

ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

**DESARROLLO DE UN SISTEMA DE TOMA DE DECISIONES PARA
APOYO A LOS INVESTIGADORES ECUATORIANOS BASADO EN
MOTORES DE BÚSQUEDA Y RECOMENDACIÓN**

**AUTOMATIZACIÓN DEL PROCESO DE EXTRACCIÓN,
DEPURACIÓN Y ALMACENAMIENTO DE DATOS DE SCOPUS
MEDIANTE LA API DE ELSEVIER**

**TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO
REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO/A EN
SOFTWARE**

JHON FERNANDO SANGOPANTA NAULA

jhon.sangopanta@epn.edu.ec

DIRECTOR: PH.D. LORENA KATHERINE RECALDE CERDA

lorena.recalde@epn.edu.ec

DQM, 25 de julio de 2024

CERTIFICACIONES

Yo, Jhon Fernando Sangopanta Naula declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

Jhon Fernando Sangopanta Naula

Certifico que el presente trabajo de integración curricular fue desarrollado por Jhon Fernando Sangopanta Naula, bajo mi supervisión.

Lorena Katherine Recalde Cerdá
DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

NOMBRE ESTUDIANTE

NOMBRE DIRECTOR

NOMBRE COLABORADOR(ES)

DEDICATORIA

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

AGRADECIMIENTO

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Índice general

1. INTRODUCCIÓN	1
1.1. Planteamiento del problema	1
1.2. Justificación teórica	1
1.3. Justificación metodológica	2
1.4. Objetivos	2
1.4.1. Objetivo General	2
1.4.2. Objetivos específicos	2
1.5. Alcance	2
1.6. Marco Teórico	2
1.6.1. Scopus	2
1.6.2. Redes de Co-Autoría	3
1.6.3. Bases de Datos Orientadas a Grafos	3
1.6.4. TF-IDF	4
1.6.5. SCRUM	5
1.6.6. Arquitectura Hexagonal	6
1.6.7. Herramientas Utilizadas	7
2. METODOLOGÍA	8
2.1. Sprint 0	8
2.2. Sprint 1	12
2.2.1. Introducción	12
2.2.2. Objetivos del Sprint	12
2.2.3. Planificación	13
2.2.4. Implementación	16
2.2.5. Revisión y Retrospectiva	22
2.3. Sprint 2	23

2.3.1. Introducción	23
2.3.2. Objetivos	24
2.3.3. Planificación	24
2.3.4. Implementación	26
2.3.5. Revisión y Retrospectiva	30
2.4. Sprint 3	31
2.4.1. Introducción	31
2.4.2. Objetivos	31
2.4.3. Planificación	31
2.4.4. Implementación	33
2.4.5. Revisión y Retrospectiva	42
2.5. Sprint 4	43
2.5.1. Introducción	43
2.5.2. Objetivos	43
2.5.3. Planificación	43
2.5.4. Implementación	44
2.5.5. Revisión y Retrospectiva	48
3. RESULTADOS, CONCLUSIONES Y RECOMENDACIONES	49
3.1. Resultados	49
3.1.1. Prueba con usuarios finales	49
3.2. Conclusiones	50
3.3. Recomendaciones	51
4. REFERENCIAS BIBLIOGRÁFICAS	52
A. Apendice A	53
B. Apendice B	55

Índice de figuras

2.1. Proyecto en Azure DevOps	10
2.2. Arquitectura Hexagonal aplicada en el modulo del motor de búsqueda	11
2.3. Arquitectura hexagonal aplicada al modulo de consenso	12
2.4. Criterios de aceptación de la historia de usuario HU-SE-01	15
2.5. Criterios de aceptación de la historia de usuario HU-SE-02	15
2.6. Planificación de tareas del Sprint 1	16
2.7. Mockup de la página de inicio	17
2.8. Estructura de carpetas de la página de inicio	18
2.9. Página de inicio	19
2.10. Mockup de la página de resultados	20
2.11. Mockup de la página de detalle del artículo	21
2.12. Routing de la página de detalle de artículo	22
2.13. Página de detalle de artículo	22
2.14. Criterios de Aceptación HU-SE-03	25
2.15. Criterios de Aceptación HU-SE-04	25
2.16. Planificación de tareas del sprint 2	26
2.17. Mockup de los detalles de un autor	26
2.18. Mockup de los artículos publicados por un autor	27
2.19. Interfaz en Typescript para la creación de la red de coautoría	28
2.20. Red de coautoría	29
2.21. Tabla de artículos publicados por un autor	30
2.22. Criterios de aceptación HU-SE-02	33
2.23. Criterios de aceptación HU-SE-05	33
2.24. Criterios de aceptación HU-SE-06	33
2.25. Archivo Dockerfile	34
2.26. Archivo Docker Compose	34

2.27. Sialida del comando docker ps	35
2.28. Aplicaciones de Django	35
2.29. Modelo de la base de datos	36
2.30. Capa de dominio de la aplicación	36
2.31. Modelo Author implementado con Neomodel	37
2.32. Repositorio AuthorRepository	38
2.33. Servicio AuthorService	39
2.34. Caso de uso FindAuthorsByQuery	40
2.35. Vista AuthorViews	40
2.36. Documentación de la API con Swagger	41
2.37. Diagrama de secuencia de la funcionalidad de búsqueda de autores	42
2.38. Base de datos Neo4j	42
2.39. Diagrama de secuencia para la generación del Corpus	45
2.40. Diagrama de secuencia para la generación del Modelo de TF-IDF	46
2.41. Diagrama de secuencia para la obtención de los Artículos Relevantes	47
2.42. Diagrama de secuencia para la obtención de la red de coautoría de un autor	47
2.43. Diagrama de secuencia para la obtención de los Artículos en los que ha colaborado un autor	48

Índice de Tablas

1.1. Herramientas utilizadas	7
2.1. Historias de Usuario del sprint 1	14
2.2. Historias de Usuario del sprint 2	24
2.3. Historias de Usuario del sprint 3	32
2.4. Historias de Usuario del sprint 4	44
3.1. Cuestionario SUS	50

RESUMEN

(Máximo 250 palabras. Colocar al final palabras clave, hasta seis)

PALABRAS CLAVE - español, test 1

ABSTRACT

(Resumen en inglés. Máximo 250 palabras. Colocar al final Keywords)

KEYWORDS - english, test 2

Capítulo 1

INTRODUCCIÓN

1.1. Planteamiento del problema

1.2. Justificación teórica

En respuesta al aumento de la producción científica y a la creciente necesidad de colaboración entre expertos, se desarrolló la aplicación web ResNet. Esta herramienta utiliza modelos de minería de datos para buscar y mostrar redes de investigadores vinculados a instituciones ecuatorianas y sus áreas académicas.

El objetivo principal de la herramienta es presentar de manera visual las redes de coautoría entre investigadores ecuatorianos, utilizando datos extraídos de Scopus.

Asimismo, ResNet facilita la búsqueda a través de diversos filtros, permitiendo a los usuarios afinar sus consultas de manera eficiente y personalizada.

En la actualidad, la aplicación ResNet se encuentra restringida a la extracción manual de datos desde Scopus, lo que ha evidenciado una necesidad apremiante de automatizar este proceso. Esta necesidad se aborda mediante la implementación de la metodología CRISP-DM (Cross-Industry Standard Process for Data Mining), que proporciona directrices sólidas para la automatización eficiente de la extracción y preprocesamiento de datos, contribuyendo así a mejorar la eficacia y la velocidad del sistema. Este enfoque permite optimizar la recopilación de información desde Scopus, agilizando el flujo de trabajo y proporcionando resultados más precisos y oportunos en la construcción de redes de investigadores y coautoría.

1.3. Justificación metodológica

1.4. Objetivos

1.4.1. Objetivo General

Integrar en una única plataforma los sistemas informáticos (independientes) Resnet y Research Decide, desarrollados previamente. Además, se busca incorporar un módulo de administrador para la gestión eficiente del sistema, permitiendo, de forma automática, la extracción y limpieza de los datos obtenidos desde Scopus.

1.4.2. Objetivos específicos

1. Analizar y comprender a detalle los sistemas Resnet y ReSerchDecide para identificar sus funcionalidades, requerimientos y posibles áreas de mejora.
2. Diseñar e implementar una arquitectura de integración que permita la interoperabilidad entre los diferentes sistemas, garantizando una coherencia de datos y compatibilidad de las plataformas.
3. Implementar la automatización del proceso de extracción, limpieza y almacenamiento de datos de Scopus con el fin de garantizar la actualización constante del sistema.

1.5. Alcance

Describir el alcance del componente de acuerdo a lo establecido en el Plan.

1.6. Marco Teórico

1.6.1. Scopus

Scopus es una destacada base de datos bibliográfica y herramienta de indexación científica ampliamente utilizada en el ámbito académico y de la investigación. Desarrollada por la editorial Elsevier, Scopus abarca diversas disciplinas académicas, incluyendo ciencias sociales, ciencias de la salud, ingeniería, ciencias naturales y ciencias exactas.

Esta plataforma proporciona un extenso repositorio de información, que incluye resúmenes y citas de artículos científicos publicados en revistas revisadas por expertos, conferencias y libros. Su alcance internacional abarca publicaciones de diversas partes del mundo, lo que la convierte en una herramienta valiosa para investigadores que buscan acceder a una amplia gama de literatura científica.

Scopus no solo indexa la información, sino que también ofrece herramientas analíticas que permiten evaluar el impacto de las publicaciones y la productividad de los investigadores. Asimismo, su capacidad para rastrear las citas entre artículos facilita el seguimiento del desarrollo de las investigaciones y la identificación de tendencias en diversas áreas del conocimiento.

1.6.2. Redes de Co-Autoría

Las redes de coautoría se construyen a partir de las publicaciones científicas, donde cada autor se representa como un nodo en la red y las colaboraciones entre autores se representan como enlaces entre los nodos correspondientes. Estas redes proporcionan una representación visual y matemática de las relaciones de colaboración en la comunidad científica. Al analizar estas redes, es posible identificar la estructura de la colaboración, encontrar comunidades de autores que colaboran estrechamente, identificar autores influyentes y recomendar nuevas colaboraciones potenciales.

Además, las redes de coautoría se utilizan para estudiar tendencias en la producción científica, identificar áreas de investigación interconectadas y evaluar la difusión del conocimiento en diferentes campos. En resumen, estas redes proporcionan una herramienta poderosa para comprender la dinámica de la colaboración científica y para facilitar la identificación de oportunidades de colaboración entre investigadores.

1.6.3. Bases de Datos Orientadas a Grafos

Las bases de datos orientadas a grafos son sistemas de gestión de bases de datos diseñados específicamente para almacenar, recuperar y gestionar datos que se pueden representar como grafos. En lugar de utilizar el modelo de datos relacional tradicional, las bases de datos orientadas a grafos utilizan estructuras de datos de grafo para representar y almacenar la información.

En un grafo, los datos se modelan como nodos (también conocidos como vértices) que están conectados por relaciones (también conocidas como aristas). Cada nodo y relación

pueden contener propiedades que describen los atributos de los datos. Este enfoque es especialmente útil para modelar y consultar datos que tienen relaciones complejas y que requieren un análisis de redes.

Las bases de datos orientadas a grafos son ampliamente utilizadas en una variedad de aplicaciones, incluyendo redes sociales, recomendaciones personalizadas, análisis de redes, sistemas de recomendación, bioinformática, gestión del conocimiento y análisis de datos interconectados. Algunas bases de datos orientadas a grafos populares incluyen Neo4j, Amazon Neptune, y JanusGraph, entre otras.

1.6.4. TF-IDF

TF-IDF (Term Frequency-Inverse Document Frequency) es una técnica de procesamiento de lenguaje natural que se utiliza para evaluar la relevancia de un término en un documento o corpus de documentos. Esta técnica se utiliza comúnmente en la recuperación de información y la minería de texto.

La técnica TF-IDF se basa en dos conceptos principales: la frecuencia del término (TF) y la frecuencia inversa del documento (IDF). La frecuencia del término se refiere al número de veces que un término aparece en un documento, mientras que la frecuencia inversa del documento se refiere a la frecuencia con la que aparece un término en todo el corpus de documentos.

La fórmula para calcular el valor TF-IDF de un término es:

$$\text{TF-IDF} = \text{TF} \cdot \log\left(\frac{N}{DF}\right)$$

Donde TF es la frecuencia del término en el documento, N es el número total de documentos en el corpus y DF es el número de documentos que contienen el término.

El valor TF-IDF de un término es alto si aparece con frecuencia en un documento específico, pero rara vez aparece en otros documentos del corpus. Esto indica que el término es importante y relevante para el documento en cuestión.

La técnica TF-IDF se utiliza comúnmente en la recuperación de información para clasificar y ordenar documentos según su relevancia para una consulta de búsqueda específica. También se utiliza en la minería de texto para identificar patrones y tendencias en grandes conjuntos de datos de texto.

1.6.5. SCRUM

Al iniciar un proyecto, es crucial que el equipo involucrado esté al tanto de sus roles, actividades y los plazos para la entrega de resultados. Scrum es un marco de trabajo que facilita este proceso, permitiendo además acelerar la entrega de valor al cliente a través de iteraciones cortas **SCRUM**.

En 2001, un grupo de desarrolladores en Salt Lake City creó el Manifiesto Ágil, que se resume en los siguientes cuatro valores:

1. Valorar a los individuos y su interacción por encima de los procesos y herramientas.
2. Valorar el software que funciona por encima de la documentación exhaustiva.
3. Valorar la colaboración con el cliente por encima de la negociación contractual.
4. Valorar la respuesta al cambio por encima del seguimiento de un plan.

El marco de trabajo SCRUM, dentro del contexto del Manifiesto Ágil, se fundamenta en aspectos como la flexibilidad, el factor humano, la colaboración y el desarrollo iterativo, con el objetivo principal de asegurar buenos resultados **SCRUM**. Scrum posee herramientas que permiten resolver problemas y mejorar la administración del proyecto, los cuales son:

- **Product Backlog:** Lista dinámica de características, requisitos, arreglos y mejoras que se deben completar.
- **Sprint Backlog:** Lista de elementos que el equipo de desarrollo debe completar durante el tiempo definido del sprint **SCRUM-Sprints**.
- **Incremento:** Entregable que surge al final del sprint.

Scrum define tres roles principales: Product Owner, Scrum Master y Team. Estos roles son esenciales en cada equipo, que se caracterizan por ser auto-organizados y multifuncionales **SCRUM-Roles**.

- **Product Owner:** Encargado de maximizar el valor del trabajo y el perfil que habla con el cliente.
- **Scrum Master:** Responsable de que las técnicas y eventos de scrum sean aplicadas.

- **Team:** Es un grupo de profesionales que trabajan juntos para entregar un incremento de producto terminado al final de cada sprint. El equipo de desarrollo es auto-organizado y multidisciplinario, lo que significa que posee todas las habilidades necesarias para crear el producto sin depender de personas externas al equipo.

1.6.6. Arquitectura Hexagonal

La arquitectura Hexagonal o también conocida como la arquitectura de puertos y adaptadores, es un patrón de arquitectura de software que nos ayuda a tener un código mantible, escalable y testable. Su característica principal es separar la lógica del dominio y que la misma no se acople a nada externo como frameworks, bases de datos, etc.

La arquitectura hexagonal se divide de 3 capas:

1. **Infraestructura:** También conocida como puertos es la capa que actúa como el punto de entrada hacia la aplicación. La misma que permite la comunicación desde y hacia el exterior.
2. **Aplicación:** También conocida como adaptadores, es el puente entre la capa de dominio y la capa de infraestructura. Esta capa sirve para transformar la comunicación entre actores externos (Capa de Infraestructura) y la lógica de la aplicación (Capa de Dominio), consiguiendo así que ambas capas sean independientes.
3. **Dominio:** Esta capa es el core de la aplicación, la cual va a contener toda la información y lógica del negocio.

Estas capas vienen acompañadas de una regla de dependencia que va desde afuera hacia adentro, teniendo en cuenta que la capa más interna es la del dominio y la más externa es la infraestructura. La regla de dependencia es muy simple.

1.6.7. Herramientas Utilizadas

Nombre	Descripción	Logo
Pycharm	IDE de desarrollo de Python para web y ciencia de datos PycharmIDE .	 The logo for PyCharm, featuring a black rounded square with the letters 'PC' in white, with a horizontal bar below it and a colorful gradient border.
Django Rest	Conjunto de herramientas para el desarrollo de APIs DRFweb .	 The logo for Django REST framework, showing the word 'django' in small blue letters above 'REST' in large red letters, with 'framework' in smaller blue letters below it.

Tabla 1.1: Herramientas utilizadas

Capítulo 2

METODOLOGÍA

Para el desarrollo del proyecto se hizo uso de SCRUM **SCRUM** como marco de trabajo a través de 4 sprints **SCRUM-Sprints** (Tomando en cuenta el Sprint 0). Cada subsección representa un Sprint en concreto.

2.1. Sprint 0

En este sprint denominado el "*Sprint 0*", se definieron aspectos fundamentales para el éxito del proyecto, con el objetivo de preparar al equipo y el entorno para los sprints de desarrollo regulares.

- Se realizó un análisis exhaustivo del stack tecnológico que se usará para el desarrollo, tanto del lado del frontend como del backend, abordando cada uno de los dos backends (ResNet Y ResearchDecide) y los cuatro componentes diferentes del sistema, los cuales son:
 - Automatización del proceso de extracción, depuración y almacenamiento de datos de Scopus mediante la API de Elsevier.
 - Desarrollo de módulo que implemente funciones del tipo red social para los usuarios del sistema.
 - Implementación de estrategia(s) de mediación o facilitación del consenso (sistema como mediador para el grupo de investigadores).
 - Desarrollo del módulo de Analítica: Dashboards y estadísticas para la toma de decisiones.

El propósito de este análisis es garantizar que tanto el frontend como el backend utilicen las tecnologías más apropiadas para satisfacer los requisitos técnicos y de negocio; y así, asegurar, y para así asegurar una integración efectiva y eficiente entre todas las partes del sistema.

En consecuencia, se tomaron las siguientes decisiones:

1. Para la integración del sistema en el lado del frontend se optó por utilizar Angular como framework de desarrollo, debido a que se puede reutilizar componentes de el sistema ResNet **RESNET**.
2. Para el lado del backend se propuso utilizar Django como framework. Con Django, se espera aprovechar su robustez, seguridad y su capacidad para desarrollar rápidamente aplicaciones web escalables y de alto rendimiento. Además, Django ofrece una amplia gama de características integradas, como autenticación de usuarios, administración de bases de datos, y un ORM¹ potente, lo que lo hace una elección sólida para el desarrollo del backend del sistema.
3. Para mantener las mismas dependencias de desarrollo y simplificar la gestión del entorno, se utilizará Docker **DOCKER**. Esto permite encapsular aplicaciones y sus dependencias en contenedores, asegurando consistencia entre entornos.
4. Además, Docker Compose se empleará para orquestar el servidor backend y la base de datos, facilitando la configuración y gestión del entorno, y simplificando el despliegue del sistema en diferentes entornos.
5. La elección de Neo4j se basa en la decisión de no alterar el modelo de datos actual. Dado que Neo4j está especialmente diseñado para modelos de datos conectados y relacionales, su adopción evita la necesidad de reestructurar el modelo existente, ahorrando tiempo y recursos mientras garantiza la continuidad del desarrollo del sistema.
6. El control de versiones se hará mediante Git y para alojar dichas versiones se utilizará GitHub. Mediante una organización en donde se alojará el código fuente de los 4 componentes.
7. Para la gestión del proyecto, se utilizó SCRUM en conjunto con Azure DevOps **AZURE-DEVOPS**. SCRUM proporciona un marco ágil para la planificación y eje-

¹ORM (Object-Relational Mapping) es una herramienta que facilita la comunicación y traducción de datos entre bases de datos relacionales y objetos en programación orientada a objetos.

cución de proyectos, mientras que Azure DevOps **AZURE-DEVOPS** ofrece herramientas integradas para facilitar todas las etapas del ciclo de vida del desarrollo de software. Esta combinación garantiza una gestión eficiente y colaborativa del proyecto, mejorando la transparencia y la entrega oportuna de resultados. En la Figura 2.1 se muestra el proyecto en Azure DevOps.

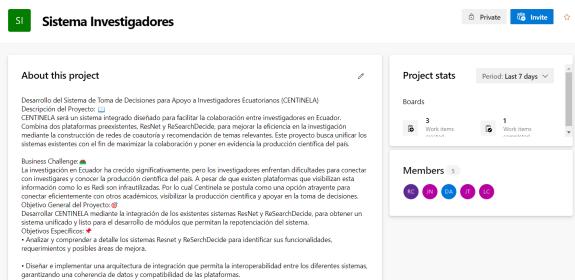


Figura 2.1: Proyecto en Azure DevOps

8. Se optó por Visual Studio Code como editor de código para el frontend y PyCharm como IDE de desarrollo especializado para el backend. Esta elección proporcionó un entorno flexible y eficaz para la colaboración en el proyecto.
- Se establecieron horarios para las reuniones diarias, las cuales se llevarán a cabo los días lunes, martes, miércoles y jueves en el horario de 14:00 a 16:00 en un espacio proporcionado por la directora del proyecto, Lorena Recalde, en el edificio de Sistemas-EPN. Durante este intervalo, se realizaron los daily meetings y se llevó a cabo el trabajo en equipo.
 - Se decidió adoptar el inglés como el idioma oficial para la codificación, los commits y toda la documentación técnica dentro de nuestro equipo. Esta medida busca estandarizar nuestras prácticas con los estándares internacionales, facilitar la colaboración con otros equipos y mejorar la accesibilidad a recursos técnicos de alta calidad.
 - Se decidió utilizar Arquitectura Hexagonal **ARQUITECTURA-HEXAGONAL** como patrón de arquitectura, definiendo así una estructura de carpetas y respetando los principios de esta arquitectura. Cabe destacar que no se ha implementado arquitectura hexagonal en su totalidad, ya que se optó por una adaptación de la misma, obteniendo así una arquitectura de dos capas.
 - **Backend:** Para el backend, en cuanto a la estructura de carpetas, se definió que por cada módulo en general se tendrá las tres capas de arquitectura como son

infraestructura, aplicación y dominio como se muestra en la Figura 2.2.

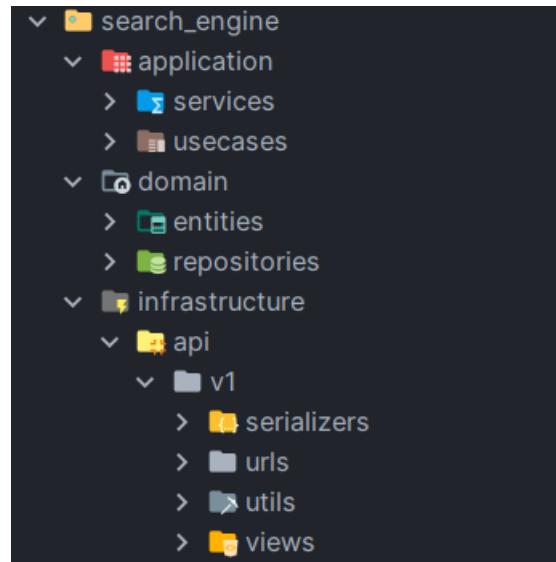


Figura 2.2: Arquitectura Hexagonal aplicada en el modulo del motor de búsqueda

- **Frontend:** Basándonos en el concepto de puertos y adaptadores, en el frontend también hemos optado por implementar la Arquitectura Hexagonal **ARQUITECTURA-HEXAGONAL** como patrón de arquitectura, con algunas adaptaciones específicas para este contexto. Dado que el frontend se enfoca en la parte gráfica, el adaptador se denominará "presentación", donde se gestionará toda la interfaz gráfica. Esta capa de presentación será el punto de entrada hacia el núcleo (core) de la aplicación frontend, facilitando así una separación clara entre la lógica de negocio y la interfaz de usuario, y promoviendo una estructura modular y fácilmente mantenible (Figura 2.3.).

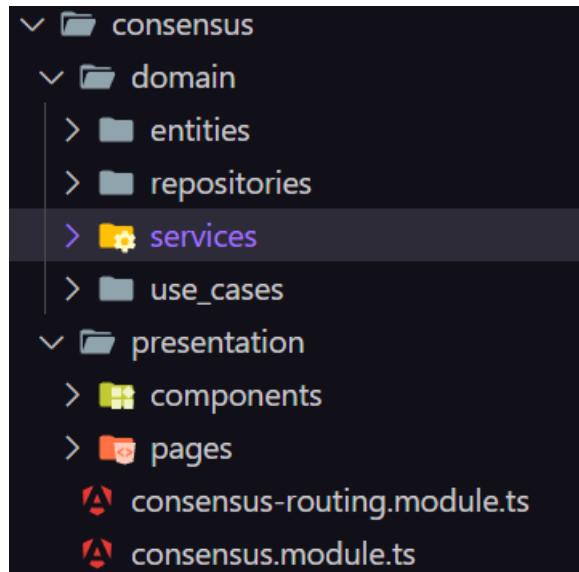


Figura 2.3: Arquitectura hexagonal aplicada al modulo de consenso

2.2. Sprint 1

2.2.1. Introducción

Durante este sprint, nuestro objetivo principal es crear diseños detallados que integren las funcionalidades de las aplicaciones Resnet (ver Sección ??) y Research Decide en una plataforma unificada. Además, nos centraremos en identificar y proponer mejoras significativas para esta aplicación combinada. Utilizaremos Figma para desarrollar los mockups, aplicando los patrones de diseño más adecuados para asegurar una experiencia óptima para el usuario final. Es importante destacar que esta fase será iterativa, permitiendo ajustes y cambios continuos durante todo el desarrollo, por lo que la propuesta final no será definitiva.

2.2.2. Objetivos del Sprint

- Diseñar el flujo de navegación de la aplicación combinada.
- Crear mockups de las interfaces de usuario de la aplicación.
- Desarrollar la estructura principal del motor de búsqueda.
- Implementar el enrutamiento de la aplicación.
- Implementar la página de inicio de la aplicación.

2.2.3. Planificación

Para este sprint, hemos seleccionado las historias de usuario con su respectivo código enfocadas principalmente en la integración gráfica de las plataformas mencionadas en la Sección ??, que se muestran en la Tabla 2.1. Se debe tomar en cuenta que esta parte del desarrollo se enfoca en los usuarios que no necesariamente están registrados en la plataforma, por lo que se ha priorizado la visualización de información relevante para estos usuarios.

Identificador	Historia de Usuario	Tareas
HU-SE-01	Como usuario no registrado deseo poder encontrar artículos relevantes dado un tema de investigación para poder acceder rápidamente a información útil y actualizada que apoye mi estudio o trabajo	<ul style="list-style-type: none"> ■ Componente para listar artículos ■ Paginación para extraer resultados limitados ■ Componente para filtrar la información por años ■ Enrutamiento para redirigir hacia la página del artículo ■ Página para visualizar información general del artículo
HU-SE-02	Como usuario no registrado deseo poder ver los investigadores que tengan colaboraciones en artículos con afiliaciones ecuatorianas para mantenerme informado sobre sus investigaciones y campos de estudio	<ul style="list-style-type: none"> ■ Interfaz para poder seleccionar distintos tipos de búsqueda ■ Componente para visualizar lista de autores que cumplan con los criterios de búsqueda ■ Página para visualizar información general del autor ■ Componente de paginación para extraer resultados limitados ■ Enrutamiento para redirigir hacia la página del autor

Tabla 2.1: Historias de Usuario del sprint 1

Todas las historias de usuario se han dividido en tareas más pequeñas y manejables. También en las Figuras 2.4 y 2.22 se muestran los criterios de aceptación de cada historia de usuario, que se utilizarán para verificar si la historia de usuario se ha completado correctamente.

Acceptance Criteria



Visualización de Investigadores:

1. El usuario debe poder acceder a una lista de investigadores que han colaborado en artículos con afiliaciones ecuatorianas.
2. La lista debe incluir al menos el nombre del investigador, su afiliación principal.

Información de Colaboraciones:

1. Cada investigador en la lista debe mostrar claramente el número de colaboraciones en artículos con afiliaciones ecuatorianas.
2. Debe ser posible ver los detalles de las colaboraciones, incluyendo el título del artículo y la fecha de publicación.

Filtros y Búsqueda:

1. El usuario debe poder filtrar la lista de investigadores por área de estudio, institución o universidad, y año de publicación.
2. El usuario tener la opción de buscar investigadores por nombre o por palabras clave relacionadas con su campo de estudio.

Acceso a Información de Artículos:

1. Debo poder hacer clic en un investigador para ver una lista detallada de los artículos en los que ha colaborado con afiliaciones ecuatorianas.
2. Cada artículo debe mostrar información relevante como resumen, coautores, y enlace a la publicación completa si está disponible.

Figura 2.4: Criterios de aceptación de la historia de usuario HU-SE-01

Acceptance Criteria



Resultados de Búsqueda:

1. Al ingresar un tema de investigación, debo recibir una lista de artículos relevantes.
2. Los resultados deben ser presentados de manera clara, mostrando al menos el título del artículo, autores principales y fecha de publicación

Filtros y Ordenación:

1. Debo poder aplicar filtros para refinar los resultados de búsqueda por criterios como fecha de publicación, tipo de artículo (revista, conferencia, etc.), y relevancia.
2. Debo poder ordenar los resultados por relevancia, fecha de publicación, o número de citas.

Acceso a Detalles del Artículo:

1. Al hacer clic en un resultado de búsqueda, debo poder ver detalles completos del artículo, incluyendo el resumen completo, la lista completa de autores, afiliaciones, y enlaces a la publicación original o al texto completo si está disponible.

Figura 2.5: Criterios de aceptación de la historia de usuario HU-SE-02

Estas historias de usuario se han priorizado en función de su importancia y complejidad, y se han asignado a los miembros del equipo de desarrollo. La Figura 2.6 muestra las tareas asignadas a cada miembro del equipo.

Title	State	Assigned To	F
✓ Como usuario no registrado deseo poder encontrar artículo... ● New			
✓ Componente para listar Articulos	● To Do	JHON FERNAN...	
✓ Paginación para extraer resultados limitados	● To Do	JHON FERNAN...	
✓ Componente para filtrar la información por años	● To Do	JOFFRE ALEXA...	
✓ Enrutamiento para redirigir hacia la página del articulo	● To Do	JOFFRE ALEXA...	
✓ Pagina para visualizar la información general de un articulo...	● To Do	JHON FERNAN...	
✓ Como usuario no registrado deseo poder ver los investigadores... ● Done			
✓ Desarrollar la interfaz para poder seleccionar distintos tipos...	● To Do	JHON FERNAN...	
✓ Componente para poder visualizar la lista de autores que...	● To Do	JOFFRE ALEXA...	
✓ Pagina para el perfil de Investigador	● To Do	DANNY RUBEN...	
✓ Restricción para usuarios no registrados	● To Do	DANNY RUBEN...	
✓ Componente de paginación para extraer resultados limitados	● To Do	JHON FERNAN...	
✓ Enrutamiento para dirigirse a la pagina de un autor seleccionado	● To Do	JOFFRE ALEXA...	

Figura 2.6: Planificación de tareas del Sprint 1

2.2.4. Implementación

Para la implementación de las historias de usuario, se ha utilizado la herramienta de diseño Figma, que permite crear prototipos de alta y baja fidelidad. A continuación, se presentan los mockups de las interfaces de usuario de la aplicación que se han desarrollado durante este sprint. Cabe destacar que se siguió un proceso iterativo, por lo que los mockups presentados no son definitivos y pueden sufrir cambios en futuras iteraciones. Además el diseño se basó en el patrón de diseño Mobile First **MOBILE-FIRST**, que consiste en diseñar primero la versión móvil de la aplicación y luego adaptarla a dispositivos de mayor tamaño.

Página de inicio

La página de inicio de la aplicación es la primera pantalla que verá el usuario al ingresar a la plataforma. En esta pantalla, se mostrarán las opciones de búsqueda y un conjunto de datos sobre la información que tiene Centinela. Así como también una barra de navegación que permitirá al usuario acceder a otras secciones de la aplicación. La figura 2.7 muestra el diseño de la página de inicio.



Figura 2.7: Mockup de la página de inicio

Como se mencionó en la sección 2.1 se utilizará una adaptación de arquitectura hexagonal para desarrollar la estructura de la aplicación. Dando como resultado la estructura de carpetas que se muestra en la figura 2.8. Puesto que la página de inicio es la primera pantalla que verá el usuario al ingresar a la plataforma, se ha decidido crear una carpeta específica para esta sección. También debido a que angular es un framework que se maneja con componentes, se ha creado un componente para la página de inicio, el cual se encuentra en la carpeta de pages. A su vez se ha creado el archivo de tipo module para la

página de inicio, el mismo nos permitirá importar los componentes necesarios para la página de inicio, sin tener que importarlos en el archivo principal de la aplicación. Este enfoque permitirá que el modulo de la página de inicio sea independiente del resto de la aplicación, que juntado con la arquitectura hexagonal facilitara la escalabilidad y mantenimiento de la aplicación.

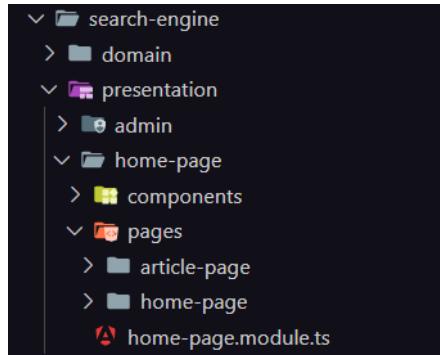


Figura 2.8: Estructura de carpetas de la página de inicio

Resultado de la implementación de la página de inicio se muestra en la figura 2.9. En la que se muestra la barra de navegación, el formulario de búsqueda y la información que tiene Centinela. El formulario de búsqueda cuenta con 3 opciones, la primera es para buscar por palabra clave a un autor, la segunda es para buscar autores relevantes dado un tópico de investigación y la tercera es para buscar artículos relevantes dado un tópico. Bajo el motor de búsqueda se muestra el resumen del numero de artículos, autores y tópicos que tiene registrado Centinela. Finalmente contiene un mapa que mostrará en que el número de artículos que se han publicado en cada provincia del Ecuador.

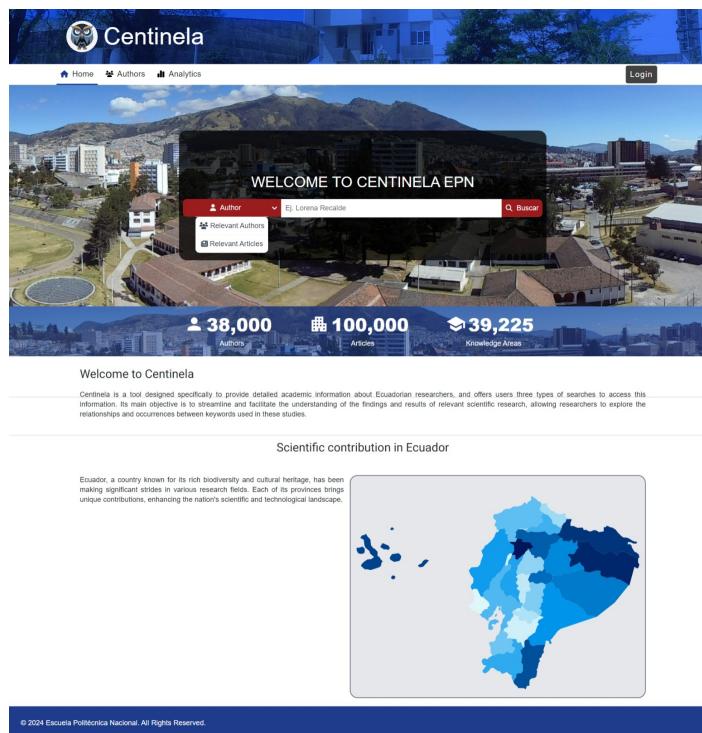


Figura 2.9: Página de inicio

Página de resultados

La página de resultados es la pantalla que se mostrará al usuario después de realizar una búsqueda en la aplicación. En esta pantalla, se mostrarán los resultados de la búsqueda, que incluirán una lista de autores y artículos relevantes, dependiendo del tipo de búsqueda que haya realizado el usuario. La figura 2.10 muestra el diseño de la página de resultados. Cabe destacar que esta página va a ser dinámica, es decir se modificará en función de los resultados de la búsqueda. También contiene el componente de paginación, que permitirá al usuario navegar entre los resultados de la búsqueda. Este último está disponible para el tipo de búsqueda de autores por palabra clave y para el tipo de búsqueda de artículos relevantes por tópico, ya que estos tipos de búsqueda pueden devolver un gran número de resultados. Y al ser una aplicación web, se ha optado por mostrar 10 resultados por página para garantizar una buena experiencia de usuario y reducir tiempos de carga.



Figura 2.10: Mockup de la página de resultados

Como se menciono en la Tabla 2.4 y 2.22 los criterios de aceptación de las historias de usuario HU-SE-01 y HU-SE-02 cada resultado tendrá que ser redirigido a una pantalla en donde se visualizará la información detallada del autor o del articulo. Para este caso en específico se muestra el diseño para la pagina que tendrá la información detallada de un autor, en la figura 2.11.

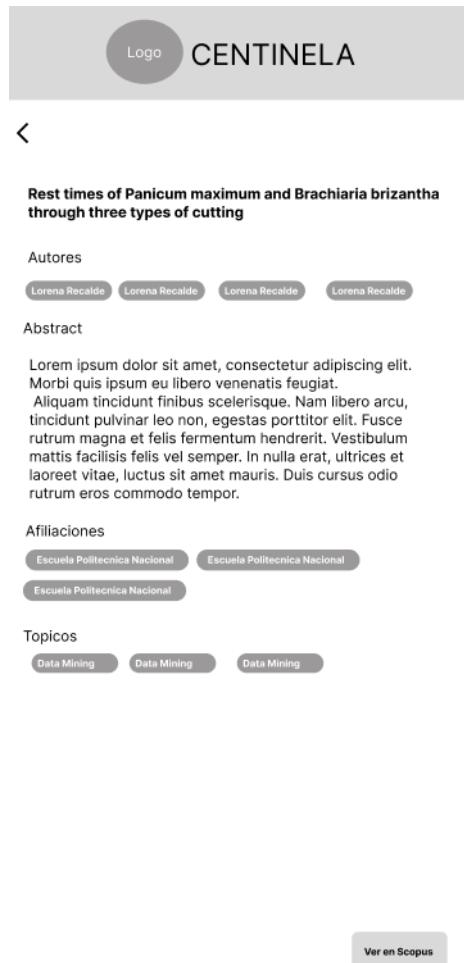


Figura 2.11: Mockup de la página de detalle del artículo

Para el enrutamiento de la aplicación se ha utilizado el modulo de enrutamiento de Angular, el cual nos permite definir las rutas de la aplicación y asociarlas con los componentes correspondientes. En la Figura 2.12 se muestra el enrutamiento hacia el componente que contendrá la información general de un articulo. El mismo que como parámetro recibirá el id del articulo que se desea visualizar.

```

    children: [
      {
        path: '',
        component: SearchResultComponent,
        // El componente app-author-retrieve se mostrará solo en la página principal
      },
      {
        path: 'article/:scopusId',
        component: ArticlePageComponent
      },
    ],
  }

```

Figura 2.12: Routing de la página de detalle de articulo

Resultado de la implementación de la página de información general de un artículo se muestra en la figura 2.13. En la que se muestra la información general de un artículo, como el título, resumen, autores, afiliaciones, palabras clave y el año de publicación. Así como también un botón que permitirá al usuario acceder al artículo completo en la página de Scopus.



Figura 2.13: Página de detalle de artículo

2.2.5. Revisión y Retrospectiva

Durante este sprint, hemos logrado completar todas las historias de usuario planificadas, lo que nos ha permitido avanzar significativamente en el desarrollo de la aplicación. Hemos creado los mockups de las interfaces de usuario de la aplicación, desarrollado la estructura principal del motor de búsqueda, implementando el enrutamiento de la aplicación y la página de inicio. Además, hemos implementado la página de resultados y la página de información general de un artículo. En general, el equipo ha trabajado de manera eficiente.

ciente y colaborativa, lo que ha permitido cumplir con los objetivos del sprint en el tiempo previsto. Sin embargo, hemos identificado algunas áreas de mejora que podrían ayudarnos a optimizar nuestro trabajo en futuros sprints.

- **Comunicación:** Aunque hemos mantenido una comunicación constante a través de reuniones diarias y canales de mensajería, es importante mejorar la comunicación entre los miembros del equipo para garantizar que todos estén al tanto de los avances del proyecto.
- **Planificación:** Aunque hemos logrado completar todas las historias de usuario planificadas, es importante revisar y ajustar nuestra planificación para futuros sprints, teniendo en cuenta la complejidad y el tiempo requerido para cada tarea.
- **Colaboración:** Hemos trabajado de manera colaborativa y eficiente durante este sprint, lo que ha permitido cumplir con los objetivos del proyecto. Es importante seguir fomentando la colaboración entre los miembros del equipo para garantizar el éxito del proyecto.
- **Retroalimentación:** Es importante recopilar y analizar la retroalimentación de los usuarios para identificar áreas de mejora y realizar ajustes en futuras iteraciones del proyecto.

Cabe destacar que las tareas referentes a la integración de la aplicación con la API de Scopus, así como la implementación de funcionalidades del lado del servidor, se han dejado para futuros sprints, ya que requieren un mayor tiempo de desarrollo y dependen de la finalización de otras tareas.

2.3. Sprint 2

2.3.1. Introducción

Durante el segundo sprint se continuó con el desarrollo de la integración de las aplicaciones mencionadas en el sprint anterior. Para este sprint en específico se trabajó en los detalles sobre los autores, así como su red de coautoría, artículos publicados y tópicos de interés.

2.3.2. Objetivos

- Implementar las ventanas para cada uno de los detalles de los autores.
- Integrar la fuerza de colaboración entre autores.
- Desarrollar la funcionalidad de artículos publicados por autor.
- Enrutamiento para las diferentes vistas de los autores.

2.3.3. Planificación

Para este sprint se selecciono un total de 2 historias de usuario, divididas en un total de 12 tareas a realizar las cuales fueron asignadas a los integrantes del equipo de desarrollo.

En la Tabla 2.2 se detallan las historias de usuario seleccionadas para este sprint.

Identificador	Historia de Usuario	Tareas
HU-SE-03	Como usuario no registrado, deseo poder ver la red de co-autoría de un autor para visualizar un grafo con los autores con los que ha colaborado, así como la fuerza de esas colaboraciones	<ul style="list-style-type: none">■ Diseñar e implementar la interfaz y el grafo interactivo■ Definir y aplicar un esquema visual para la fuerza de la colaboración■ Implementar funcionalidades interactivas y filtros■ Agregar un panel de detalles con información adicional sobre el autor
HU-SE-04	Como usuario no registrado quiero poder ver los artículos de un investigador para conocer su trabajo y las publicaciones en las que ha contribuido	<ul style="list-style-type: none">■ Diseñar la interfaz y enruteamiento para la lista de artículos■ Implementar la tabla de artículos y la funcionalidad de redirección

Tabla 2.2: Historias de Usuario del sprint 2

Al igual que en el sprint anterior, cada historia de usuario se dividió en tareas más

pequeñas, las cuales se asignaron a los integrantes del equipo de desarrollo. Así también se definieron los criterios de aceptación que se muestran en la Figura 2.14 y en la Figura 2.15. para cada una con el fin de asegurar la calidad del desarrollo y que cada miembro del equipo tuviera claro lo que se esperaba de la tarea asignada.

Acceptance Criteria

Visualización del Grafo de Coautoría:

- El usuario no registrado debe poder acceder a una página o sección donde se muestre un grafo interactivo de la red de coautoría de un autor específico.
- El grafo debe mostrar nodos que representen a los autores y enlaces que representen las colaboraciones entre ellos

Información del Nodo:

- Al pasar el cursor sobre un nodo (autor) en el grafo, debe aparecer información básica sobre el autor, como su nombre completo y afiliación.
- Al hacer clic en un nodo, se debe mostrar un panel con detalles adicionales del autor, como el número de publicaciones y las áreas de investigación principales.

Fuerza de la Colaboración:

- Los enlaces entre nodos deben variar en grosor o color para indicar la fuerza de la colaboración, basada en el número de publicaciones conjuntas o la duración de la colaboración.
- Debe haber una leyenda que explique el significado de los diferentes grosores o colores de los enlaces.

Interactividad:

- El grafo debe ser interactivo, permitiendo al usuario hacer zoom y desplazar la vista para explorar diferentes partes de la red de coautoría.
- El usuario debe poder filtrar las colaboraciones mostradas por año o relevancia.

Figura 2.14: Criterios de Aceptación HU-SE-03

Acceptance Criteria

Visualización de Artículos:

- El usuario debe poder acceder a una página o sección donde se muestran los artículos de un investigador específico.
- La lista de artículos debe incluir el título del artículo y la fecha de publicación.

Detalle del Artículo:

- Al hacer clic en un artículo de la lista, el usuario debe ser redirigido a una página de detalles que muestre información adicional del artículo, como el resumen, los coautores, y enlaces al texto completo si están disponibles.

Figura 2.15: Criterios de Aceptación HU-SE-04

A continuación en la Figura 2.16 se muestran las tareas de cada historia de usuario. Así como en Azure DevOps se crearon los *work items* para cada tarea y se asignaron a los miembros del equipo.

Title		State	Assigned To	F
Como usuario no registrado, deseo poder ver la red de coautoría de un autor para visualizar un grafo con los autores con los que ha colaborado, así como la fuerza de esas colaboraciones		New		
Diseñar e implementar la interfaz y el grafo interactivo		To Do	JOFFRE ALEXA...	
Definir y aplicar un esquema visual para la fuerza de la colaboración		To Do	JOFFRE ALEXA...	
Implementar funcionalidades interactivas y filtros		To Do	JHON FERNAN...	
Agregar un panel de detalles con información adicional sobre el autor		To Do	JHON FERNAN...	
Como usuario no registrado quiero poder ver los artículos de un investigador para conocer su trabajo y las publicaciones en las que ha contribuido.		New		
Diseñar la interfaz y enrutamiento para la lista de artículos		To Do	JHON FERNAN...	
Implementar la tabla de artículos y la funcionalidad de redirección		To Do	JOFFRE ALEXA...	

Figura 2.16: Planificación de tareas del sprint 2

2.3.4. Implementación

Al igual que el sprint anterior, se utilizó Figma para el diseño de los mockups de las vistas que posteriormente se implementarán utilizando Angular. En la Figura 2.17 se muestra el diseño de la vista de los detalles de un autor. Así como los detalles que va tener su red de coautoría.

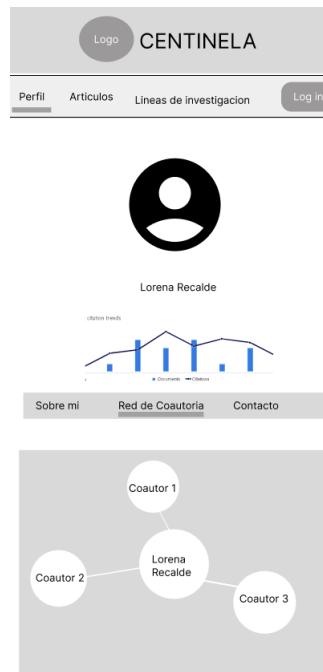


Figura 2.17: Mockup de los detalles de un autor

Como se muestra en la Figura 2.17, se diseñó la vista de los detalles de un autor, en la cual se muestra la red de coautoría del autor seleccionado. En la red de coautoría se muestra un grafo con los autores con los que ha colaborado, la fuerza de colaboración con cada autor se refleja en el grosor de la línea que los une.

En la Figura 2.18 se muestra el diseño de la vista de los artículos publicados por un autor. Los mismos están representados en una tabla con la información de cada uno y contendrán la interactividad que permitirá al usuario redirigir al mockup de la Figura 2.11.



Figura 2.18: Mockup de los artículos publicados por un autor

Para implementar la red de coautoría se utilizó la librería D3.js, la cual permite la creación de gráficos interactivos en la web. Para ello primero debemos definir los nodos y las relaciones entre ellos, en este caso los autores y la fuerza de colaboración entre ellos. En la Figura 2.19 se muestra la interfaz en Typescript que servirá para la creación de la red de coautoría.



```
1
2 export interface Link {
3   source: number;
4   target: number;
5   collabStrength: number;
6 }
7
8 export interface Coauthors {
9   links: Link[];
10  nodes: AuthorNode[];
11  affiliations: { scopusId: number, name: string }[]
12 }
13 export interface AuthorNode {
14   scopus_id: number
15   initials: string
16   first_name: string
17   last_name: string
18   weight: number
19 }
20
```

Figura 2.19: Interfaz en Typescript para la creación de la red de coautoría

Debido a que el actual componente no se encargara de la creación de la red, sino que se encargará de la visualización de la misma. No se ahondará en la implementación de la red de coautoría, ya que se encuentra fuera del alcance de este documento. El resultado final de la red de coautoría se muestra en la Figura 2.20.



Figura 2.20: Red de coautoría

Cabe mencionar que para armar la red de coautoría que se muestra en la Figura 2.20 se utilizó un conjunto de datos de prueba, ya que de momento la aplicación no cuenta con la información necesaria para armar la red de coautoría.

Ahora en la Figura 2.21 se muestra la tabla de los artículos publicados por un autor. Esta tabla también contendrá la interactividad que permitirá al usuario redirigir al mockup de la Figura 2.11.



Figura 2.21: Tabla de artículos publicados por un autor

2.3.5. Revisión y Retrospectiva

Como resultado de este sprint se logró implementar las ventanas para cada uno de los detalles de los autores, la integración de la fuerza de colaboración entre autores, la funcionalidad de los artículos publicados por autor y el enrutamiento para las diferentes vistas de los autores. Se logró cumplir con los objetivos planteados para este sprint, sin embargo, se presentaron algunos problemas durante el desarrollo de las tareas asignadas. Uno de los problemas que se presentó fue la falta de información necesaria para la creación de la red de coautoría, ya que la aplicación no cuenta con la información necesaria para armar la red de coautoría. Debido a que la implementación de la extracción de la información desde Scopus se realizará en el siguiente sprint, sin embargo los servicios que se encargan de construir la red de coautoría ya se encuentran implementados y listos para recibir la información necesaria.

Al igual que en el Sprint anterior, las tareas relacionadas con la integración con Scopus quedaron pendientes, ya que se decidió que la implementación de la extracción de la información desde Scopus se realizará iniciara en el siguiente Sprint.

2.4. Sprint 3

2.4.1. Introducción

A partir de este Sprint, se comenzará a implementar la integración con Scopus y toda lógica relacionada con el servidor.

Para este Sprint específico, se establecerá el ambiente de desarrollo con las herramientas necesarias para la implementación. Además, se rediseñará la base de datos orientada a grafos utilizando Neo4j, Django y Neomodel como ORM, basándonos en el diseño en Neo4j propuesto en Resnet. Se trabajará en la implementación de los modelos en Neomodel.

Finalmente, se pretende exponer el uso de Arquitectura Hexagonal para la implementación de la funcionalidad de búsqueda de autores. Cabe destacar que este será el único sprint en el que se ahondará en la Arquitectura Hexagonal; los demás sprints no tendrán ese nivel de detalle.

2.4.2. Objetivos

- Levantar el ambiente de desarrollo
- Implementar y rediseñar los modelos de la base de datos en Neo4j a través de Neomodel.
- Desarrollar la funcionalidad de búsqueda de autores.

2.4.3. Planificación

Para este sprint se ha seleccionado las historias de usuario que se muestran en la tabla 2.3.

Identificador	Historia de Usuario	Tareas
HU-SE-02	Como usuario no registrado deseo poder ver los investigadores que tengan colaboraciones en artículos con afiliaciones ecuatorianas para mantenerme informado sobre sus investigaciones y campos de estudio	<ul style="list-style-type: none"> ■ Crear servicios para el modelo de autor ■ Diseñar el endpoint que maneje las solicitudes del autor
HU-SE-05	Como desarrollador de software, quiero migrar nuestro modelo de base de datos actual a Neomodel, para que podamos aprovechar las capacidades del ORM Neomodel para simplificar la gestión de datos y mejorar la integración con nuestro proyecto Django.	<ul style="list-style-type: none"> ■ Definir los modelos, relaciones y propiedades de los nodos. ■ Hacer las migraciones desde Django a Neo4j.
HU-SE-06	Como desarrollador, quiero configurar un entorno de desarrollo replicable, para asegurar que podamos contar con un entorno completamente aislado y consistente que optimice el flujo de trabajo	<ul style="list-style-type: none"> ■ Crear el contenedor de Docker para al aplicación de backend ■ Crear un orquestador de contenedores ■ Orquestar los servicios de la aplicación y la base de datos

Tabla 2.3: Historias de Usuario del sprint 3

A continuación en las Figuras 2.22, 2.23 y 2.24 se muestran los criterios de aceptación para las historias de usuario seleccionadas.

Acceptance Criteria	↗ ↘
<ul style="list-style-type: none"> • El sistema debe permitir la búsqueda de autores utilizando cualquier atributo relevante, como nombre, apellido, ID de Scopus, o cualquier otro identificador único o atributo significativo • El sistema debe permitir la búsqueda de autores utilizando cualquier atributo relevante, como nombre, apellido, ID de Scopus, o cualquier otro identificador único o atributo significativo • Si no se encuentran resultados que coincidan con los criterios de búsqueda, el sistema debe devolver una lista vacía • Los resultados de la búsqueda deben ser devueltos con paginación para manejar grandes volúmenes de datos y mejorar la eficiencia 	

Figura 2.22: Criterios de aceptación HU-SE-02

Acceptance Criteria	↗ ↘
<ul style="list-style-type: none"> • El modelo de datos actual debe ser migrado a Neomodel sin pérdida de información ni alteraciones en la estructura de los datos. • Los modelos en Neomodel deben reflejar fielmente la estructura y las relaciones definidas en el modelo de datos actual. • La integración entre Neomodel y el proyecto Django debe estar configurada correctamente. 	

Figura 2.23: Criterios de aceptación HU-SE-05

Acceptance Criteria	↗ ↘
<ul style="list-style-type: none"> • El entorno de desarrollo debe estar configurado utilizando Docker y Docker Compose para asegurar la replicabilidad • Debe existir un archivo <code>Dockerfile</code> para definir la imagen del contenedor y un archivo <code>docker-compose.yml</code> para la configuración de los servicios. • Los contenedores deben estar configurados para comunicarse entre sí de manera segura y eficiente • Toda información sensible debe ser guardada en variables de entorno 	

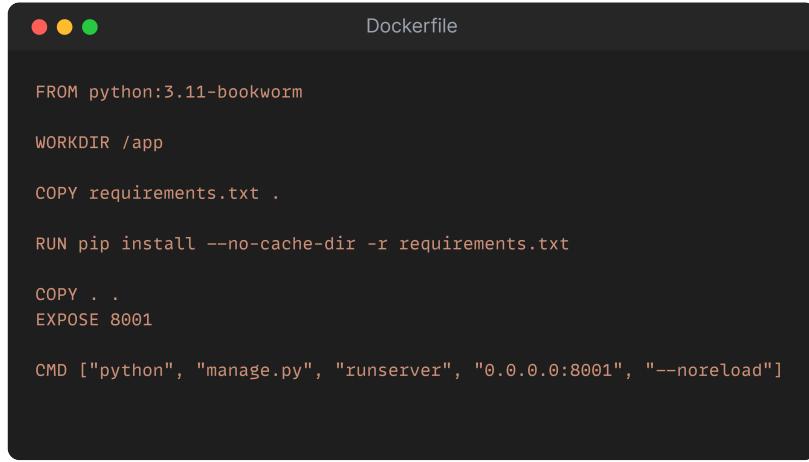
Figura 2.24: Criterios de aceptación HU-SE-06

2.4.4. Implementación

Para el desarrollo del Sprint 3 se empezó con la configuración del entorno de desarrollo, se utilizó Docker para la creación de contenedores y Docker Compose para la orquestación de los servicios. Para dockerizar la aplicación se creó un archivo Dockerfile en la raíz del proyecto, en el cual se especifican las instrucciones necesarias para la creación de la imagen de la aplicación, tal como se muestra Figura 2.25.

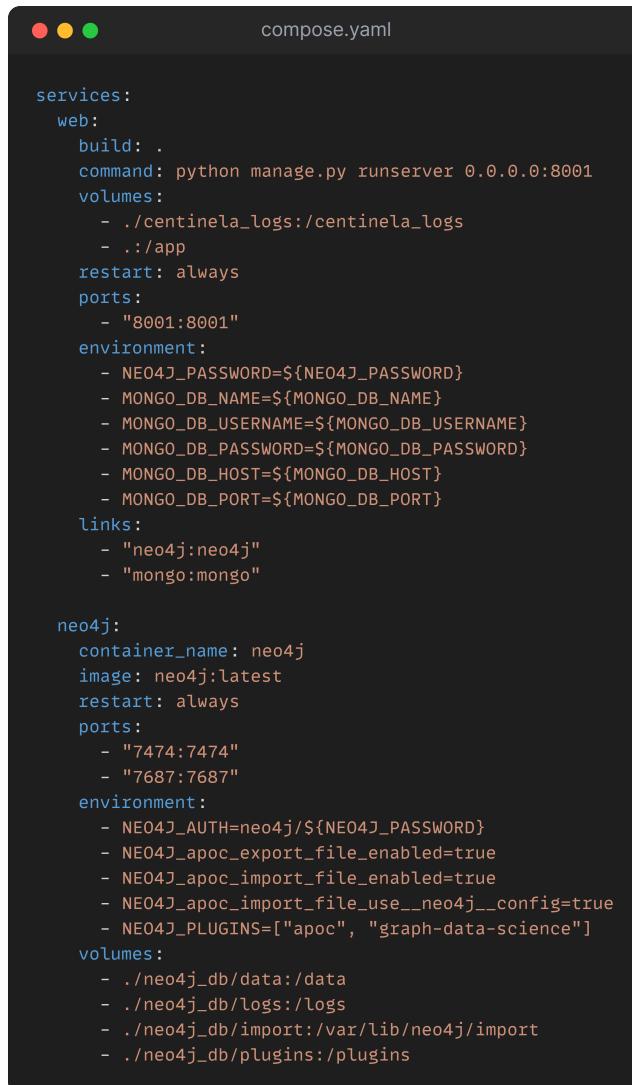
En la Figura 2.26 se muestra el archivo de configuración de Docker Compose. El mismo se encarga de orquestar los servicios de la aplicación y la base de datos. También se asegura de persistir los datos de la base de datos en un volumen de Docker. De la misma manera se especifican las variables de entorno necesarias para la configuración y el correcto funcionamiento de la aplicación.

Una vez configurado los archivos de Docker y Docker Compose, se procedió a levantar los contenedores con el comando `docker-compose up --build`. Este comando se encarga



```
FROM python:3.11-bookworm
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY . .
EXPOSE 8001
CMD ["python", "manage.py", "runserver", "0.0.0.0:8001", "--noreload"]
```

Figura 2.25: Archivo Dockerfile



```
services:
  web:
    build: .
    command: python manage.py runserver 0.0.0.0:8001
    volumes:
      - ./centinela_logs:/centinela_logs
      - ./app
    restart: always
    ports:
      - "8001:8001"
    environment:
      - NEO4J_PASSWORD=${NEO4J_PASSWORD}
      - MONGO_DB_NAME=${MONGO_DB_NAME}
      - MONGO_DB_USERNAME=${MONGO_DB_USERNAME}
      - MONGO_DB_PASSWORD=${MONGO_DB_PASSWORD}
      - MONGO_DB_HOST=${MONGO_DB_HOST}
      - MONGO_DB_PORT=${MONGO_DB_PORT}
    links:
      - "neo4j:neo4j"
      - "mongo:mongo"

  neo4j:
    container_name: neo4j
    image: neo4j:latest
    restart: always
    ports:
      - "7474:7474"
      - "7687:7687"
    environment:
      - NEO4J_AUTH=neo4j/${NEO4J_PASSWORD}
      - NEO4J_apoc_export_file_enabled=true
      - NEO4J_apoc_import_file_enabled=true
      - NEO4J_apoc_import_file_use_neo4j__config=true
      - NEO4J_PLUGINS=["apoc", "graph-data-science"]
    volumes:
      - ./neo4j_db/data:/data
      - ./neo4j_db/logs:/logs
      - ./neo4j_db/import:/var/lib/neo4j/import
      - ./neo4j_db/plugins:/plugins
```

Figura 2.26: Archivo Docker Compose

de construir y levantar los servicios de la aplicación y la base de datos. Para verificar que los contenedores se encuentran en ejecución se utiliza el comando `docker ps`. En la Figura 2.27 se muestra la salida del comando `docker ps`.

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
8be816bcc986	search_engine_backend-web	"python manage.py runserver"	5 days ago	Up About a minute	0.0.0.0:8001->8001/tcp	search_engine_back
end-web-1						
0737fb90afb3	neo4j:latest	"tini -g -- /startup.sh"	5 days ago	Up About a minute	0.0.0.0:7474->7474/tcp, 7473/tcp, 0.0.0.0:7687->7687/tcp	neo4j
3bf4dd908f5c	mongo:latest	"docker-entrypoint.sh"	5 days ago	Up About a minute	0.0.0.0:27017->27017/tcp	mongo

Figura 2.27: Sialida del comando docker ps

Con el entorno de desarrollo configurado. Se procedió a la creación de las aplicaciones de Django. Django denomina aplicaciones a un conjunto de funcionalidades que se pueden reutilizar en diferentes proyectos. En este caso se crearon 3 aplicaciones *dashboards*, *scopus_integration* y *search_engine*. Las mismas que se pueden observar en la Figura 2.28.

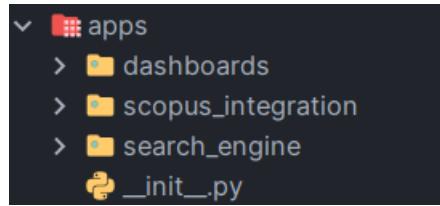


Figura 2.28: Aplicaciones de Django

Las mismas que tendrán la estructura de carpetas en base a Arquitectura Hexagonal, como se menciono en la Sección 2.1. Cada aplicación tendrá su propia capa de dominio, aplicación e infraestructura. Así como cada una tendrá una función específica en la aplicación, *dashboards* se encargara de la visualización de los datos, *scopus_integration* de la integración con Scopus y *search_engine* de la funcionalidad del motor de búsqueda.

Una vez finalizada la creación de las aplicaciones, procedió a hacer el rediseño de la base de datos previo a la implementación de los modelos en Neonmodel. Como se menciono en la Sección 2.4.1 se tomo como base el diseño en Neo4j propuesto en Resnet. Únicamente se agrego nuevos atributos a los nodos y relaciones existentes. Mas específicamente se agregaron atributos al nodo de *Autor* y a la relación *COAUTHORED*. En la Figura 2.29 se muestra el diseño de la base de datos.

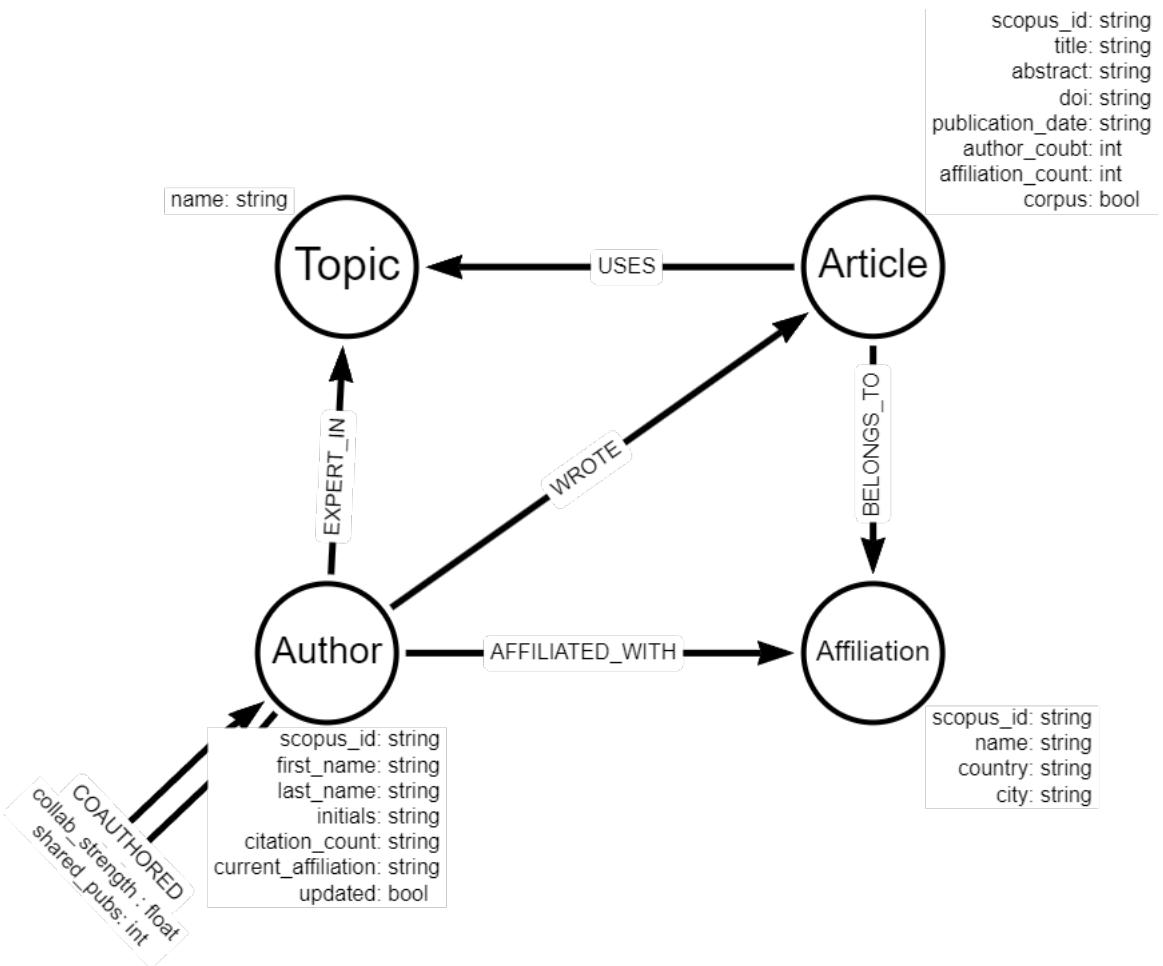


Figura 2.29: Modelo de la base de datos

Con estas actualizaciones en el modelo, se procedió a la implementación de los mismos en Neomodel. Como se menciono en el Sprint 0, para el desarrollo de la aplicación se utilizará Arquitectura Hexagonal. Bajo este enfoque se crearon los modelos de la base de datos en la capa de dominio. En la Figura 2.30 se muestra la capa de dominio de la aplicación que se encargara del motor de búsqueda.

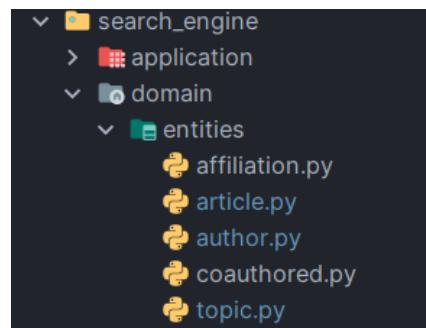


Figura 2.30: Capa de dominio de la aplicación

Basandonos en el diseño de la base de datos que se muestra en la Figura 2.29, se muestra el modelo *Author* implementado con Neomodel en la Figura 2.31. En terminos generales para asegurar la integridad de los datos se hace uso de las restricciones de unicidad de Neomodel, con el tipo de dato *UniqueIdProperty*. Esto será de gran utilidad para evitar la creación de nodos duplicados en la base de datos. Asimismo se utilizan las propiedades de tipo *Relationship* y *RelationshipTo* para definir las relaciones entre los nodos.



```
author

class Author(DjangoNode):
    scopus_id = UniqueIdProperty()
    first_name = StringProperty()
    last_name = StringProperty()
    auth_name = StringProperty()
    initials = StringProperty()
    citation_count = IntegerProperty(default=0)
    current_affiliation = StringProperty()
    affiliations = RelationshipTo('apps.search_engine.domain.entities.affiliation.Affiliation', 'AFFILIATED_WITH')
    articles = RelationshipTo('apps.search_engine.domain.entities.article.Article', 'WROTE')
    co_authors = Relationship('Author', 'CO_AUTHORED', model=CoAuthored)
    topics = RelationshipTo('apps.search_engine.domain.entities.topic.Topic', 'EXPERT_IN')
    updated = BooleanProperty(default=False)
```

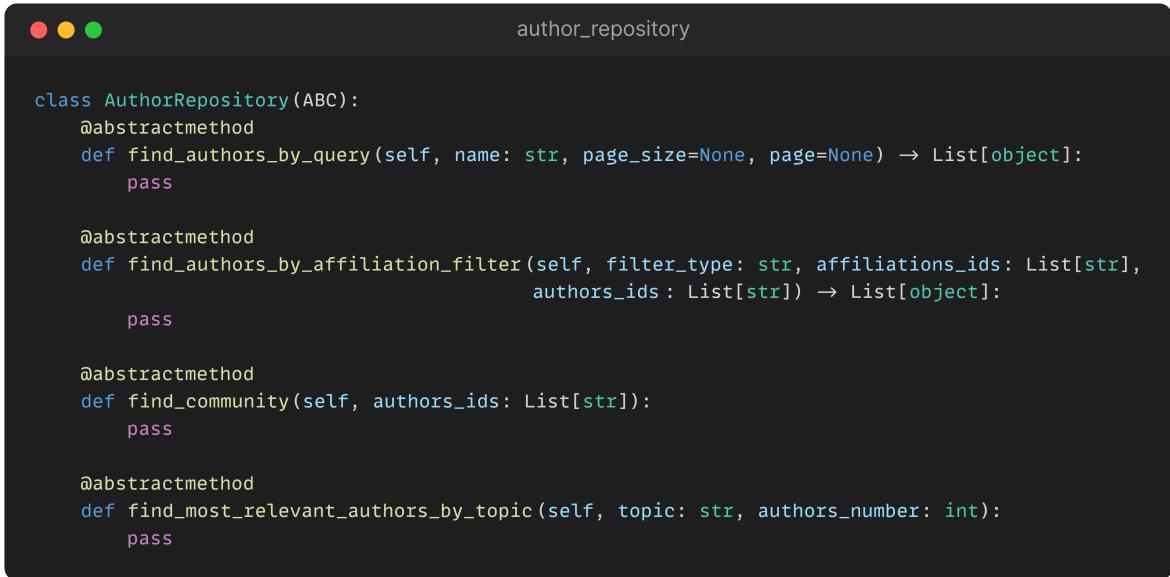
Figura 2.31: Modelo Author implementado con Neomodel

Finalmente la migración de estos modelos a la base de datos se realiza con el comando `python manage.py install_labels`. Aunque este comando es opcional es recomendable ejecutarlo para asegurar que los modelos se encuentren en la base de datos, o si en su defecto se requiere hacer una actualización de los modelos.

Siguiendo la Arquitectura Hexagonal, se procedió a implementar la funcionalidad de búsqueda de autores. Para ello, se creó el repositorio *AuthorRepository* en la capa de dominio, encargado de definir los métodos necesarios para la búsqueda de autores.

El uso del patrón de diseño Repository garantiza la separación entre la lógica de negocio y la lógica de persistencia. Posteriormente, en la capa de aplicación, se implementará la lógica de negocio correspondiente a cada una de las funciones definidas en el repositorio.

En la Figura 2.32 se muestra un extracto del repositorio *AuthorRepository* con algunos de los métodos que nos servirán para la búsqueda de autores.



```
author_repository

class AuthorRepository(ABC):
    @abstractmethod
    def find_authors_by_query(self, name: str, page_size=None, page=None) → List[object]:
        pass

    @abstractmethod
    def find_authors_by_affiliation_filter(self, filter_type: str, affiliations_ids: List[str],
                                           authors_ids : List[str]) → List[object]:
        pass

    @abstractmethod
    def find_community(self, authors_ids: List[str]):
        pass

    @abstractmethod
    def find_most_relevant_authors_by_topic(self, topic: str, authors_number: int):
        pass
```

Figura 2.32: Repositorio AuthorRepository

Una vez definidos los repositorios pasamos a la capa de aplicación, en la cual se implementa la lógica de negocio correspondiente a cada uno de los métodos definidos en el repositorio. La Figura 2.33 muestra un extracto del servicio *AuthorService* el mismo que hereda de la clase *AuthorRepository* con la implementación de la función *find_authors_by_query*.

```

class AuthorService(AuthorRepository):
    def find_authors_by_query(self, name: str, page_size=1, page=10) -> (List[object], int):
        custom_name = unidecode(name).strip().lower()
        skip = (page - 1) * page_size
        query = """
            MATCH (au:Author)
            WHERE toLower(au.first_name) CONTAINS $custom_name OR
                  toLower(au.last_name) CONTAINS $custom_name OR
                  toLower(au.first_name) + " " + toLower(au.last_name) CONTAINS $custom_name OR
                  toLower(au.last_name) + " " + toLower(au.first_name) CONTAINS $custom_name OR
                  toLower(au.auth_name) CONTAINS $custom_name OR
                  toLower(au.initials) CONTAINS $custom_name OR
                  toLower(au.email) CONTAINS $custom_name OR
                  au.scopus_id CONTAINS $custom_name
            RETURN count(au) as total
        """
        params = {'custom_name': custom_name}
        results, meta = db.cypher_query(query, params=params)

        total = results[0][0]
        query = """
            MATCH (au:Author)
            WHERE toLower(au.first_name) CONTAINS $custom_name OR
                  toLower(au.last_name) CONTAINS $custom_name OR
                  toLower(au.first_name) + " " + toLower(au.last_name) CONTAINS $custom_name OR
                  toLower(au.last_name) + " " + toLower(au.first_name) CONTAINS $custom_name OR
                  toLower(au.auth_name) CONTAINS $custom_name OR
                  toLower(au.initials) CONTAINS $custom_name OR
                  toLower(au.email) CONTAINS $custom_name OR
                  au.scopus_id CONTAINS $custom_name
            RETURN au
            ORDER BY au.citation_count DESC
            SKIP $skip LIMIT $page_size
        """
        params = {'custom_name': custom_name, 'skip': skip, 'page_size': page_size}
        results, meta = db.cypher_query(query, params=params)
        authors = [Author.inflate(row[0]) for row in results]
        return authors, total

```

Figura 2.33: Servicio AuthorService

Dado que en este caso se está utilizando una base de datos orientada a grafos, se empleó el lenguaje de consulta Cypher para realizar la búsqueda de autores. Usar Cypher permite aprovechar las ventajas de las bases de datos orientadas a grafos, como la rapidez en la consulta de datos y la facilidad para encontrar relaciones entre los nodos.

Sin embargo, existe el riesgo de Cypher Injection, por lo que es fundamental tener cuidado al construir las consultas. Para todas las consultas, se utilizó el método *cypher_query* de Neomodel, que permite ejecutar consultas Cypher directamente en la base de datos, enviando los parámetros no en la cadena de texto sino como argumentos de la función. Esto evita la inyección de Cypher en la base de datos, sanitizando las consultas y protegiéndolas de posibles inyecciones de Cypher.

Una vez con el servicio listo se procedió a la implementación del caso de uso *FindAuthorsByQuery* en la capa de aplicación. En la Figura 2.34 se muestra el caso de uso que se encargara de llamar a los repositorios necesarios para la búsqueda de autores.

```

class AuthorByQueryUseCase:
    def __init__(self, author_repository: AuthorRepository):
        self.author_repository = author_repository

    def execute(self, name: str, page_size=10, page=1) -> list[object]:
        return self.author_repository.find_authors_by_query(name, page_size=page_size, page=page)

```

Figura 2.34: Caso de uso FindAuthorsByQuery

A continuación se procedió a la implementación de la capa de infraestructura. Esta capa se va a encargar de la comunicación con el exterior de la aplicación, en este caso con la API REST. Para ello se creó la vista *AuthorViews* que se encargará de recibir las peticiones HTTP y llamar a los casos de uso correspondientes. En la Figura 2.35 se muestra un extracto de la vista *AuthorViews* con la implementación del caso de uso *FindAuthorsByQuery*. Como se puede observar, la vista únicamente se encarga de recibir la petición HTTP y llamar al caso de uso correspondiente. No maneja la lógica de negocio. Esto permite que la vista sea independiente de la lógica de negocio y de la lógica de persistencia. De esta manera, se garantiza la separación de responsabilidades y se facilita la reutilización de código, logrando que la aplicación sea más mantenible y escalable.

```

class AuthorViews(viewsets.ViewSet):
    # inject needed services
    author_service = AuthorService()
    affiliation_service = AffiliationService()

    @extend_schema(
        description="Find authors by query",
        responses=RetrieveAuthorSerializer(many=True),
        tags=['Authors'],
        parameters=[
            OpenApiParameter(name='query', type=str, required=True),
            OpenApiParameter(name='page_size', type=int, required=False),
            OpenApiParameter(name='page', type=int, required=False)
        ]
    )
    @action(detail=False, methods=['get'], url_path='find_by_query')
    def find_by_query(self, request, *args, **kwargs):
        try:
            query = request.query_params.get('query', '')
            page_size = int(request.query_params.get('page_size', 10))
            page = int(request.query_params.get('page', 1))
            author_by_query_use_case = AuthorByQueryUseCase(author_repository=self.author_service)
            authors, total = author_by_query_use_case.execute(name=query, page_size=page_size, page=page)
            serializer = RetrieveAuthorSerializer(authors, many=True)
            pagination_info = build_pagination_urls(request, page, page_size, serializer.data)
            data = serializer.data
            return Response({'total': total, 'next_page': pagination_info.get('next_page'),
                            'previous_page': pagination_info.get('previous_page'), 'data': data})
        except ValueError as ve:
            return Response({'error': str(ve)}, status=status.HTTP_400_BAD_REQUEST)
        except Exception as e:
            return Response({'error': str(e)}, status=status.HTTP_400_BAD_REQUEST)

```

Figura 2.35: Vista AuthorViews

También se implementó la serialización de los datos en la capa de infraestructura. Para

ello se creo el serializador *AuthorSerializer* que se encargara de serializar los datos de los autores. Ademas, se utilizo Swagger para documentar la API REST. En la Figura 2.35 se puede visualizar el decorador ² *extend_schema* que permite la documentación automática de la API, en este caso se documenta el endpoint *find_by_query*. El resultado de la documentación se muestra en la Figura 2.36. La cual tiene una interfaz gráfica que permite probar los endpoints de la API.

The screenshot shows the Swagger UI interface for a REST API endpoint. At the top, it says "GET /api/v1/authors/authors/find_by_query/". Below that is a title "Find authors by query". Under "Parameters", there are three fields: "page" (integer, query), "page_size" (integer, query), and "query" (string, query). A "Try it out" button is on the right. Under "Responses", there is a table for the 200 status code. It shows "Media type: application/json" and a dropdown menu set to "application/json". Below this is a note "Controls Accept header." and "Example Value: Schema". The schema example is shown in a dark box:

```
[{"scopus_id": "string", "name": "string", "affiliations": "string", "articles": "string", "topics": "string", "current_affiliation": "string", "citation_count": 0, "updated": true}]
```

Figura 2.36: Documentación de la API con Swagger

Todo lo descrito anteriormente para la implementación de la funcionalidad de búsqueda de autores se puede resumir con un diagrama de secuencia. En la Figura 2.37 se muestra el diagrama de secuencia de la funcionalidad de búsqueda de autores. En resumen con este diagrama de secuencia se puede observar el flujo de la aplicación. La petición HTTP llega a la vista *AuthorViews*, la vista llama al caso de uso *FindAuthorsByQuery*, el caso de uso llama al repositorio *AuthorRepository* y este a su vez ejecuta la consulta en la base de datos. También podemos distinguir las capas por sus colores por ejemplo la capa de dominio esta representada en color naranja, la capa de aplicación en color amarillo y la capa de infraestructura en color verde. A partir de este punto se representará únicamente el flujo de la aplicación, sin entrar en detalles de la implementación.

²Un decorador es una función que toma otra función y extiende su funcionalidad sin modificarla.

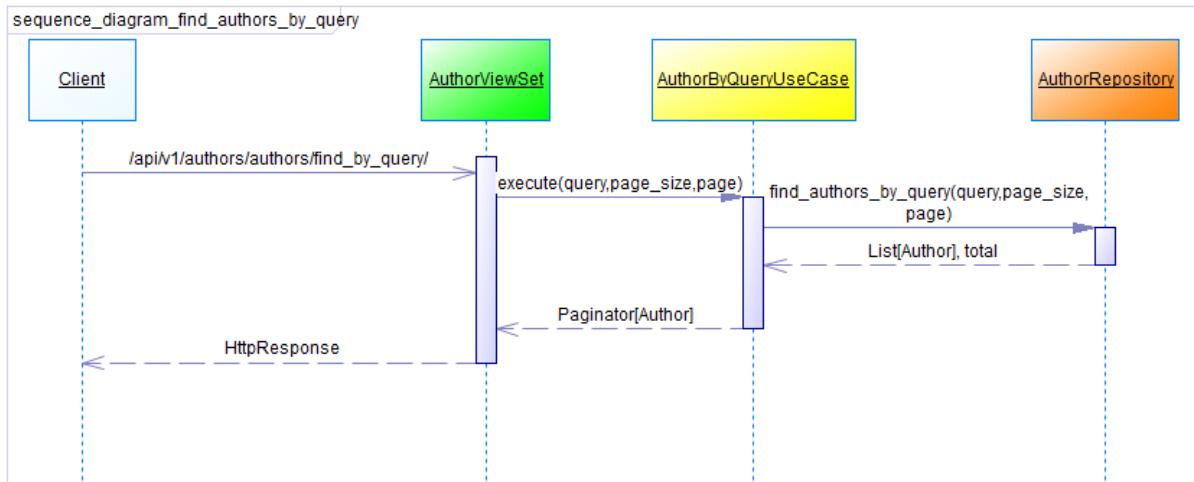


Figura 2.37: Diagrama de secuencia de la funcionalidad de búsqueda de autores

2.4.5. Revisión y Retrospectiva

En la revisión del Sprint 3 se verificó que se cumplieran los criterios de aceptación de las historias de usuario seleccionadas. Se comprobó que la funcionalidad de búsqueda de autores funcionara correctamente y que la documentación de la API REST estuviera actualizada. Además, se verificó que los modelos de la base de datos se encontraran en la base de datos. En la Figura 2.38 se muestra la base de datos con los nodos y relaciones creados.

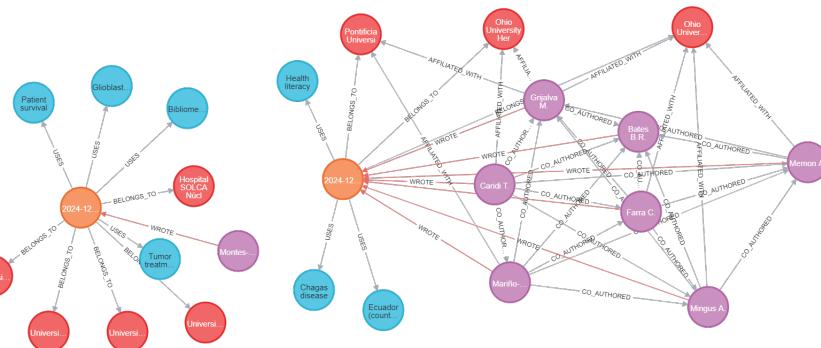


Figura 2.38: Base de datos Neo4j

Se considera que se cumplió con los objetivos planteados para este sprint, ya que se logró implementar la funcionalidad de búsqueda de autores y se rediseñó la base de datos en Neo4j utilizando Neomodel. Además, se logró levantar el ambiente de desarrollo con Docker y Docker Compose.

Sin embargo, se presentaron algunos problemas durante el desarrollo de las tareas asignadas. Uno de los problemas más significativos fue la falta de documentación de Neomodel, lo que dificultó la implementación de los modelos en la base de datos. La falta de datos de prueba también fue un problema, ya que se tuvo que trabajar con datos ficticios para realizar las pruebas de la funcionalidad de búsqueda de autores.

2.5. Sprint 4

2.5.1. Introducción

El cuarto sprint tiene como objetivo alcanzar el estado final de Resnet, es decir, lograr que Centinela esté completamente funcional y lista para ser utilizada por los usuarios. En el contexto del motor de búsqueda, se espera que Centinela sea capaz de realizar las búsquedas que Resnet podía realizar en su estado final, alcanzando así el nivel de funcionalidad de la versión anterior.

2.5.2. Objetivos

- Implementar la automatización para generar el Corpus y el Modelo de TF-IDF
- Implementar la funcionalidad de búsqueda de autores relevantes dado un tópico
- Implementar la funcionalidad de búsqueda de artículos relevantes dado un tópico
- Implementar la funcionalidad para obtener la red de coautoría de un autor
- Implementar la funcionalidad para obtener un Artículo dado su ID de Scopus
- Implementar la funcionalidad para obtener un Autor dado su ID de Scopus

2.5.3. Planificación

Para este sprint se terminaron las tareas que quedaron pendientes en el Sprint 1 (Ver sección 2.2) y en el Sprint 2 (Ver sección 2.3). En la tabla 2.4 se presentan las historias de usuario que se abordarán en este sprint con sus respectivas tareas.

Los criterios de aceptación para las historias de usuario HU-SE-01, HU-SE-03 y HU-SE-04 han sido definidos y se pueden consultar en las siguientes figuras: Figura 2.4, Figura 2.14 y Figura 2.15.

Identificador	Historia de Usuario	Tareas
HU-SE-01	Como usuario no registrado deseo poder encontrar artículos relevantes dado un tema de investigación para poder acceder rápidamente a información útil y actualizada que apoye mi estudio o trabajo	<ul style="list-style-type: none"> ■ Generar el corpus ³ de datos ■ Generar el modelo de TF-IDF a partir del corpus ■ Crear el modelo de Artículo para la base de datos ■ Implementar el filtro de artículos por año ■ Crear un servicio de búsqueda de artículos ■ Desarrollar un endpoint para manejar las solicitudes entrantes
HU-SE-04	Como usuario no registrado quiero poder ver los artículos de un investigador para conocer su trabajo y las publicaciones en las que ha contribuido	<ul style="list-style-type: none"> ■ Implementar un endpoint para recuperar el perfil de un investigador y sus publicaciones ■ Crear un servicio de búsqueda de investigadores y sus artículos
HU-SE-03	Como usuario no registrado, deseo poder ver la red de coautoría de un autor para visualizar un grafo con los autores con los que ha colaborado, así como la fuerza de esas colaboraciones	<ul style="list-style-type: none"> ■ Implementar un endpoint para recuperar los datos de colaboración entre autores ■ Crear un servicio que genere los datos del grafo de coautoría de un autor

Tabla 2.4: Historias de Usuario del sprint 4

2.5.4. Implementación

Para la implementación de ciertas tareas en este Sprint, se optó por reutilizar el código de la versión anterior de ResNet, ya que se consideró más eficiente y rápido que comenzar

desde cero. La optimización de funcionalidades existentes no está incluida en el alcance de este componente. Las funcionalidades que se reutilizarán son la generación del corpus y el modelo de TF-IDF, dado que estas tareas son las más demandantes en términos de tiempo y se determinó que no era necesario reimplementarlas. A estas funcionalidades solo se les añadirá una capa de infraestructura, ya que la lógica de negocio ya está implementada en la versión anterior de ResNet. Esto garantizará un punto de entrada para generar tanto el corpus como el modelo de TF-IDF, permitiendo su utilización en las búsquedas de autores y artículos.

Tarea: *HU-SE-01: Generar el corpus de datos* Dado que las funcionalidades de buscar autores y artículos relevantes dependen del corpus de datos, y del modelo de TF-IDF, estas tareas se realizarán en primer lugar. Para poder generar el corpus de datos, el endpoint `/api/v1/generate-corpus/` se encargará de recibir la petición mediante el método POST. Este llamará al caso de uso *GenerateCorpusUseCase* que se encargará de generar y ejecutar la lógica de negocio para generar el corpus de datos. El mismo que toma datos de Autores, Artículos, y Tópicos de la base de datos Neo4j para crear los documentos y almacenarlos en un archivo Pickle. En la figura 2.39 se muestra el diagrama de secuencia para la generación del corpus de datos.

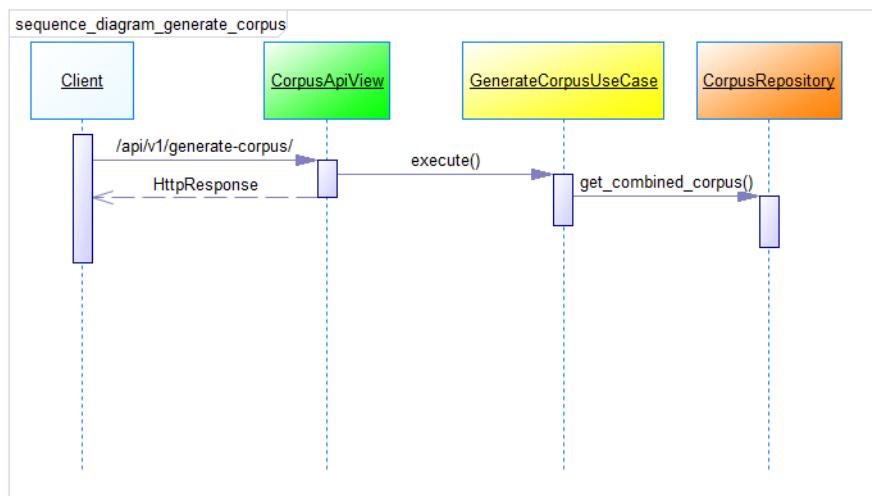


Figura 2.39: Diagrama de secuencia para la generación del Corpus

Tarea: *HU-SE-01: Generar el modelo de TF-IDF* Una vez que el corpus de datos ha sido generado, se procederá a generar el modelo de TF-IDF. Para ello, el endpoint `/api/v1/generate-model/` se encargará de recibir la petición mediante el método POST. Este llamará al caso de uso *GenerateTfidfUseCase* que se encargará de ejecutar la lógica de negocio para generar el modelo de TF-IDF. El mismo que toma el corpus de datos generado anteriormente

y lo procesa para generar el modelo de TF-IDF, para guardar el modelo en un archivo Pickle. En la figura 2.40 se muestra el diagrama de secuencia para la generación del modelo de TF-IDF.

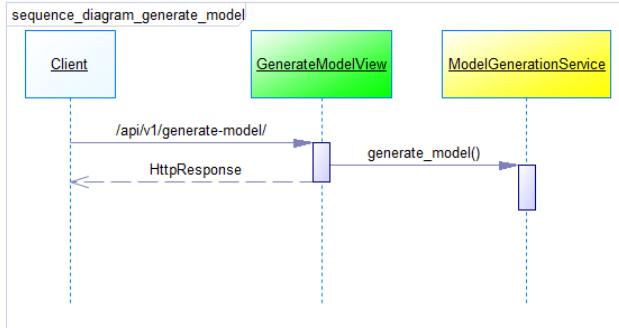


Figura 2.40: Diagrama de secuencia para la generación del Modelo de TF-IDF

Una vez que el corpus de datos y el modelo han sido generados se puede proceder a realizar las implementaciones tanto de autores relevantes como de artículos relevantes. Para la funcionalidad de Artículos relevantes la ruta `/api/v1/articles/most-relevant-articles-by-topic/` (POST) recibira en su cuerpo los siguientes datos:

- query: El Tópico por el cual se desea buscar los Artículos relevantes
- size: El límite de Artículos que se desea obtener, por defecto es 10
- page: La página de Artículos que se desea obtener, por defecto es 1
- type: Si se desea incluir o excluir años de publicación
- years: Años de publicación a incluir o excluir

Este endpoint llamará al caso de uso `GetMostRelevantArticlesByTopicUseCase` que se encargará de ejecutar la lógica de negocio para obtener los Artículos relevantes. El proceso comienza tomando el tema y pasándolo al modelo TF-IDF para identificar los IDs de los artículos relevantes. A continuación, se consulta la base de datos Neo4j para obtener los datos de esos artículos. Utilizando los parámetros de la petición, se buscan y retornan los artículos relevantes en una lista paginada, todo esto dentro de una respuesta Http. En la figura 2.41 se muestra el diagrama de secuencia para la obtención de los Artículos relevantes.

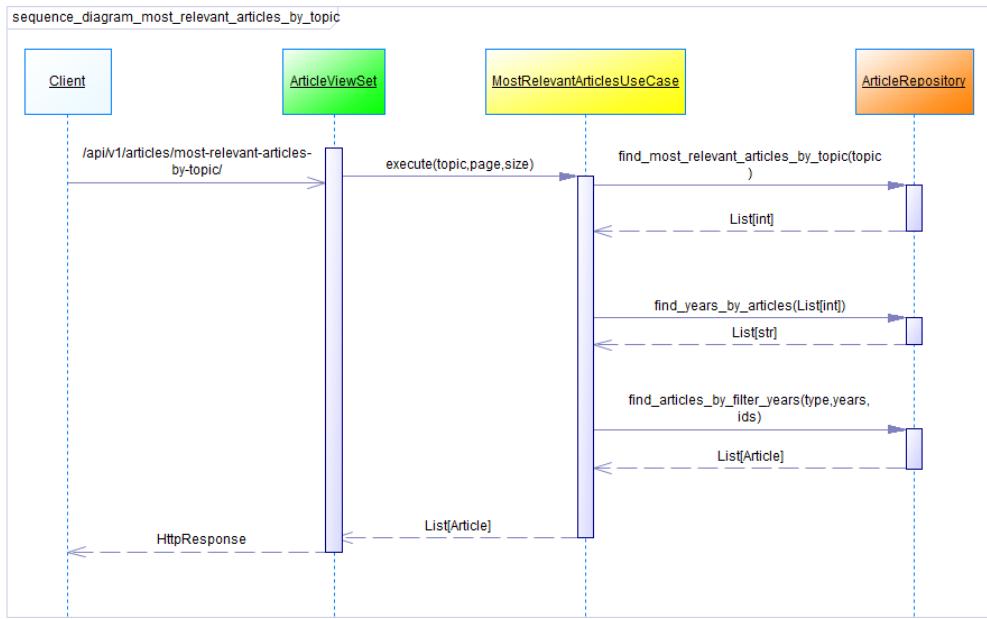


Figura 2.41: Diagrama de secuencia para la obtención de los Artículos Relevantes

HU-SE-03: Obtener la red de coautoría de un autor Para la funcionalidad de obtener la red de coautoría de un autor, la ruta `/api/v1/coauthors/coauthors/id/coauthors_by_id/` (GET) recibirá en su URL el ID del autor del cual se desea obtener la red de coautoría. Este endpoint llamará al caso de uso `GetCoauthorsByAuthorIdUseCase` que se encargará de ejecutar la lógica de negocio para obtener la red de coautoría de un autor. El proceso comienza tomando el ID del autor y consultando la base de datos Neo4j para obtener los nodos (Autores) y links (collab_strength) de la red de coautoría de ese autor. A continuación, se consultan los datos de esos coautores y se retornan en una respuesta Http. En la figura 2.42 se muestra el diagrama de secuencia para la obtención de la red de coautoría de un autor.

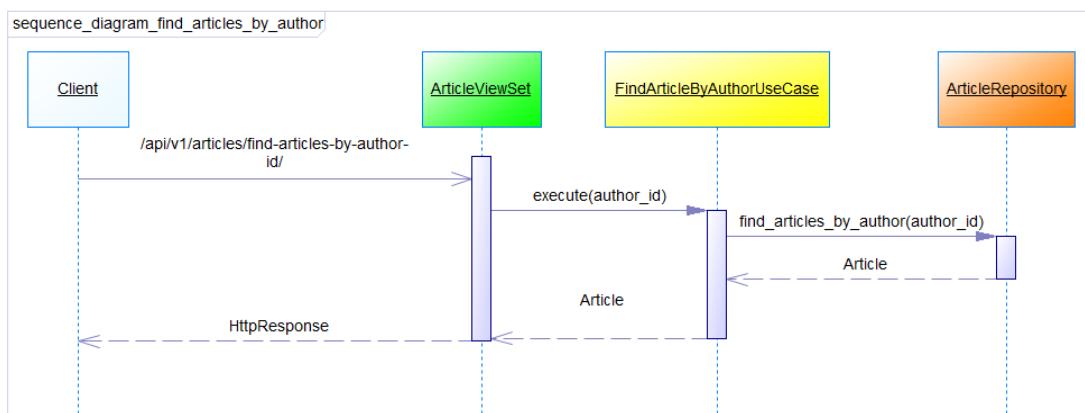


Figura 2.42: Diagrama de secuencia para la obtención de la red de coautoría de un autor

HU-SE-03: Para obtener los Artículos en los que ha colaborado un autor, la ruta `/api/v1/articles/find-articles-by-author-id/` (GET) recibirá en su URL el ID del autor del cual se desea obtener los artículos en los que ha colaborado. Este endpoint llamará al caso de uso `FindArticlesByAuthorIdUseCase` que se encargará de ejecutar la lógica de negocio para obtener los Artículos en los que ha colaborado un autor. En la Figura 2.43 se muestra el diagrama de secuencia para la obtención de los artículos en los que ha colaborado un autor.

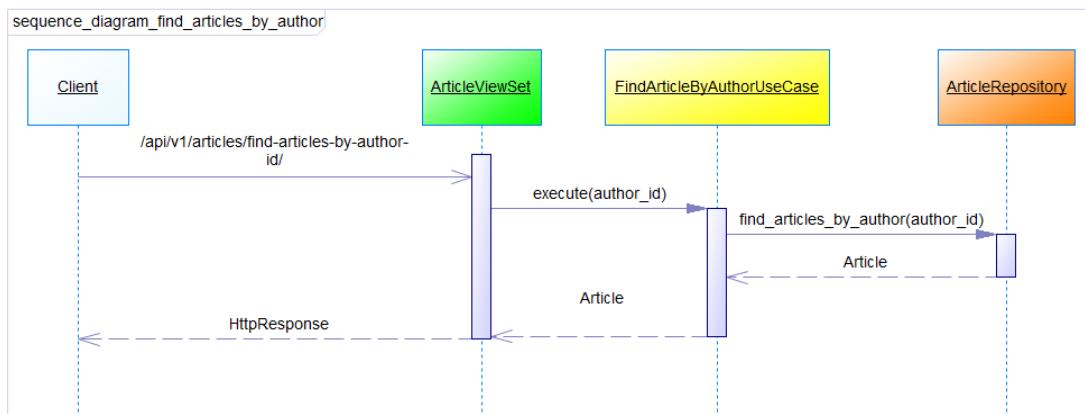


Figura 2.43: Diagrama de secuencia para la obtención de los Artículos en los que ha colaborado un autor

2.5.5. Revisión y Retrospectiva

Al finalizar el sprint, se logró implementar las funcionalidades de búsqueda de autores y artículos relevantes, así como la obtención de la red de coautoría de un autor y la obtención de los artículos en los que ha colaborado un autor. Mismas que habían quedado pendientes en los sprints anteriores. Así mismo, se logró evidenciar que la reutilización de código de la versión anterior de ResNet fue una buena decisión, ya que permitió acelerar el desarrollo de las funcionalidades. También el uso de Arquitectura Hexagonal permitió diagramar de manera clara y concisa la implementación de las funcionalidades.

Capítulo 3

RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1. Resultados

3.1.1. Prueba con usuarios finales

Para evaluar la usabilidad de la aplicación se utilizó el cuestionario SUS¹, se trata de un cuestionario de 10 preguntas como se muestra en la Tabla con una escala de 5 puntos, donde 1 es muy en desacuerdo y 5 muy de acuerdo.

¹SUS por sus siglas en inglés System Usability Scale permite medir qué tan usable es un sistema mediante un cuestionario de 10 preguntas

Numero	Pregunta
1	Creo que me gustaría usar esta aplicación frecuentemente.
2	Encontré la aplicación innecesariamente compleja.
3	Creo que la aplicación es fácil de usar.
4	Creo que necesitaría el apoyo de un experto para usar esta aplicación.
5	Encontré que las diferentes funciones de la aplicación estaban bien integradas.
6	Creo que la aplicación es muy inconsistente.
7	Imagino que la mayoría de la gente aprendería a usar esta aplicación muy rápidamente.
8	Encontré la aplicación muy difícil de usar.
9	Me sentí muy confiado al usar la aplicación.
10	Necesité aprender muchas cosas antes de poder empezar a usar la aplicación.

Tabla 3.1: Cuestionario SUS

El cuestionario fue aplicado a 14 usuarios, los resultados obtenidos se muestran en la tabla 3.1.1.

3.2. Conclusiones

El análisis de ResNet y ResearchDecide permitió identificar puntos clave en cada una de estas herramientas. A partir de este análisis, se definió una estrategia de integración que mantiene la independencia de cada sistema, mientras se facilita una comunicación eficiente entre las mismas. Esta estrategia asegura que ambos sistemas puedan colaborar sin comprometer su autonomía, optimizando el flujo de información y mejorando la eficacia global del proceso.

El uso de la Arquitectura Hexagonal en la versión 1 de Centinela, bajo el enfoque de ‘arquitectura limpia’, permitió abordar de manera eficiente la separación de responsabilidades y la integración de componentes. Este enfoque facilitó la escalabilidad del sistema, mejoró la mantenibilidad del código y aseguró que los módulos centrales permanecieran independientes de las tecnologías externas, permitiendo adaptaciones futuras sin afectar la lógica de negocio. Aunque su implementación requirió un mayor esfuerzo en la fase de diseño, se logró una mayor eficiencia en la fase de desarrollo.

La integración automatizada vía API con Scopus ha convertido a Centinela en una herramienta más robusta y completa. Esta integración permite acceder a la información más

actualizada y confiable de la base de datos de Scopus. Gracias a este proceso automatizado, un administrador puede extraer información directamente desde Scopus, asegurando que la base de datos de Centinela se mantenga siempre actualizada de manera eficiente y sin intervención manual. Además que se garantiza la calidad, integridad y disponibilidad de los datos, lo que mejora la confiabilidad de los resultados obtenidos por los usuarios de Centinela.

3.3. Recomendaciones

- Se sugiere obtener credenciales de acceso para el API Elsevier, con mayor capacidad de consultas, en especial para la obtención de información de Autores, que se hace mediante la API de Author Retrieval. Actualmente se cuenta con la limitación de 5000 consultas por semana, lo que puede ser ineficiente para la obtención de información de autores en grandes cantidades.
- Se recomienda mantener el enfoque de Arquitectura Hexagonal para futuras funcionalidades y mejoras en el sistema, ya que permite una fácil escalabilidad y mantenimiento del sistema.
- Se recomienda revisar constantemente la documentación de Neomodel, ya que al ser una herramienta relativamente nueva esta en constante actualización y puede presentar cambios en su funcionamiento y en la forma de implementar ciertas funcionalidades.
- La funcionalidad de buscar Autores dado un query (nombre, apellido, ID, etc) en ocasiones no retorna resultados esperados, se recomienda revisar la lógica de búsqueda y la forma en que se realiza la consulta a la base de datos de Neo4j.

Capítulo 4

REFERENCIAS BIBLIOGRÁFICAS

Apéndice A

Annexe Title Example B

 Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

 Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

 Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar

lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Apéndice B

Annexe Title Example B

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar

lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.