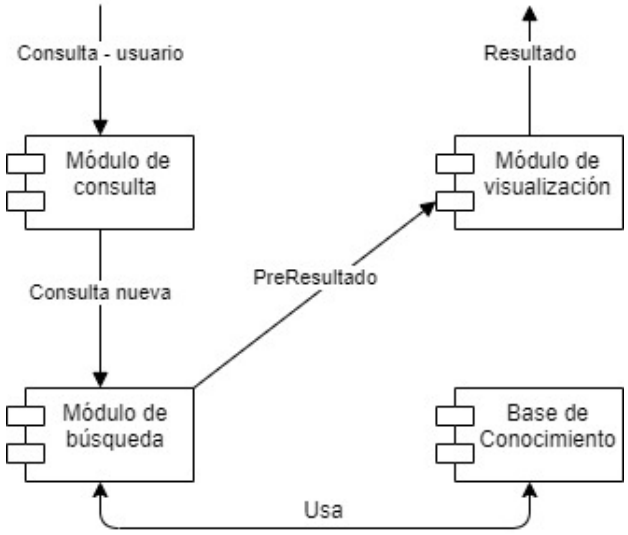


Figura 3.1: Relación de módulos Propuestos. Elaboración propia.

Vista física

Para la vista física, la Figura 3.2 muestra el diagrama de despliegue de la aplicación. El modelo es de tipo cliente servidor y consta de:

- Servidor Web: Un servidor el cual contiene los recursos necesarios para ejecutar la interfaz gráfica del buscador. La tecnología que soporta debe ser javascript que a su vez utiliza HTML/CSS para el diseño de los componentes.
- Servidor de Aplicación: Un servidor de aplicación el cual contiene los recursos necesarios para la ejecución de la lógica del negocio. Este contiene la especificación de las servicios Restful, las carpetas de los recursos desarrollados para el tratamiento de la consulta y su respuesta y la dependencia a los módulos necesarios que permiten el cargue y la consulta a la base del conocimiento.
- Comunicación : La comunicación es de tipo cliente-servidor y se realiza usando los lineamientos de arquitectura REST por medio del intercambio de archivos en formato JSON.

Vista lógica

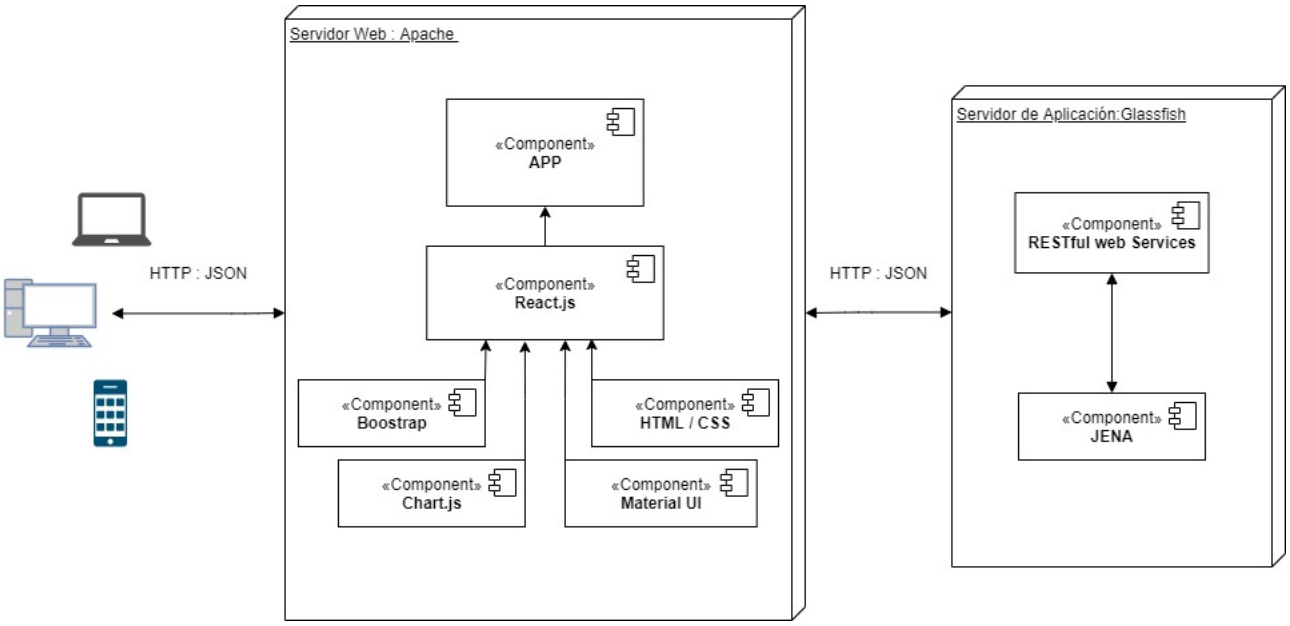


Figura 3.2: Diagrama de despliegue. Elaboración propia.

Con el fin de verificar el comportamiento de los distintos componentes implicados en la consulta de un usuario, la Figura 3.3 muestra un diagrama de secuencia. El diagrama de secuencia inicia cuando un usuario realiza una consulta y procede a dar clic en buscar, esta acción activa el componente de javascript que lanza la consulta del usuario y por medio del consumo de servicios REST transfiere la solicitud activando el componente de respuesta del lado del servidor. El componente receptor (identificado como configurador) funciona como camino principal para importar y solucionar las rutas de los recursos y así direccionar la consulta al endPoint correspondiente (Especificación RestFul). Una vez llega la consulta el endPoint llama al método que la transforma a una sintaxis que sea comprendida por el módulo búsqueda (Query) y la retorna. Una vez el método dispone de la consulta en el formato adecuado realiza el llamado al paquete (Ontology) donde se definen el método para el cargue del modelo, la ejecución de la consulta y el retorno de los resultados. En cada uno de los anteriores pasos se hace uso de las dependencias que proveen métodos de cargue y consulta sobre la base de conocimiento.

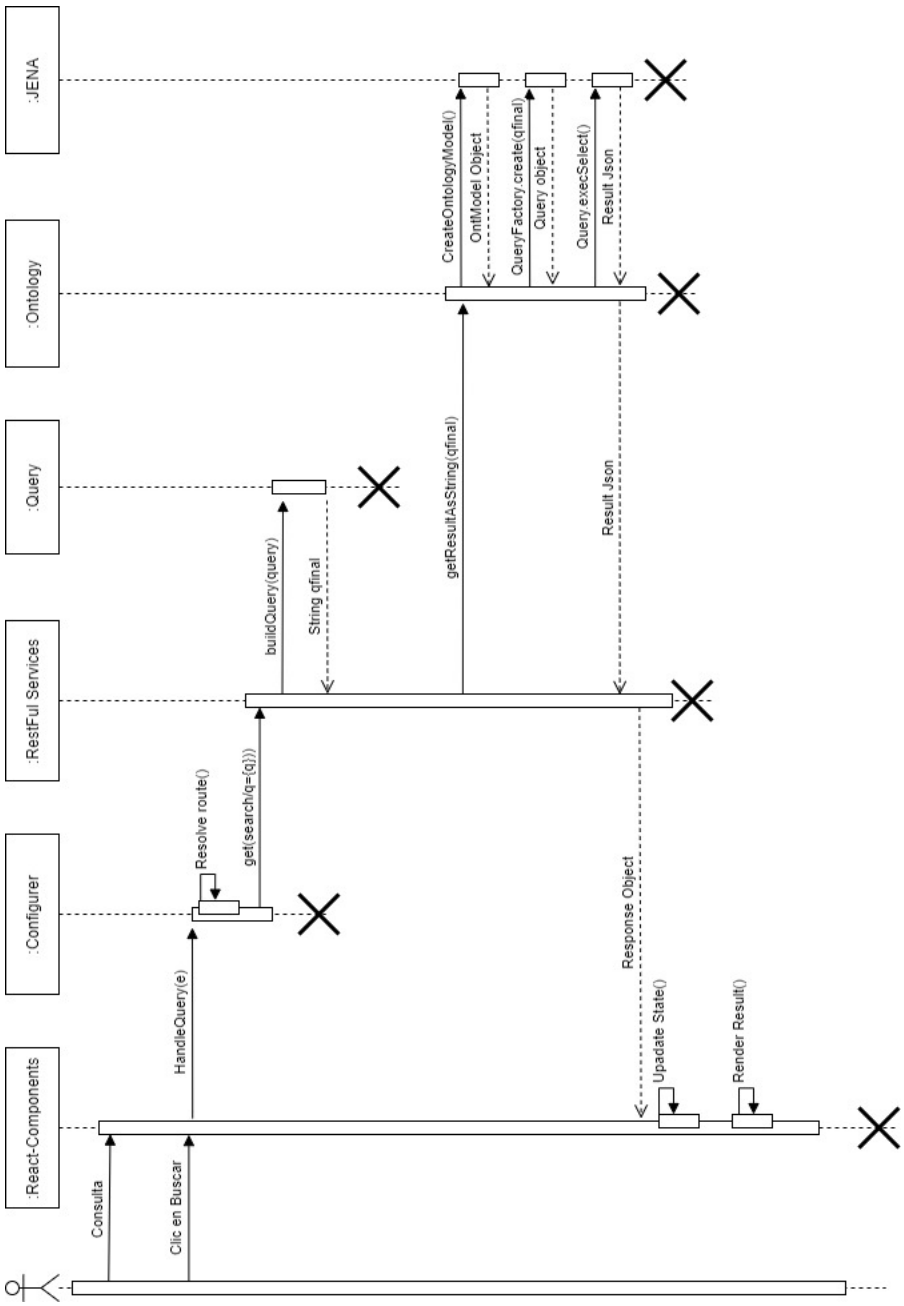


Figura 3.3: Diagrama de secuencia para una consulta. Elaboración propia.

3.2. Implementación

3.2.1. Base de Conocimiento

En este módulo se hace énfasis en el desarrollo del dominio del conocimiento para el buscador. Para la recuperación de la información la semántica o relación de los contenidos en las búsquedas es importante. Los buscadores tradicionales regularmente almacenan índices para hacer búsquedas. La complejidad del lenguaje natural hace que estas búsquedas basadas en contenido difícilmente puedan garantizar buenas medidas en *precision*[1] y *recall*[2] (Fang et al., 2005). Lo anterior se ha intentado equilibrar por medio del uso de otras técnicas como la expansión de consultas o LSA *(Latent Semantic Analysis)* (Chauhan et al., 2013). Sin embargo, el problema no se ataca en la raiz. Es en este punto donde definir el dominio del conocimiento aparece como una clave para dar soporte a los buscadores con características semánticas (Fang et al., 2005).

Según Fang et al. (2005) existen tres pasos fundamentales que se deben tener en cuenta para la elaboración del dominio de conocimiento para un buscador, estas son:

1. **El modelo de representación del conocimiento:** La representación del conocimiento está referida a la forma como se organiza y se recopila la información ya que existen diversas maneras para realizarlo. Algunas de ellas son redes semánticas, representación basada en la lógica, representación de procedimiento, entre otras. Este paso es importante ya que la manera como se ordena tiene consecuencias en los procedimientos u operaciones que podrán realizarse para poder manipular dicha información.

2. **El lenguaje para definir el modelo:** Como lo especifica el Diccionario de la Real Academia Española RAE (Española, Madrid, 1970), el lenguaje en la informática es el 'Conjunto de signos y reglas que permite la comunicación'. Hay distintos lenguajes según los modelos de representación, algunos con distintos sublenguajes que acotan la expresividad de los mismos según las necesidades o requerimientos. Es aquí donde se elige el conjuntos de signos que permite expresar el modelo de representación del conocimiento.

3. **El enfoque del modelo:** Para terminar, es importante conocer cómo el modelo puede apoyar el objetivo, en este caso, poder implementarlo dentro de un buscador y así lograr de forma especifica sacar provecho de las estructuras ahí definidas.

Para este caso, y como se plantea inicialmente en el desarrollo del proyecto, el modelo de representación del conocimiento se hace por medio de una ontología. Una ontología, especifícame de dominio, es un conjunto de conceptos que describen algún tipo de contexto.

El lenguaje para definir el modelo es OWL, de sus siglas en inglés *Ontology Web Language*, una herramienta diseñada para la exposición de ontologías en la *World Wide Web* (McGuinness et al.,

[1] En el campo de la recuperación de información, *precision* es la fracción de los documentos recuperados que son relevantes para la consulta

[2] En la recuperación de información, *recall* es la fracción de los documentos pertinentes que se recuperan con éxito.

2004).

Por último, el enfoque de este modelo es crear una estructura para las experiencias industriales de Ingeniería de Líneas de Productos que se han documentado y de acuerdo a ello compartir el conocimiento recuperado por la ontología con la comunidad interesada. La siguiente sección muestra el proceso que se lleva a cabo para la inmersión en el dominio del conocimiento y de esa forma realizar la ontología.

3.2.1.1. Proceso de Acercamiento al Dominio del Conocimiento

Para hacer el acercamiento al dominio del conocimiento, se lleva a cabo un proceso de inmersión. Según la literatura es necesario conocer y relacionarse con la información que se ofrece en las distintas experiencias documentadas (Braun, Clarke, 2006; Staples, Niazi, 2008). Por esto, siguiendo un proceso sistemático e incremental se realizó un acercamiento a la identificación de patrones en las distintas experiencias. En esta labor fue de utilidad el uso de la herramienta MAXQDA, la cual sirve para el análisis cualitativo y cuantitativo sobre información no estructurada. [3]. La Figura 3.4 muestra el proceso propuesto de acercamiento al dominio del conocimiento.

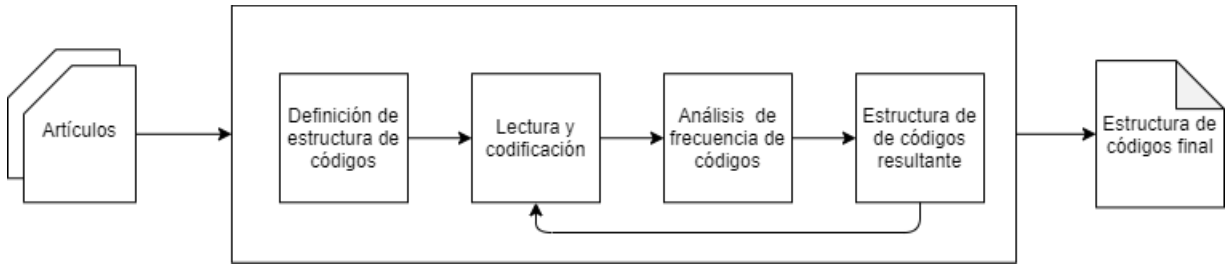


Figura 3.4: Proceso de acercamiento al dominio del conocimiento. Elaboración propia.

La primera fase fue la elección de los documentos. Estos son un subconjunto de las experiencias industriales publicadas en la Conferencia de Ingeniería de Líneas de Productos SPLC hasta el año 2020. Con un total de cuarenta (43) documentos, se tomaron un porcentaje aproximadamente del 12.5 % por cada iteración. La iteración inicial tiene como objetivo obtener la primera estructura de códigos [4] mediante la lectura linea a linea de cada documento. Este proceso es conocido en la síntesis temática [5] como codificación inductiva, donde la asignación o creación de códigos se hace respecto a como van surgiendo conceptos o información relevante en la lectura (Cruzes, Dyba, 2011). En las iteraciones siguientes, el punto de partida es la estructura anterior (codificación deductiva), sin

[3] https://es.maxqda.com
[4] Un código es un término utilizado para nombrar aspectos fundamentales en un texto o imagen
[5] Conjunto de ideas o conceptos importantes y más relevantes acerca del tema o texto

embargo, la estructura previa puede estar sometida a cambios.

Los códigos obtenidos en cada ciclo son comparados con los inmediatamente previos, esto por medio de un análisis de frecuencias ilustrado en la Figura 3.5. El proceso propuesto anteriormente permite obtener la generalidad de la taxonomía respecto a los documentos estudiados. Cada comparación de estructuras genera una nueva estructura, con más o menos códigos, y si es el caso, subcódigos. Se realiza lo anterior hasta terminar el conjunto de documentos elegido y obtener una taxonomía más refinada. La nomenclatura del color de la Figura 3.5 se expone en la Figura 3.6.

Porcentaje acumulado de Documentos					Resultados
10	Códigos anterior	total	Códigos posterior	total	
25,00%	Thesis.mx20	105	Thesis1.mx20	186	
	Team size	2	TEAM SIZE	1	bajó
	scoping	5	SCOPING	1	bajó
	approaches	5	APPROACHES	0	bajó
	Domain Analysis Approach	2	Domain Analysis Aproach	2	igual
	Challenges	6	CHALLENGES	4	bajó
	categories of peo	1	CATEGORIES OF PEOPLE	14	subió

Figura 3.5: Comparación de frecuencias de códigos. Elaboración propia.

	Nomenclatura sistema de códigos MAXQDA -Análisis de contenido Buscador.
1	Códigos padre primera generación están en mayuscula y de color verde.
2	Códigos hijo tendrán la primera en letra en mayúscula y serán de color azúl
3	Códigos hijos que son padré tendrán la primera letra mayúscula y serán de color azul claro

Figura 3.6: Nomenclatura de códigos. Elaboración propia.

Con el fin de observar los resultados arrojados por la herramienta MAXQDA la Figura 3.7 presenta el listado de códigos padre versus la frecuencia de cada uno. Este gráfico permite la realización del análisis de aquellas temáticas que más se encuentran dentro de las experiencias documentadas de Ingeniería de Líneas de Productos. Con esa base, se puede orientar el enfoque de la construcción de la ontología, puesto que se puede evidenciar que para códigos como lo son herramientas, resultados y retos el contenido es mucho mayor y por ende se puede obtener más información de esas codificaciones. El Cuadro 3.3 describe cada una de la agrupaciones.

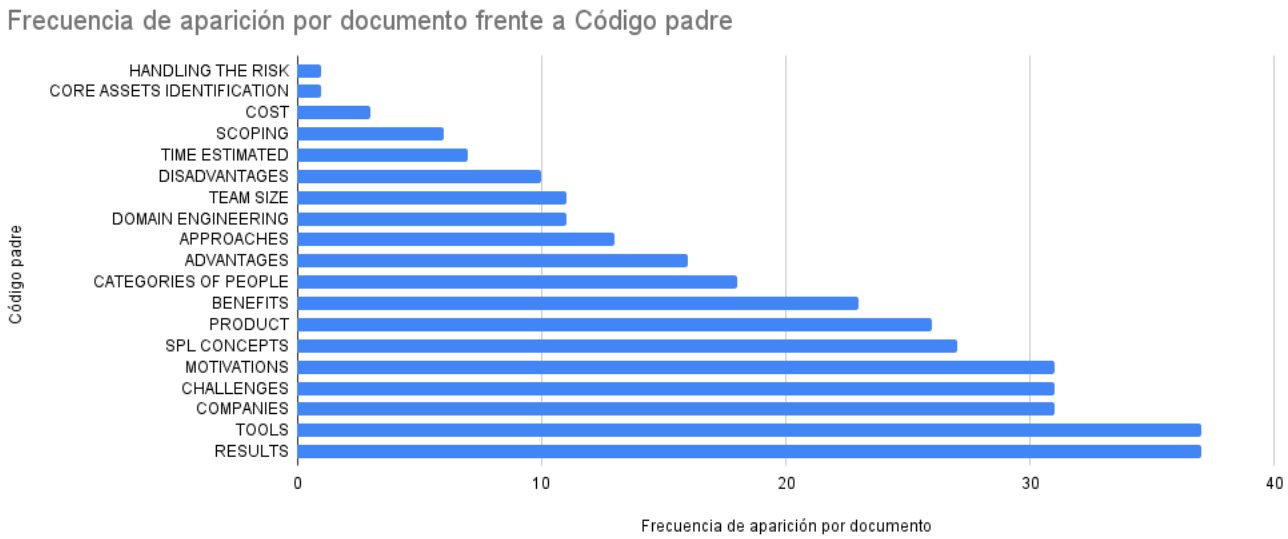


Figura 3.7: Código padre vs Frecuencias. Elaboración propia

Finalizado el proceso de acercamiento a la base de conocimiento y una vez definida la taxonomía de aquellos conceptos y partes que más se encontraban dentro de los artículos revisados, se procede a la elaboración de la ontología. Para este fin se utiliza Protégé, una herramienta para la creación y edición de ontologías. Para la implementación de la ontología se sigue la metodología de ingeniería del conocimiento expuesta por Noy, McGUINNESS (2005). La siguiente subsección presenta los pasos seguidos en el desarrollo de la ontología.

Descripción de temáticas (códigos)	
Temática	**Descripción**
Handling the risk	Agrupa el manejo del riesgo que reportaban las compañías en el procesos de adopción
Core assets identification	Agrupa la forma como la compañía identifica los núcleos en una Línea de Productos
Costo	Agrupa costos reportados por la compañías en el proceso de adopción
Scoping	Agrupa los alcances que plantean las compañías en los procesos de adopción
Time estimated	Agrupa los tiempos que las compañías estimaban en el paso a Líneas de Productos
Disadvantages	Agrupa las desventajas de utilizar Líneas de Productos
Team Size	Agrupa la cantidad de personas por área o proyecto que trabajaron en el proceso de migración o implementación de la Líneas de Productos.
Domain Engineering	Agrupa las empresas que reportaron el análisis de Dominio para el proceso de adopción
Approaches	Agrupa lo relacionado a los acercamientos que se han tenido para optar por la utilización de una Linea de Productos
Advantages	Agrupa ventajas de utilizar Líneas de Productos sobre otro paradigma
Catagories of people	Nombre de profesiones y grupos de personas implicadas en los procesos de adopción
Benefits	Beneficios de usar Líneas de Productos
Products	Agrupa los tipos de productos que manejan las empresas
Concepts	Definiciones sobre términos de Líneas de Productos dadas por cada estudio
Motivations	Agrupa lo que motiva a la empresa o compañía a migrar hacia una Línea de Productos.
Challenges	Agrupa los retos que reportaron las compañías al adoptar una Línea de productos
Companies	Nombres de las empresas donde se realizó el proceso de adopción
Tools	Agrupa pasos, estrategias, metodologías y herramientas de software utilizadas o exploradas en la adopción a Líneas de Productos
Results	Resultados obtenidos del proceso de adopción a Líneas de productos

Cuadro 3.3: Descripción de temáticas resultantes. Elaboración propia.

3.2.1.2. Implementación de la Ontología

Noy, McGUINNESS (2005) crearon una guía con 7 pasos para la elaboración de una ontología, la cual ha sido adoptada y aceptada en la literatura siendo muy usada. El primer paso es la

determinación del dominio y el alcance de la ontología. Seguidamente se realiza la consideración de usar ontologías previamente creadas, si pueden aplicar, y listar los conceptos relevantes para el dominio. También, se definen clases y jerarquías, propiedades, relaciones, facetas y slots. Para finalizar se crean los individuos y se pobla el modelo. Las siguientes subsecciones describen a detalle el anterior proceso.

1. **Determinar el dominio y alcance de la ontología**

   El dominio de la ontología es la representación de las experiencias industriales de Ingeniería de Líneas de Productos. Esta ontología se planeó para apoyar el proceso de búsqueda para que aquellas nuevas empresas que desean migrar a Líneas de Productos. Los conceptos que describen las experiencias industriales están ligados a factores como lo son estrategias, metodologías, conceptos y resultados que a su vez están asociados a la empresa y tipos de productos descritos en la experiencia industrial. Son entonces las herramientas y resultados de la aplicación o acercamiento de Ingeniería de Líneas de Productos en una empresa lo que figurará en esta ontología. Ya que la ontología va a ser utilizada como un punto de referencia para otras empresas es necesario incluir, pasos, estrategias, metodologías, ventajas, desventajas, recursos entre otros que permitan ampliar la información.

2. **Considerar la reutilización de ontologías**

   Es importante considerar si ya existen ontologías relacionadas al tema. Reusar ontologías puede ser un requerimiento para algunas aplicaciones si estas deben interactuar en conjunto con otras. Para este fin se consultaron otras ontologías relacionadas a Líneas de Productos, entre ellas OntoExper-SPL (Vignando et al., 2020). Sin embargo, no fue encontrada en la literatura alguna con un enfoque parecido a las experiencias industriales, que es el objetivo de este trabajo. Además, se ha considerado esta como una base que se incorpora al prototipo que se desarrolla en este proyecto y se deja a disposición de la comunidad para futuras integraciones.

3. **Enumerar términos importantes para la ontología**

   Realizar una lista de términos relacionados al dominio de la ontología es clave. Además, según la guía es un buen ejercicio pensar en términos sin preocuparse por los distintas características que puedan tener y luego empezar a realizar jerarquías, propiedades y tipos de datos. La lista de términos propuesta a continuación es una parte de la resultante en el proceso de acercamiento al dominio del conocimiento. A partir de esa lista de códigos se inicia la creación de clases o propiedades para la ontología. Algunos de los términos son:

   - Conceptos
   - Herramientas
   - Productos
   - Compañías
   - Ventajas
   - Desventajas
   - Artículos
   - URL
   - Variabilidad
   - Autores
   - Resultados
   - Beneficios

- Nombre
- Título
- Reuso

4. **Definir las clases y la jerarquía de las clases**

   Según Noy, McGUINNESS (2005), existen 3 formas de acercamiento para la definición de jerarquías de clases dentro de la ontología, estas son:

   - **Top-down**: Para este proceso de desarrollo la definición de clases empieza por los conceptos más generales a los más específicos.
   - **Bottom-up**: Por otro lado, el proceso Bottom-up empieza con la definición de los conceptos de forma más específica hasta llegar a los más generales.
   - **Combinado o Híbrido**: El proceso Combinado o Híbrido permite empezar por conceptos tanto generales como específicos, pero que contendrán clases intermedias que van a relacionarlos.

   En este trabajo el proceso de desarrollo de jerarquía de clases es de tipo **Top-down**, ya que es este método el que se adoptó en el proceso de codificación de los artículos en la fase de acercamiento al dominio del conocimiento con el fin de obtener una taxonomía que va de conceptos generales a específicos. El Cuadro 3.4 muestra las clases definidas con su respectiva descripción para la implementación. Se aclara que los nombres se conservan en inglés, ya que la literatura se encuentra en dicho idioma y así se evitan ambigüedades de traducción.

   Cada una de las distintas clases equivalen a agrupaciones de elementos que se encuentran en el proceso de acercamiento al dominio del conocimiento en cada una de las experiencias documentadas. La Figura 3.8 muestra la representación de clases implementadas dentro de la ontología en el editor Protégé.

Clase	Descripción
Autor	Agrupa los autores de los documentos
Challenges	Agrupa los retos de migrar Líneas de Producto
Companies	Empresas donde se realizó el proceso de adopción. Subclases: HardwareCompany, softwareCompany
CompanyDomain	Dominio al que pertence la compañía
Concepts	Definiciones encontradas sobre términos de Líneas de Productos. Subclases: ArchitectureConcepts, ConfiguratorConpcept, GeneralConcepts, ReuseConcepts, VariabilityConcepts
Consequences	Efectos de usar Líneas de Productos. Subclases: Advantages, Benefits, Disadvantages, Results
Motivations	Agrupa lo que motiva a la empresa o compañía a migrar hacia una Línea de Productos
Papers	Artículos que hacen parte del grupo de la base de conocimiento explorada en el desarrollo del proyecto. SubClases: PreAdoption, Adoption y Maintenance
Products	Agrupa productos implicados en el proceso de migración a Líneas de Productos. Subclases: SoftwareProduct
Resources	Recursos usados en una experiencia documentada. Subclases: Cost, Team y Time
Tools	Agrupa pasos, estrategias, metodologías y herramientas de software utilizadas o exploradas en la mirgración a Líneas de Productos. Subclases: Methodologies, Models, Software y Strategies

Cuadro 3.4: Descripción clases de la Ontología. Elaboración propia.

5. **Definir las propiedades de las clases: slots**

   Las propiedades son atributos que describen la clase. Estas propiedades permiten crear una estructura interna para cada individuo que pertenece a una clase específica con el fin de almacenar la información suficiente. Existen propiedades que describen atributos de la clase (*Data Properties*) y otras que describen relaciones entre individuos con otras clases (*Object Properties*). Por ejemplo, para la clase **PAPERS** se definieron 3 atributos y una relación. Los atributos son **Title**, **Url** y **Year** y la relación es **writeBy**. El objetivo de cada slot o atributo es poder almacenar cada una de las características para cada instancia de clase, y en el caso de la relación, enlazar cada instancia de tipo **PAPERS** con instancias de tipo **AUTOR**, configurando los autores(dominio) de cada artículo(rango).

   El cuadro 3.5 muestra la relaciones definidas en la ontología con su respectiva descripción. Se destaca que el editor Protégé permite definir distintos tipos de relaciones. Las relaciones en

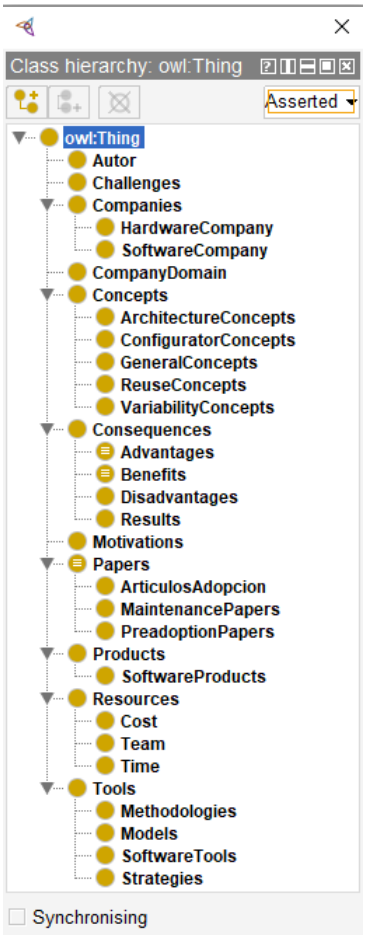


Figura 3.8: Representación de la ontología en la editor Protégé. Elaboración propia

este proyecto se ven como grafos dirigidos o funciones entre individuos donde las clases son el dominio o el rango. La Figura 3.9 muestra cómo se ven las relaciones o Object Properties en el editor de ontologías Protégé.

Relaciones	Descripción	Dominio-Rango	Inversa
reportedBenefitBy	Relaciona artículos con los beneficios que reporta	Papers - Benefits	benefitReportedBy
hasDomain	Relaciona compañias con el dominio al que pertenecen	Companies - CompanyDomain	
autorOf	Relaciona autores con sus artículos	Autors - Papers	writeBy
challengeOf	Relaciona retos con la compañía que los reporta	Challenges - Companies	hasChallenge
hasCompany	Relaciona artículos con la compañía que reportan	Papers - Company	hasPaper
hasConsequence	Relaciona herramientas con las consequencias obtenidas	Tools - Consequences	isConsequenceOf
hasType	Relaciona los productos con elemento de un arreglo	Products - [software, hardware]	
implementedIn	Relaciona las herramientas implementadas en los artículos	Tools - Papers	useA
introduceA	Relaciona artículos con los conceptos de Líneas de Productos que describe	Concepts - Papers	introducesBy
inverseOf	Relaciona ventajas con desventajas	Advantages - Disadvantages	inverseOf
manufacturedBy	Relaciona productos con la compañía que lo produce	Products - Companies	manufactures
motivartionShowBy	Relaciona motivaciones con las compañías que las exponen	Motivations - Companies	showMotivation
relatedWith	Relaciona ventajas con beneficios	Advantages - Benefits	relatedWith
resourcesUseBy	Relaciona Recursos usados en un artículo para el proceso de migración o adopción	Resources - Papers	useResources

Cuadro 3.5: Descripción de las relaciones de la Ontología. Elaboración propia.

6. **Definir las facetas(tipos de datos) de los slots**

   Los slots o propiedades de las clases poseen varias facetas. Las facetas describen el tipo de valor, valores admitidos y el número de valores de cada atributo (Noy, McGUINNESS, 2005). No todas las facetas deben estar estrictamente definidas para cada atributo. En este proyecto

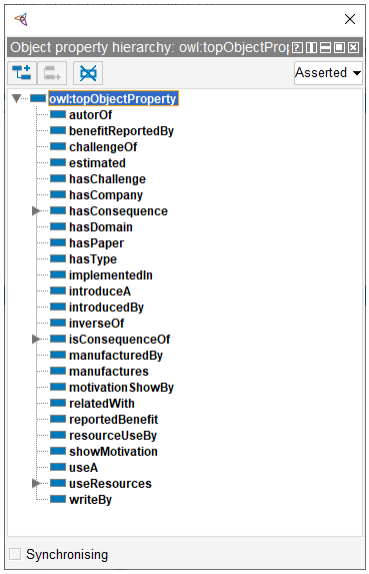


Figura 3.9: Representación de las relaciones de la ontología en el editor Protégé. Elaboración propia.

es esencial definir el tipo de dato de cada atributo, ya que permite separar datos cualitativos de los cuantitativos al momento de recuperar la información y de esta manera mostrarlos de la forma más adecuada. Por ejemplo, para la clase **PAPERS** los atributos o slots definidos son **Title** y **Url**. La faceta del slot **Title** es de tipo cadena de caracteres, mientras que el atributo **Year** es de tipo entero. La Figura 3.10 muestra los slots definidos en el proyecto. Se resalta que estos propiedades son reutilizables dentro de diferentes clases.

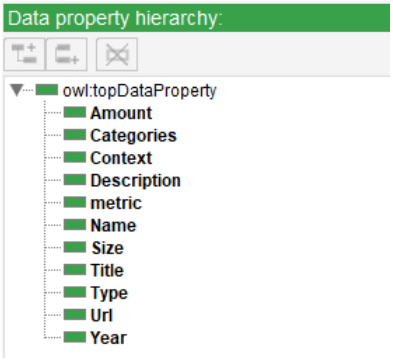


Figura 3.10: Representación de los atributos de la ontología en el editor Protégé. Elaboración propia.

7. **Crear Instancias**

   Para finalizar, es necesario poblar la ontología. Este paso consiste en la creación de individuos de cada clase. Establecer un individuo requiere del llenado de las propiedades necesarias para el mismo. El nombramiento de individuos debe ser intuitivo. Por ejemplo, las instancias de la clase **AUTOR** deben tener el nombre de pila del autor. Seguidamente, se llenan las propiedades objeto y datos, que para el caso de un individuo de la clase Autor son **Name**, correspondiente al nombre del autor, y **autorOf**, que relaciona los individuos de tipo **PAPERS** que la instancia que Autor escribió. En este proyecto la creación de individuos se realiza de forma manual por medio del editor de ontología Protégé y se ilustra en la Figura 3.11.

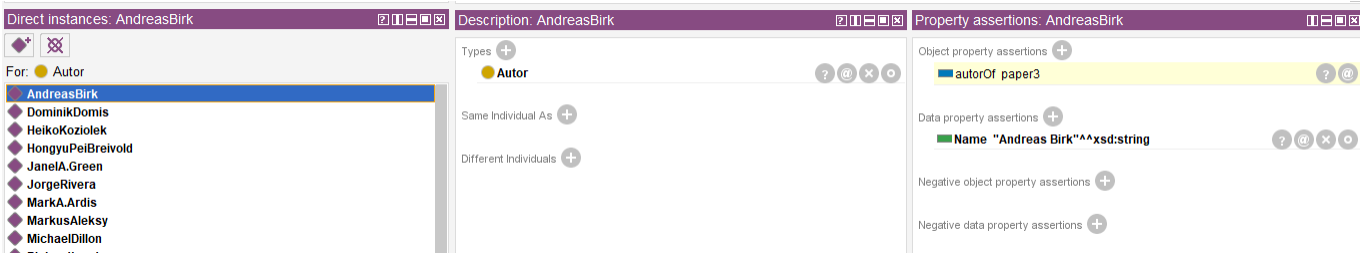


Figura 3.11: Visualización de una instancia de tipo Autor en Protégé. Elaboración propia.

3.2.1.3. Resultados Ontología

Para finalizar el proceso de acercamiento al dominio del conocimiento, se obtuvo una ontología con 34 clases, 35 relaciones y 14 atributos, los datos son verificados por el editor de ontologías Protégé (Figura 3.12). La Figura 3.13 se muestra una sección de la ontología que muestra clases, subclases y relaciones.

Metrics

Class count	34
Object property count	35
Data property count	15

Figura 3.12: Métricas de la ontología, editor Protégé. Elaboración propia.

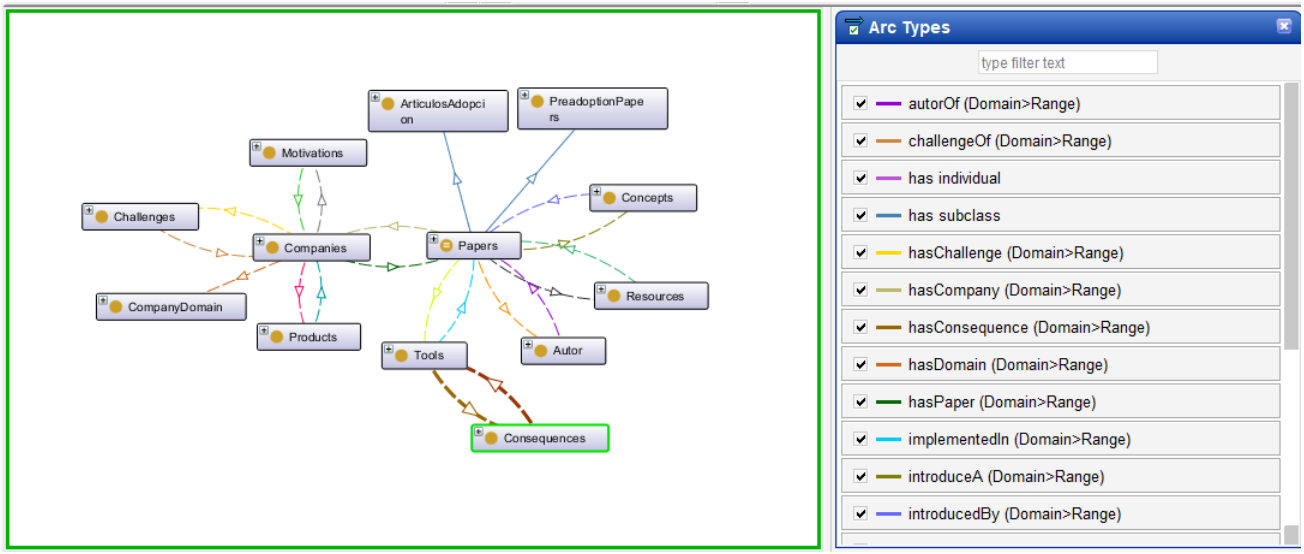


Figura 3.13: Sección de la ontología desarrollada en el editor Protégé. Elaboración propia.

3.2.1.4. Preguntas de prueba a la ontología

Como parte del proceso de comprobar el funcionamiento de la ontología desarrollada se realizan preguntas en SPARQL Query, herramienta del editor Protégé para consultar información. Algunas de las preguntas que se realizan son:

1. ¿Cúales son los retos reportados?

   - Consulta SPARQL

     Ene

   - Resultado

class	Challenge	Description
Challenges	"CommonUnderstanding"@en	"he technology core team needs to communicate the vision on a regula
Challenges	"IdentifyComponentsToRefactor"@en	"the main focus is to identify components that need to be refactored to f
Challenges	"Cost-effectivelyManageMigration"@en	"An important and challenging type of software maintenance and evolut
Challenges	"MigrationActivities"@en	""Architecture workshops need be conducted, where the stakeholders c
Challenges	"ArchitectureRequirements"@en	"Architecture workshops need be conducted, where the stakeholders di
Challenges	"CommonalityAmongProducts"@en	"Fundamentally, the problem is that reuse happens when a project star

Figura 3.14: Resultado pregunta 1. Elaboración propia.

2. ¿Cuáles son los escritores de los artículos?

   - Consulta SPARQL

```
SELECT Distinct ?class ?Title ?Autor
WHERE {?class a owl:Class.
      ?class rdfs:label "Papers"@en.
      ?paper a ?class.
      ?paper  uri:Title ?Title.
      ?paper uri:writeBy ?Autor}
```

   - Resultado

class	Title	Autor
Papers	"Successful Introduction of Domain Engineering into Software	JanelA.Green
Papers	"Successful Introduction of Domain Engineering into Software	MarkA.Ardis
Papers	"A Methodical Approach to Product Line Adoption"^^<http://www	JorgeRivera
Papers	"A Methodical Approach to Product Line Adoption"^^<http://www	MichaelDillon
Papers	"A Methodical Approach to Product Line Adoption"^^<http://www	RowlandDarbin
Papers	"Customizing Domain Analysis For Assessing The Reuse Pot	MarkusAleksy
Papers	"Customizing Domain Analysis For Assessing The Reuse Pot	ThomasGamer
Papers	"Customizing Domain Analysis For Assessing The Reuse Pot	HeikoKoziolek
Papers	"Customizing Domain Analysis For Assessing The Reuse Pot	StephanSehestedt
Papers	"Customizing Domain Analysis For Assessing The Reuse Pot	DominikDomis
Papers	"Three Case Studies on Initiating Product Lines: Enablers and	AndreasBirk
Papers	"Migrating Industrial Systems towards Software Product Lines:	RichardLand

Figura 3.15: Resultado pregunta 2. Elaboración Propia.

3.2.2. Módulo de Búsqueda

El módulo de búsqueda se encarga de recibir la consulta y realizar operaciones como el cargue del modelo y la ejecución de las búsquedas sobre el mismo. Para este objetivo se proponen distintas herramientas que apoyan las anteriores actividades. Solr, Lucene y Jena son las herramientas propuestas y las cuales son una referencia importante como motores y/o frameworks para la recuperación de información. Con el fin de cumplir los objetivos propuestos en el proyecto, la herramienta de recuperación deben tener las siguientes características:

- Soportar modelos ontológicos, en este caso expresado en lenguaje OWL en un archivo RDF/XML.
- Soportar consultas de tipo SparQL.
- Proveer métodos que simplifiquen la necesidades requeridas sobre recuperación de información.

Se debe aclarar que Solr está basado en Lucene. Es decir, Solr refina Lucene adicionando más beneficios. Por ello, es Solr el que clasifica para la posible elección dentro del proyecto junto a Jena. Tanto Jena como Solr poseen características importantes ya que proveen API's (Interfaz de Programación de Aplicaciones) que permiten accesos más sencillos a sus funcionalidades a nivel de desarrollo back-end. Sin embargo, Solr al ser un motor de búsqueda consolidado requiere de pasos adicionales como la definición de plantillas XML para el acoplamiento de una ontología. Por otro lado, Jena es según la literatura, la herramienta por excelencia para el manejo de ontologías y web semántica (McBride, 2002). Además, Jena provee API's para consultas SparQL, cargue y almacenamiento de modelos en RDF, motor de inferencia y almacenamiento. La Figura 3.16 muestra la arquitectura de Jena.

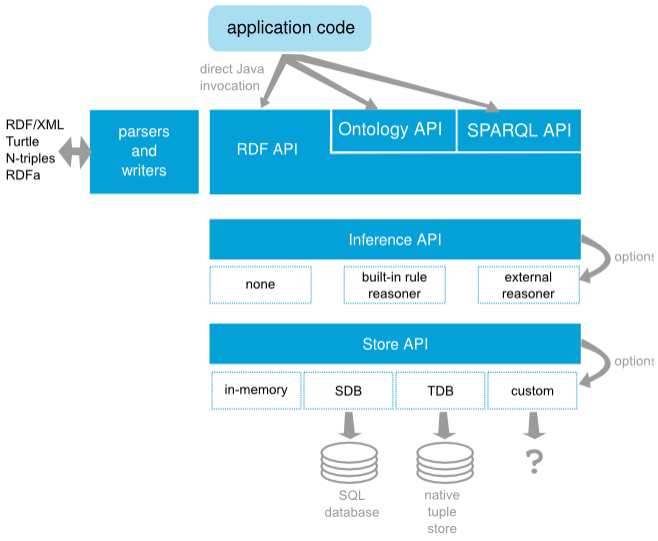


Figura 3.16: Arquitectura Jena. Tomado de (McBride, 2002).

Jena provee beneficios que a futuro pueden apoyar la transformación del prototipo del buscador para expandir sus funcionalidades, por ejemplo , añadir nuevas clases a la ontología. Por lo anterior y por contar con las características necesarias para esta versión del proyecto Jena es elegido como el framework para el manejo de la recuperación de información.

Por otro lado, se plantearon Java o Node.js como lenguajes de programación para Back-end. Aunque Node.js es uno de los lenguajes más usados del lado del servidor para aplicaciones web en la actualidad, la incorporación que hay de Jena con este lenguaje solo se puede realizar por medio de librerías de terceros y con funcionalidades limitadas, ya que aún están en proceso de construcción. En compensación, Jena está hecho en Java, por lo cual, su incorporación y acoplamiento es directo con dicho lenguaje de programación. Jena se convierte en una dependencia del proyecto Java donde se pueden aprovechar la totalidad de sus funcionalidades. Con Java como Lenguaje

de programación elegido y Jena como motor de búsqueda se inicia el desarrollo Back-End para el manejo de las búsquedas.

Siguiendo la arquitectura planteada en la fase de Diseño, se crea un proyecto WEB. Para este fin se utilizó Apache NetBeans como entorno de desarrollo (IDE) [6]. Este Entorno de desarrollo entrega una arquitectura predefinida orientada al desacoplamiento y las buenas prácticas del desarrollo Web. Siguiendo dicho esquema y orientado en la propuesta inicial de arquitectura, se desarrollaron los paquetes predefinidos para de ese modo resolver las funcionalidades del buscador. La Figura 3.17 muestra los paquetes creados en el desarrollo Backend.

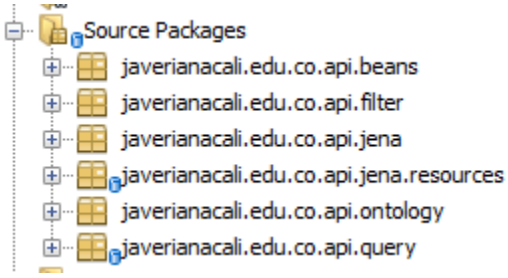


Figura 3.17: Estructura de paquetes desarrollados en NetBeans. Elaboración propia.

```
1  public static OntModel onto() {
2          OntModel mode = null;
3          mode = ModelFactory.createOntologyModel( OntModelSpec.OWL_MEM_RULE_INF );
4          java.io.InputStream in = FileManager.get().open(OwlFile);
5          if (in == null) {
6              throw new IllegalArgumentException("Archivo no encontrado");
7          }
8              System.out.println("Cargó");
9              return  (OntModel) mode.read(in, "");
10     }
```

Figura 3.18: Cargue del modelo Ontológico en Jena. Elaboración propia.

[6] https://netbeans.apache.org/

La Figura 3.18 muestra uno de los métodos que se desarrollan de la clase ontology. Onto() se crea para el proceso de cargue de la ontología. El método retorna un modelo de tipo OntModel propio del método ModelFactory de la API de Jena. Es de notar que el cargue de la ontología se realiza por medio de la lectura de un archivo .OWL que previamente debe ser importado del editor de ontologías Protégé y el cual corresponde a la ontología desarrollada en la sección anterior. Un archivo como persistencia de información no es la mejor práctica, pero la implementación de la ontología en un Base de Datos va más allá del alcance del proyecto.

3.2.3. Módulo de Consulta

Cuando un usuario utiliza una aplicación de búsqueda, generalmente escribe la consulta en lenguaje natural a través de un formulario. La interfaz que contiene el formulario se encarga de enviar dicha consulta al servidor que contiene el motor de búsqueda. Una vez recibida, se debe transformar dicho lenguaje natural a una consulta formal que pueda ser interpretada por el motor de búsqueda (Benítez Andrades, 2013). Este módulo es el encargado de realizar el proceso de tratamiento de la consulta para que posteriormente sea entregada a los métodos que necesiten ejecutarla.

Teniendo en cuenta que el modelo de representación del conocimiento se realiza por medio del uso de una ontología, se proponen dos opciones de lenguajes de consulta que soportan este tipo de modelos, SPARQL y SERQL. Con el fin de elegir el lenguaje más expresivo y más utilizado, Marquez Solis (2007); Plaza (2009) presenta una comparación entre SERQL Y SPARQL, los cuales poseen funcionalidades y propiedades muy parecidas. Sin embargo, SparQL se destaca por poseer la forma de realizar recursión y tener una amplia gama de tipos de datos, además de contar con una gran comunidad debido a su uso. Asimismo, Jena que es el framework de recuperación de información elegido posee una API para la ejecución de este tipo de consultas sobre el modelo de ontología desarrollado.

Por lo anterior, SPARQL se elige como lenguaje de consulta para el proyecto. SPARQL está basado en comparación de tripletas o patrones triples, así como las tripletas RDF pero con la opción de una variable de consulta en lugar de un término RDF. Combinando las tripletas se puede obtener un patrón para proceder a hacer la comparación. La sintaxis de una consulta SparQL se muestra en el Cuadro 3.6 expuesto en (Benítez Andrades, 2013).

La función de la palabra PREFIX es equivalente a la declaración de namespace en XML, es decir, donde se asocia el contexto a una etiqueta. Existen una gran similitud en la sintaxis de SPARQL con SQL por las palabras claves utilizadas. Al inicio de una consulta SPARQL se encuentra la palabra SELECT, semejante a SQL, donde se definen los datos que se requieren en la respuesta. Seguido va la sentencia FROM, el cual es opcional y donde se identifica los datos sobre los que se ejecutará la consulta. Luego se encuentra la sentencia WHERE, donde en caso de no exitir FROM, puede ir especificado el namespace como una URI y se especifica el patrón de la

Prologue (optional)	▪ BASE <IRI> ▪ PREFIX prefix: <IRI>(repeatable)
Query Results forms (required, pick 1)	▪ SELECT (DISTINCT)sequence of ?variable ▪ SELECT (DISTINCT)* ▪ DESCRIBE sequence of ?variable or <IRI> ▪ DESCRIBE * ▪ CONSTRUCT [graph pattern] ▪ ASK
Query Dataset Sources (optional)	Add triple to the background graph (repeatable): ▪ FROM <IRI> Add a named graph (repeatable): ▪ FROM NAMED <IRI>
Graph Pattern (optional, required for ASK)	▪ WHERE [graph pattern z]
Query Results Ordering(optional)	▪ ORDER BY ..
Query Results Selection(optional)	▪ LIMIT n, OFFSET m

Cuadro 3.6: Sintaxis Básica de una consulta SPARQL. Tomado de (Benítez Andrades, 2013)

tripleta con el cual se van a filtrar las tripletas RDF del modelo completo.

Las consultas SPARQL cuentan con formas de ordenamiento y filtros. Además, tienen la capa-

cidad de preguntar si los elementos consultados están en el modelo. Esta opción es realizada por medio del método ASK, el cual devuelve un valor booleano. La estructura básica de una consulta SparQL es la siguiente:

```
SELECT <variables>
WHERE {
   <graph pattern>
}
```

Donde las variables siguen la sintaxis *?nombreVariable* con el signo de interrogración como indicador y el patrón del grafo sigue la estructura $\langle$ *sujeto* $\rangle$ $\langle$ *predicado* $\rangle$ $\langle$ *objeto* $\rangle$.

En el proyecto el módulo encargado de realizar la tratamiento de la consulta de usuario es el paquete api.query. Los métodos de la clase *Query* se ocupan de recibir el body del endPoint correspondiente y retornar la estructura SELECT predefinida en lenguaje de consulta SPARQL.

3.2.4. Módulo de Visualización

Actualmente, con el auge de la gran cantidad de datos, se le ha dado importancia a la manera de poder presentarlos para brindar información de utilidad. Por lo anterior surge la visualización de datos como una herramienta que ayuda a presentar la información de forma más clara (Duke et al., 2005).

El módulo de visualización se encarga de abordar la forma como se presenta la información resultante de una consulta a un usuario. En este módulo se abarca la importancia de la presentación de la información de tal manera que esta pueda ser efectiva. EL campo de la visualización de datos posee una gran fortaleza ya que según Infogram (2018) las personas pueden procesar hasta 60 mil veces más rápido contenido visual que texto.

Hablar de técnicas de visualización de información es referirse al conjunto de procedimientos o recursos que se utilizan para poder representar datos de forma gráfica. Datos de carácter cualitativo o cuantitativo que necesitan ser agrupados para una mejor comprensión. Existen 3 aspectos relevantes para representar información. Estos son:

1. **Relaciones de datos**

   Las relación de datos está definida como la relación entre uno o más puntos de datos (Infogram, 2018). Estas relaciones pueden llegar a ser sencillas o complejas. Por ejemplo, medir el número de personas que entra a distintos centros comerciales en un lapso de tiempo o poder determinar cuantas de ellas eligieron un color de camisa específico para cada día de la semana. Las relaciones de datos están clasificadas en (Infogram, 2018) de la siguiente forma:

- **Comparación nominal**: Es una comparación de datos cuantitativos por subcategorias.
- **Series de tiempo**: Muestra el cambio de un valor en una misma métrica a través del tiempo.
- **Correlación:** Se da cuando los datos tiene dos o más variables que pueden demostrar una correlación positiva o negativa una con otra.
- **Clasificación**: Muestra el tamaño relativo de la relación entre dos o más valores.
- **Desviación**: Examina cómo se interrelacionan los puntos de datos y, en particular, cuánto difieren los datos de la media.
- **Distribución**: A menudo muestra la distribución de datos respecto a una pieza informativa, valor o dato central .
- **Parte a parte**: Esto muestra un subconjunto de datos en comparación con el conjunto Padre.

2. **Forma de visualización**

   La forma de visualización hace énfasis en los gráficos que se pueden usar para representar determinada información. Entre los gráficos más conocidos para la representación de datos cuantitativos se encuentran:

   - Gráficos de barras
   - Gráficos Circulares
   - Gráfico de Líneas
   - Gráfico de Área
   - Gráfico de Dispersión
   - Gráfico de Burbujas
   - Mapas de Calor

   Saber elegir cual gráfico utilizar es un proceso que está determinado por la relación de los datos. Cada relación puede tener una o varias formas de poder representarse. Sin embargo, existen características importantes que tienen efectos sobre la claridad de la representación, por ejemplo la efectividad de la percepción. Según Cleveland et al. (1986) "las personas realizan las tareas perceptivas asociadas a la interpretación de presentaciones gráficas con diferentes grados de precisión". La Figura 3.19 tomada de dicho acercamiento sobre datos cualitativos, representa el resultado de las pruebas experimentales en las cuales se basa la clasificación. Esta es importante porque expone que tanto el gráfico elegido como sus formas tienen efecto en la precisión de efectividad del entendimiento.

   Como resultado, se pude notar que las personas tienen más precisión sobre gráficos que involucren posición, longitud y ángulos o pendientes y menor precisión sobre aquellas con volúmenes o densidades.

3. **Diseño de la visualización**

   Elegir un esquema según la relación de los datos y la efectividad de sus formas no son suficientes sino se tiene en cuenta el diseño. Aspectos como el color, orden lógico y resaltado

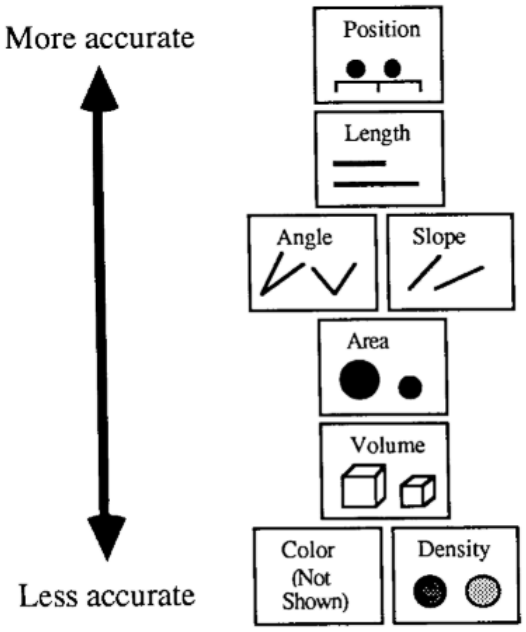


Figura 3.19: Clasificación de la precisión de las tareas perceptivas cuantitativas. Tomado de (Cleveland et al., 1986).

de etiquetas potencia la claridad de lo que desea expresar en el gráfico. Por esta razón en Infogram (2018) presentan los siguientes consejos:

- Etiquetar correctamente los cuadros y gráficos
- Destacar o resaltar la información importante
- Utilizar esquemas de color coherentes y atractivos
- Ordenar los conjuntos de datos de forma lógica
- Evitar los gráficos en 3D
- Elegir rangos de datos adecuados

3.2.4.1. Aplicación de técnicas de visualización

Teniendo en cuenta los aspectos fundamentales para la visualización de información y el alcance del proyecto se empieza por realizar un análisis de aquellas datos que podrían ser expuestos en gráficos. La mayor parte de la información en la Ontología es de tipo cualitativo. Por tal razón se le da relevancia a la frecuencia de las apariciones en la base de conocimiento.

Con el objetivo de garantizar la comunicación de gráficos atractivos para el usuario se llevó a cabo un análisis de estructura del idioma del gráfico propuesto por Munzner (2019). En el se relaciona la combinación de marcas (formas geométricas primitivas), canales (Posición, forma, tamaño, color e inclinación) y datos con la efectividad y la expresividad de cada gráfico.

Por ejemplo, el gráfico que muestra la Figura 3.20 se elije ya que los datos son de carácter categórico nominal. La estructura del idioma del gráfico tiene como marca el área y posee una combinación de canales que son color y forma. En conjunto este idioma del gráfico logra cumplir un principio de expresividad al relacionar estos atributos categóricos con regiones y colores. También, cumple un principio de efectividad al codificar los atributos con uno de los canales que según la escala de precisión a tareas perceptivas (Figura 3.19) posee un buen ranking.

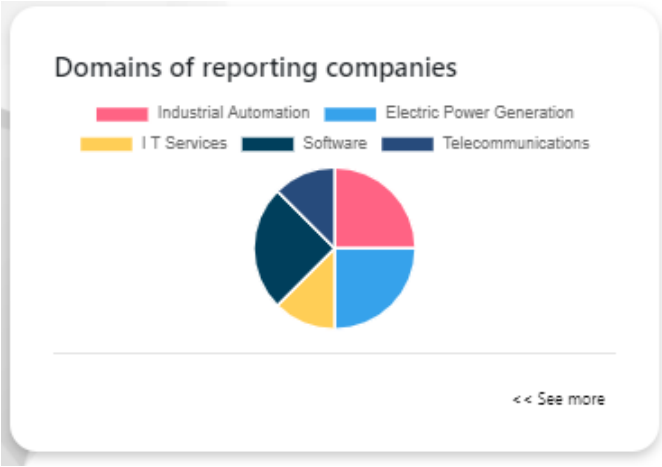


Figura 3.20: Gráfico en estado normal. Elaboración propia.

Por otro lado, las Figura 3.21 muestra una característica importante. Tiene un apartado en el cual el usuario puede ampliar información y se encuentra clasificada. Esta característica se conoce como detalles a la carta. En la Figura 3.22 se destaca la posibilidad de suprimir algunas de las variables del gráfico para brindar mayor ajuste al interés del usuario. Esto está relacionado al zoom y filtro del la información renderizada. La alineación de los textos de forma horizontal, que el gráfico sea plano y no 3D sin justificación y que además se ajuste de forma resposive a su entorno es para Munzner (2019) parte de las 'reglas de oro' de visualización de información y las cuales se han integrado en los gráficos de este proyecto.

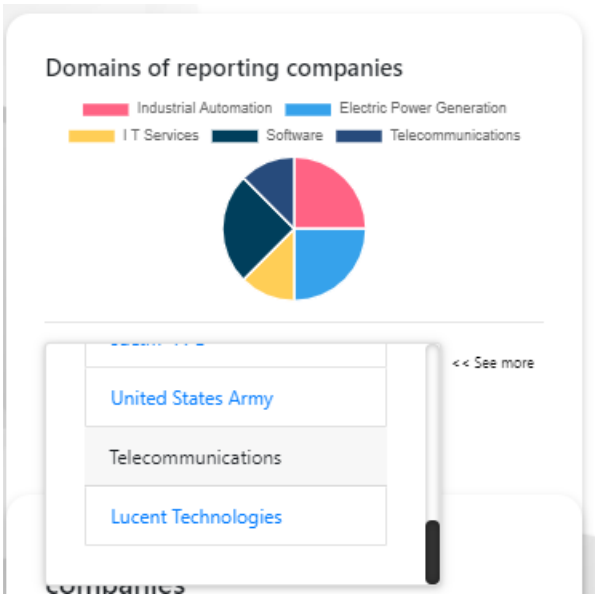


Figura 3.21: Gráfico que amplía información. Elaboración propia.

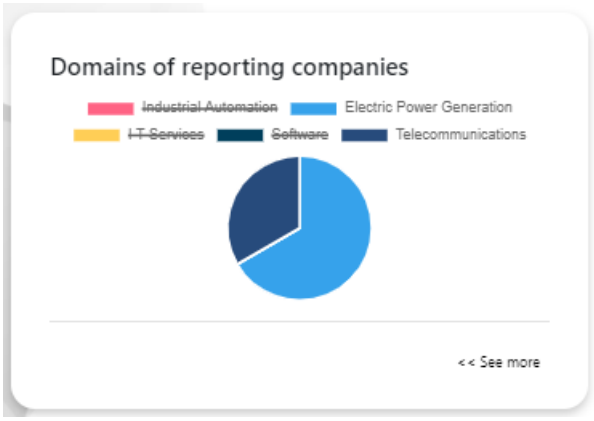


Figura 3.22: Gráfico que permite filtrar variables. Elaboración propia.

3.2.4.2. Incorporación al desarrollo Front-end

Una vez realizado el recorrido del análisis de datos y su respectiva estructura de idioma, se procede a la incorporación dentro de la interfaz de usuario. Cumpliendo a los requerimientos de búsquedas exploratorias y búsquedas sencillas se enfoca el desarrollo de la interfaz de usuario en proveer estas dos funcionalidades, traducidas como exploración de datos y búsqueda sencilla. La exploración de datos es donde el usuario puede visualizar los gráficas resultantes e interactuar con

ellos. La búsqueda avanzada es para que el usuario pueda ir más allá y buscar por cualquiera de las categorías definidas en la ontología.

La herramienta que se usa para el desarrollo de la interfaz de usuario es React.js, una librería de JavaScript que, aunque no es un framework completo, aborda la necesidad de actualizar una vista cuando sea necesario de una forma más inteligente por medio del DOM virtual, que otros frameworks o librerías (Alvarez, 2020). Para la visualización de información se utiliza Chart.js, una librería de JavaScript que proporciona gráficos que permiten personalización y así poder moldear los gráficos con las técnicas sugeridas. La figura 3.23 muestra la estructuración del proyecto front-end.

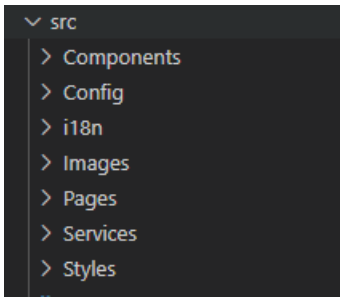


Figura 3.23: Estructura de proyecto Front-end. Elaboración propia.

El sentido de ser de la organización del proyecto que se ve en la Figura 3.23 tiene que ver con la escalabilidad y mantenibilidad del software. Es necesario tener una estructura organizada que nos permita navegar y entender como se conforma el proyecto. React.js está orientado al desarrollo de componentes reutilizables, es por esta razón que aquellas piezas más pequeñas se agrupan para ser reutilizadas dentro de las páginas que se requieran. Como los datos que se muestra vienen desde un servidor diferente se crea la estructura de servicios la cual usa la carpeta Config donde se define el archivo de configuración de endpoints. Para finalizar, y como un adicional al proyecto, la carpeta i18n contiene la especificación de la internacionalización. Esta opción es la que permite cambiar el lenguaje de la aplicación de inglés a español, haciendo uso de diccionarios de mensajes.

3.3. Resultados

Como resultado se obtiene el prototipo funcional del Buscador sobre Casos industriales de aplicación de Ingeniería de Líneas de Productos. El prototipo incorpora cada uno de los de módulos abordados en la sección anterior. Con la retroalimentación dada por la stakeholder y directora del proyecto Luis Rincón, se obtuvo una versión final del prototipo. Las Figuras 3.24, 3.25, 3.26, 3.28 muestran las páginas desarrolladas para el prototipo. Por otro lado, las figuras 3.29, 3.30 muestran funcionalidades de búsqueda.

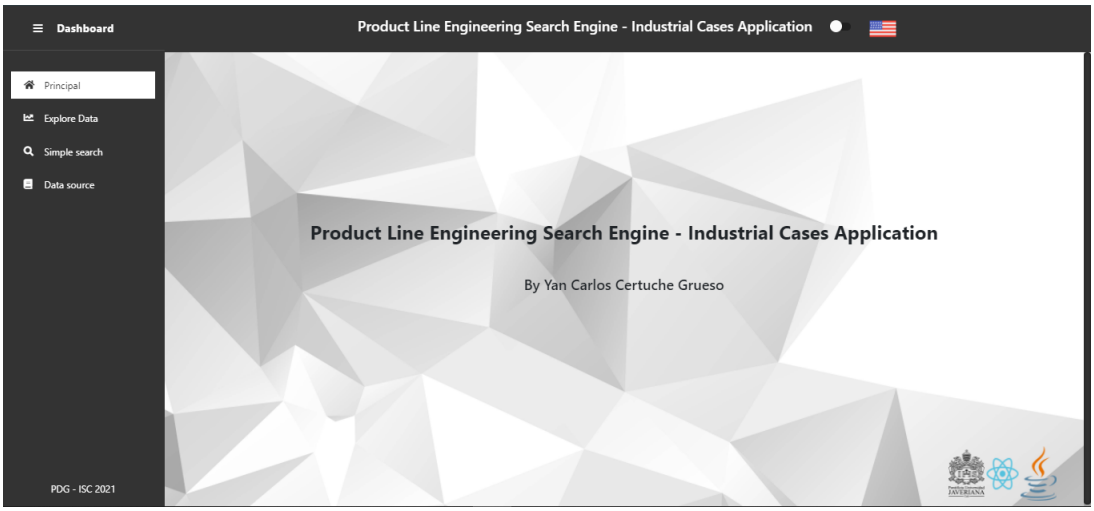


Figura 3.24: Página principal. Elaboración propia.

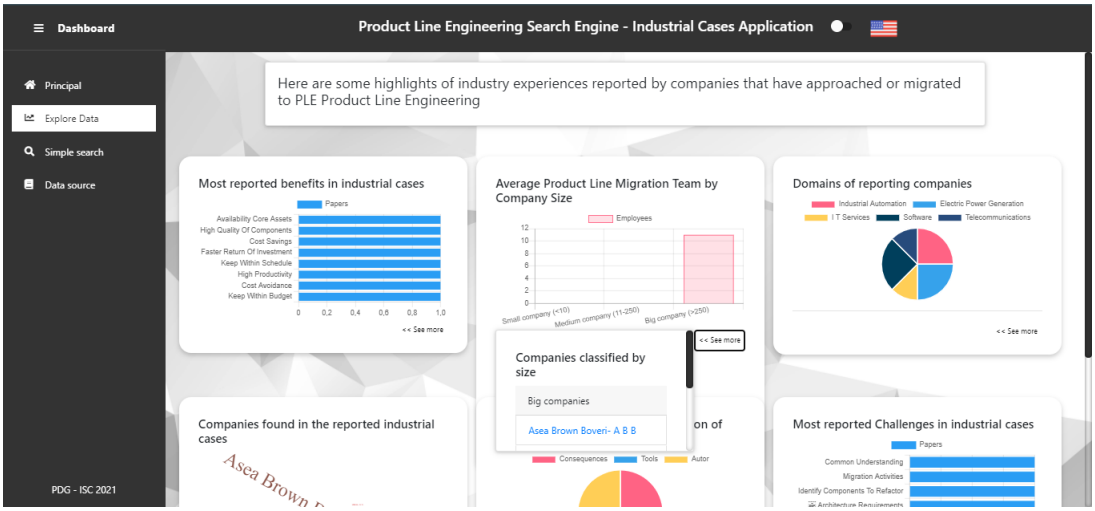


Figura 3.25: Página explorar datos. Elaboración propia.

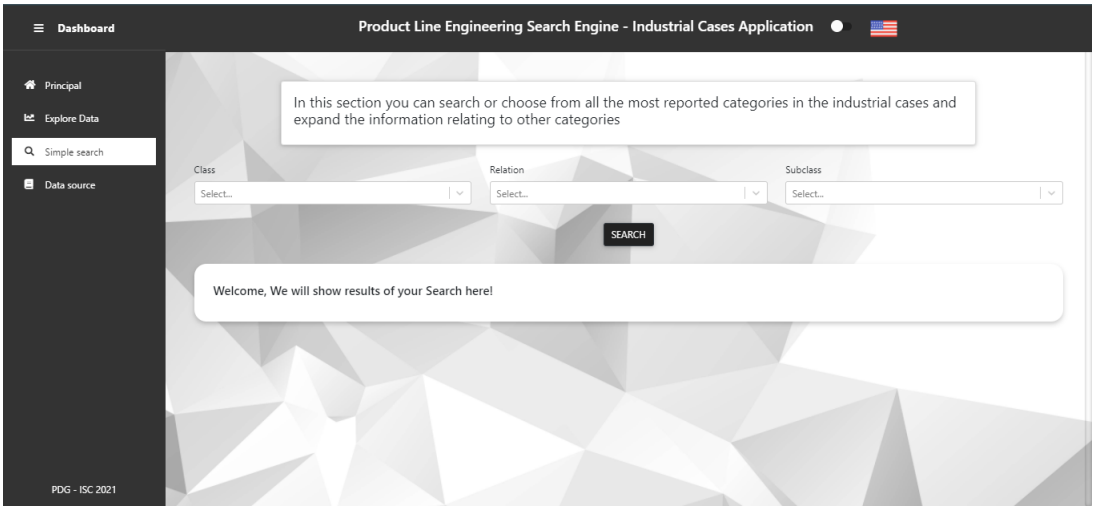


Figura 3.26: Página búsqueda Sencilla. Elaboración propia.

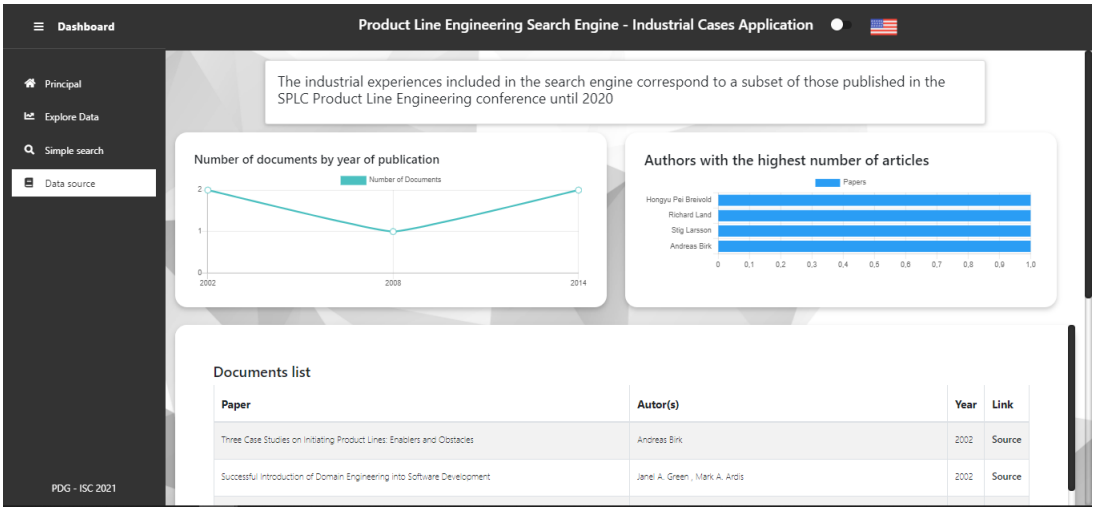


Figura 3.27: Página fuentes de Datos. Elaboración propia.

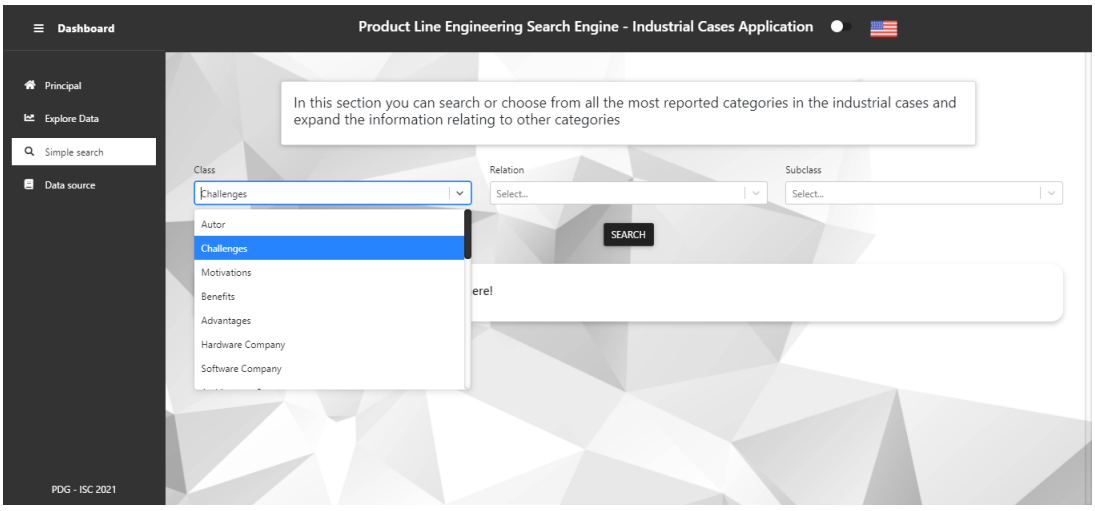


Figura 3.28: Página búsqueda sencilla, selección de clases. Elaboración propia.

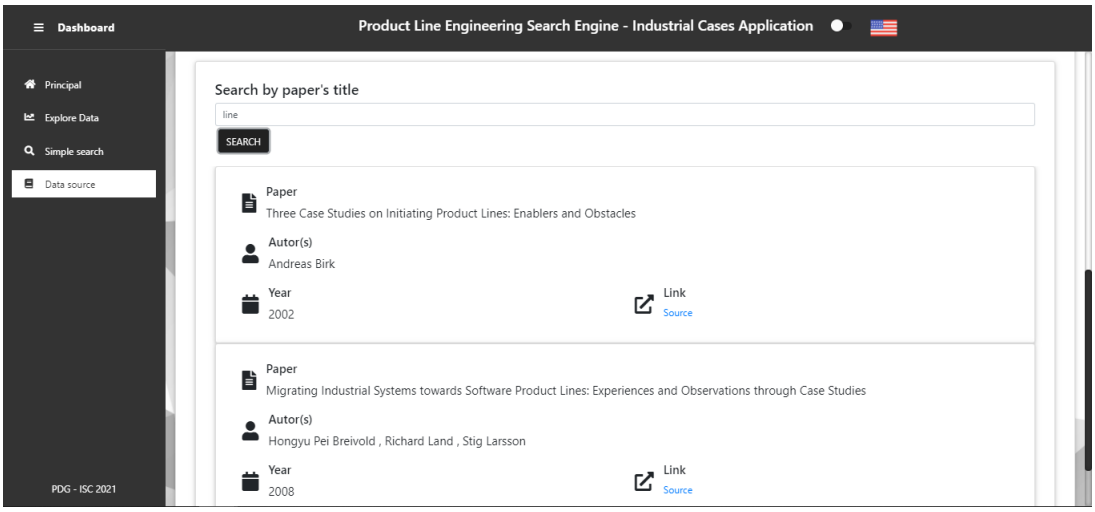


Figura 3.29: Página fuentes de datos, búsqueda de artículos por título. Elaboración propia.

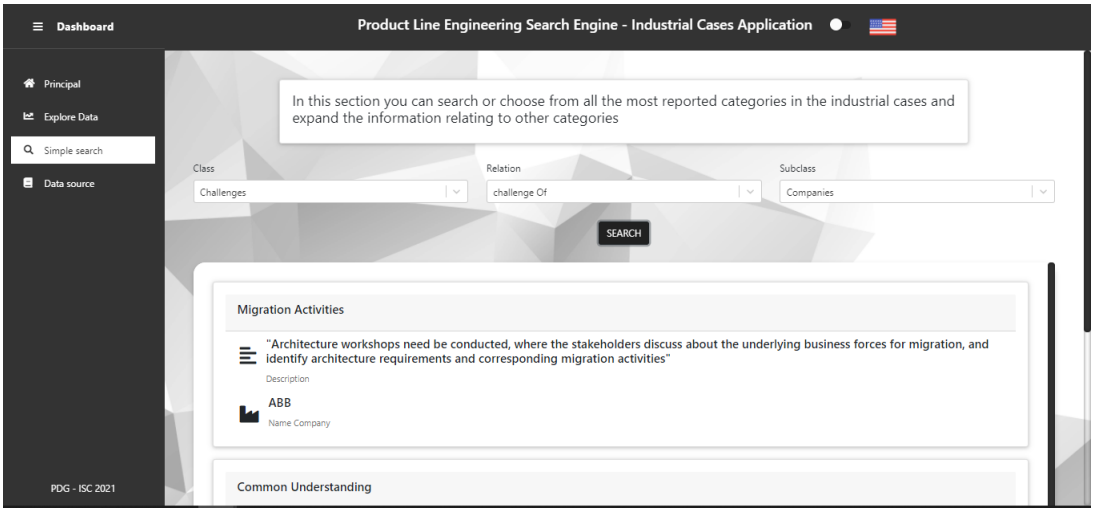


Figura 3.30: Página búsqueda sencilla, búsqueda de retos y la empresa que lo reportó. Elaboración propia.

Capítulo 4

Pruebas de Software

Como parte de las buenas prácticas del desarrollo de software, se realizan pruebas funcionales, en específico de componentes. Estas pruebas están orientadas a la comprobación de que el resultado sea el requerido. Este tipo de prueba es utilizada tanto para probar interfaces de usuario como módulos de código (Vargas, 2020). Las pruebas tienen el objetivo de garantizar el buen funcionamiento del software desarrollado. En el proyecto se realizaron casos de prueba teniendo en cuenta casos de éxito y de fallo a propósito.

4.1. Pruebas Back-end

Al desarrollarse sobre lineamientos de arquitectura Rest, y debido al enfoque de la prueba funcional, el plan es probar los servicios expuestos. Con la ayuda de la herramienta Postman [1] que hace las veces de cliente, se puede realizar esta actividad de una manera más cómoda. Sobre Postman se ejecutan las pruebas de cada uno de los servicios. Por ejemplo, para el caso de prueba CP-01 que se muestra en la Figura 4.4 la respuesta de Postman fue la expuesta en la Figura 4.1, la cual fue exitosa y retorna un *body* con el contenido y un *status code* 200. Un código de estado es la respuesta a la ejecución de una solicitud HTTP específica (Mozila, 2005) y el 200 equivale a ejecuciones exitosas. Por otro lado, para el caso fallido, Postman muestra lo ilustrado en la Figura 4.2, siendo un *status code* 400 *bad request* ya que la petición GET no solicita *body*, por tanto está mal construida.

[1] https://www.postman.com/

Caso de prueba	
Código Caso de prueba	CP-01
Código Requerimiento	RF-001
Propósito	Verificar retorno de datos
Pre - Requisito	Modelo OWL
Datos de prueba	–
Resultados esperados	Lista de clases del modelo en formato JSON
Resultado	Exitoso

Cuadro 4.1: Caso de prueba CP-01. Elaboración propia.

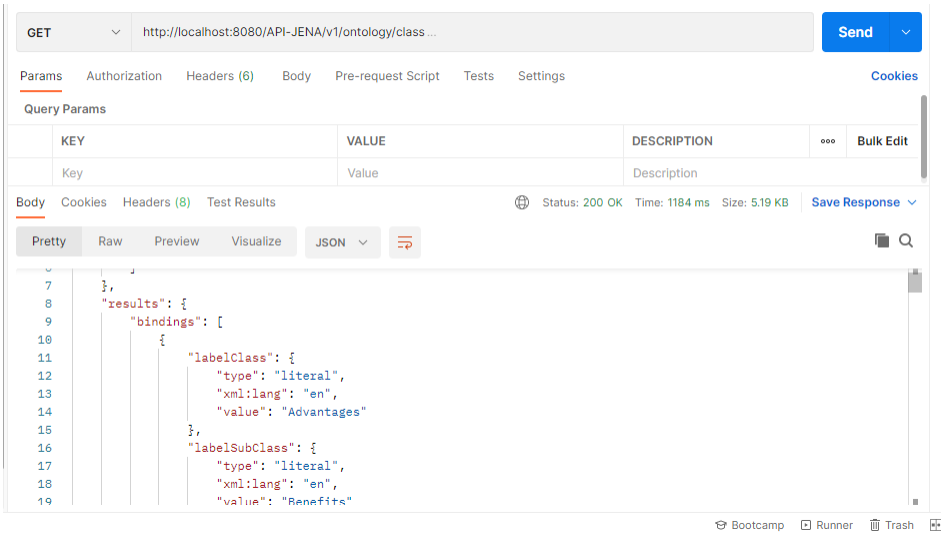


Figura 4.1: Resultado prueba exitosa. Elaboración propia.

Una de los principales retos en alineación con los requisitos no funcionales son los tiempos de respuesta. Los cuales se revisan en esta etapa de prueba, y que gracias a la interfaz que provee Postman se pueden visualizar. En etapas tempranas del desarrollo estos tiempos superaban los 10 segundos. Sin embargo, con el refinamiento del modelo ontológico, las consultas y por último la subdivisión de servicios los tiempos estuvieron por debajo de el tiempo límite propuesto de 5 segundos. También, se tiene en cuenta el nombramiento de recursos y la cohesión con el método de petición. A continuación se presentan otros casos de prueba:

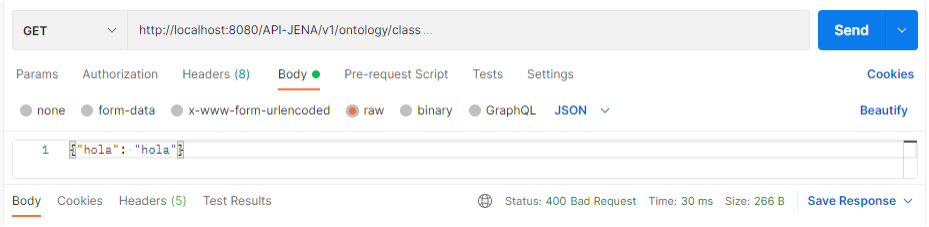


Figura 4.2: Resultado prueba fallida. Elaboración propia.

Caso de prueba	
Código Caso de prueba	CP-02
Código Requerimiento	RF-002
Propósito	Verificar retorno de datos
Pre - Requisito	Modelo OWL
Datos de prueba	request Body, formato { 'class' : 'NombreClase'}
Resultados esperados	Lista con las relaciones de la clase entregada
Resultado	Exitoso

Cuadro 4.2: Caso de prueba CP-02. Elaboración propia.

Caso de prueba	
Código Caso de prueba	CP-03
Código Requerimiento	RF-003
Propósito	Verificar retorno de datos
Pre - Requisito	Modelo OWL
Datos de prueba	request Body, formato { 'class' : 'NombreClase'}
Resultados esperados	Lista de individuos de la clase enviada con sus propiedadas
Resultado	Exitoso

Cuadro 4.3: Caso de prueba CP-03. Elaboración propia.

Caso de prueba	
Código Caso de prueba	CP-04
Código Requerimiento	RF-004
Propósito	Verificar retorno de datos
Pre - Requisito	Modelo OWL, Existencia de parametros.
Datos de prueba	request Body, formato { 'classIn1' : 'NombreClase' , 'relation' : 'relacion', 'classIn2' : 'NombreClase}
Resultados esperados	retorno de los individuos de la relación especificada entre las dos clases
Resultado	Exitoso

Cuadro 4.4: Caso de prueba CP-04. Elaboración propia.

4.2. Pruebas Front-end

Para el desarrollo de las pruebas de la interfaz gráfica se tiene en cuenta aquellas partes donde, a consideración, el usuario va a tener más interacción con la herramienta más allá de un clic. En este caso, las entradas de texto. En este punto se requiere que la interfaz comunique al usuario cuando algún campo es o no requerido a fin de minimizar los errores, ya que no se debe ejecutar, por ejemplo, una búsqueda vacía. El caso de prueba 4.5 abarca la situación descrita anteriormente y su resultado es exitoso (Figura 4.3).

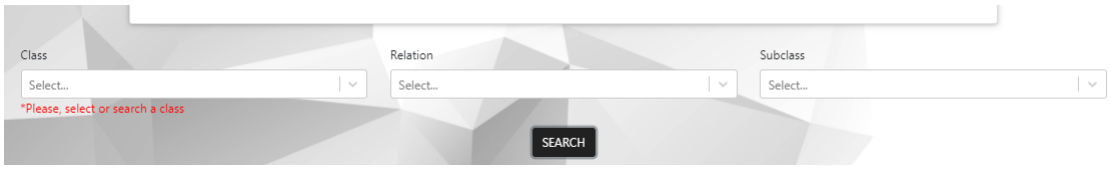


Figura 4.3: Resultado prueba exitosa front-end. Elaboración propia.

En etapas tempranas del desarrollo Front-end se fueron dando cambios respecto a la retroalimentación de la directora y *stakeholder* del proyecto. Fue interesante el avance de las entradas de texto. En principio era una entrada de texto convencional. Sin embargo, los usuarios potenciales podrían no saber exactamente qué buscar y se optó por entradas de selección o *dropdown*. Los dropdown listan las clases de la ontología desarrollada. Desafortunadamente la lista de las clases o relaciones puede crecer, lo cual hace de esta lista desplegable algo poco práctico para el usuario, ya que debería hacer scroll hasta encontrarlo. Fue de esta manera como surge la selección y filtrado

Caso de prueba	
Código Caso de prueba	CP-05
Código Requerimiento	RF-005
Propósito	Usuario ejecuta una consulta sin seleccionar ninguna entrada
Pre - Requisito	–
Datos de prueba	–
Resultados esperados	La interfaz debe solicitarle seleccionar o buscar una clase y no ejecutar las consulta
Resultado	Exitoso

Cuadro 4.5: Caso de prueba CP-05. Elaboración propia.

en un solo componente. Esta entrada de texto permite buscar entre los elementos existentes en la lista de selección o seleccionar de los elementos listados.

4.3. Pruebas Integración

Verificar el funcionamiento en conjunto del software es fundamental. Una vez probado cada componente funcional, se debe verificar el funcionamiento general. Los casos de prueba 4.6, 4.7, 4.8 son algunos de los que cumplen dicha meta y se exponen a continuación.

Caso de prueba	
Código Caso de prueba	CP-06
Código Requerimiento	RF-006
Propósito	Validar respuesta vacia, sin resultados de búsqueda
Pre - Requisito	–
Datos de prueba	–
Resultados esperados	Renderizar un mensaje que comunique al usuario que no se encontraron coincidencias con su búsqueda
Resultado	Exitoso

Cuadro 4.6: Caso de prueba CP-06. Elaboración propia.

Caso de prueba	
Código Caso de prueba	CP-07
Código Requerimiento	RF-007
Propósito	Verificar renderizado de datos en la gráficas y correspondencia al modelo
Pre - Requisito	–
Datos de prueba	–
Resultados esperados	Gráficas con datos acordes a los retornados por el backend
Resultado	Exitoso

Cuadro 4.7: Caso de prueba CP-07. Elaboración propia.

Como se puede observar los casos fueron exitosos y en ellos se revisan tanto el modelo, como el back-end y front-end. Además, se rectifica de forma visual la congruencia de datos mostrados en las gráficas (RNF-002) y los resultados de búsqueda. Por otro lado, no solo que sea funcional es importante, también se necesita que sea eficiente. En concordancia con el requisito no funcional RNF-003 sobre tiempos de respuesta, se adjunta en la Figura 4.4, la cual es tomada de las herramientas de desarrollo del navegador donde se ejecuta la aplicación y la cual entrega una lista de tiempos de las solicitudes http de la interfaz de usuario hacia la lógica del negocio.

Caso de prueba	
Código Caso de prueba	CP-08
Código Requerimiento	RF-009
Propósito	Verificar que se rendericen la propiedades de acuerdo a los datos de los individuos retornados, cards de respuesta dinámicas.
Pre - Requisito	–
Datos de prueba	–
Resultados esperados	Al ejecutar una búsqueda por una clase debe traer el individuo con sus propiedades, no dibujar datos sin información
Resultado	Exitoso

Cuadro 4.8: Caso de prueba CP-08. Elaboración propia.

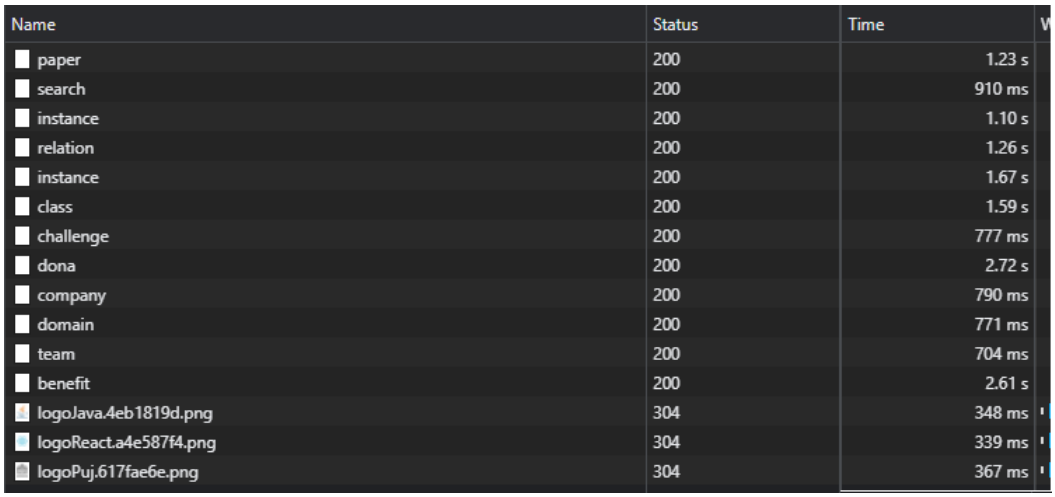


Figura 4.4: Recursos utilizados por la aplicación . Elaboración propia.

Teniendo como referencia la Figura 4.4 podemos observar que los tiempos de cada petición no superan los 5 segundos. Otros de los datos que brinda las herramientas de desarrollador del navegador son los tiempos de carga de las páginas. El *DOMContentLoaded* y el *Load* son eventos que

se disparan cuando el árbol del DOM está completamente construido y cuando se han descargado los recursos necesarios, como scripts, imágenes y estilos (Manz, 2019). La figura 4.5 evidencia que los tiempos de carga del proyecto para el *DOMContentLoaded* es de 3.57 segundos y para el *Load* es de 3.72 segundos, para los cuales, aunque no existe un tiempo límite este debería ser lo más bajo posible ya que puede inferir en el rebote del usuario.

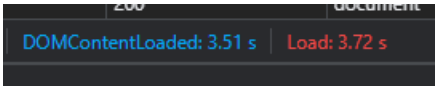


Figura 4.5: Carga de Dom, herramientas de desarrollador del navegador. Elaboración propia.

Capítulo 5

Conclusiones

5.1. Conclusiones

El objetivo de este trabajo fue acercar la experiencias industriales de aplicación de Líneas de Productos a las personas interesadas. Es importante ver en los reportes como las empresas han ido realizando un ajuste de distintos modelos, metodologías y pasos para poder incorporar Líneas de Productos dentro de su compañías. Todos estas estrategias, metodologías, pasos ajustados a las necesidades particulares se convierten en nuevas herramientas para otras empresas que quieren conocer sobre Ingeniería de Línea de productos y las cuales pueden ser encontradas usando el prototipo de buscador que propone este trabajo.

En el desarrollo de este trabajo se diseñó, se implementó y se probó funcionalmente un prototipo de buscador, con características de la web semántica usando una ontología como fuente de consulta, usando técnicas de visualización de información para la presentación de gráficos y siguiendo las etapas de la creación de software.

Según lo identificado en el proceso de acercamiento al dominio del conocimiento se encuentra que las temáticas que más se abordan en las experiencias industriales son los resultados, las herramientas, los retos y las motivaciones de las compañías. Por otro lado, las menos mencionadas son el manejo del riesgo, la identificación de core assets y los costos de la adopción de una Líneas de Productos.

También, en el proceso de acercamiento al dominio del conocimiento se puede notar que las Líneas de Productos más que un paradigma, herramienta o tecnología es toda una filosofía que se crea desde la empresa que adopta para llevar a cabo la adopción exitosamente.

5.2. Lecciones Aprendidas

En el desarrollo del proyecto existieron un gran número de retos. En principio, esta la familiarización con la lectura de textos científicos de las experiencias industriales. El área de Líneas de Productos maneja un gran número de conceptos que pueden ser moldeados al ámbito industrial en el que se aplica, por ende comprender el contexto es fundamental. Aprender del manejo de MAXQDA, una herramienta de análisis cualitativo, la cual fue de gran ayuda en el momento de

clasificar, visualizar y comparar los textos científicos abordados. Por último, aprender sobre web semántica, en especial sobre la creación de ontologías, un tema desconocido para mi, pero que me adentró en el mundo de la Ingeniería del Conocimiento y las grandes virtudes que posee para la información creciente en la *World Wide Web* y sus poderosos usos para herramientas especializadas.

Una de las ventajas a resaltar de utilizar una ontología en comparación a una base de datos convencional es que gracias al razonador incorporado en la herramienta de desarrollo Protégé, no es necesario especificar todas las relaciones de un individuo, ya que el modelo es capaz de inferir relaciones a partir de las relaciones existentes.

5.3. Trabajo Futuro

De cara a la forma de poblar la ontología, se podría automatizar este proceso por medio de la creación de una plataforma de gestión que reciba nueva información y la incorpore con la ya existente. Por otro lado, se puede llevar la ontología aquí desarrollada a un agente web y realizar un proceso de refinamiento para realizar web scraping.

Como parte de la prueba del prototipo, se podrían involucrar personas que hagan parte de un grupo de usuarios potenciales del buscador con el fin de realizar pruebas respecto a la percepción de utilidad, facilidad de uso e intención de uso, así como obtener retroalimentación del grupo objetivo al cual está dirigida la herramienta. Este tipo de personas se podrían identificar por ejemplo en la comunidad de la ingeniería de líneas de productos, en la industria del software y en los cursos de posgrado que se interesan en abordar entre sus temáticas la ingeniería de líneas de productos.

Bibliografía

Alvarez Miguel Angel. Qué es React. Por qué usar React. Oct 2020.

Antoniou Grigoris, Van Harmelen Frank. Web ontology language: Owl // Handbook on ontologies. 2004. 67–92.

Ardis Mark A, Green Janel A. Successful introduction of domain engineering into software development // Bell Labs Technical Journal. 1998. 3, 3. 10–20.

Propuesta de Búsqueda Semántica: Aplicación al catálogo de mapas, planos y Dibujos del Archivo General de Simancas. // . 2013.

Birk Andreas. Three Case Studies on Initiating Product Lines: Enablers and Obstacles // Proc. PLEES 2002 Product Line Engineering Workshop. 2002. 19–25.

Braun Virginia, Clarke Victoria. Using thematic analysis in psychology // Qualitative research in psychology. 2006. 3, 2. 77–101.

Breivold Hongyu Pei, Larsson Stig, Land Rikard. Migrating industrial systems towards software product lines: Experiences and observations through case studies // 2008 34th Euromicro Conference Software Engineering and Advanced Applications. 2008. 232–239.

Cal Bruno, O'Neill Henrique. Software Product Lines: a realistic path to software development industrialization? // Atas da Conferência da Associação Portuguesa de Sistemas de Informação. 11. 2014.

Castells Pablo. La web semántica // Sistemas interactivos y colaborativos en la web. 2003. 195–212.

Chauhan Rashmi, Goudar Rayan, Sharma Robin, Chauhan Atul. Domain ontology based semantic search for efficient information retrieval through automatic query expansion // 2013 international conference on intelligent systems and signal processing (ISSP). 2013. 397–402.

Clements Paul, Northrop Linda M. Software Product Lines: Practices and Patterns. 2001. 1st.

Cleveland W. S., McGill R., Mackinlay Jock. Automating the design of graphical presentations of relational information // Acm Transactions On Graphics (Tog). 1986. 5, 2. 110–141.

Cruzes Daniela S, Dyba Tore. Recommended steps for thematic synthesis in software engineering // 2011 international symposium on empirical software engineering and measurement. 2011. 275–284.

Dillon Michael, Rivera Jorge, Darbin Rowland. A methodical approach to product line adoption // Proceedings of the 18th International Software Product Line Conference-Volume 1. 2014. 340–349.

Dordowsky Frank, Hipp Walter. Adopting software product line principles to manage software variants in a complex avionics system // Proceedings of the 13th International Software Product Line Conference. 2009. 265–274.

Duke David J, Brodlie Ken W, Duce David A, Herman Ivan. Do you see what I mean?[Data visualization] // IEEE Computer Graphics and Applications. 2005. 25, 3. 6–9.

Española Real Academia, Madrid España. Diccionario de la lengua española. 19. 1970.

Fajar Mohammad, Nakanishi Tsuneo, Tagashira Shigeaki, Fukuda Akira. Introducing software product line development for wireless sensor/actuator network based agriculture systems // AFITA2010 International Conference, The Quality Information for Competitive Agricultural Based Production System and Commerce. 2010. 83–88.

Falvo Venilton, Duarte Filho Nemésio F, Oliveira Edson, Barbosa Ellen Frantine. A contribution to the adoption of software product lines in the development of mobile learning applications // 2014 IEEE Frontiers in Education Conference (FIE) Proceedings. 2014. 1–8.

Fang Wei-Dong, Zhang Ling, Wang Yan-Xuan, Dong Shou-Bin. Toward a semantic search engine based on ontologies // 2005 International Conference on Machine Learning and Cybernetics. 3. 2005. 1913–1918.

Ferreira Nuno, Machado Ricardo J, Gasevic Dragan. An ontology-based approach to model-driven software product lines // 2009 Fourth International Conference on Software Engineering Advances. 2009. 559–564.

Foguem B Kamsu, Coudert Thierry, Béler Cédrick, Geneste Laurent. Knowledge formalization in experience feedback processes: An ontology-based approach // Computers in Industry. 2008. 59, 7. 694–710.

Gil Teresa González, Arana Alejandra Cano. Los softwares como recurso de apoyo al procesamiento y organización de los datos cualitativos // 47. 2010.

Hamza Haitham S, Martinez Jabier, Alonso Carmen. Introducing Product Line Architectures in the ERP Industry: Challenges and Lessons Learned. // SPLC Workshops. 2010. 263–266.

Infogram . [Ebook Download] Presenting Data People Can't Ignore. Mar 2018.

Kircher Michael, Schwanninger Christa, Groher Iris. Transitioning to a software product family approach-challenges and best practices // 10th International Software Product Line Conference (SPLC'06). 2006. 9–pp.

Knauber Peter, Muthig Dirk, Schmid Klaus, Widen Tanya. Applying product line concepts in small and medium-sized companies // IEEE Software. 2000. 17, 5. 88–95.

Krsek Martin, Zyl Jay van, Redpath Robert, Clohesy Ben, Dean Nick. Experiences of large banks: Hurdles and enablers to the adoption of software product line practices in large corporate organisations // 2008 12th International Software Product Line Conference. 2008. 161–169.

Krueger Charles W. Introduction to the emerging practice of software product line development // Methods and Tools. 2006. 14, 3. 3–15.

Lim Edward HY, Liu James, Lee Raymond. Knowledge Seeker-Ontology Modelling for Information Search and Management. 2013.

Linden Frank van der, Schmid Klaus, Rommes Eelco. Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering. 2007. 340.

Implementación de un buscador semántico de documentos en el dominio de la lingüística. // . 2017.

Manz . ¿Qué ES El Dom? - javascript en español. Jun 2019.

Web semántica y servicios web semanticos. // . 2007.

McBride Brian. Jena: A semantic web toolkit // IEEE Internet computing. 2002. 6, 6. 55–59.

McBride Brian. The resource description framework (RDF) and its vocabulary description language RDFS // Handbook on ontologies. 2004. 51–65.

McGuinness Deborah L, Van Harmelen Frank, others . OWL web ontology language overview // W3C recommendation. 2004. 10, 10. 2004.

Mohan Kannan, Ramesh Balasubramaniam. Ontology-based support for variability management in product and families // 36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the. 2003. 9–pp.

Montilva Jonás A. Desarrollo de Software Basado en Líneas de Productos de Software // IEEE Computer Society. 2006. 1–34.

Mozila . Códigos de Estado De Respuesta HTTP - http: Mdn. 2005.

Munzner Tamara. Visualization Analysis Design All Book/Teaching Slides // University of Brirish Columbia. 2019.

Noy Natalya F, McGUINNESS Deborah L. Desarrollo de Ontologías-101: guía para crear tu primera ontología // traducido del inglés por: E. Antezana,, http://protege. stanford. edu/publications/ontology_development/ontology101-es. pdf. 2005.

Ordoñez Hugo, Cobos Carlos. OntoGhobi-Meta buscador semántico que incorpora una ontología de dominio general (WordNet) y perfil de usuario // XXXVI Conferencia Latinoamericana de Informática (XXXVI CLEI). 18. 2010. 1–15.

Peis Eduardo, Herrera-Viedma Enrique, Castillo José M Morales-del. Aproximación a la web semántica desde la perspectiva de la Documentación // Investigación bibliotecológica. 2007. 21, 43. 47–71.

Estudio Comparativo de Lenguajes para la Búsqueda y Recu-peración de Información Semántica. // . 2009.

Pohl Klaus, Böckle Günter, Linden Frank van der. Software Product Line Engineering. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. 467.

Quilty Gerard, Cinnéide Mel Ó. Experiences with software product line development in risk management software // 2011 15th International Software Product Line Conference. 2011. 251–260.

Rabiser Rick, Schmid Klaus, Becker Martin, Botterweck Goetz, Galster Matthias, Groher Iris, Weyns Danny. A study and comparison of industrial vs. academic software product line research published at SPLC // Proceedings of the 22nd International Systems and Software Product Line Conference-Volume 1. 2018. 14–24.

Raval Vishwas, Kumar Padam. SEReleC (Search Engine Result Refinement and Classification)-a Meta search engine based on combinatorial search and search keyword based link classification // IEEE-International Conference On Advances In Engineering, Science And Management (ICAESM-2012). 2012. 627–631.

Rogers Everett M. Diffusion of Innovations. 1983.

Silver Christina, Lewins Ann. Using software in qualitative research: A step-by-step guide. 2014.

Staples Mark, Niazi Mahmood. Systematic review of organizational motivations for adopting CMM-based SPI // Information and software technology. 2008. 50, 7-8. 605–620.

Vargas Connie. Tipos de pruebas funcionales de software testing • trycore. Oct 2020.

Vignando Henrique, Furtado Viviane R, Teixeira Lucas O, OliveiraJr Edson. OntoExper-SPL: An Ontology for Software Product Line Experiments. // ICEIS (2). 2020. 401–408.

Apéndice A

Anexos

A.1. Especificación de requerimientos

A.1.1. Requisitos funcionales Back-End

Especificación de Requisito	
Código Requisito	RF-001
Nombre	Listar Clases
Prioridad	Alta
Entrada	Modelo OWL
Salida	Lista de clases
Fuente Datos	Ontología
Destino Datos	Front-end
Restricciones	La salida debe estar en formato JSON
Descripción	Desarrollar un servicio REST tipo GET para consultar las clases disponibles en la ontología y retornarlas en formato JSON.
Proceso	Cargar modelo OWL al back-end y realizar la búsqueda de las clases sobre ese modelo.
Efecto Colateral	–

Especificación de Requisito	
Código Requisito	RF-002
Nombre	Listar relaciones de una clase
Prioridad	Alta
Entrada	Una clase en formato string
Salida	Lista Relaciones de una clase en formato JSON
Fuente Datos	Ontología
Destino Datos	Front-end
Restricciones	La clase debe ir en el BODY de la petición.
Descripción	Desarrollar un servicio REST tipo POST para consultar las relaciones de una clases en la ontología y retornarlas en formato JSON.
Proceso	N/A
Efecto Colateral	N/A

Especificación de Requisito	
Código Requisito	RF-003
Nombre	Obtener individuos de una clase
Prioridad	Alta
Entrada	Una clase en formato String
Salida	Lista de individuos en formato JSON
Fuente Datos	Ontología
Destino Datos	Front-end
Restricciones	Cada individuo debe traer toda la información relacionada a el. La clase debe viajar en el BODY de la petición
Descripción	Desarrollar un servicio REST tipo POST para consultar los individuos de una clases en la ontología y retornarlas en formato JSON.
Proceso	N/A
Efecto Colateral	N/A

Especificación de Requisito	
Código Requisito	RF-004
Nombre	Obtener individuos de una clase relacionada con otra clase
Prioridad	Alta
Entrada	Dos clases y una relación entre ellas en formato string
Salida	Lista de individuos en formato JSON correspondientes a los de la relación de la clases.
Fuente Datos	Ontología
Destino Datos	Front-end
Restricciones	Cada individuo debe traer toda la información relacionada a el. Las clases y su relación debe viajar en el BODY de la petición
Descripción	Desarrollar un servicio REST tipo POST para consultar los individuos de una clase relacionada con otra clase y retornarlas en formato JSON.
Proceso	N/A
Efecto Colateral	N/A

A.1.2. Requisitos funcionales Front-End

Especificación de Requisito	
Código Requisito	RF-005
Nombre	Entrada de busqueda
Prioridad	Alta
Entrada	Selección de usuario
Salida	Renderizar los individuos retornados en la card de resultados
Fuente Datos	Back-end
Destino Datos	Selector
Restricciones	RF-001 y RF-003
Descripción	Deben haber 3 selectores ubicados de forma horizontal para que el usuario seleccione CLASE - RELACIÓN - CLASE
Proceso	El usuario debe seleccionar una clase o categoría y seguidamente ejecutar el consumo al servicio RestFul correspondiente
Efecto Colateral	–

Especificación de Requisito	
Código Requisito	RF-006
Nombre	Card de resultados
Prioridad	Alta
Entrada	Archivo JSON
Salida	Renderizado en cards tipo stack
Fuente	Back-end
Destino	Front-end
Restricciones	RF-005
Descripción	La card debe contener información URL de donde se encontró la consulta, junto a las propiedades almacenadas en la ontología.
Proceso	–
Efecto Colateral	–

Especificación de Requisito	
Código Requisito	RF-007
Nombre	Card de gráficos
Prioridad	media
Entrada	Componente React
Salida	renderizar componente en la card
Fuente Datos	–
Destino Datos	Card de gráfico
Restricciones	Card responsive, se debe adaptar al gráfico de entrada.
Descripción	Deben haber mínimo dos gráficos en pantalla, máximo 6
Proceso	–
Efecto Colateral	–

Especificación de Requisito	
Código Requisito	RF-008
Nombre	Personalización de gráficos
Prioridad	media
Entrada	Componente React chart.js
Salida	Component React chart.js Personalizado
Fuente Datos	Back-end
Destino Datos	Card de gráfico
Restricciones	RF-004.
Descripción	Deben haber mínimo tres gráficos resaltando alguna propiedad de los datos de la ontología.
Proceso	–
Efecto Colateral	–

Especificación de Requisito	
Código Requisito	RF-009
Nombre	Página de Resultados
Prioridad	Alta
Entrada	Datos tipo JSON
Salida	Renderizado en componentes previamente desarrollados
Fuente Datos	BackEnd
Destino Datos	FrontEnd
Restricciones	RF-005, RF-006, RF-007, RF-008
Descripción	La página de resultados contendrá resultados enlaces los artículos y gráficos descriptivos
Proceso	–
Efecto Colateral	–