



UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
CESIM, FACULTAD DE TECNOLOGÍAS INTERACTIVAS

MÓDULO DE GESTIÓN DOCUMENTAL PARA LA PLATAFORMA GAPID

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Orlando Farid Giquez Tarafa

Tutores: Ing. Luis Manuel Valera Pérez

Ing. José Ramón Fidalgo García

Ing. Romario Yunior Lorenzo Arcia

La Habana, 2024

A photograph of Cristiano Ronaldo in a white Real Madrid jersey, smiling and celebrating with his right fist raised. The background is a blurred stadium with yellow lights.

**EL TALENTO NO LO ES TODO. LO
PUEDES TENER DESDE LA CUNA, PERO
ES NECESARIO APRENDER EL OFICIO
PARA SER EL MEJOR.**

- CRISTIANO RONALDO

Dedicatoria

A mi familia, gracias por ser mi refugio y mi motor. A mis padres May-Ling y Orlando, cuyo apoyo inquebrantable me permitió alcanzar esta meta. A mis abuelos Juanito, Teresa y Miriam, que siempre me motivaron a seguir adelante con sus palabras llenas de amor y aliento. A mi novia Lili, por su comprensión, apoyo y compañía en cada paso de este camino. A mi primo Alejandro, cuya inspiración fue clave para estudiar esta carrera. Este logro es para ustedes.

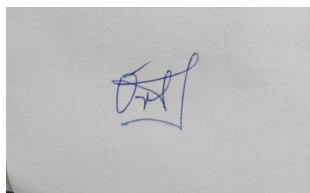
Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas las personas que, de una manera u otra, han contribuido a la culminación de esta etapa. A mis padres May-Ling y Orlando, por su amor, sacrificio y apoyo incondicional en cada paso de mi vida. Su confianza en mí ha sido el motor que me impulsó a seguir adelante. A mis abuelos Juanito, Teresa y Miriam, por sus palabras de aliento, su sabiduría y por ser un ejemplo de fortaleza y dedicación. A mi novia Lili, por su paciencia, apoyo constante y por estar siempre a mi lado, incluso en los momentos más difíciles. A mi primo Alejandro, cuya inspiración fue clave para elegir esta carrera y un modelo a seguir en muchos aspectos. A mi suegra Noila, por su cariño, comprensión y apoyo durante este proceso. A todos mis familiares, quienes con su amor y aliento han sido una fuente de fortaleza y alegría en mi vida. A mis amigos Kleyris, Driel, el Javi, el Dani y Leo gracias por estar siempre presentes, por sus palabras de ánimo y por hacer más llevadero este camino con su compañía. A todos los compañeros de clase y demás amigos que estuvieron a mi lado durante este recorrido, su apoyo, compañerismo y los momentos compartidos dejaron una huella imborrable en esta etapa de mi vida. Finalmente, a mis tutores Luis Manuel, Romario y José Ramón, por su guía, paciencia y valiosos consejos que enriquecieron este trabajo. Este logro es el resultado del esfuerzo conjunto de todas estas personas a quienes estaré eternamente agradecido.

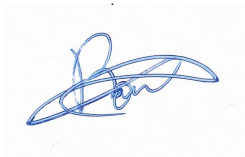
Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales sobre esta, con carácter exclusivo.

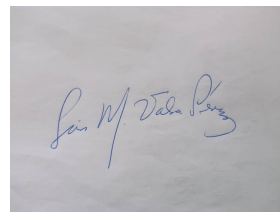
Para que así conste firmamos la presente a los 27 días del mes de noviembre del año 2024.



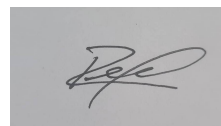
Orlando Farid Giquez Tarafa



Ing. José Ramón Fidalgo García
Tutor



Ing. Luis Manuel Valera Pérez
Tutor



Ing. Romario Yunior Lorenzo Arcia
Tutor

El desarrollo del módulo de gestión documental para la plataforma GAPID aborda las crecientes necesidades de las instituciones científicas y tecnológicas en Cuba para optimizar la gestión de programas y proyectos de investigación. Este módulo permite centralizar y organizar la documentación, superando las limitaciones de los métodos manuales y descentralizados que dificultan el acceso y control de la información. La investigación se basa en un marco teórico que incluye conceptos clave sobre gestión documental y mejores prácticas internacionales. Se utilizaron herramientas modernas como Visual Paradigm para el diseño UML, Visual Studio Code para el desarrollo, y Next.js junto con React para la interfaz de usuario, lo que permitió un desarrollo ágil. La validación del módulo se realizó a través de pruebas exhaustivas que garantizan su funcionalidad e integridad. Los resultados indicaron mejoras significativas en los flujos de trabajo y en la satisfacción del usuario. Este módulo no solo responde a las necesidades actuales de gestión documental, sino que también se alinea con las estrategias de digitalización necesarias para impulsar el avance científico y tecnológico en el país. Al proporcionar un sistema centralizado GAPID facilita el acceso rápido a la información, mejora la colaboración entre equipos y asegura la trazabilidad de los documentos, estableciendo así un precedente importante para futuras iniciativas en la gestión documental dentro del contexto cubano.

Palabras clave: documento, gestión, interfaces, proyectos de investigación, plataforma tecnológica.

The development of the document management module for the GAPID platform addresses the growing needs of scientific and technological institutions in Cuba to optimize the management of research programs and projects. This module allows centralizing and organizing documentation, overcoming the limitations of manual and decentralized methods that hinder access to and control over information. The research is based on a theoretical framework that includes key concepts about document management and international best practices. Modern tools such as Visual Paradigm for UML design, Visual Studio Code for development, and Next.js along with React for the user interface were used to enable agile development. The module's validation was carried out through exhaustive testing that guarantees its functionality and integrity. The results indicated significant improvements in workflows and user satisfaction. This module not only responds to the current needs of document management but also aligns with the necessary digitization strategies to drive scientific and technological advancement in the country. By providing a centralized system, GAPID facilitates quick access to information, enhances collaboration among teams, and ensures document traceability, thus establishing an important precedent for future initiatives in document management within the Cuban context.

Keywords: *document, management, interfaces, research projects, technological platform.*

Introducción	1
1 Fundamentos y referentes teórico-metodológicos sobre el objeto de estudio	5
1.1 Conceptos asociados a la investigación	5
1.2 Análisis de las soluciones homólogas	9
1.2.1 Soluciones homólogas internacionales	9
1.2.2 Soluciones homólogas nacionales	11
1.2.3 Valoración de las soluciones homólogas analizadas	13
1.3 Metodología de desarrollo de software	14
1.4 Herramientas y Tecnologías	17
1.4.1 Herramientas CASE	18
1.4.2 Editor de código	18
1.4.3 Desarrollo de la Interfaz de Usuario	19
1.4.4 Control de Versiones	21
1.4.5 Pruebas	22
Conclusiones parciales	22
2 Análisis y diseño de la propuesta de solución	24

2.1	Propuesta de Solución	24
2.1.1	Modelo de Dominio	25
2.2	Análisis de Requisitos	27
2.2.1	Requisitos Funcionales	27
2.2.2	Requisitos No Funcionales	28
2.3	Historias de usuario	29
2.4	Plan de iteraciones	36
2.5	Arquitectura	37
2.6	Modelo de datos	38
2.7	Patrones de diseño	39
2.8	Tarjetas CRC	43
	Conclusiones parciales	48
3	Implementación y pruebas de la solución	49
3.1	Estándar de codificación	49
3.2	Tareas de ingeniería	52
3.3	Diagrama de componentes	53
3.4	Diagrama de despliegue	54
3.5	Pruebas	55
3.5.1	Estrategia de pruebas	55
3.5.2	Pruebas unitarias	56
3.5.3	Pruebas de aceptación	58
	Conclusiones parciales	61
	Conclusiones	62

Recomendaciones	63
Referencias bibliográficas	64
Apéndices	69
A Tareas de ingeniería	70
B Pruebas unitarias	72
C Pruebas de aceptación	75

Índice de figuras

1	Modelo de Dominio. Fuente: Elaboración Propia.	26
2	Clean Architecture. Fuente: Elaboración Propia.	38
3	Modelo de datos. Fuente: Elaboración Propia.	38
4	Expert. Fuente: Elaboración Propia.	40
5	Composite. Fuente: Elaboración Propia.	41
6	Repository. Fuente: Elaboración Propia.	42
7	Inyección de Dependencias. Fuente: Elaboración Propia.	43
8	Estructura de carpetas. Fuente: Elaboración Propia.	50
9	Diagrama de Componentes. Fuente: Elaboración Propia.	54
10	Diagrama de Despliegue. Fuente: Elaboración Propia.	55
11	Test al Data Source para obtener carpetas. Fuente: Elaboración Propia.	56
12	Test al estado global para actualizar documentos dentro de sus carpetas. Fuente: Elaboración Propia.	57
13	Análisis de las pruebas unitarias. Fuente: Elaboración Propia.	58
14	Análisis de las pruebas de aceptación. Fuente: Elaboración Propia.	60
15	Test al store que maneja las carpetas seleccionadas. Fuente: Elaboración Propia.	72
16	Test a la funcionalidad de buscar documentos. Fuente: Elaboración Propia.	73

17	Test al componente DocumentsView. Fuente: Elaboración Propia.	73
18	Test a la funcionalidad de visualizar documento. Fuente: Elaboración Propia.	74
19	Test a la funcionalidad de descargar documento. Fuente: Elaboración Propia.	74

Índice de tablas

1	Comparación de soluciones de gestión documental	13
2	Requisitos Funcionales del Módulo de Gestión Documental	27
3	Historia de Usuario 1: Visualizar Estructura de Carpetas y Documentos	29
4	Historia de Usuario 2: Filtrar Documentos	30
5	Historia de Usuario 3: Buscar Documentos	31
6	Historia de Usuario 4: Visualizar Documento	32
7	Historia de Usuario 5: Descargar Documento	33
8	Historia de Usuario 6: Ordenar Documentos	34
9	Historia de Usuario 7: Renombrar Documento	35
10	Plan de Iteraciones	36
11	Tarjeta CRC: “DocumentationDataSource“	43
12	Tarjeta CRC: “DocumentationRepository“	43
13	Tarjeta CRC: “DocumentationRepositoryImpl“	44
14	Tarjeta CRC: “useDocumentationStore“	44
15	Tarjeta CRC: “Folder“	45
16	Tarjeta CRC: “Document“	46
17	Tarjeta CRC: “DocumentsView“	46

18	Tarjeta CRC: “DocumentsGrid“	47
19	Tarjeta CRC: “DocumentsTable“	47
20	Tarea de Ingeniería 1: Visualización de Estructura de Carpetas y Documentos	52
21	Tarea de Ingeniería 2: Filtrado de Documentos	52
22	Tarea de Ingeniería 3: Búsqueda de Documentos	53
23	Prueba de Aceptación para Visualizar Estructura de Carpetas y Documentos	58
24	Prueba de Aceptación para Filtrar Documentos	59
25	Prueba de Aceptación para Buscar Documentos	59
26	Tarea de Ingeniería 4: Visualización de Documento	70
27	Tarea de Ingeniería 5: Descarga de Documento	70
28	Tarea de Ingeniería 6: Ordenación de Documentos	70
29	Tarea de Ingeniería 7: Renombrar Documentos	71
30	Prueba de Aceptación para Visualizar Documento	75
31	Prueba de Aceptación para Descargar Documento	75
32	Prueba de Aceptación para Ordenar Documentos	76
33	Prueba de Aceptación para Renombrar Documento	76

La gestión de proyectos de ciencia, tecnología e innovación en Cuba ha adquirido una relevancia crítica en los últimos años, impulsada por la necesidad de optimizar recursos y asegurar el cumplimiento de estándares internacionales de calidad y eficiencia. En este contexto, el Ministerio de Ciencia, Tecnología y Medio Ambiente (CITMA) ha jugado un papel clave como entidad reguladora y facilitadora. Desde su creación en la década de 1990, el CITMA ha coordinado las políticas públicas relacionadas con la investigación científica, el desarrollo tecnológico y la protección medioambiental, alineándose con las estrategias de desarrollo sostenible del país (CITMA, 2020).

Con la implementación de normativas como el Decreto Ley 7/2020 y la Resolución 287/2019, el CITMA ha establecido un marco regulatorio robusto que obliga a las instituciones a gestionar sus proyectos de una manera transparente. Estas normativas no solo definen los parámetros de ejecución de proyectos, sino que también refuerzan la necesidad de un manejo adecuado de la información y la optimización constante de recursos (CITMA, 2019). Sin embargo, muchas instituciones cubanas aún enfrentan desafíos significativos en la gestión documental relacionada con sus proyectos.

El uso tradicional de herramientas ofimáticas como Microsoft Word y Excel ha demostrado ser insuficiente para manejar la complejidad actual. La dispersión de información, las dificultades en el control de versiones y los problemas para acceder rápidamente a documentos clave continúan siendo barreras importantes (M. González y T. López, 2020). Además, el creciente volumen de datos generados por los proyectos exige un enfoque más estructurado que permita procesar y analizar esta información eficazmente.

La creciente complejidad en la gestión de proyectos exige implementar herramientas que no solo administren los proyectos, sino que también faciliten el acceso a información crítica. La capacidad para manejar grandes volúmenes de datos se vuelve cada vez más necesaria en una sociedad basada en el conocimiento (R. López y Fernández, 2021). Cada proyecto aporta datos históricos que permiten dar seguimiento a los procesos, evaluar condiciones que provocaron el fracaso o éxito, y aprender de experiencias anteriores.

Para abordar estos problemas, el CITMA ha incentivado el desarrollo de soluciones tecnológicas que optimicen la gestión administrativa de los programas de ciencia y tecnología. Un ejemplo destacado es el Sistema

para la Gestión Administrativa de Programas y Proyectos de Ciencia, Tecnología e Innovación (GAPID). Esta plataforma ha sido diseñada para centralizar la gestión de proyectos, ofreciendo herramientas que automatizan procesos como la planificación, el seguimiento, el control económico y la gestión de recursos humanos. Esto permitirá a las instituciones cumplir con los requisitos normativos mientras optimizan sus operaciones internas (Cruz; A. Pérez y R. Martínez, 2021).

Uno de los componentes clave de GAPID es su módulo de gestión documental, diseñado para superar las limitaciones de los métodos tradicionales. Este módulo permitirá centralizar, organizar y controlar todo el ciclo de vida de los documentos asociados a los proyectos, garantizando su trazabilidad y facilitando el acceso a la información. Su diseño se basará en las mejores prácticas internacionales en gestión documental adaptadas al contexto cubano, asegurando así la confidencialidad y seguridad de los datos (R. López y Fernández, 2021).

El desarrollo del módulo no solo responderá a las demandas inmediatas de las instituciones científicas cubanas; también se alinearán con la estrategia del CITMA para digitalizar y modernizar los procesos administrativos. En un mundo donde la información es un recurso clave para la innovación, contar con herramientas que permitan una gestión documental efectiva es crucial para el éxito en investigación y el avance científico-tecnológico del país (M. González y T. López, 2020).

Además, Cuba ha estado vinculada a la gestión científica desde hace varios años. Su origen se remonta a políticas establecidas por el CITMA que fomentan no solo la investigación científica sino también el desarrollo sostenible mediante leyes como el Decreto Ley 290 del 2012 sobre invenciones y modelos industriales. Estas iniciativas buscan crear un ecosistema propicio para la innovación tecnológica en diversas áreas.

Problemática

Actualmente, la gestión documental se realiza de manera manual y descentralizada, utilizando herramientas ofimáticas como Word y Excel. Esto resulta en la dispersión de la información, dificultades para acceder y controlar los documentos, así como ineficiencias en los flujos de trabajo y la colaboración entre los usuarios involucrados en los proyectos de investigación.

Problema de investigación

¿Cómo contribuir en la mejora de la gestión documental de forma que permita centralizar, organizar y controlar el ciclo de vida de los documentos asociados a programas y proyectos de investigación en la plataforma GAPID?

Objeto de Estudio

Las aplicaciones y sistemas de gestión documental.

Campo de Acción

La gestión documental de programas y proyectos en la plataforma GAPID.

Objetivo General

Desarrollar un módulo de gestión documental para la plataforma GAPID que permita centralizar, organizar y facilitar el acceso a los documentos asociados a programas y proyectos de investigación.

Tareas de Investigación

1. Elaboración del marco teórico de la investigación mediante el análisis del estado del arte, con el fin de facilitar la comprensión del estudio de los sistemas de gestión documental.
2. Análisis de soluciones de gestión documental homologas.
3. Identificación de requisitos funcionales y no funcionales para establecer la propuesta de solución.
4. Implementación de las funcionalidades del módulo de gestión documental.
5. Validación de los resultados obtenidos a partir de pruebas de software definidas para la investigación.

Métodos Científicos

Para alcanzar los objetivos propuestos, se utilizaron varios métodos científicos:

Métodos Teóricos

- **Analítico-Sintético:** Este método es fundamental en la investigación científica, ya que permite la descomposición de un fenómeno complejo en sus elementos constitutivos (análisis) para luego reconstruir el conocimiento (síntesis). En el caso de la gestión documental en GAPID, este método facilita el análisis de las funciones y características necesarias del módulo, comparando las soluciones teóricas con las aplicables en la práctica. A través de este enfoque, es posible identificar las mejores prácticas y optimizar el sistema de gestión documental. El análisis-síntesis es clave para la construcción de soluciones integrales en campos técnicos y científicos (Sampieri, 2021).
- **Histórico-Lógico:** Este método examina la evolución de los sistemas de gestión documental, permitiendo extraer las mejores prácticas para el desarrollo del módulo en GAPID. A través de este enfoque, se logra una comprensión lógica de cómo los sistemas han evolucionado y cómo pueden mejorarse en base a experiencias previas (Francis, 2020).

Métodos Empíricos

- **Modelación:** Es una herramienta esencial para representar sistemas complejos mediante una versión simplificada o abstracta de la realidad. Permite estudiar el sistema sin observarlo directamente, facilitando el análisis y la toma de decisiones. A través de modelos físicos o conceptuales, se destacan los aspectos críticos, omitiendo elementos no esenciales, lo que ayuda en el análisis y en la búsqueda de soluciones (Francis, 2020). En el contexto de GAPID, la modelación ayudó a simplificar los flujos de trabajo y requisitos, permitiendo optimizar la gestión documental en función de las necesidades de los usuarios (Sampieri, 2021).

Estructura Capitular

- **Capítulo 1: Fundamentos y referentes teórico-metodológicos sobre el objeto de estudio.** Se lleva a cabo un análisis del estado del arte y se presentan los fundamentos que respaldan la investigación. Se describen los conceptos clave necesarios para facilitar la comprensión del trabajo que se va a realizar. También se detalla la selección de las tecnologías que se emplearán para abordar el problema, así como la justificación de la metodología de desarrollo de software elegida para orientar el proceso de creación de la solución propuesta.
- **Capítulo 2: Análisis y diseño de la propuesta de solución.** Se ofrece una breve descripción de la solución propuesta. Se desarrollan historias de usuario que especifican tanto los requisitos funcionales como los no funcionales, proporcionando una descripción concisa de cada uno. Además, se definen los patrones de diseño que se implementarán y se establece la arquitectura y el modelo de datos del sistema. Se crean tarjetas Clase-Responsabilidades-Colaboradores (CRC) para detallar las clases que componen el sistema.
- **Capítulo 3: Implementación y pruebas de la solución.** Se realiza una planificación iterativa para la implementación de la aplicación, donde se describen las tareas de ingeniería asociadas a cada historia de usuario. Se llevan a cabo un conjunto de pruebas unitarias, además de una serie de pruebas de aceptación para cada una de estas historias de usuarios, finalizando con un gráfico que resume los resultados obtenidos en las pruebas realizadas.

Fundamentos y referentes teórico-metodológicos sobre el objeto de estudio

En este capítulo se analiza en profundidad el tema de los sistemas de gestión, incluyendo las definiciones y conceptos clave que sustentan esta investigación. Además, se presentarán tanto la metodología de desarrollo de software, como las distintas tecnologías y herramientas elegidas para el desarrollo del sistema, explicando su relevancia y cómo cada una contribuye al cumplimiento de los objetivos planteados en el estudio.

1.1. Conceptos asociados a la investigación

Con el fin de profundizar en el dominio del problema, se llevó a cabo un estudio en el cual se examinaron diversos términos, tales como:

Gestión Documental

La gestión documental se ha convertido en una pieza fundamental para las organizaciones que operan en un entorno digitalizado, donde el volumen de información crece exponencialmente. La eficiencia en la creación, almacenamiento, organización, recuperación y disposición de los documentos es clave para mantener la agilidad y competitividad de las organizaciones. Una gestión documental efectiva no solo mejora la productividad al reducir el tiempo dedicado a la búsqueda de información, sino que también ayuda a minimizar riesgos asociados al cumplimiento normativo, especialmente en sectores altamente regulados como la sanidad, finanzas y tecnología (Zhang; Chen y Liu, 2020).

La transformación digital ha hecho que la gestión documental evolucione, integrando no solo documentos físicos, sino también activos digitales y metadatos que facilitan la trazabilidad de la información (Lee y Chu, 2022). Hoy en día, la implementación de sistemas de gestión documental electrónica (SGDE) es fundamen-

tal, dado que permite un acceso rápido y controlado a la información crítica, lo cual incrementa la toma de decisiones basada en datos, así como la eficiencia operativa en entornos colaborativos (M. Johnson, 2020). Además, estos sistemas garantizan la seguridad de los datos, facilitando el control de accesos y auditorías internas (Smith, 2021).

Digitalización de Documentos

La digitalización de documentos implica mucho más que simplemente convertir archivos físicos en formatos digitales. Este proceso es una transformación integral que conlleva la implementación de tecnologías avanzadas, como el reconocimiento óptico de caracteres (OCR). Esta tecnología no solo permite el almacenamiento eficiente de documentos, sino que también facilita la extracción de información estructurada, lo que mejora significativamente la capacidad de búsqueda y análisis de datos (Petrov y Möller, 2019). Al convertir documentos en un formato digital, las organizaciones pueden acceder a la información de manera más rápida y precisa, lo que resulta crucial en un entorno empresarial cada vez más competitivo.

Desde una perspectiva organizacional, la digitalización ofrece una variedad de beneficios significativos. En primer lugar, optimiza el uso del espacio físico al reducir la necesidad de almacenamiento de documentos en papel. Esto no solo libera espacio valioso, sino que también contribuye a la sostenibilidad al disminuir el consumo de papel. Además, la digitalización permite una mayor accesibilidad a la información, facilitando el trabajo colaborativo y la toma de decisiones informadas en tiempo real. La digitalización también juega un papel fundamental en la preservación y seguridad de la información. Al almacenar documentos en formatos digitales, las organizaciones pueden implementar medidas de seguridad avanzadas, como el cifrado y las copias de seguridad automáticas, para proteger la información crítica. Esto es particularmente relevante en el contexto actual, donde las amenazas cibernéticas son cada vez más sofisticadas.

Un estudio reciente de (García y Alonso, 2022) destaca la sinergia que se puede lograr al integrar la digitalización con sistemas de gestión documental basados en tecnologías emergentes como la nube y blockchain. Estas tecnologías no solo mejoran la trazabilidad y seguridad de los datos, sino que también aseguran el cumplimiento de normativas sobre protección de datos y gestión de archivos, un aspecto vital en entornos internacionales donde las organizaciones deben adherirse a regulaciones estrictas. La capacidad de auditar el acceso y las modificaciones de los documentos en tiempo real, junto con la inmutabilidad de la información almacenada en blockchain, ofrece a las organizaciones una capa adicional de confianza en sus procesos de gestión documental.

Sistemas de Gestión Documental Electrónica

Los Sistemas de Gestión Documental Electrónica (SGDE) se han consolidado como herramientas esenciales para las organizaciones modernas, transformando la manera en que se manejan y almacenan los documentos. Estos sistemas permiten el manejo centralizado de grandes volúmenes de documentos, lo que facilita su integración con otros sistemas empresariales y optimiza los flujos de trabajo. Al centralizar la información,

los SGDE no solo mejoran la eficiencia operativa, sino que también ofrecen una plataforma cohesiva para la colaboración entre equipos, lo que es fundamental en un entorno empresarial cada vez más dinámico y globalizado.

Una de las características más destacadas de los SGDE es su capacidad para controlar exhaustivamente el ciclo de vida de los documentos, desde su creación hasta su eliminación. Esta gestión integral asegura que las organizaciones cumplan con los requisitos legales y normativos vigentes, lo que es esencial para evitar sanciones y proteger la reputación corporativa (Davies y Brown, 2020). La posibilidad de establecer políticas de retención y eliminación de documentos garantiza que la información se maneje de manera ética y conforme a las regulaciones pertinentes, especialmente en sectores altamente regulados como el financiero y el de salud.

Un informe de (Forrester Research, 2021) destaca que las organizaciones que han implementado SGDE han experimentado reducciones significativas en sus costos operativos. Esto se debe en gran medida a la automatización de procesos manuales y a una gestión más ágil de documentos clave para el negocio. La agilidad que proporcionan estos sistemas permite a las empresas adaptarse rápidamente a cambios en el mercado y a demandas de los clientes, lo que, a su vez, mejora su competitividad.

Además, los SGDE ofrecen herramientas robustas de seguridad que son fundamentales para proteger datos sensibles. Mediante el control de accesos, las organizaciones pueden limitar quién puede ver y editar documentos, lo que minimiza el riesgo de filtraciones de información. La encriptación de documentos sensibles, una funcionalidad integral en muchos SGDE, asegura que incluso si los datos son interceptados, no puedan ser leídos sin las credenciales adecuadas. Esta capa adicional de seguridad es crucial para garantizar la protección de datos confidenciales y mantener la confianza de los clientes y socios comerciales.

Automatización de Procesos Documentales

La automatización de procesos documentales se refiere a la utilización de tecnologías avanzadas para llevar a cabo tareas relacionadas con la gestión de documentos que, tradicionalmente, se ejecutaban manualmente. Este enfoque incluye la clasificación automática de documentos, la generación de metadatos y la integración con otros sistemas de gestión empresarial, lo cual permite a las organizaciones aumentar su eficiencia operativa y reducir costos (Williams, 2019). La automatización no solo minimiza el tiempo y los errores asociados con la gestión manual, sino que también libera recursos humanos para que se concentren en actividades más estratégicas y de mayor valor añadido.

La implementación de tecnologías como el aprendizaje automático y el procesamiento de lenguaje natural permite analizar y procesar documentos de manera inteligente. Esto se traduce en aplicaciones prácticas como:

- **Clasificación automática:** Los documentos se organizan en categorías predefinidas según su conte-

nido y metadatos, lo que facilita su recuperación posterior.

- **Extracción de datos:** Los sistemas pueden extraer datos clave de documentos estructurados, como facturas y contratos, para su procesamiento y análisis.
- **Generación de metadatos:** Se crean automáticamente metadatos descriptivos, administrativos y estructurales, mejorando la búsqueda y recuperación de documentos.
- **Flujos de trabajo inteligentes:** Los documentos se enrutan automáticamente a través de procesos de revisión y aprobación, basados en reglas preestablecidas, agilizando así la toma de decisiones.

No obstante, la automatización de procesos documentales exige la implementación de soluciones tecnológicas avanzadas y una clara definición de reglas y modelos que guíen el procesamiento. Además, es crucial proporcionar capacitación al personal sobre el uso de estas tecnologías y asegurar la integración con otros sistemas empresariales para maximizar la eficiencia de los flujos de trabajo. La adopción de sistemas automatizados puede aumentar la productividad en un 30 %, permitiendo a las organizaciones adaptarse rápidamente a cambios en el mercado (Rivera; Sanchez y Torres, 2021).

Gestión Documental en proyectos de Ciencia y Tecnología

En el ámbito de los proyectos de ciencia y tecnología, la gestión documental desempeña un papel esencial al garantizar que toda la información relevante esté organizada y sea fácilmente accesible. Esto es especialmente crítico en iniciativas que requieren colaboración entre múltiples entidades y cumplimiento de normativas específicas. Una gestión documental eficiente no solo facilita la planificación, ejecución y evaluación de proyectos, sino que también contribuye al éxito general de las iniciativas científicas y tecnológicas (T. Miller, 2020).

Los proyectos de ciencia y tecnología abarcan una amplia gama de documentos, que incluyen propuestas, informes de avance, resultados de investigación, publicaciones y documentación financiera. Estos documentos deben ser organizados y almacenados de manera segura, permitiendo el acceso a los miembros del equipo y a las partes interesadas autorizadas. La conformidad con normativas y requisitos de financiamiento es fundamental, lo que implica mantener registros precisos y detallados de todas las actividades del proyecto.

La implementación de sistemas de gestión documental electrónica en este contexto ofrece numerosos beneficios, como:

- **Mejora en la colaboración:** Facilita la interacción entre los miembros del equipo, permitiendo un intercambio de información más fluido.
- **Reducción de la duplicación de esfuerzos:** Minimiza el riesgo de que se realicen trabajos redundantes, lo que ahorra tiempo y recursos.

- **Optimización de flujos de trabajo:** Los sistemas automatizados ayudan a definir y seguir procesos estandarizados, mejorando la eficiencia operativa.
- **Generación de informes y análisis:** Permiten la elaboración de informes detallados y análisis del progreso del proyecto, lo cual es fundamental para la toma de decisiones estratégicas.

Un informe de la Organización para la Cooperación y el Desarrollo Económicos (OCDE) destaca que la gestión documental adecuada es un determinante clave para el éxito en proyectos de innovación, dado que asegura la transparencia y la responsabilidad, aspectos críticos para la evaluación de resultados (OCDE, 2022).

1.2. Análisis de las soluciones homólogas

El estudio de soluciones homólogas se llevó a cabo con el objetivo de identificar y analizar sistemas de gestión documental similares que han sido implementados en diferentes contextos, tanto a nivel internacional como nacional. Este análisis es fundamental para comprender las mejores prácticas y los desafíos enfrentados por otros sistemas, lo que permitirá informar el desarrollo del módulo de gestión documental para la plataforma GAPID. Al examinar estas soluciones, se busca extraer lecciones valiosas que puedan ser aplicadas en el diseño y la implementación del nuevo sistema, asegurando así su eficacia y pertinencia en el contexto cubano.

1.2.1. Soluciones homólogas internacionales

SharePoint

Desarrollado por Microsoft, SharePoint se ha consolidado como una de las plataformas más extendidas para la gestión documental a nivel global, destacándose por su robustez y la integración fluida con otras herramientas del ecosistema Microsoft, como Word, Excel y Outlook. Esta integración permite a los usuarios crear, almacenar y colaborar en documentos de manera efectiva dentro de un entorno seguro y controlado. La plataforma no solo mejora la eficiencia organizacional, sino que también permite el trabajo colaborativo en tiempo real, lo que es esencial en el entorno laboral actual, donde la agilidad y la capacidad de respuesta son cruciales.

Una de las características más valoradas de SharePoint son sus capacidades avanzadas de control de acceso y auditoría. Estas funcionalidades permiten a las organizaciones cumplir con rigurosos requisitos de seguridad y normativas, garantizando la protección de la información sensible (Microsoft, 2020). Además, SharePoint facilita la creación de flujos de trabajo personalizados, permitiendo a los usuarios automatizar procesos que

de otro modo serían manuales, reduciendo así el riesgo de errores y mejorando la productividad.

SharePoint se distingue por su escalabilidad y personalización, lo que lo convierte en una opción viable para grandes proyectos de gestión documental. Las organizaciones pueden adaptar la plataforma según sus necesidades específicas, integrando módulos adicionales para funciones como la gestión de contenido y el trabajo colaborativo. Este nivel de personalización es especialmente valioso para empresas multinacionales y organismos gubernamentales, donde la gestión de información compleja y la colaboración entre diferentes departamentos y ubicaciones son esenciales para el éxito operativo (Microsoft, 2020). Además, su amplia comunidad de usuarios y el soporte constante de Microsoft aseguran actualizaciones continuas y mejoras en la funcionalidad, manteniéndola a la vanguardia en el campo de la gestión documental.

DocuWare

DocuWare es un sistema de gestión documental especializado en la automatización de flujos de trabajo y la digitalización de documentos. Su diseño intuitivo permite que los usuarios de diferentes niveles técnicos puedan aprovechar sus funcionalidades, lo que lo convierte en una herramienta accesible y eficiente. Este sistema ha sido adoptado en diversas industrias, como la salud, la educación y la manufactura, donde la necesidad de manejar grandes volúmenes de documentación es fundamental.

Una de las principales ventajas de DocuWare es su enfoque en la seguridad de los datos. El sistema proporciona controles de acceso granulares, cifrado de documentos y capacidades de auditoría, lo que asegura que la información sensible esté protegida contra accesos no autorizados (DocuWare, 2021). Además, su capacidad para integrarse con otros sistemas empresariales, como ERP y CRM, permite que las organizaciones utilicen DocuWare como parte de un ecosistema más amplio de gestión empresarial. Esto no solo mejora la productividad al centralizar la gestión de documentos, sino que también reduce los costos operativos al eliminar la redundancia y facilitar el acceso a la información.

DocuWare también destaca por sus herramientas de automatización de procesos, que permiten a las organizaciones digitalizar flujos de trabajo, desde la creación de documentos hasta la aprobación y el almacenamiento. Esta automatización no solo ahorra tiempo, sino que también asegura que los procesos cumplan con los estándares y regulaciones, un aspecto esencial en industrias altamente reguladas. En un contexto donde la digitalización y la automatización son esenciales, DocuWare proporciona una infraestructura robusta para el manejo eficiente de documentos, adaptándose a las necesidades cambiantes de las organizaciones (DocuWare, 2021).

M-Files

M-Files es una solución innovadora que utiliza un sistema basado en metadatos para organizar y recuperar documentos, eliminando la necesidad de estructuras de carpetas tradicionales. Este enfoque permite a los usuarios acceder rápidamente a la información relevante, mejorando la toma de decisiones y la eficiencia

operativa. La flexibilidad que ofrece M-Files al organizar documentos mediante metadatos permite una búsqueda más intuitiva y eficaz, reduciendo el tiempo que los empleados pasan buscando información (M-Files, 2019).

La singularidad de M-Files radica en su capacidad para aprender del comportamiento del usuario y adaptar la organización de los documentos según patrones de uso. Esto significa que a medida que los usuarios interactúan con el sistema, M-Files se ajusta para facilitar el acceso a la información que más utilizan. Este enfoque no solo mejora la experiencia del usuario, sino que también maximiza la eficiencia en entornos laborales donde el tiempo y la precisión son cruciales.

Además, M-Files se integra fácilmente con plataformas existentes, lo que permite a las organizaciones implementar esta solución sin necesidad de una revisión completa de sus sistemas actuales. Su enfoque intuitivo basado en metadatos no solo optimiza la gestión documental, sino que también mejora la colaboración entre equipos, permitiendo un acceso más fácil y rápido a la información relevante. Este nivel de innovación y adaptabilidad hace de M-Files una opción atractiva para organizaciones que buscan simplificar la gestión documental y mejorar la productividad, al tiempo que fomentan una cultura de colaboración efectiva (M-Files, 2019).

1.2.2. Soluciones homólogas nacionales

GESPRO

GESPRO se ha establecido como una de las soluciones más relevantes en el contexto cubano para la gestión de proyectos, especialmente dentro del sector informático. Diseñado como una suite orientada a la web, este sistema permite la planificación, monitoreo y control de proyectos, facilitando la coordinación de equipos y la asignación de recursos. GESPRO proporciona funcionalidades que ayudan a los gestores de proyectos a establecer cronogramas, asignar tareas, y realizar un seguimiento del progreso, lo cual es crucial para el éxito de cualquier proyecto.

Sin embargo, aunque GESPRO ofrece herramientas útiles para la gestión de documentos, su enfoque se centra principalmente en proyectos de software. Esto implica que, si bien puede manejar ciertos aspectos de la documentación asociada a la planificación y ejecución de proyectos, no aborda de manera integral las necesidades de gestión documental requeridas por el sistema de Programas y Proyectos del CITMA (García, 2020). Esta limitación se vuelve evidente cuando se consideran las exigencias de documentación y certificación de actividades establecidas por el CITMA, que requieren un control documental exhaustivo y normativo.

La falta de una estructura adecuada para el manejo de documentos técnicos y administrativos significa que los usuarios deben recurrir a otras herramientas para cumplir con los requisitos de documentación. Esto

puede generar ineficiencias y redundancias en el flujo de trabajo, afectando la efectividad global de la gestión de proyectos. Además, la carencia de características avanzadas como el control de versiones, auditorías de cambios y gestión de accesos limita aún más la capacidad de GESPRO para adaptarse a las demandas específicas del entorno de ciencia y tecnología en Cuba. Por lo tanto, aunque GESPRO es una herramienta valiosa para la gestión de proyectos, su enfoque específico en el sector informático y la insuficiencia en la gestión documental integral pueden comprometer su utilidad en un contexto más amplio de investigación y desarrollo (García, 2020).

eXcriba

eXcriba es otro sistema significativo en Cuba, que utiliza el repositorio de contenido Alfresco para gestionar documentos administrativos. Este sistema ha sido adoptado principalmente en entornos empresariales, donde su eficacia en la gestión de documentos como contratos, informes y comunicaciones es notable. La interfaz de usuario de eXcriba es intuitiva, permitiendo a los empleados manejar documentos de manera eficiente y colaborar en línea, lo cual es esencial en un mundo cada vez más digitalizado.

Sin embargo, a pesar de su eficacia en la gestión de documentos administrativos, eXcriba no proporciona la estructura necesaria para el control documental específico requerido por el sistema de Programas y Proyectos del CITMA (D. Martínez, 2021). Esto limita su efectividad en la gestión integral de la documentación necesaria para la ejecución de proyectos de ciencia y tecnología en el país. En el contexto de proyectos de investigación y desarrollo, se requieren capacidades específicas, como la gestión de versiones de documentos técnicos, trazabilidad de cambios, y cumplimiento con normativas de archivo y seguridad de datos.

La ausencia de estas funcionalidades puede dificultar la capacidad de los equipos de investigación para documentar adecuadamente sus hallazgos y cumplir con los estándares establecidos. Esto es especialmente crítico en proyectos financiados o regulados por entidades gubernamentales, donde la transparencia y la documentación exhaustiva son imperativas. Además, la falta de integración con otras plataformas de gestión de proyectos puede resultar en un manejo fragmentado de la información, lo que dificulta la colaboración efectiva entre diferentes departamentos y disciplinas. Aunque eXcriba es una herramienta útil para la gestión documental administrativa, su limitada aplicabilidad en el ámbito de la ciencia y la tecnología resalta la necesidad de un enfoque más robusto y especializado para cumplir con los requisitos específicos del contexto cubano (D. Martínez, 2021).

Solución	Libre de costo	Código abierto	Cumplimiento normativo del CITMA	Enfoque en gestión documental
SharePoint	No	No	No	Sí
DocuWare	No	No	No	Sí
M-Files	No	No	No	Sí
GESPRO	Sí	Sí	No	Parcial
eXcriba	Sí	Sí	No	Sí

Tabla 1. Comparación de soluciones de gestión documental

1.2.3. Valoración de las soluciones homólogas analizadas

El análisis de las soluciones homólogas, tanto internacionales como nacionales, ha permitido identificar las fortalezas y debilidades de cada sistema en el contexto de la gestión documental. Las soluciones internacionales, como SharePoint, DocuWare y M-Files, ofrecen características avanzadas que facilitan la colaboración, la automatización de flujos de trabajo y la organización de documentos. Sin embargo, estas soluciones pueden no estar completamente alineadas con las necesidades específicas del contexto cubano, donde las normativas y procedimientos del CITMA requieren un enfoque adaptado y personalizado.

Por otro lado, las soluciones nacionales como GESPRO y eXcriba, aunque son relevantes en el contexto cubano, presentan limitaciones significativas. GESPRO está más orientado a la gestión de proyectos informáticos y no aborda de manera integral las necesidades de gestión documental requeridas por el sistema de Programas y Proyectos del CITMA. eXcriba, aunque es útil para la gestión de documentos administrativos, no proporciona la estructura necesaria para el control documental específico requerido en el ámbito de la ciencia y la tecnología. Ambas soluciones no logran satisfacer las exigencias de documentación y certificación de actividades, lo que pone de manifiesto la necesidad de un sistema que integre estas funcionalidades de manera efectiva.

El desarrollo del módulo de gestión documental para la plataforma GAPID se presenta como la solución ideal para resolver estos problemas. El módulo de gestión documental de la plataforma GAPID está diseñado específicamente para abordar las particularidades del entorno cubano, integrando funcionalidades que permiten la generación automática de documentos, la gestión centralizada de expedientes y el cumplimiento de las normativas del CITMA. Este enfoque no solo garantiza que el sistema sea eficaz y eficiente, sino que también asegura que se alinee con las expectativas y requerimientos de los usuarios en el contexto cubano.

Además, se beneficiará de la experiencia adquirida a partir del análisis de soluciones homólogas, permitiendo la adopción de mejores prácticas y tecnologías que han demostrado su eficacia en otros contextos. Al integrar las lecciones aprendidas de sistemas como SharePoint y DocuWare, junto con un enfoque adaptado

a las necesidades locales, tiene el potencial de convertirse en una herramienta clave para la gestión documental en proyectos de ciencia y tecnología en Cuba. En conclusión, el desarrollo de este módulo no solo responde a la necesidad de un sistema de gestión documental robusto y eficiente, sino que también representa una oportunidad para impulsar la innovación y el desarrollo en el ámbito de la ciencia y la tecnología en el país, asegurando que los proyectos se gestionen de manera efectiva y cumplan con los estándares requeridos por el CITMA.

1.3. Metodología de desarrollo de software

En el desarrollo de sistemas de software de alta calidad, la elección de la metodología adecuada es esencial para estructurar, organizar y gestionar el proceso completo, desde la fase de diseño hasta la implementación y pruebas finales. En el caso del módulo de gestión documental para la plataforma GAPID, se aplicó una metodología robusta que integró técnicas, procedimientos y herramientas que aseguraron la construcción de un sistema eficaz, mantenible y adaptable a los requisitos cambiantes.

La metodología en el desarrollo de software implica un marco organizado que guía al equipo en cada fase del proyecto, proporcionando una estructura que permite gestionar tareas complejas y garantizar la calidad del producto final. Al adoptar una metodología sólida, los equipos de desarrollo no solo optimizan su eficiencia, sino que también mitigan riesgos, mejoran la colaboración y aseguran una entrega satisfactoria (Sommerville, 2016).

RUP (Rational Unified Process)

Rational Unified Process (RUP) es una metodología de desarrollo de software creada por Rational Software (ahora parte de IBM) que se centra en el ciclo de vida completo del desarrollo de un sistema, desde la concepción inicial hasta la entrega del producto final. RUP se caracteriza por ser iterativo e incremental, lo que permite un desarrollo evolutivo del software y la constante retroalimentación del cliente (S. W. Ambler, 2006). Entre sus principales características se encuentran:

- **Fases claramente definidas:** Divide el desarrollo en cuatro fases principales: Inicio, Elaboración, Construcción y Transición, lo que permite un enfoque sistemático y controlado para cada etapa.
- **Enfoque orientado a objetos:** Utiliza el modelado UML (Unified Modeling Language) como base para el diseño de sistemas, lo cual es útil en proyectos complejos donde se requiere una estructura clara y coherente.
- **Gestión de riesgos:** Al integrar actividades de gestión de riesgos desde el inicio del proyecto, busca identificar y mitigar problemas antes de que afecten el desarrollo.

- **Flexibilidad adaptativa:** Aunque RUP es una metodología formal, puede ser ajustada y adaptada a diferentes contextos y tamaños de proyecto. Esto es particularmente importante en proyectos grandes, donde la gestión de la complejidad es un factor crucial.

Sin embargo, RUP puede presentar ciertas desventajas en proyectos de menor escala o en entornos que requieren flexibilidad extrema, ya que su implementación puede ser compleja y requerir un alto nivel de planificación y documentación. Esto a menudo lleva a que sea percibida como una metodología rígida y pesada en términos de tiempo y recursos (S. W. Ambler, 2006).

XP (Extreme Programming)

Extreme Programming (XP) es una metodología ágil que se enfoca principalmente en mejorar la calidad del software y la capacidad del equipo de adaptarse a los requisitos cambiantes del cliente. XP surge como una respuesta a las limitaciones de los métodos tradicionales en términos de flexibilidad y la necesidad de entregar software de manera rápida y eficiente. Sus principios claves incluyen:

- **Desarrollo basado en la retroalimentación constante:** Promueve ciclos de desarrollo cortos (normalmente en iteraciones de una o dos semanas), lo que permite adaptarse rápidamente a los cambios en los requisitos del cliente.
- **Programación en pares:** Esta técnica fomenta la colaboración estrecha entre dos desarrolladores que trabajan juntos en el mismo código, lo cual mejora la calidad y promueve el aprendizaje compartido.
- **Refactorización continua:** Pone un énfasis particular en la mejora continua del código, asegurando que este sea siempre limpio, mantenible y eficiente, lo que reduce los costos de mantenimiento a largo plazo.
- **Entrega frecuente:** Promueve la entrega frecuente de versiones funcionales del software, lo que permite al cliente tener un producto usable en todo momento y reduce el riesgo de desalineación con las expectativas del cliente.

Una de las grandes fortalezas de XP es su capacidad para gestionar proyectos donde los requisitos no están completamente definidos desde el inicio o cambian con frecuencia. Sin embargo, al ser un enfoque altamente colaborativo, su éxito depende de que todos los miembros del equipo estén alineados con las prácticas ágiles y puedan trabajar en un ambiente de comunicación abierta y constante (Pressman y Maxim, 2015). Además, XP tiende a ser menos formal en términos de documentación y procesos, lo que podría ser una desventaja en proyectos donde la formalización y el control son críticos.

Metodología AUP UCI

La metodología AUP (Agile Unified Process) es una variación ágil del Proceso Unificado de Rational (RUP),

diseñada para facilitar la entrega incremental e iterativa de software de alta calidad. AUP simplifica algunos de los aspectos más rígidos de RUP, permitiendo una mayor flexibilidad y adaptabilidad a las demandas de los clientes, integrando principios ágiles como la interacción continua con los usuarios, la entrega frecuente de software funcional y la rápida adaptación a los cambios (S. Ambler, 2006).

En la Universidad de las Ciencias Informáticas (UCI), se adoptó una variante de AUP, denominada AUP UCI, la cual está adaptada al contexto y las necesidades de los proyectos desarrollados en la universidad. Esta metodología ha demostrado ser particularmente eficaz en entornos académicos, donde los recursos y los tiempos pueden ser limitados, pero donde la calidad del software sigue siendo una prioridad (Delgado; R. Pérez y A. González, 2017).

Fases de AUP UCI

El ciclo de vida de AUP UCI se divide en tres fases principales, que permiten gestionar el proyecto de manera estructurada y eficiente:

- **Inicio:** En esta fase, se definen los objetivos y el alcance del proyecto. Se identifican los requisitos iniciales y se elabora un plan de proyecto de alto nivel, que incluye la evaluación de riesgos y la planificación de su mitigación. Este es un paso esencial para asegurar que todas las partes interesadas comprendan claramente los objetivos del proyecto, proporcionando una base sólida para el desarrollo posterior.
- **Ejecución:** Es la fase central del desarrollo, donde se construye el sistema de manera iterativa e incremental. Se llevan a cabo análisis detallados, el diseño de la arquitectura y la implementación de los componentes clave. El uso de prototipos durante esta fase es fundamental para validar los diseños y asegurar que el sistema evoluciona de acuerdo con las expectativas del cliente.
- **Cierre:** Durante esta última fase, el software se entrega al cliente y se realiza la transición hacia su operación en un entorno productivo. Aquí se incluyen pruebas de aceptación, capacitación de usuarios y la resolución de problemas encontrados durante la implementación. Esta fase garantiza que el producto final esté listo para ser utilizado de manera eficiente por los usuarios finales.

Escenarios de AUP UCI

AUP UCI define cuatro escenarios para abordar el modelado de los requisitos del sistema, proporcionando distintas perspectivas sobre cómo capturar, analizar y priorizar las funcionalidades.

- **Escenario 1:** Se utiliza una combinación de casos de uso del negocio (CUN) y del sistema (CUS) para modelar los requisitos desde las perspectivas del usuario final y del sistema técnico, respectivamente. Esto asegura que las interacciones del sistema con sus usuarios y otros sistemas externos estén bien definidas.

- **Escenario 2:** Se emplea el modelo conceptual (MC) para capturar los conceptos fundamentales del negocio, y los casos de uso del sistema (CUS) para detallar las interacciones técnicas del sistema. Este enfoque asegura que tanto la lógica del negocio como las especificaciones técnicas estén bien alineadas.
- **Escenario 3:** Se prioriza la descripción de procesos de negocio (DPN) y los requisitos de procesos (DRP), con el fin de modelar los flujos de trabajo del negocio y cómo el sistema los apoya. Esto es crucial para asegurar que el software automatice eficientemente los procesos clave.
- **Escenario 4:** Se prefieren las historias de usuario (HU) para capturar los requisitos funcionales desde la perspectiva del usuario final. Este enfoque facilita una mayor flexibilidad en el manejo de cambios en los requisitos, ya que se basa en la entrega incremental y en la colaboración constante con los usuarios para adaptar el software a sus necesidades reales.

Para el desarrollo del módulo de gestión documental de GAPID, se seleccionó el Escenario 4, ya que permitió un enfoque centrado en el usuario, que priorizó la entrega frecuente de funcionalidades y la colaboración continua con los interesados. Esto garantizó que el software se alineara con las expectativas del cliente y que las funcionalidades fueran entregadas de manera incremental y con alta calidad.

La elección de AUP UCI no solo responde a su capacidad para combinar la estructura robusta de RUP con la flexibilidad de XP, sino también porque:

1. Es la metodología utilizada en el centro CESIM, lo que facilita su adopción dentro del proyecto.
2. Su enfoque en la entrega incremental asegura que las funcionalidades sean implementadas de manera iterativa, similar a XP, permitiendo que el cliente interactúe con el sistema de forma constante.
3. Tiene un enfoque ajustado al contexto cubano, lo que permite gestionar los recursos y la infraestructura tecnológica disponible de manera más efectiva.

1.4. Herramientas y Tecnologías

Para el desarrollo del módulo de gestión documental de la plataforma GAPID, se seleccionaron cuidadosamente diversas herramientas y tecnologías que optimizaron cada fase del proyecto, asegurando la creación de un sistema robusto, escalable y mantenible. Estas herramientas no solo mejoraron la eficiencia del desarrollo, sino que también facilitaron el trabajo en equipo y garantizaron altos estándares de calidad en la implementación y pruebas.

1.4.1. Herramientas CASE

Se realizó un estudio comparativo de tres herramientas CASE (Computer-Aided Software Engineering): **Visual Paradigm** (versión 8.0), **Excalidraw** (versión 2023.10.2) y **Lucidchart** (versión 1.62).

1. **Visual Paradigm (8.0)**: Es una herramienta que permite modelar sistemas mediante diagramas UML (Unified Modeling Language). Su capacidad para generar código automáticamente a partir de los diagramas acelera el desarrollo y reduce errores derivados de la interpretación manual. Esto es especialmente valioso en entornos donde el tiempo es un factor crítico. A pesar de su complejidad, que puede ser un reto para nuevos usuarios, su funcionalidad para compartir modelos facilita la colaboración entre los miembros del equipo, promoviendo una comunicación efectiva (Paradigm, 2023).
2. **Excalidraw (2023.10.2)**: Destaca por su interfaz intuitiva y la creación de diagramas a mano alzada, lo que la convierte en una opción ideal para la rápida visualización de ideas y conceptos. Su facilidad de uso permite a los usuarios sin experiencia técnica crear representaciones visuales efectivas en poco tiempo. Sin embargo, carece de funcionalidades avanzadas que ofrecen otras herramientas CASE, lo que puede limitar su aplicación en proyectos más complejos donde se requieren diagramas más detallados (Excalidraw, 2023).
3. **Lucidchart (1.62)**: Es una herramienta basada en la web que permite la creación colaborativa de diagramas. Su interfaz amigable y sus integraciones con aplicaciones como Google Drive y Microsoft Office hacen que sea útil para equipos distribuidos geográficamente. Esto facilita el trabajo conjunto en tiempo real, lo cual es esencial en proyectos donde los miembros del equipo pueden estar ubicados en diferentes lugares. Sin embargo, la versión gratuita tiene limitaciones significativas en cuanto a la cantidad de documentos y funciones disponibles, lo que puede restringir su uso en entornos más exigentes (Lucidchart, 2023).

Después de evaluar estas opciones, se eligieron Visual Paradigm y Excalidraw por su versatilidad y facilidad de uso. Visual Paradigm se eligió por su capacidad para generar código automáticamente, mientras que Excalidraw fue valorada por su simplicidad para crear diagramas rápidos. Lucidchart fue descartada debido a las limitaciones de la versión gratuita y porque las funcionalidades críticas ya estaban cubiertas por las otras dos herramientas seleccionadas.

1.4.2. Editor de código

Se llevó a cabo un estudio sobre varios editores de códigos antes de seleccionar el adecuado para el proyecto.

1. **Visual Studio Code (1.71)**: Es ampliamente utilizado debido a su flexibilidad, ligereza y robustez.

Ofrece soporte para múltiples lenguajes y una amplia gama de extensiones que permiten personalizar el entorno de desarrollo según las necesidades específicas del proyecto. Además, su integración con Git facilita la gestión del código fuente (Microsoft, 2023b). La interfaz intuitiva permite a los desarrolladores adaptar el entorno a sus necesidades específicas, utilizando extensiones para mejorar el autocompletado, la depuración de código y el análisis estático.

2. **Sublime Text (4)**: Es valorado por su velocidad y ligereza, lo que lo hace ideal para trabajar con archivos grandes. Su capacidad de personalización mediante configuraciones avanzadas permite a los desarrolladores optimizar su flujo de trabajo según sus preferencias individuales. Sin embargo, la versión gratuita muestra recordatorios constantes para adquirir una licencia, lo cual puede ser molesto para algunos usuarios (Sublime HQ, 2023). Sublime Text también incluye características como "Goto Anything" para navegación rápida entre archivos y "Multiple Selections" para editar múltiples líneas simultáneamente.
3. **Atom (1.58)**: Es un editor altamente personalizable desarrollado por GitHub. Permite a los usuarios modificar casi cualquier aspecto del editor mediante paquetes y temas, lo que fomenta una experiencia adaptada a sus necesidades particulares. Sin embargo, puede ser más lento en comparación con otras opciones como Visual Studio Code y Sublime Text cuando se trabaja con archivos grandes (GitHub, 2023). Atom incluye características como un sistema integrado para gestión de paquetes y un editor colaborativo llamado Teletype que permite a varios desarrolladores trabajar en el mismo archivo simultáneamente.

Finalmente, Visual Studio Code fue seleccionado como el editor de código a utilizar debido a su extensibilidad, comunidad activa y la integración con Git, lo que permitió al equipo concentrarse en el desarrollo y la colaboración sin interrupciones.

Todas las herramientas y tecnologías que se mencionarán a continuación fueron elegidas por el equipo del GAPID en el centro CESIM.

1.4.3. Desarrollo de la Interfaz de Usuario

Next.js (14)

En su versión 14, fue el framework utilizado para el desarrollo de la interfaz de usuario del módulo de gestión documental. Next.js es una herramienta que permite la creación de aplicaciones web mediante React, ofreciendo además renderización del lado del servidor (SSR) y generación de sitios estáticos (SSG), características que mejoran tanto la velocidad de carga como la optimización para motores de búsqueda (SEO). Esto es especialmente relevante en aplicaciones que manejan grandes volúmenes de datos o donde el acceso

rápido a la información es crucial, como en un sistema de gestión documental (Vercel, 2023).

Next.js también destaca por su capacidad de manejar enrutamiento automático y su simplicidad para implementar páginas y secciones de la aplicación, lo que redujo la complejidad del desarrollo y mejoró la mantenibilidad a largo plazo. Además, su capacidad para dividir el código y cargar solo lo necesario en cada vista permitió mejorar el rendimiento general de la aplicación.

React (18.2)

La biblioteca de JavaScript desarrollada por Facebook, fue utilizada como el motor principal para construir la interfaz de usuario. En su versión 18.2, React ofrece un enfoque basado en componentes, lo que permite dividir la interfaz en pequeñas piezas reutilizables. Esta característica no solo mejora la organización del código, sino que también facilita su mantenimiento y escalabilidad, ya que cada componente puede desarrollarse, probarse y actualizarse de manera independiente (Meta, 2023b).

La capacidad de React para actualizar de manera eficiente el DOM virtual asegura que los cambios en la interfaz se reflejen rápidamente, lo que es crucial para proporcionar una experiencia fluida y altamente interactiva. Esto es especialmente importante en el contexto del módulo donde los usuarios interactúan constantemente con la aplicación para buscar, visualizar y manipular documentos.

TypeScript (5.1)

Es un superconjunto de JavaScript que añade tipado estático y otras características avanzadas que permiten un desarrollo más seguro y eficiente. En su versión 5.1, se utilizó en combinación con React para escribir código más legible, estructurado y menos propenso a errores. El tipado estático ayuda a los desarrolladores a detectar problemas en tiempo de compilación, lo que reduce los errores en tiempo de ejecución y mejora la autocompletación y documentación del código (Microsoft, 2023a).

El uso de TypeScript también permitió definir interfaces y tipos personalizados para representar más claramente las estructuras de datos en el módulo, contribuyendo a un código más robusto y mantenible.

HTML5 y CSS3

El desarrollo de la interfaz también involucró el uso de HTML5 y CSS3. HTML5 proporcionó una estructura semántica sólida y eficiente para las páginas web del sistema, facilitando la accesibilidad y mejorando la interoperabilidad entre dispositivos. CSS3, con sus capacidades avanzadas como transiciones y animaciones, permitió estilizar la interfaz atractivamente.

Tailwind CSS (3.2)

En su versión 3.2, fue el framework elegido para gestionar el estilo de la interfaz. Tailwind se destaca por su enfoque basado en utilidades; esto permite a los desarrolladores aplicar estilos directamente a los elementos HTML sin necesidad escribir largas hojas personalizadas CSS. Este enfoque incrementó la velocidad desarrollo ya que los estilos básicos podían aplicarse rápidamente sin sacrificar personalización diseño (Labs, 2023).

La utilización Tailwind CSS también permitió mantener una estética limpia profesional asegurando funcionalidades sistema fueran accesibles manera clara ordenada usuarios finales.

Next UI (2.4.8)

Es una biblioteca de componentes diseñada específicamente para aplicaciones construidas en Next.js y React. Su versión 2.4.8 fue empleada este proyecto integrar componentes predefinidos como botones formularios modales aceleraron proceso desarrollo; utilización Next UI aseguró interfaz coherente estéticamente agradable mientras componentes predefinidos mantuvieron consistencia visual funcional (UI, 2023).

1.4.4. Control de Versiones

Git (2.41)

En su versión 2.41, se utilizó como sistema de control de versiones para gestionar el código del módulo de gestión documental. Git es una herramienta fundamental para rastrear el historial de versiones, permitiendo a los desarrolladores trabajar en paralelo en diferentes características o corregir errores sin interferir en el trabajo de los demás. Este sistema distribuye copias completas del repositorio a cada miembro del equipo, lo que permite trabajar de manera descentralizada, sin necesidad de conexión constante a un servidor.

Git fue especialmente útil en este proyecto para crear ramas de desarrollo separadas, cada una enfocada en diferentes módulos o funcionalidades. A través del proceso de pull requests y revisiones de código, se garantizó la calidad del código antes de integrarlo en la rama principal.

1.4.5. Pruebas

Jest (29)

En su versión 29 fue el framework utilizado para implementar pruebas unitarias, se caracteriza por su facilidad de uso y configuración mínima, lo que permitió realizar pruebas de manera rápida y eficiente. Con este, se verificó el funcionamiento correcto de los componentes de forma aislada, asegurando que cada unidad del sistema cumpliera con sus requisitos antes de la integración. Además, su capacidad para ejecutar pruebas de manera paralela y generar informes detallados facilitó el proceso de depuración y la validación del código. Su simplicidad también permitió realizar simulaciones (mocking) de funciones y objetos, lo que ayudó a aislar los componentes y garantizar que cada parte del sistema se comportara como se esperaba (Meta, 2023a).

Conclusiones parciales

En este capítulo se abordaron aspectos fundamentales para el desarrollo del módulo de gestión documental para la plataforma GAPID. Se estableció la importancia de contar con un marco metodológico sólido que no solo facilite la organización del trabajo, sino que también mejore la comunicación entre los miembros del equipo y la satisfacción del cliente.

- La fundamentación teórica estableció el contexto y la importancia de contar con un marco metodológico sólido. Una metodología bien definida no solo facilita la organización del trabajo, sino que también mejora la comunicación entre los miembros del equipo y la satisfacción del cliente.
- El análisis de soluciones homólogas permitió identificar herramientas y enfoques existentes en el ámbito del desarrollo de software. Esto proporcionó una base sólida para la selección de tecnologías adecuadas, como Visual Paradigm para el modelado y Visual Studio Code como editor de código fuente.
- La elección de tecnologías como Next.js, React y TypeScript, junto con frameworks de diseño como Tailwind CSS y Next UI, demostró ser crucial para crear una interfaz de usuario eficiente y atractiva. Estas herramientas optimizan el flujo de trabajo del desarrollador y garantizan una experiencia de usuario fluida.
- La metodología AUP UCI fue presentada como el enfoque elegido para guiar el desarrollo del módulo. Este enfoque permite un desarrollo ágil y flexible, facilitando la adaptación a cambios en los requisitos a lo largo del proceso. Se trabajará con el Escenario 4, que utiliza historias de usuario para modelar

las funcionalidades desde la perspectiva del usuario, asegurando que el desarrollo se alinee con las necesidades reales.

Análisis y diseño de la propuesta de solución

En este capítulo se definen los requisitos tanto funcionales como no funcionales que la aplicación web debe cumplir para garantizar su correcta operación y desempeño. Además, se identifican los principales artefactos de ingeniería de software que se generarán a lo largo del proyecto, basados en la metodología adoptada. Estos artefactos, que incluyen diagramas, modelos y documentos técnicos, son fundamentales para asegurar que el desarrollo siga una estructura ordenada y coherente, facilitando tanto la implementación como la futura evolución del sistema.

2.1. Propuesta de Solución

La propuesta de solución para el módulo de gestión documental en la plataforma GAPID se centra en desarrollar un sistema integral que permita a los usuarios interactuar de manera efectiva con los documentos relacionados con programas, proyectos y entidades. Este sistema facilitará el acceso a documentos organizados de forma estructurada en carpetas, mejorando significativamente la gestión de la documentación asociada a iniciativas de investigación. Con este enfoque, se busca garantizar que la información sea fácilmente accesible y esté bien organizada. Se crearán carpetas que contendrán todos los documentos relevantes, permitiendo a los usuarios acceder rápidamente a la información necesaria. Esta flexibilidad en la organización no solo optimiza la búsqueda de documentos, sino que también mejora la trazabilidad y el control de la información a lo largo del tiempo.

El módulo de gestión documental permitirá a los usuarios realizar diversas acciones clave:

- **Acceso a Documentos:** Los usuarios podrán navegar por los proyectos y acceder a las carpetas donde

encontrarán todos los documentos organizados de manera clara y coherente.

- **Recuperación de Documentos:** Se implementarán funcionalidades avanzadas de búsqueda que permitirán localizar documentos específicos dentro de las carpetas, utilizando filtros como tipo de documento o fecha.
- **Visualización de Documentos:** Los usuarios podrán visualizar los documentos directamente desde la plataforma, facilitando así la revisión y el análisis de la información sin necesidad de descargar archivos.

La implementación de este módulo es esencial en el contexto actual, donde las instituciones dedicadas a la ciencia, tecnología e innovación enfrentan crecientes desafíos en la gestión eficiente de programas y proyectos. La necesidad de cumplir con normativas específicas, optimizar el uso de recursos y garantizar la integridad y accesibilidad de la información generada hace que un sistema centralizado y eficiente sea una prioridad.

Además, el desarrollo del módulo se fundamenta en las mejores prácticas internacionales adaptadas a las necesidades específicas del contexto cubano. Esto garantiza no solo la seguridad y confidencialidad de la información, sino también que se alineen con las estrategias locales de digitalización y modernización en las instituciones científicas y tecnológicas. En un entorno donde la información es un recurso crítico para la innovación, contar con herramientas que permitan una gestión más efectiva del documento se convierte en un factor clave para el éxito de los proyectos de investigación y, en última instancia, para el avance científico y tecnológico del país.

2.1.1. Modelo de Dominio

El concepto de modelo de dominio es una herramienta clave en el desarrollo de software, utilizada para representar de manera conceptual las entidades, relaciones y comportamientos de un sistema dentro de un contexto específico. Este tipo de modelo no solo actúa como un puente entre el mundo real y el diseño del software, sino que también facilita la comprensión de los requisitos del negocio y la alineación con las soluciones técnicas a implementar. Un modelo de dominio bien definido es esencial para garantizar que tanto los desarrolladores como las partes interesadas compartan una visión común del sistema a construir, mejorando la comunicación y reduciendo posibles malentendidos (Gysel; Kälbener; Giersche y Zimmermann, 2016).

El modelo de dominio es especialmente relevante en enfoques como el Diseño Dirigido por el Dominio (DDD), donde las entidades y sus relaciones se estructuran de manera iterativa, asegurando que reflejen fielmente las necesidades del negocio. Estas entidades suelen agruparse en objetos de valor y agregados, que encapsulan la lógica de negocio y mantienen la consistencia de los datos dentro del sistema (Seidewitz, 2003).

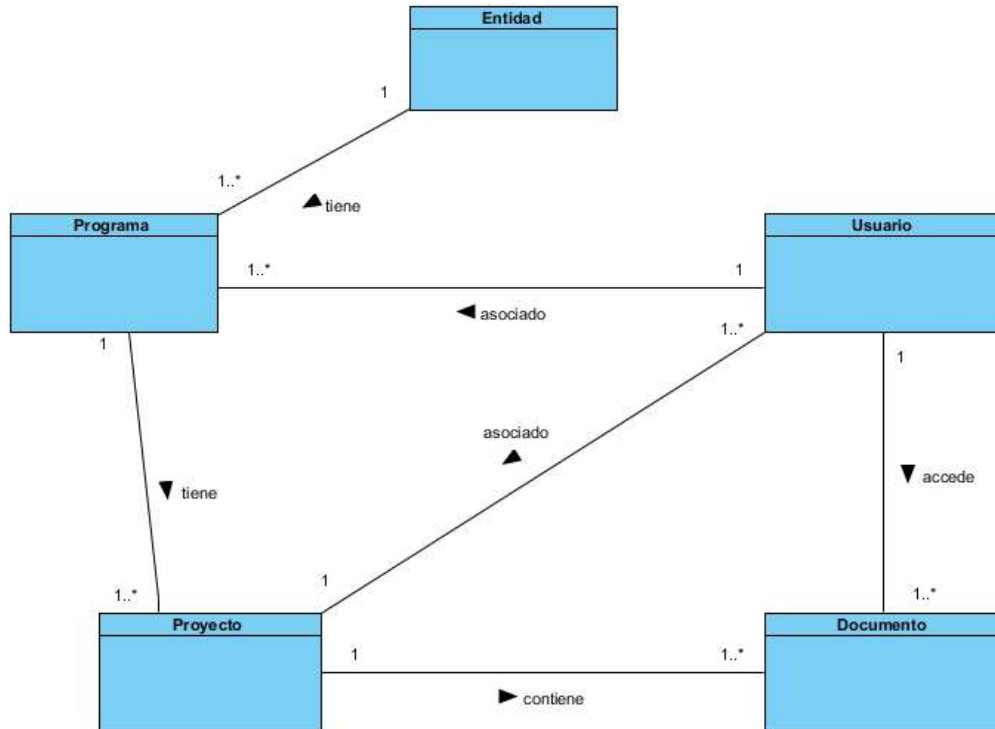


Figura 1. Modelo de Dominio. Fuente: Elaboración Propia.

Entidades del Modelo

- **Usuario:** Un usuario es una persona que interactúa con la plataforma GAPID. Los usuarios tienen la capacidad de gestionar documentos asociados a programas y proyectos.
- **Documento:** Un documento es cualquier archivo o registro que contiene información relevante para los programas y proyectos gestionados en GAPID. Esto incluye informes, propuestas, resultados de investigación y otros tipos de archivos que son esenciales para la comunicación y toma de decisiones dentro de los proyectos.
- **Programa:** Un programa es una estructura organizativa que coordina y gestiona la planificación de varios proyectos interrelacionados. Su propósito es asegurar que estos proyectos se alineen con los objetivos estratégicos de la entidad, optimizando recursos y facilitando la comunicación entre los equipos involucrados.
- **Proyecto:** Un proyecto se define como una planificación que consiste en un conjunto de actividades y objetivos interrelacionados y coordinados. Estos elementos requieren la evaluación y supervisión por parte de expertos asignados a cada proyecto, asegurando que se cumplan los estándares de calidad y se alcancen los resultados deseados.

- **Entidad:** Una entidad es una organización, institución o grupo que participa en la gestión de programas y proyectos dentro de la plataforma GAPID.

2.2. Análisis de Requisitos

El análisis de requisitos es una etapa esencial en el desarrollo de sistemas, pues define las capacidades que el sistema debe cumplir para satisfacer las necesidades de los usuarios y cumplir con normativas o estándares específicos. Un requisito se entiende como una condición o funcionalidad que debe implementarse en un sistema para cumplir con un conjunto definido de necesidades, ya sean operativas, contractuales o técnicas. Los requisitos deben ser “claros, completos y verificables” para garantizar que el producto final esté alineado con las expectativas del cliente. Los requisitos ambiguos o incompletos pueden llevar a malentendidos durante el desarrollo, retrasos y, en consecuencia, a un producto que no satisface las expectativas de los usuarios (Pressman, 2014).

En la gestión documental, un análisis exhaustivo de los requisitos es crucial para asegurar que el sistema no solo sea funcional, sino también adaptable a las especificaciones de los usuarios y las normativas vigentes. Esto es especialmente relevante en áreas donde la integridad y accesibilidad de la información son esenciales, como en instituciones de ciencia y tecnología, donde la gestión eficiente de documentos asegura el éxito de los proyectos de investigación (Kotonya y Sommerville, 1998).

2.2.1. Requisitos Funcionales

Los requisitos funcionales describen las funciones específicas que un sistema debe realizar. Estos requisitos se centran en las acciones que el sistema debe permitir a los usuarios y en cómo debe comportarse en diferentes situaciones. En el contexto del módulo de gestión documental de GAPID, se han identificado los siguientes requisitos funcionales:

Tabla 2. Requisitos Funcionales del Módulo de Gestión Documental

NO	Requisito	Prioridad	Descripción
1	Visualizar Estructura de Carpetas y Documentos	Alta	Permitir a los usuarios ver un listado de las carpetas disponibles y los documentos contenidos en cada carpeta.
2	Filtrar Documentos	Alta	Facilitar la aplicación de filtros para visualizar documentos específicos según criterios como tipo de documento o fecha de creación.

3	Buscar Documentos	Alta	Incluir un campo de búsqueda que permita a los usuarios buscar documentos utilizando palabras clave, proporcionando resultados relevantes en tiempo real.
4	Visualizar Documento	Alta	Ofrecer la opción de abrir y ver el contenido de un documento directamente desde el sistema sin necesidad de descargarlo.
5	Descargar Documento	Alta	Permitir a los usuarios descargar un documento de manera sencilla y accesible.
6	Ordenar Documentos	Media	Incluir una opción para ordenar los documentos según criterios como nombre, fecha de creación o tipo de archivo.
7	Renombrar Documento	Media	Permitir a los usuarios cambiar el nombre de un documento existente para mejorar la organización y la identificación.

2.2.2. Requisitos No Funcionales

Los requisitos no funcionales (RNF) son propiedades o cualidades que el sistema debe tener. Se refieren a las características, restricciones y atributos de calidad del software. A diferencia de los requisitos funcionales, los RNF no determinan una funcionalidad específica del sistema, sino que definen cómo debe comportarse el sistema. Los RNF deben ser específicos, cuantificables y verificables (Sommerville, 2011). A continuación, se muestran los RNF identificados:

Usabilidad

- RNF 1: El sistema debe presentar una interfaz agradable, intuitiva y fácil de usar para los usuarios.
- RNF 2: El sistema debe ser accesible desde los principales navegadores web, como Chrome, Firefox, Safari y Opera, en sus versiones más recientes.

Rendimiento

- RNF 3: El tiempo de respuesta del sistema en cada transición o acción del usuario no debe exceder los 3 segundos.
- RNF 4: El sistema debe ser capaz de manejar un alto volumen de usuarios concurrentes sin degradar su rendimiento.

Mantenibilidad

- RNF 5: El código fuente del sistema debe estar bien documentado y seguir las mejores prácticas de programación para facilitar su mantenimiento y evolución.
- RNF 6: El sistema debe tener una arquitectura modular que permita la incorporación de nuevas funcionalidades y la corrección de errores de manera eficiente.

Escalabilidad

- RNF 7: El sistema debe ser escalable horizontalmente para adaptarse a un crecimiento futuro en la cantidad de usuarios y datos.

Seguridad

- RNF 8: El sistema debe cumplir con las regulaciones y estándares de seguridad aplicables, como la protección de datos personales.

2.3. Historias de usuario

Tabla 3. Historia de Usuario 1: Visualizar Estructura de Carpetas y Documentos

Historia de Usuario	
Número: 1	Nombre: Visualizar Estructura de Carpetas y Documentos
Programador: Orlando Farid Giquez Tarafa	Iteración Asignada: 1
Prioridad en negocio: Alta	Tiempo Estimado: 5 días
Riesgo en Desarrollo: N/A	—
Descripción: • Esta funcionalidad es esencial para que los usuarios accedan a una representación visual de la estructura de carpetas y documentos dentro del sistema. • Permite una navegación eficiente entre los diferentes niveles de la jerarquía de proyectos y sus respectivas carpetas. • La visualización debe ser clara y organizada, facilitando el acceso rápido a la información necesaria.	
Observaciones: • La interfaz debe ser intuitiva y facilitar la navegación sin complicaciones. • Es fundamental que la visualización se actualice en tiempo real al realizar cambios en la estructura de carpetas.	

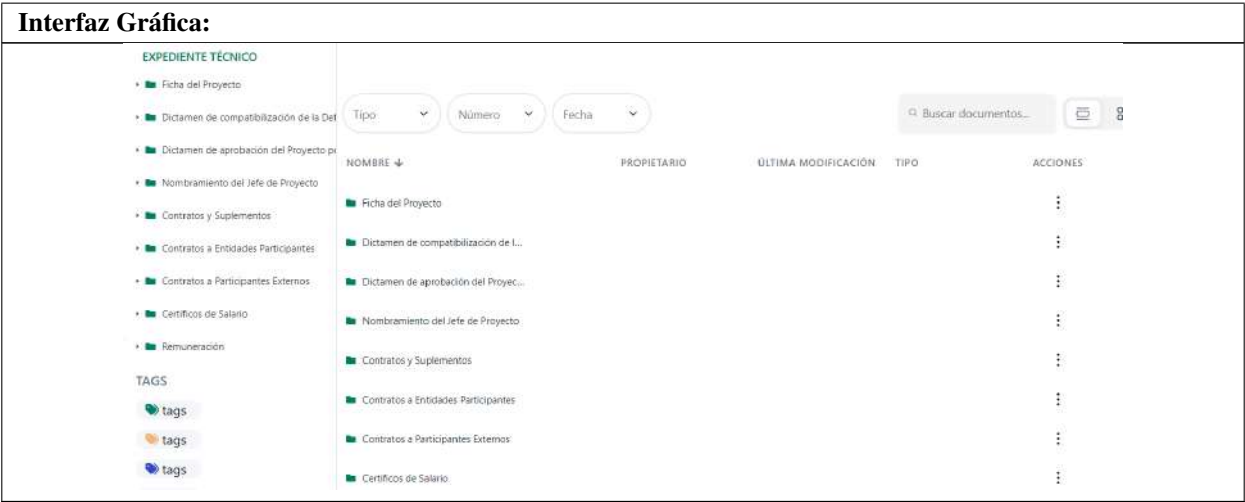


Tabla 4. Historia de Usuario 2: Filtrar Documentos

Historia de Usuario	
Número: 2	Nombre: Filtrar Documentos
Programador: Orlando Farid Giquez Tarafa	Iteración Asignada: 1
Prioridad en negocio: Alta	Tiempo Estimado: 5 días
Riesgo en Desarrollo: N/A	—
Descripción: • Esta funcionalidad permite a los usuarios aplicar filtros para visualizar documentos específicos según criterios como tipo de documento, fecha de creación o modificación. • Los resultados deben actualizarse en tiempo real, mejorando la eficiencia en la localización de documentos relevantes. • La implementación de filtros debe ser intuitiva, permitiendo a los usuarios seleccionar y combinar múltiples criterios de búsqueda.	
Observaciones: • Es importante que la interfaz de filtrado sea clara y fácil de entender. • Se debe garantizar que los filtros aplicados no afecten negativamente el rendimiento del sistema.	
Interfaz Gráfica:	

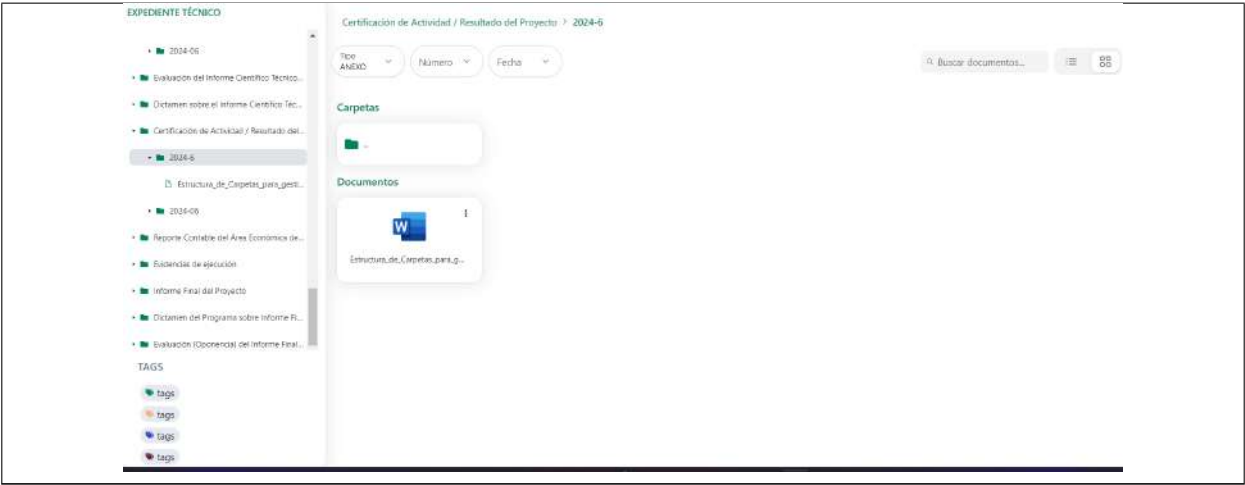


Tabla 5. Historia de Usuario 3: Buscar Documentos

Historia de Usuario	
Número: 3	Nombre: Buscar Documentos
Programador: Orlando Farid Giquez Tarafa	Iteración Asignada: 1
Prioridad en negocio: Alta	Tiempo Estimado: 4 días
Riesgo en Desarrollo: N/A	—
Descripción: - Esta funcionalidad incluye un campo de búsqueda que permite a los usuarios buscar documentos utilizando palabras clave. - El sistema debe implementar un algoritmo de búsqueda eficiente que considere tanto el contenido del documento como sus metadatos. - Los resultados de la búsqueda deben presentarse de manera clara, permitiendo a los usuarios identificar rápidamente los documentos relevantes.	
Observaciones: - Es esencial que la búsqueda sea rápida y eficiente, incluso con grandes volúmenes de documentos. - La funcionalidad de búsqueda debe ser capaz de manejar errores tipográficos y ofrecer sugerencias de búsqueda.	
Interfaz Gráfica:	

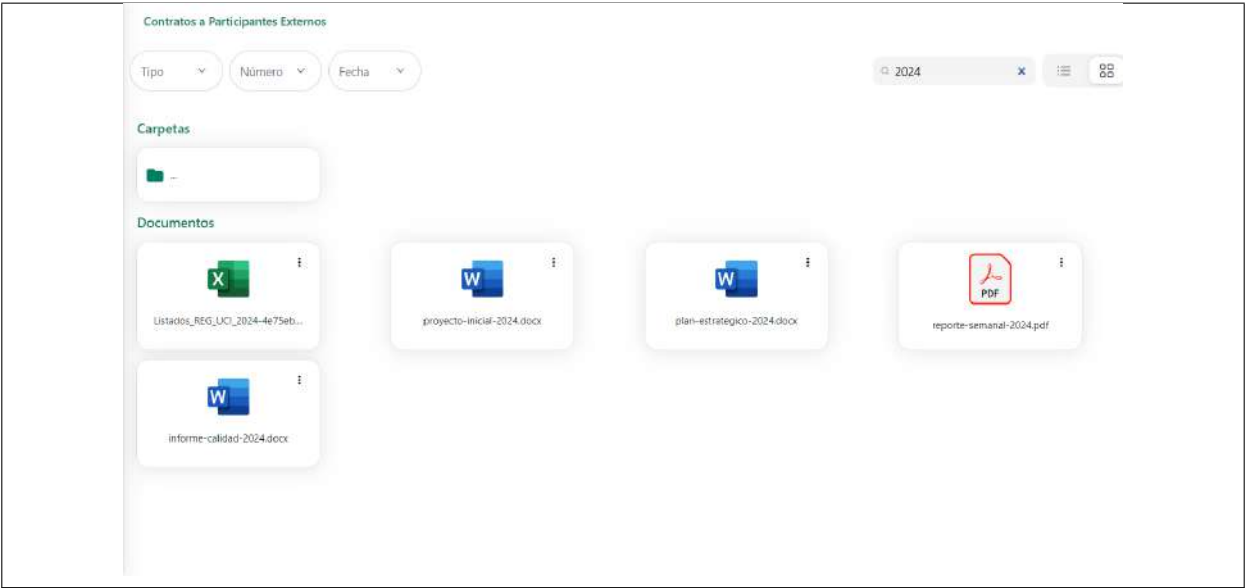


Tabla 6. Historia de Usuario 4: Visualizar Documento

Historia de Usuario	
Número: 4	Nombre: Visualizar Documento
Programador: Orlando Farid Giquez Tarafa	Iteración Asignada: 2
Prioridad en negocio: Alta	Tiempo Estimado: 5 días
Riesgo en Desarrollo: N/A	—
Descripción:	
- El usuario desea visualizar documentos almacenados en el sistema de gestión documental sin necesidad de descargarlos o modificarlos. - El sistema debe ofrecer un visor que permita la navegación por el documento. - No se permitirá la edición del documento durante su visualización.	
Observaciones:	
- El sistema validará los permisos del usuario antes de permitir la visualización del documento. - El visor deberá cargar los documentos de manera eficiente, incluso si son de gran tamaño.	
Interfaz Gráfica:	

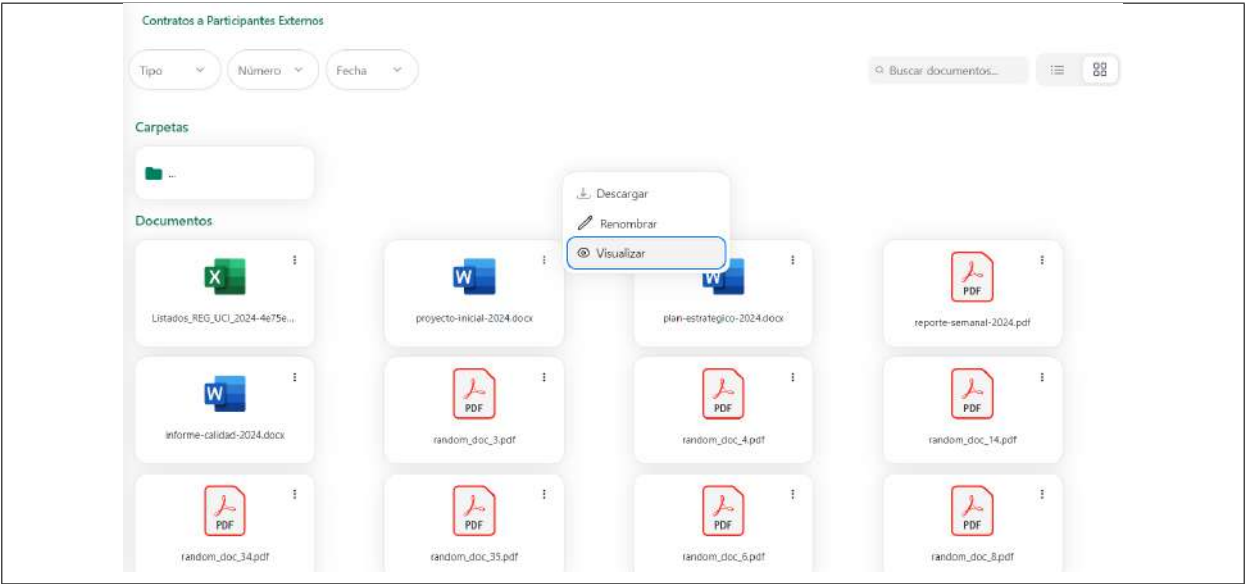


Tabla 7. Historia de Usuario 5: Descargar Documento

Historia de Usuario	
Número: 5	Nombre: Descargar Documento
Programador: Orlando Farid Giquez Tarafa	Iteración Asignada: 2
Prioridad en negocio: Alta	Tiempo Estimado: 3 días
Riesgo en Desarrollo: N/A	—
Descripción: - Esta funcionalidad permite a los usuarios descargar documentos de manera sencilla y accesible. - El sistema debe ofrecer opciones de descarga en formatos comunes, como PDF o Microsoft Office Word. - La interfaz de descarga debe ser clara, permitiendo a los usuarios seleccionar fácilmente los documentos que desean descargar.	
Observaciones: - Se debe garantizar que la descarga sea segura y no interrumpa el flujo de trabajo del usuario. - Es importante que el sistema notifique al usuario una vez que la descarga se haya completado.	
Interfaz Gráfica:	

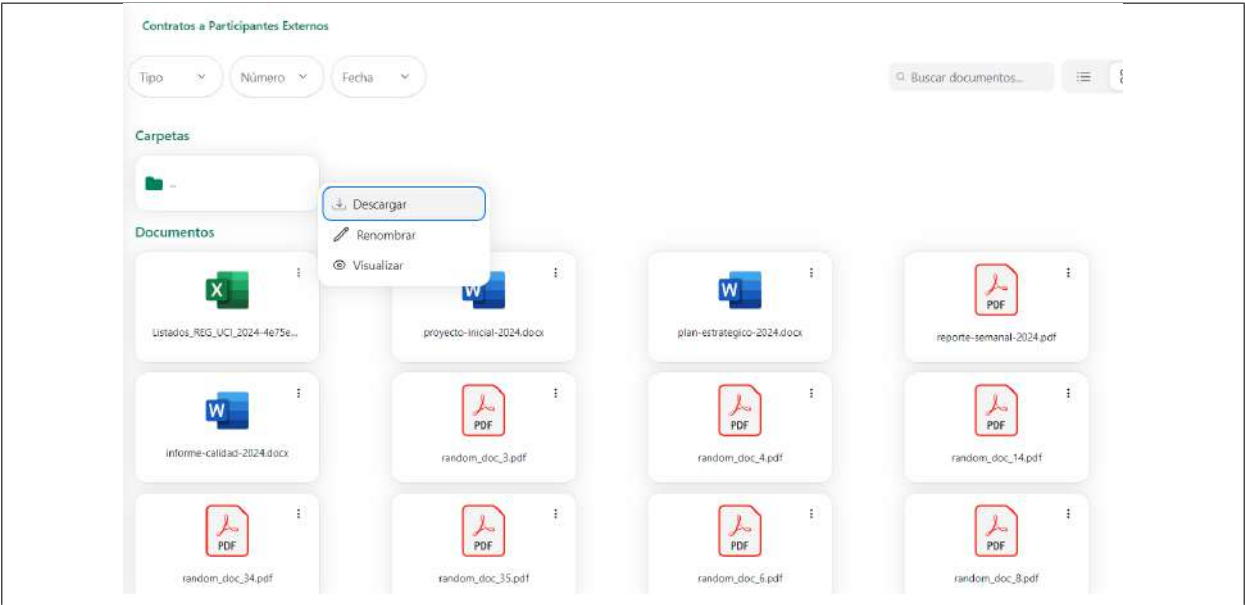


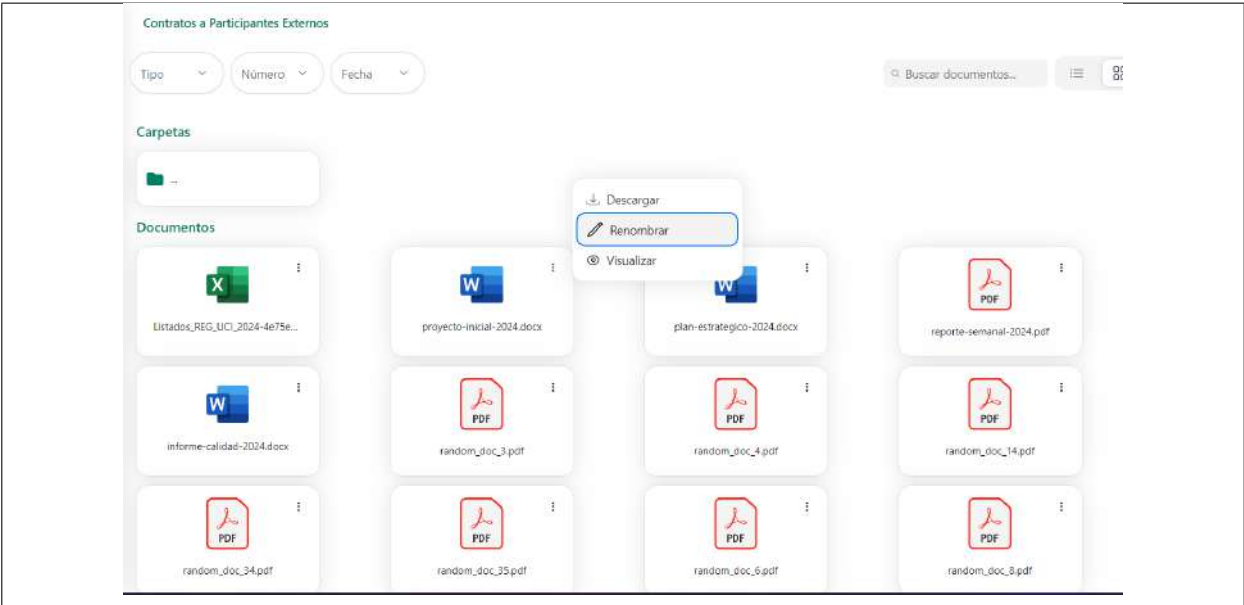
Tabla 8. Historia de Usuario 6: Ordenar Documentos

Historia de Usuario	
Número: 6	Nombre: Ordenar Documentos
Programador: Orlando Farid Giquez Tarafa	Iteración Asignada: 2
Prioridad en negocio: Media	Tiempo Estimado: 3 días
Riesgo en Desarrollo: N/A	—
Descripción: - El usuario necesita la capacidad de ordenar documentos almacenados en el sistema según diferentes criterios, como fecha, nombre o tipo de documento. - El sistema debe proporcionar una interfaz intuitiva que permita al usuario seleccionar el criterio de ordenación y visualizar los resultados en tiempo real. - Se debe garantizar que la ordenación no afecte la integridad de los documentos ni la organización de la biblioteca digital.	
Observaciones: - El sistema debe permitir al usuario revertir la ordenación para volver al orden original de los documentos. - Es recomendable incluir opciones de filtrado adicional para ayudar al usuario a encontrar documentos específicos rápidamente.	
Interfaz Gráfica:	

Nombre	Propietario	Última modificación	Tipo	Acciones
informe-calidad-2024.docx	Usuario	2024-09	Contratos a Participantes Externos 19	
listados_REG_UCL_2024-4e75eb9b...	Usuario	2024-6	Contratos a Participantes Externos	
plan-estrategico-2024.docx	Usuario	2024-08	Contratos a Participantes Externos 8	
proyecto-inicial-2024.docx	Usuario	2024-02	Contratos a Participantes Externos 2	
random_doc_100.pdf	Usuario	2024-06	Contratos a Participantes Externos 8	
random_doc_102.pdf	Usuario	2024-02	Contratos a Participantes Externos 6	
random_doc_107.pdf	Usuario	2024-07	Contratos a Participantes Externos 16	
random_doc_109.pdf	Usuario	2024-02	Contratos a Participantes Externos 3	
random_doc_117.pdf	Usuario	2024-02	Contratos a Participantes Externos 6	
random_doc_121.pdf	Usuario	2024-11	Contratos a Participantes Externos 7	

Tabla 9. Historia de Usuario 7: Renombrar Documento

Historia de Usuario	
Número: 7	Nombre: Renombrar Documento
Programador: Orlando Farid Giquez Tarafa	Iteración Asignada: 2
Prioridad en negocio: Media	Tiempo Estimado: 3 días
Riesgo en Desarrollo: N/A	—
Descripción:	
- Esta funcionalidad permite a los usuarios cambiar el nombre de un documento existente de manera sencilla. - El sistema debe proporcionar un campo de entrada donde los usuarios puedan ingresar el nuevo nombre del documento. - La interfaz debe ser intuitiva, asegurando que los usuarios puedan renombrar documentos sin complicaciones.	
Observaciones:	
- Se debe garantizar que el nuevo nombre no cause conflictos con otros documentos existentes. - Es importante que el sistema notifique al usuario una vez que el documento haya sido renombrado exitosamente.	
Interfaz Gráfica:	



2.4. Plan de iteraciones

Tabla 10. Plan de Iteraciones

Iteración	Historias de Usuario	Duración (semanas)
1	1. Visualizar Estructura de Carpetas y Documentos 2. Filtrar Documentos 3. Buscar Documentos	3
2	4. Visualizar Documento 5. Descargar Documento 6. Ordenar Documentos 7. Renombrar Documento	3
Total		6

2.5. Arquitectura

La arquitectura de software se refiere a la estructura y organización fundamental de un sistema de software, que define sus componentes, sus interacciones y las pautas para su diseño y desarrollo. Esta arquitectura actúa como un plano que guía el proceso de construcción del software, asegurando que los diferentes elementos del sistema trabajen juntos de manera coherente. Una buena arquitectura de software no solo aborda los requisitos funcionales, sino que también considera aspectos no funcionales como la escalabilidad, la seguridad y el rendimiento. Además, la arquitectura de software es crucial para facilitar el mantenimiento y la evolución del sistema a lo largo del tiempo, permitiendo adaptaciones a cambios en los requisitos o en el entorno tecnológico (Bass; Clements y Kazman, 2012). En el desarrollo del sistema GAPID, se implementó una Clean Architecture en la parte del frontend, lo que permitió una separación clara de las responsabilidades y una mejor organización del código. La Clean Architecture es un enfoque que promueve la independencia de los frameworks, la base de datos y cualquier otra preocupación externa, lo que facilita la prueba y el mantenimiento del software. Este modelo arquitectónico se basa en la idea de que las reglas de negocio deben ser independientes de los detalles de implementación, lo que permite a los desarrolladores realizar cambios en el sistema sin afectar su lógica central (R. C. Martin, 2017a). La aplicación de este enfoque en GAPID ha permitido que el módulo de gestión documental sea más escalable y fácil de entender. Al estructurar el frontend siguiendo los principios de la Clean Architecture, se logró una mayor modularidad, lo que facilita la integración con otros módulos del sistema y mejora la experiencia del usuario. Además, esta arquitectura ayuda a mantener un código más limpio y organizado, lo que a su vez reduce el tiempo necesario para realizar pruebas y correcciones.

- **Capa de Dominio:** Esta capa contiene los modelos de datos y la lógica del negocio, que son fundamentales para el funcionamiento del sistema. Aquí se definen las reglas y comportamientos que rigen el dominio específico del sistema.
- **Capa de Datos:** En esta capa se gestionan las operaciones relacionadas con el almacenamiento y recuperación de datos. Utiliza repositorios para interactuar con la base de datos, permitiendo realizar consultas y transacciones de manera eficiente. La separación de la lógica de acceso a datos ayuda a mantener el código limpio y facilita el mantenimiento.
- **Capa de Presentación (Interfaz de Usuario):** Esta capa es responsable de la presentación y la interacción con el usuario. Se encarga de mostrar la información al usuario y recibir sus entradas. La capa de UI se diseñó para ser intuitiva y accesible, mejorando la experiencia del usuario al interactuar con el sistema.

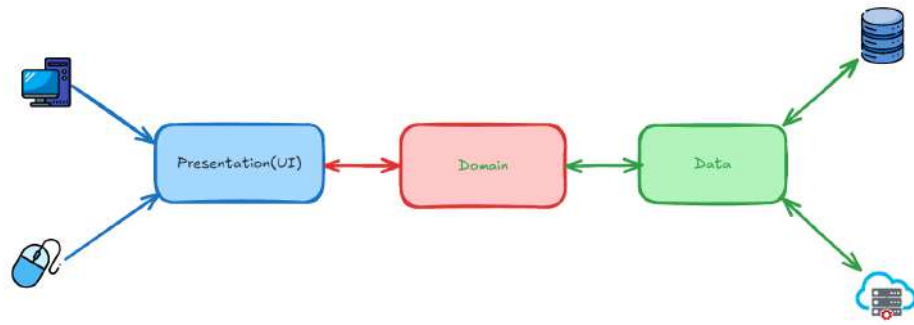


Figura 2. Clean Architecture. Fuente: Elaboración Propia.

2.6. Modelo de datos

Un modelo de datos es una representación abstracta que describe cómo se organizan, almacenan y gestionan los datos dentro de un sistema. Este modelo define las estructuras de datos, las relaciones entre ellas y las reglas que rigen su uso, lo que permite a los desarrolladores y analistas comprender la información que el sistema manejará. Un modelo de datos bien diseñado es fundamental para garantizar la integridad y accesibilidad de la información, facilitando su implementación en bases de datos y asegurando que los datos sean coherentes a lo largo del tiempo (Elmasri y Navathe, 2015). Además, un modelo de datos eficaz proporciona una base sólida para el análisis y optimización de las operaciones sobre los datos, mejorando el rendimiento del sistema y asegurando la escalabilidad del mismo (Silberschatz; Korth y Sudarshan, 2019).

En la siguiente figura se muestra el modelo de datos correspondiente a un documento en la base de datos del GAPID:

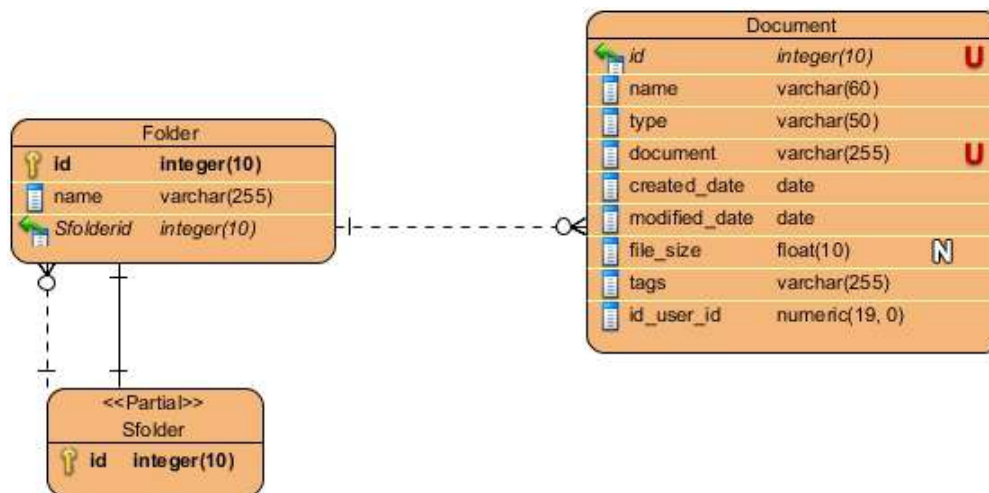


Figura 3. Modelo de datos. Fuente: Elaboración Propia.

2.7. Patrones de diseño

Un patrón de diseño es una solución general y reutilizable a un problema recurrente que se presenta en un contexto específico dentro del diseño de software. Estos patrones sirven como guías que ayudan a los desarrolladores a crear sistemas más robustos, mantenibles y escalables. Los patrones de diseño son “soluciones típicas a problemas comunes en el diseño de software” y proporcionan un vocabulario común para discutir y documentar el diseño del software (Gamma; Helm; R. Johnson y Vlissides, 1994).

GRASP (General Responsibility Assignment Software Patterns)

GRASP es un conjunto de principios que guían la asignación de responsabilidades en el diseño orientado a objetos. Introducido por Craig Larman en su libro *Applying UML and Patterns* (Larman, 2004), GRASP ayuda a los diseñadores a decidir qué clase o objeto debe asumir una determinada responsabilidad dentro del sistema. Este enfoque no solo mejora la organización del código, sino que también promueve la claridad y la cohesión en el diseño del software.

Patrones GRASP utilizados:

1. Expert (Experto en Información):

- **Descripción:** El patrón Expert asigna tareas a la clase que tiene la información necesaria para realizarlas, mejorando la cohesión y reduciendo la dependencia entre objetos (E. Freeman y Robson, 2020).
- **Aplicación:** En el módulo de gestión documental, el patrón Expert se implementa en la clase *DocumentationDataSourcesImpl*, que centraliza la responsabilidad de obtener documentos y carpetas desde el backend. La clase tiene acceso a toda la información requerida (URLs y parámetros) para ejecutar las solicitudes HTTP, por lo que es la “experta” en esta tarea, siguiendo el principio de asignación de responsabilidades a la entidad que tiene los datos necesarios.

```
export const DocumentationDataSourcesImpl = {
  restClient: AxiosInstance,
}: DocumentationDataSources => ({
  getFolders: async function (): Promise<FolderModel[]> {
    const { path } = generateUrlAndQuery({
      version: VERSION,
      basePath: 'folder/',
    });
    console.log(path);

    return (await restClient.get(path)).data;
  },

  getDocuments: async function (
    folderId: string,
  ): Promise<Document[] | Record<string, Document[]>> {
    const { path } = generateUrlAndQuery({
      version: VERSION,
      basePath: `folder/${folderId}/${PROJECT_ID}`,
    });
    return (await restClient.get(path)).data;
  },
});
```

Figura 4. Expert. Fuente: Elaboración Propia.

2. High Cohesion (Alta Cohesión):

- **Descripción:** La alta cohesión se refiere a la medida en que los elementos dentro de un módulo o componente están relacionados y contribuyen a un único propósito o responsabilidad bien definida. Los módulos con alta cohesión tienen funcionalidades enfocadas, lo que facilita su comprensión, mantenimiento y reutilización (Gazar, 2024).
- **Aplicación:** El componente *DocumentsTable* tiene una responsabilidad clara ,mostrar una tabla de documentos y facilitar la navegación entre carpetas. La lógica de búsqueda también está integrada dentro del mismo componente, lo que refuerza su alta cohesión y simplifica el mantenimiento del código (Gazar, 2024).

3. Creator (Creador):

- **Descripción:** El patrón Creator establece que una clase debe ser responsable de la creación de instancias de otra clase si tiene la información o la lógica necesaria para crearla. Este patrón ayuda a mantener una cohesión alta dentro del sistema al agrupar la lógica de creación con el objeto que la utiliza (E. Freeman y Robson, 2020).
- **Aplicación:** En la implementación de *DocumentationRepositoryImpl*, el patrón Creator se aplica al designar al repositorio como el responsable de la creación de instancias de documentos y carpetas. Al estar directamente relacionado con la gestión de los datos de documentación, el repositorio tiene el conocimiento necesario para instanciar estos objetos, lo que facilita su creación

y gestión.

Patrones de Diseño del Gang of Four (GoF)

Los patrones de diseño del Gang of Four (GoF) son un conjunto influyente de patrones ampliamente utilizados en el desarrollo de software orientado a objetos. Estos patrones se dividen en tres categorías principales:

1. **Patrones Creacionales:** Se centran en la creación de objetos y ayudan a abstraer el proceso de instanciación (Robert C. Martin, 2018). Ejemplos incluyen el patrón Singleton, el patrón Repository y el patrón de Inyección de Dependencias (E. Freeman y Robson, 2020).
2. **Patrones Estructurales:** Se ocupan de cómo las clases y objetos se componen para formar estructuras más grandes. Ejemplos incluyen Adapter, Composite y Decorator (Shallow, 2019).
3. **Patrones Comportamentales:** Se centran en cómo los objetos interactúan y comunican entre sí. Ejemplos incluyen Strategy, Command y Template Method (J. Miller, 2021).

Los patrones GoF proporcionan soluciones probadas para problemas comunes en el diseño orientado a objetos, mejorando así la reutilización del código y facilitando la comunicación entre los desarrolladores sobre las mejores prácticas en el diseño (E. Freeman y Robson, 2020).

Patrones GOF utilizados:

1. Composite (Objeto Compuesto):

- **Descripción:** El patrón Composite permite representar jerarquías de partes-todo, compitiendo objetos en estructuras de árbol, lo que permite a los clientes tratar objetos individuales y composiciones de objetos de manera uniforme (Shallow, 2019).
- **Aplicación:** En el código, la estructura *FolderOrDocument* se utiliza para manejar tanto documentos como carpetas. Las carpetas pueden contener otros documentos o carpetas, formando una estructura de árbol. Esto permite que el componente *DocumentsTable* maneje la visualización y la lógica de navegación de manera uniforme, ya que tanto las carpetas como los documentos pueden ser tratados como elementos dentro de la misma jerarquía.

```
import { Folder } from './Folder';  
import { Document } from './Document';  
  
export type FolderOrDocument = Folder | Document
```

Figura 5. Composite. Fuente: Elaboración Propia.

Repository (Repositorio):

- **Descripción:** El patrón Repository actúa como un intermediario entre la lógica de negocio y la persistencia de datos, facilitando el acceso a los datos y promoviendo la separación de preocupaciones (Robert C. Martin, 2018).
- **Aplicación:** En el código, *DocumentationRepositoryImpl* se encarga de interactuar con las fuentes de datos para obtener documentos y carpetas. Este patrón permite abstraer los detalles de la implementación de la persistencia, lo que facilita el mantenimiento y las pruebas del sistema.

```
export const DocumentationRepositoryImpl = (
  documentationDataSources: DocumentationDataSources,
): DocumentationRepository => ({
  getDocuments: async function (
    folderId: string,
  ): Promise<Either<string, Document[] | Record<string, Document[]>>> {
    try {
      const response = await documentationDataSources.getDocuments(folderId);

      return SuccessData(response);
    } catch (error) {
      return ErrorData('Error en la petición');
    }
  },
  getFolders: async function (): Promise<Either<string, Folder[]>> {
    try {
      const response = await documentationDataSources.getFolders();
      return SuccessData(mappedResponse(response));
    } catch (error) {
      return ErrorData('Error en la petición');
    }
  }
});
```

Figura 6. Repository. Fuente: Elaboración Propia.

Dependency Injection (Inyección de Dependencias):

- **Descripción:** La inyección de dependencias es un patrón que permite inyectar dependencias en un objeto en lugar de que este las cree por sí mismo, lo que favorece la flexibilidad y la prueba del código (J. Miller, 2021).
- **Aplicación:** En el código la inyección de dependencias se manifiesta a través de la integración del *documentationRepository* en el *useDocumentationStore*. Esta práctica facilita la separación de responsabilidades, ya que el store no necesita preocuparse por la lógica de acceso a datos; en su lugar, delega esa tarea al repositorio. Esto no solo mejora la testabilidad del código al permitir la simulación de dependencias durante las pruebas, sino que también fomenta una arquitectura más limpia, donde los componentes son fácilmente intercambiables y modificables sin afectar a otros.

```
export const documentationRepository = injector<DocumentationRepository>(  
  DocumentationSymbols.DOCUMENTATION_REPOSITORY,  
);
```

Figura 7. Inyección de Dependencias. Fuente: Elaboración Propia.

2.8. Tarjetas CRC

Tabla 11. Tarjeta CRC: “DocumentationDataSource“

Tarjeta CRC:	
Clase	DocumentationDataSource
Responsabilidad	• getFolders(): Permite recuperar de la fuente de datos, la lista de carpetas disponibles en el sistema de gestión documental. • getDocuments(): Permite obtener de la fuente de datos, el listado de documentos disponibles asociados a una carpeta en particular.
Colaboración	• DocumentationRepositoryImpl • Folder • Document

Tabla 12. Tarjeta CRC: “DocumentationRepository“

Tarjeta CRC:	
Clase	DocumentationRepository
Responsabilidad	• getFolders(): Define la operación de obtener una lista de carpetas, pero no especifica cómo deben ser obtenidas. • getDocuments(): Define la operación de obtener una lista de documentos asociados a una carpeta, pero no especifica cómo deben ser obtenidas.

Colaboración	• DocumentationRepositoryImpl • DocumentationDataSource • useDocumentationStore • Folder • Document
---------------------	---

Tabla 13. Tarjeta CRC: “DocumentationRepositoryImpl”

Tarjeta CRC:	
Clase	DocumentationRepositoryImpl
Responsabilidad	• getFolders(): Implementa el método getFolders() definido en el repository, para recuperar los datos y manejará cualquier procesamiento adicional, como filtrado o transformación, si es necesario. • getDocuments(): Implementa el método getDocuments() definido en el repository, que se encarga de obtener los documentos asociados a una carpeta específica desde una fuente de datos.
Colaboración	• DocumentationDataSource • DocumentationRepository • Folder • Document

Tabla 14. Tarjeta CRC: “useDocumentationStore”

Tarjeta CRC:	
Clase	useDocumentationStore

Responsabilidad	• getFolders(): Obtiene todas las carpetas desde el repository y actualiza el estado global (data) con los resultados. Maneja posibles errores y establece un mensaje de error en el estado (error) en caso de fallos. • updateDocumentsIntoFolders(): Implementa un método que permite a la UI obtener los documentos asociados a una carpeta específica. Mantiene el estado de estos documentos y los actualiza dinámicamente según la carpeta seleccionada.
Colaboración	• Document • Folder • DocumentsView • DocumentsGrid • MenuBar • DocumentsTable

Tabla 15. Tarjeta CRC: “Folder“

Tarjeta CRC:	
Clase	Folder
Responsabilidad	

Colaboración	• Document • DocumentationRepositoryImpl • DocumentationRepository • DocumentationDataSources • DocumentsTable • DocumentsGrid • DocumentsBreadCrumb • MenuBar
---------------------	---

Tabla 16. Tarjeta CRC: “Document“

Tarjeta CRC:	
Clase	Document
Responsabilidad	
Colaboración	• Folder • DocumentationRepositoryImpl • DocumentationRepository • DocumentationDataSources • DocumentsTable • DocumentsGrid • DocumentsBreadCrumb • MenuBar

Tabla 17. Tarjeta CRC: “DocumentsView“

Tarjeta CRC:	
Clase	DocumentsView

Responsabilidad	
Colaboración	• MenuBar • DocumentsGrid • DocumentsTable • DocumentsBreadCrumbs • useDocumentationStore

Tabla 18. Tarjeta CRC: “DocumentsGrid”

Tarjeta CRC:	
Clase	DocumentsGrid
Responsabilidad	
Colaboración	• Document • Folder • useDocumentationStore • DocumentsView

Tabla 19. Tarjeta CRC: “DocumentsTable”

Tarjeta CRC:	
Clase	DocumentsTable
Responsabilidad	
Colaboración	• Document • Folder • useDocumentationStore

Conclusiones parciales

En este capítulo se exploraron diversas dimensiones esenciales en el desarrollo del módulo de gestión documental para GAPID, lo que permitió obtener conclusiones significativas sobre el diseño de la solución propuesta. La implementación de una arquitectura bien definida y un enfoque centrado en el usuario han sido pilares fundamentales para el éxito del proyecto, asegurando que cada aspecto del sistema esté alineado con las necesidades de los usuarios finales.

- La definición de Historias de Usuario ha sido crucial para comprender las necesidades y expectativas de los usuarios finales. Este enfoque ha proporcionado una guía sólida para el desarrollo de funcionalidades, asegurando que el producto final se alinee con las expectativas del cliente.
- La estimación del tiempo para la implementación de las Historias de Usuario es un componente crítico en el proceso de desarrollo. Al analizar y cuantificar el tiempo necesario para cada HU, se logró una planificación más precisa y un control eficiente del proyecto, facilitando la entrega final al cliente.
- La implementación de la Clean Architecture ha permitido una clara separación entre las capas del sistema, facilitando su mantenimiento y escalabilidad. Esto asegura que los cambios se realicen sin afectar otras partes del sistema, lo que es crucial en un entorno dinámico.
- La adopción de patrones de diseño ha permitido organizar los componentes del sistema de manera efectiva. Esto no solo optimiza el desarrollo inicial, sino que también simplifica el proceso de mantenimiento a largo plazo, ayudando a reducir la deuda técnica.

Implementación y pruebas de la solución

Este capítulo se centra en la implementación y validación de la solución propuesta. Se presenta el diagrama que ilustra el despliegue del sistema a implementar. Además se exponen las tareas de ingeniería generadas para cada HU definida y se analizan los diferentes tipos de pruebas realizadas y los resultados obtenidos, con el objetivo de asegurar la calidad del programa y verificar que los requisitos del sistema se estén cumpliendo adecuadamente.

3.1. Estándar de codificación

Un estándar de codificación es un conjunto de reglas y directrices que establecen cómo se debe escribir y estructurar el código fuente de un programa. Estos estándares son fundamentales para garantizar la calidad, legibilidad y mantenibilidad del software, facilitando la colaboración entre diferentes desarrolladores. Al seguir un estándar de codificación, se promueve la consistencia en el estilo de programación, lo que reduce la probabilidad de errores y mejora la comprensión del código por parte de otros programadores (McConnell, 2004). Además, los estándares pueden incluir convenciones sobre la nomenclatura de variables, la organización de archivos y el uso de comentarios, lo que contribuye a crear un entorno de desarrollo más eficiente y profesional (R. C. Martin, 2017b).

Estructura de carpetas

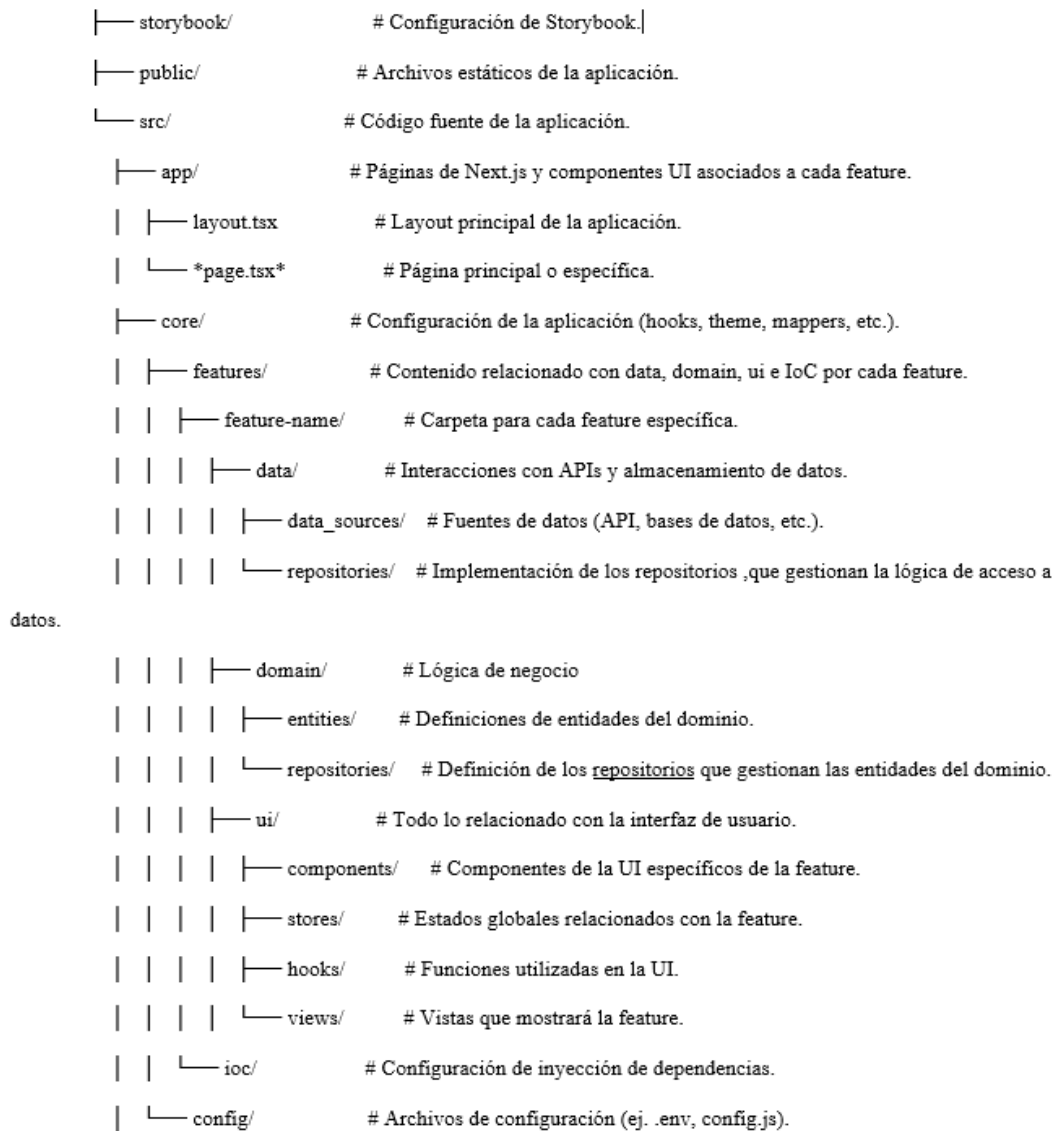


Figura 8. Estructura de carpetas. Fuente: Elaboración Propia.

Nombrado de Archivos y Carpetas

- Los nombres deben ser en minúsculas y separados por guiones bajos.
 - `my_hook.ts`, `my_components`
- Los nombres deben ser descriptivos.
- Los nombres deben ser en inglés.

- Los nombres de los archivos que sean componentes deben ser en `UpperCamelCase`.
 - `TextFieldComponent.tsx`

Nombrado de Variables y Funciones

- Los nombres deben ser en `lowerCamelCase`, descriptivos y en inglés.
 - `myVariable`
 - `myFunction()`

Nombres de Componentes

- Los nombres deben ser en `UpperCamelCase`, descriptivos y en inglés.
- Los nombres deben incluir el sufijo `Component`.
 - `FilterDocumentComponent`

Estructura de los Componentes

- Se declaran los componentes con la palabra clave `function`.
- Es necesario que los componentes que usen props tengan un tipo definido para las mismas.

Ejemplo:

```
import React from 'react';
type MyComponentProps = {
  myProp: string;
};
export default function MyComponent(props: MyComponentProps) {
  return (
    <div>
      <h1>{props.myProp}</h1>
    </div>
  );
}
```

Estilos

- Se usa Tailwind CSS para los estilos.
- Se aplican las clases de Tailwind directamente en los componentes.

Ejemplo:

```
import React from 'react';
interface MyComponentProps {
  myProp: string;
}
function MyComponent({myProp}: MyComponentProps) {
  return (
    <div className="bg-gray-100 p-5 rounded-lg">
      <h1 className="text-xl font-bold">{myProp}</h1>
    </div>
  );
}
export default MyComponent;
```

3.2. Tareas de ingeniería

Tabla 20. Tarea de Ingeniería 1: Visualización de Estructura de Carpetas y Documentos

Tarea de Ingeniería	
Número de tarea: 1	Número de Historia de Usuario: 1
Nombre de la tarea: Visualización de Estructura de Carpetas y Documentos	
Tipo de tarea: Desarrollo	Tiempo Estimado: 5 días
Programador responsable: Orlando Farid Giquez Tarafa	
Descripción: Implementar la interfaz que permita a los usuarios visualizar las carpetas disponibles y los documentos contenidos en cada carpeta.	

Tabla 21. Tarea de Ingeniería 2: Filtrado de Documentos

Tarea de Ingeniería	
Número de tarea: 2	Número de Historia de Usuario: 2
Nombre de la tarea: Filtrado de Documentos	
Tipo de tarea: Desarrollo	Tiempo Estimado: 5 días
Programador responsable: Orlando Farid Giquez Tarafa	
Descripción: Desarrollar las funcionalidades y la interfaz necesaria para aplicar filtros a la lista de documentos según criterios seleccionados por el usuario.	

Tabla 22. Tarea de Ingeniería 3: Búsqueda de Documentos

Tarea de Ingeniería	
Número de tarea: 3	Número de Historia de Usuario: 3
Nombre de la tarea: Búsqueda de Documentos	
Tipo de tarea: Desarrollo	Tiempo Estimado: 4 días
Programador responsable: Orlando Farid Giquez Tarafa	
Descripción: Crear un campo de búsqueda que permita a los usuarios buscar documentos utilizando palabras clave, mostrando resultados relevantes en tiempo real.	

El resto de las tareas de ingeniería se pueden observar en el apéndice *A de los anexos*.

3.3. Diagrama de componentes

Los diagramas de componentes son herramientas utilizadas para representar la vista estática de un sistema. Estos diagramas ilustran la organización y las dependencias entre un conjunto de componentes. No es necesario que un diagrama incluya todos los componentes del sistema; generalmente, se crean en partes (Sommerville, 2011). El diagrama de componentes es particularmente útil para comprender la arquitectura física de un sistema, ya que muestra cómo se relacionan los diferentes módulos y cómo interactúan a nivel de implementación, lo que ayuda a visualizar las dependencias y las interfaces entre los componentes (Pressman, 2014).

La siguiente ilustración muestra el diagrama de componentes del módulo:

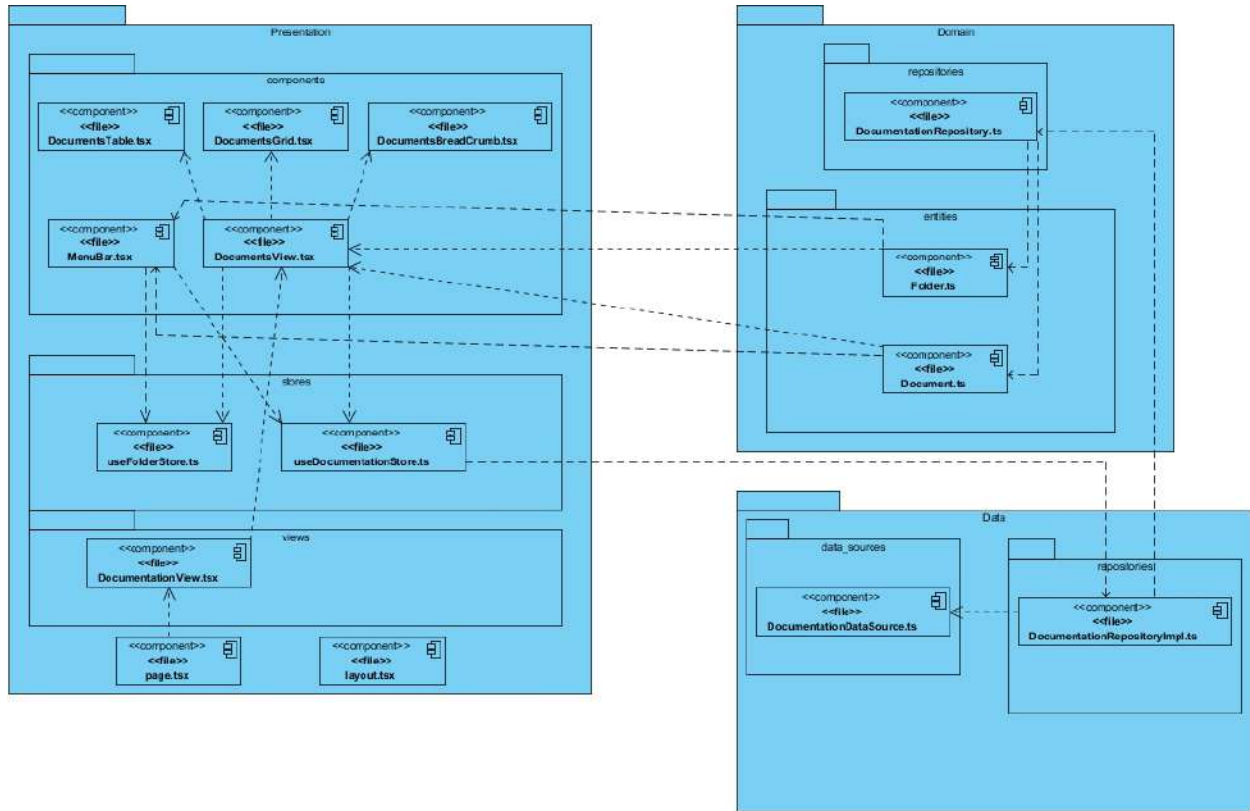


Figura 9. Diagrama de Componentes. Fuente: Elaboración Propia.

3.4. Diagrama de despliegue

Un diagrama de despliegue es una representación visual que ilustra la arquitectura física de un sistema, mostrando cómo los componentes de software se distribuyen en el hardware. Este tipo de diagrama es fundamental para entender la disposición de los distintos elementos del sistema y sus interacciones en un entorno específico. Permite a los desarrolladores y arquitectos de software visualizar la relación entre los componentes, como servidores, bases de datos y dispositivos de red, así como las conexiones entre ellos (Sommerville, 2011). Al proporcionar una vista clara de la infraestructura del sistema, el diagrama de despliegue facilita la planificación, implementación y mantenimiento del sistema, asegurando que todos los elementos funcionen correctamente en conjunto (Pressman, 2014).

En la siguiente figura, que ilustra el diagrama de despliegue, se representan los componentes de hardware y sus interconexiones necesarios para implementar el sistema. Este modelo de despliegue incluye tres nodos principales: al menos una computadora del usuario, el servidor de la plataforma GAPID y el servidor de base de datos. Todas las computadoras cliente podrán conectarse a este servidor utilizando el protocolo de

comunicación HTTPS, lo que permitirá a los usuarios acceder a la plataforma desde cada estación de trabajo.

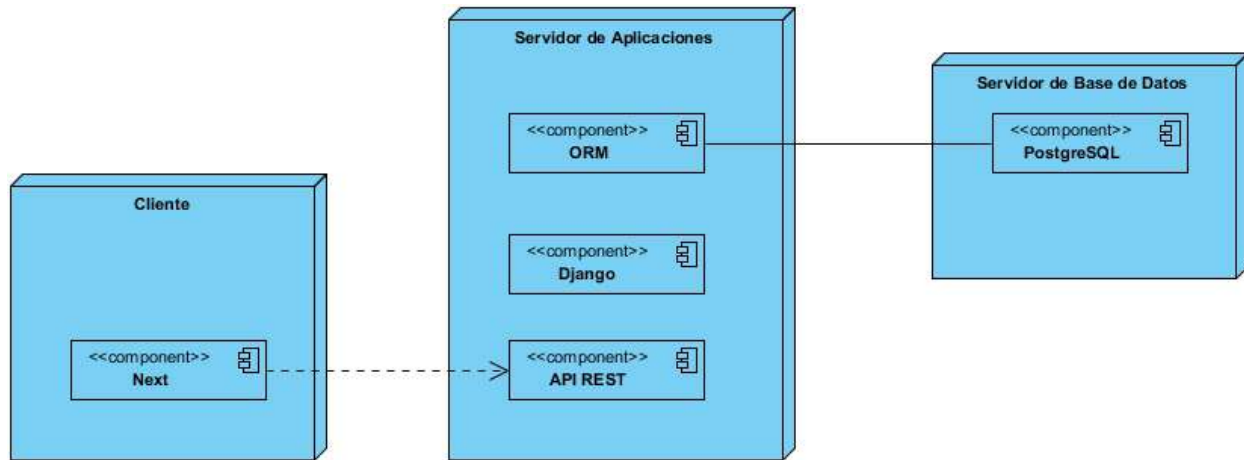


Figura 10. Diagrama de Despliegue. Fuente: Elaboración Propia.

3.5. Pruebas

Las pruebas de software son un conjunto de actividades sistemáticas diseñadas para evaluar la funcionalidad, rendimiento y calidad de un software antes de su implementación final. Estas pruebas son cruciales para identificar errores y fallos, asegurando que el software cumpla con los requisitos establecidos y funcione correctamente en diversas condiciones (Beck, 2001). Existen diferentes tipos de pruebas, como las pruebas unitarias, que verifican componentes individuales y las pruebas de aceptación, que determinan si el software satisface las necesidades del usuario final. La realización de pruebas exhaustivas no solo mejora la calidad del producto final, sino que también contribuye a la satisfacción del usuario al garantizar que el software cumpla con sus expectativas (Myers; Sandler y Badgett, 2011).

3.5.1. Estrategia de pruebas

Una estrategia de prueba es un plan detallado que define el enfoque y los métodos a utilizar para llevar a cabo las pruebas de un software, asegurando que se cumplan los requisitos de calidad y funcionalidad. Esta estrategia abarca la identificación de los tipos de pruebas a realizar, como pruebas unitarias y de aceptación, así como la definición de criterios de éxito y las herramientas que se emplearán. Al establecer una estrategia de prueba clara, se busca optimizar el proceso de prueba, minimizar riesgos y garantizar que el software funcione correctamente en diferentes escenarios antes de su lanzamiento (Desikan y Ramesh, 2006). Una estrategia bien formulada no solo mejora la calidad del producto final, sino que también contribuye a la eficiencia en el ciclo de desarrollo del software (Kaner; Falk y Nguyen, 1999).

3.5.2. Pruebas unitarias

Las pruebas unitarias son un tipo de prueba de software que evalúa el correcto funcionamiento de componentes individuales del sistema, conocidos como unidades, en un ambiente aislado. Estas pruebas, por lo general, se enfocan en verificar la lógica interna de funciones y métodos, identificando fallos de implementación antes de que las unidades se integren con otros componentes del sistema. Su importancia radica en que facilitan la detección temprana de errores, lo cual reduce costos y complejidad en fases posteriores del desarrollo (Meszaros, 2007). Además, el uso de pruebas unitarias fomenta un código más modular y mantenible, ya que al diseñar el software en componentes fácilmente testeables se promueve una estructura menos acoplada (Bach y Bolton, 2019).

Un aspecto fundamental de las pruebas unitarias es que ayudan a reforzar la confianza en la estabilidad del software, permitiendo a los desarrolladores hacer cambios en el código con menor riesgo de introducir nuevos errores (Fowler, 2021). Sin embargo, para que sean efectivas, deben aplicarse siguiendo buenas prácticas, como centrarse en una única funcionalidad de cada unidad y abarcar tanto casos regulares como escenarios límites o excepcionales (S. Freeman y Pryce, 2017).

Prueba de que el *DocumentationDataSources* obtiene las carpetas correctamente:

```
describe("DocumentationDataSourcesImpl", () => {
  let restClient: AxiosInstance;
  let dataSources: ReturnType<typeof DocumentationDataSourcesImpl>;

  beforeEach(() => {
    restClient = {
      get: jest.fn(),
    } as unknown as AxiosInstance;
    dataSources = DocumentationDataSourcesImpl(restClient);
  });

  it("should fetch folders", async () => {
    const mockFolders: FolderModel[] = [{ id: "1", name: "Folder 1" }];
    const expectedPath = "folder/";
    (restClient.get as jest.Mock).mockResolvedValueOnce({ data: mockFolders });

    const folders = await dataSources.getFolders();

    expect(folders).toEqual(mockFolders);
    expect(restClient.get).toHaveBeenCalledWith(expectedPath);
  });
});
```

Figura 11. Test al Data Source para obtener carpetas. Fuente: Elaboración Propia.

Prueba de que el estado global *useDocumentationStore* actualiza los documentos dentro de las carpetas al interactuar con estas:

```
jest.mock("../domain/repositories/DocumentationRepository");
jest.mock("../hooks");

describe("useDocumentationStore", () => {
  beforeEach(() => {
    jest.clearAllMocks();
  });

  it("should update documents into folders", async () => {
    const mockFolders: Folder[] = [{ id: "1", name: "Folder 1", children: [] }];
    (documentationRepository.getFolders as jest.Mock).mockResolvedValueOnce({
      _tag: "Right",
      right: mockFolders,
    });
    (fetchFoldersWithDocuments as jest.Mock).mockResolvedValueOnce([
      { id: "2", title: "Document 1" },
    ]);

    const { result } = renderHook(() => useDocumentationStore());

    await act(async () => {
      await result.current.getFolders();
    });

    await act(async () => {
      await result.current.updateDocumentsIntoFolders("1");
    });

    expect(result.current.data[0].children).toEqual([
      { id: "2", title: "Document 1" },
    ]);
    expect(result.current.updatedFolder).toEqual(result.current.data[0]);
    expect(result.current.isLoading).toBe(false);
  });
});
```

Figura 12. Test al estado global para actualizar documentos dentro de sus carpetas. Fuente: Elaboración Propia.

El resto de las pruebas unitarias se pueden observar en el apéndice *B de los anexos*.

Se llevaron a cabo tres iteraciones de pruebas unitarias. A continuación, se presenta un gráfico que resume los resultados obtenidos en cada iteración.

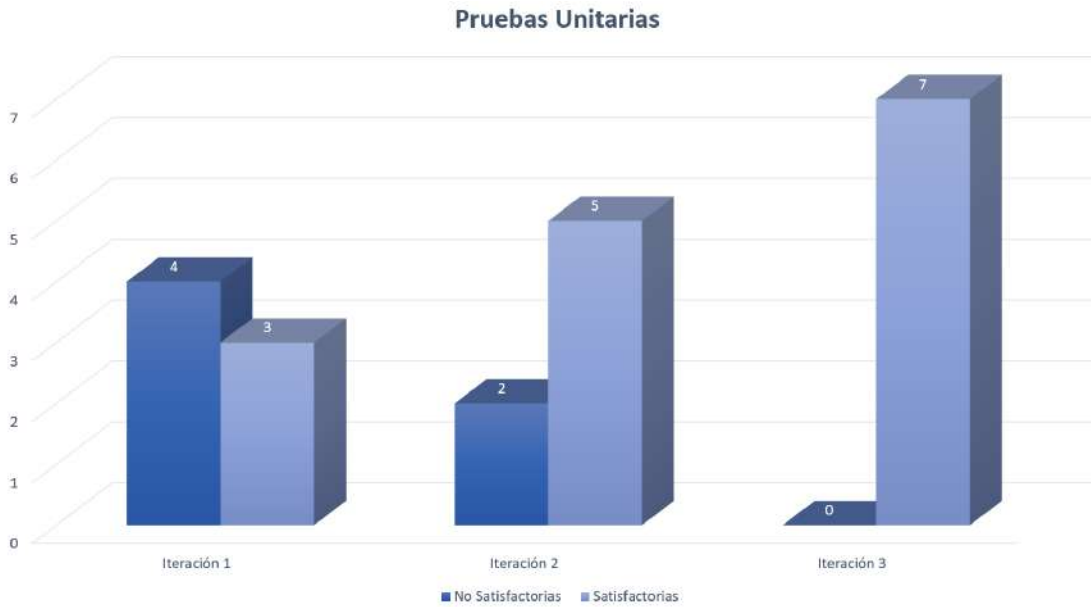


Figura 13. Análisis de las pruebas unitarias. Fuente: Elaboración Propia.

3.5.3. Pruebas de aceptación

Las pruebas de aceptación en el desarrollo de software representan una fase crítica en la validación del producto, asegurando que las funcionalidades cumplen con los criterios y requisitos establecidos por el usuario o cliente (Gockenbach, 2020). Este tipo de pruebas se centra en validar la experiencia final del usuario y la satisfacción de los objetivos del negocio, siendo generalmente realizadas en el entorno final del sistema (Evans, 2021). En su implementación, las pruebas de aceptación también permiten la detección de defectos que pudieron no ser evidentes en las etapas de pruebas unitaria, facilitando un enfoque de desarrollo más ágil y centrado en la calidad (Wake, 2019). A continuación, se describen los casos de pruebas de aceptación definidos:

Tabla 23. Prueba de Aceptación para Visualizar Estructura de Carpetas y Documentos

Caso de Prueba de Aceptación	
Código: HU1_P1	Historia de Usuario: 1
Nombre: Visualizar Estructura de Carpetas y Documentos	
Descripción: Verificar que los usuarios puedan acceder a la estructura de carpetas y documentos.	
Condiciones de Ejecución: El usuario debe estar autenticado y tener permisos adecuados.	

Pasos de Ejecución: 1. Iniciar sesión en el sistema. 2. Navegar a la sección "Programas". 3. Acceder a la sección "Proyectos". 4. Ir a la sección "Documentación". 5. Observar la representación visual de la estructura.
Resultado: Satisfactorio

Tabla 24. Prueba de Aceptación para Filtrar Documentos

Caso de Prueba de Aceptación	
Código: HU2_P1	Historia de Usuario: 2
Nombre: Filtrar Documentos	
Descripción: Verificar que los usuarios puedan aplicar filtros para visualizar documentos específicos.	
Condiciones de Ejecución: El usuario debe estar autenticado y tener acceso a documentos.	
Pasos de Ejecución: 1. Iniciar sesión en el sistema. 2. Navegar a la sección "Programas". 3. Acceder a la sección "Proyectos". 4. Ir a la sección "Documentación". 5. Aplicar filtros según criterios seleccionados. 6. Observar los resultados actualizados.	
Resultado: Satisfactorio	

Tabla 25. Prueba de Aceptación para Buscar Documentos

Caso de Prueba de Aceptación	
Código: HU3_P1	Historia de Usuario: 3
Name: Buscar Documentos	
Descripción: Verificar que el sistema permita buscar documentos utilizando palabras clave.	

Condiciones de Ejecución: El usuario debe estar autenticado.
Pasos de Ejecución: 1. Iniciar sesión en el sistema. 2. Navegar a la sección "Programas". 3. Acceder a la sección "Proyectos". 4. Ir a la sección "Documentación". 5. Ingresar palabras clave en el campo de búsqueda. 6. Observar los resultados presentados.
Resultado : Satisfactorio

El resto de las pruebas de aceptación se pueden observar en el apéndice *C de los anexos*.

Análisis de las pruebas de aceptación

Se llevaron a cabo tres iteraciones de pruebas de aceptación. A continuación, se presenta un gráfico que resume los resultados obtenidos en cada iteración.

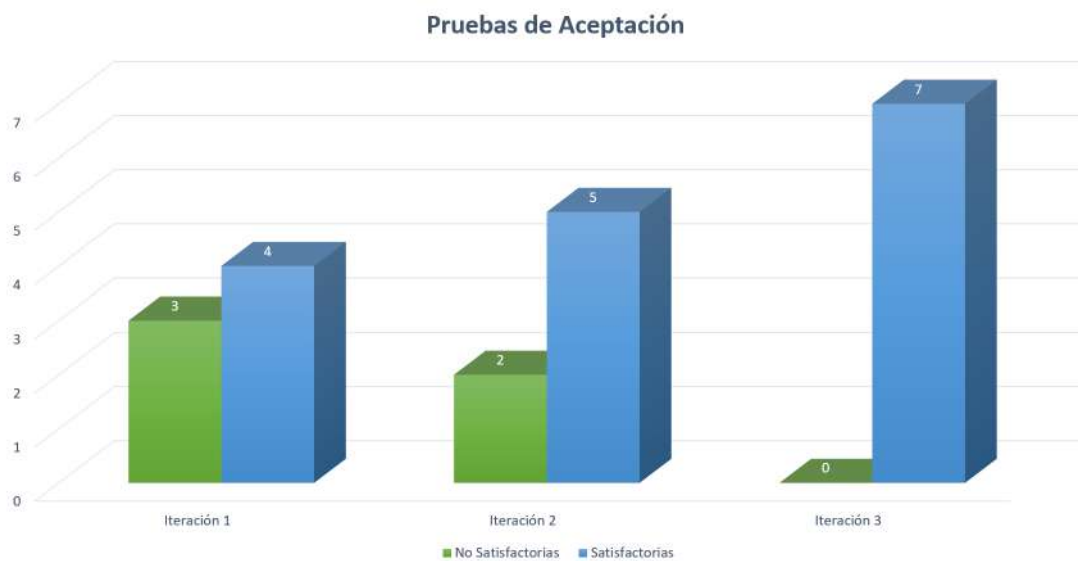


Figura 14. Análisis de las pruebas de aceptación. Fuente: Elaboración Propia.

Conclusiones parciales

Este capítulo ha sido fundamental en el desarrollo del módulo de gestión documental para la plataforma GAPID

- Se llevaron a cabo varias tareas de ingeniería orientadas a garantizar la calidad y el funcionamiento adecuado de los componentes del proyecto.
- Las pruebas de software desempeñaron un papel crucial en el desarrollo del sistema, garantizando que el producto final cumpliera con los requisitos establecidos y funcionara adecuadamente en diversas condiciones. La implementación de diferentes tipos de pruebas, como las unitarias y de aceptación, permitieron identificar y corregir errores en etapas tempranas del ciclo de desarrollo, mejorando así la calidad del software y la satisfacción del usuario.
- Las pruebas de software no solo constituyeron un requisito técnico, sino que también se consideraron una estrategia integral que contribuyó al éxito del desarrollo del módulo de gestión documental en GAPID. A través de un enfoque sistemático y bien estructurado en las pruebas, se logró no solo cumplir con los estándares de calidad, sino también proporcionar un producto final que estuviera acorde a las expectativas y necesidades de los usuarios finales.

Conclusiones

- La elaboración del marco teórico y metodológico permitió alcanzar una mayor comprensión del alcance del módulo de gestión documental dentro de la plataforma GAPID. Esto clarificó su objeto de estudio y garantizó la confiabilidad y seguridad de los documentos digitales.
- La adopción de herramientas y tecnologías modernas, como Visual Paradigm, Visual Studio Code, Next.js, React y TypeScript, ha optimizado cada fase del desarrollo. Estas herramientas no solo han mejorado la eficiencia del proceso, sino que también han facilitado el trabajo en equipo y garantizado altos estándares de calidad en la implementación y pruebas.
- Un análisis de sistemas homólogos demostró que la integración de un componente propio en GAPID es la manera más eficaz de eliminar la dependencia de herramientas externas. Esta estrategia minimiza los costos asociados a integraciones a través de API y garantiza que la funcionalidad se ajuste de manera más precisa a las necesidades específicas del sistema.
- A partir de los requisitos identificados en la investigación, se desarrolló un módulo que permite centralizar, organizar y controlar el ciclo de vida de los documentos asociados a programas y proyectos. La definición clara de la arquitectura del sistema y el diseño del diagrama de componentes fueron esenciales para establecer una base estructural sólida para la implementación del componente.
- La validez de la solución fue confirmada mediante una estrategia de pruebas, que permitió identificar y corregir discrepancias entre el comportamiento esperado del software y su funcionamiento real. Esto asegura la calidad, fiabilidad y seguridad del sistema antes de su lanzamiento.

Recomendaciones

- Diseñar e implementar un plan de capacitación para los usuarios finales, asegurando que aprovechen al máximo las funcionalidades implementadas.
- Explorar la integración del sistema con plataformas externas, como servicios de almacenamiento en la nube o sistemas de gestión documental utilizados en la industria.
- Incluir mecanismos de clasificación automática mediante técnicas de inteligencia artificial, como aprendizaje automático, para organizar los documentos por categorías.

Referencias bibliográficas

- AMBLER, S., 2006. *The Agile Unified Process (AUP)*. Ambyssoft Inc (vid. pág. 16).
- AMBLER, Scott W., 2006. *The Rational Unified Process: An Introduction*. Addison-Wesley Professional (vid. págs. 14, 15).
- BACH, James y BOLTON, Michael, 2019. *Lessons Learned in Software Testing: A Context-Driven Approach*. Hoboken, NJ: Wiley (vid. pág. 56).
- BASS, L.; CLEMENTS, P. y KAZMAN, R., 2012. *Software Architecture in Practice (3rd ed.)* Addison-Wesley (vid. pág. 37).
- BECK, K., 2001. *Manifesto for Agile Software Development*. Agile Alliance (vid. pág. 55).
- CITMA, 2019. *Resolución 287/2019 sobre Normas Técnicas*. Url: <https://www.citma.gob.cu/> (vid. pág. 1).
- CITMA, 2020. *Decreto Ley 7/2020 sobre Ciencia, Tecnología e Innovación*. Url: <https://www.citma.gob.cu/> (vid. pág. 1).
- CRUZ, J.; PÉREZ, A. y MARTÍNEZ, R., 2021. Gestión del conocimiento en instituciones científicas: Retos actuales. *Revista Cubana de Ciencias Sociales* (vid. pág. 2).
- DAVIES, J. y BROWN, R., 2020. The future of document management: Trends and strategies for the digital age. *International Journal of Information Management*. Vol. 55, págs. 102-118 (vid. pág. 7).
- DELGADO, J.; PÉREZ, R. y GONZÁLEZ, A., 2017. Adaptación de metodologías ágiles en proyectos universitarios: AUP UCI en la práctica. *Revista Cubana de Informática*. Vol. 32, n.º 2, págs. 50-65 (vid. pág. 16).
- DESIKAN, S. y RAMESH, G., 2006. *Software Testing: Principles and Practices*. Pearson Education India (vid. pág. 55).
- DOCUWARE, 2021. *DocuWare: Document management and workflow automation*. Url: <https://www.docuware.com> (vid. pág. 10).
- ELMASRI, R. y NAVATHE, S. B., 2015. *Fundamentals of Database Systems (7th ed.)* Pearson (vid. pág. 38).

- EVANS, Brian, 2021. *Acceptance Test Driven Development: Agile Methods in Practice*. Hoboken, NJ: Wiley (vid. pág. 58).
- EXCALIDRAW, 2023. *Herramienta de Diagramación Excalidraw*. Url: <https://excalidraw.com> (vid. pág. 18).
- FORRESTER RESEARCH, 2021. *Maximizing business efficiency through electronic document management systems*. Forrester Insights (vid. pág. 7).
- FOWLER, Martin, 2021. *Refactoring: Improving the Design of Existing Code*. 2nd. Boston, MA: Addison-Wesley Professional (vid. pág. 56).
- FRANCIS, Taylor &, 2020. *Research Methodologies in Software Engineering*. Taylor & Francis Group (vid. págs. 3, 4).
- FREEMAN, E. y ROBSON, E., 2020. *Head First Design Patterns: Building Extensible and Maintainable Object-Oriented Software*. 2nd. O'Reilly Media (vid. págs. 39-41).
- FREEMAN, Steve y PRYCE, Nat, 2017. *Growing Object-Oriented Software, Guided by Tests*. Boston, MA: Addison-Wesley Professional (vid. pág. 56).
- GAMMA, E.; HELM, R.; JOHNSON, R. y VLISSIDES, J. F., 1994. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley (vid. pág. 39).
- GARCÍA, M., 2020. GESPRO: Un sistema de gestión de proyectos en el contexto cubano. *Revista Cubana de Informática*. Vol. 25, n.º 1, págs. 30-45 (vid. págs. 11, 12).
- GARCÍA, M. y ALONSO, R., 2022. Blockchain and cloud-based solutions for document management in a digital economy. *Journal of Digital Transformation*. Vol. 12, n.º 3, págs. 154-165 (vid. pág. 6).
- GAZAR, 2024. *High Cohesion in Software Development: Best Practices and Implementation Strategies*. Url: <https://gazar.dev/clean-code/high-cohesion-in-software-development-best-practice> (vid. pág. 40).
- GITHUB, 2023. *Atom Editor*. Url: <https://atom.io/> (vid. pág. 19).
- GOCKENBACH, John, 2020. *Software Quality Assurance and Testing*. Boston, MA: Cengage Learning (vid. pág. 58).
- GONZÁLEZ, M. y LÓPEZ, T., 2020. La importancia del manejo documental en proyectos científicos. *Journal of Information Management* (vid. págs. 1, 2).
- GYSEL, M.; KÄLBENER, L.; GIERSCHE, W. y ZIMMERMANN, O., 2016. En: *Service Cutter: A Systematic Approach to Service Decomposition*. Springer, vol. 9846. Lecture Notes in Computer Science (vid. pág. 25).
- JOHNSON, M., 2020. Document control and compliance: Managing risk in a digital environment. *International Journal of Information Security*. Vol. 45, n.º 6, págs. 567-579 (vid. pág. 6).

- KANER, C.; FALK, J. y NGUYEN, H. Q., 1999. *Testing Computer Software (2nd ed.)* John Wiley & Sons (vid. pág. 55).
- KOTONYA, G. y SOMMERVILLE, I., 1998. *Requirements Engineering: Processes and Techniques*. Wiley (vid. pág. 27).
- LABS, Tailwind, 2023. *Tailwind CSS (v3.3.2)*. Url: <https://tailwindcss.com> (vid. pág. 21).
- LARMAN, C., 2004. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd ed.)* Prentice Hall (vid. pág. 39).
- LEE, K. y CHU, H., 2022. Metadata-based document management systems: Enhancing document retrieval in digital enterprises. *Journal of Information Science*. Vol. 48, n.º 4, págs. 671-690 (vid. pág. 5).
- LÓPEZ, R. y FERNÁNDEZ, S., 2021. Documentación científica: Un enfoque hacia su gestión eficaz. *Revista Cubana de Información* (vid. págs. 1, 2).
- LUCIDCHART, 2023. *Herramienta de Diagramación Lucidchart*. Url: <https://www.lucidchart.com> (vid. pág. 18).
- MARTIN, R. C., 2017a. *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Prentice Hall (vid. pág. 37).
- MARTIN, R. C., 2017b. *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall (vid. pág. 49).
- MARTIN, Robert C., 2018. *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Prentice Hall (vid. págs. 41, 42).
- MARTÍNEZ, D., 2021. eXcriba: Gestión documental en el sector público cubano. *Journal of Cuban Studies*. Vol. 15, n.º 2, págs. 112-128 (vid. pág. 12).
- MCCONNELL, S., 2004. *Code Complete: A Practical Handbook of Software Construction (2nd ed.)* Microsoft Press (vid. pág. 49).
- MESZAROS, Gerard, 2007. *XUnit Test Patterns: Refactoring Test Code*. Boston, MA: Addison-Wesley Professional (vid. pág. 56).
- META, 2023a. *Jest: Testing Framework (v29.6)*. Url: <https://jestjs.io> (vid. pág. 22).
- META, 2023b. *React: A JavaScript library for building user interfaces*. Url: <https://reactjs.org> (vid. pág. 20).
- M-FILES, 2019. *M-Files: Intelligent information management*. Url: <https://www.m-files.com> (vid. pág. 11).
- MICROSOFT, 2020. *SharePoint: Document management and collaboration solution*. Url: <https://www.microsoft.com/sharepoint> (vid. págs. 9, 10).
- MICROSOFT, 2023a. *TypeScript (v5.1.6): JavaScript con tipos*. Url: <https://www.typescriptlang.org> (vid. pág. 20).

- MICROSOFT, 2023b. *Visual Studio Code (v1.71)*. Url: <https://code.visualstudio.com> (vid. pág. 19).
- MILLER, John, 2021. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley (vid. págs. 41, 42).
- MILLER, T., 2020. Document Management in Scientific Research: Best Practices and Methodologies. *Journal of Research Management*. Vol. 34, n.º 2, págs. 45-62 (vid. pág. 8).
- MYERS, G. J.; SANDLER, C. y BADGETT, T., 2011. *The Art of Software Testing*. John Wiley & Sons (vid. pág. 55).
- OCDE, 2022. *Effective document management for innovation in science and technology*. OECD Publishing (vid. pág. 9).
- PARADIGM, Visual, 2023. *Modelado UML con Visual Paradigm 8.0*. Url: <https://www.visual-paradigm.com> (vid. pág. 18).
- PETROV, M. y MÖLLER, L., 2019. OCR technology and the future of digital document management. *International Journal of Digital Management*. Vol. 27, n.º 3, págs. 115-128 (vid. pág. 6).
- PRESSMAN, R. S., 2014. *Ingeniería de software: Un enfoque práctico*. McGraw-Hill Education (vid. págs. 27, 53, 54).
- PRESSMAN, R. S. y MAXIM, B., 2015. *Software Engineering: A Practitioner's Approach*. 8th. McGraw-Hill Education (vid. pág. 15).
- RIVERA, J.; SANCHEZ, P. y TORRES, L., 2021. The impact of document automation on organizational productivity: A quantitative analysis. *Journal of Business and Management Studies*. Vol. 28, n.º 4, págs. 102-116 (vid. pág. 8).
- SAMPIERI, Hernández, 2021. *Metodología de la Investigación*. 7th. McGraw-Hill (vid. págs. 3, 4).
- SEIDEWITZ, E., 2003. What models mean. *IEEE Software*. Vol. 20, n.º 5, págs. 26-32 (vid. pág. 25).
- SHALLOW, L., 2019. *Design Patterns for Object-Oriented Software Development*. Springer (vid. pág. 41).
- SILBERSCHATZ, A.; KORTH, H. F. y SUDARSHAN, S., 2019. *Database System Concepts (7th ed.)*. McGraw-Hill (vid. pág. 38).
- SMITH, A., 2021. Data security in document management: Best practices and emerging technologies. *Journal of Information Security*. Vol. 50, n.º 2, págs. 137-153 (vid. pág. 6).
- SOMMERVILLE, I., 2011. *Software Engineering (9th ed.)*. Addison-Wesley (vid. págs. 28, 53, 54).
- SOMMERVILLE, I., 2016. *Software Engineering*. 10th. Pearson Education (vid. pág. 14).
- SUBLIME HQ, 2023. *Sublime Text*. Url: <https://www.sublimetext.com/> (vid. pág. 19).
- UI, Next, 2023. *Next UI (v1.3.0): Component library for Next.js*. Url: <https://nextui.org> (vid. pág. 21).
- VERCEL, 2023. *Next.js Framework (v14)*. Url: <https://nextjs.org> (vid. pág. 20).

WAKE, William, 2019. *The Art of Agile Development*. Sebastopol, CA: O'Reilly Media (vid. pág. 58).

WILLIAMS, R., 2019. Document automation: The key to operational efficiency in the digital age. *International Journal of Document Management*. Vol. 12, n.º 1, págs. 78-89 (vid. pág. 7).

ZHANG, L.; CHEN, W. y LIU, Z., 2020. Digital Document Management Systems: Efficiency and Risk Management in the Modern Organization. *Journal of Business Informatics*. Vol. 44, n.º 1, págs. 89-105 (vid. pág. 5).

Apéndices

Tareas de ingeniería

Tabla 26. Tarea de Ingeniería 4: Visualización de Documento

Tarea de Ingeniería	
Número de tarea: 4	Número de Historia de Usuario: 4
Nombre de la tarea: Visualización de Documento	
Tipo de tarea: Desarrollo	Tiempo Estimado: 5 días
Programador responsable: Orlando Farid Giquez Tarafa	
Descripción: Desarrollar la funcionalidad que permita a los usuarios abrir y ver el contenido de un documento sin necesidad de descargarlo.	

Tabla 27. Tarea de Ingeniería 5: Descarga de Documento

Tarea de Ingeniería	
Número de tarea: 5	Número de Historia de Usuario: 5
Nombre de la tarea: Descarga de Documento	
Tipo de tarea: Desarrollo	Tiempo Estimado: 3 días
Programador responsable: Orlando Farid Giquez Tarafa	
Descripción: Implementar la opción que permita a los usuarios descargar un solo documento a la vez.	

Tabla 28. Tarea de Ingeniería 6: Ordenación de Documentos

Tarea de Ingeniería	
Número de tarea: 6	Número de Historia de Usuario: 6
Nombre de la tarea: Ordenación de Documentos	
Tipo de tarea: Desarrollo	Tiempo Estimado: 3 días
Programador responsable: Orlando Farid Giquez Tarafa	
Descripción: Crear la funcionalidad que permita a los usuarios ordenar los documentos según criterios como nombre, fecha de creación o tipo de archivo.	

Tabla 29. Tarea de Ingeniería 7: Renombrar Documentos

Tarea de Ingeniería	
Número de tarea: 7	Número de Historia de Usuario: 7
Nombre de la tarea: Renombrar Documentos	
Tipo de tarea: Desarrollo	Tiempo Estimado: 3 días
Programador responsable: Orlando Farid Giquez Tarafa	
Descripción: Implementar la opción que permita a los usuarios renombrar documentos.	

Pruebas unitarias

```
it("should set selected folder and update history", () => {  
  const { result } = renderHook(() => useFolderStore());  
  const testFolder = { id: "1", name: "Test Folder", parent: null };  
  
  act(() => {  
    result.current.setSelectedFolder(testFolder);  
  });  
  
  expect(result.current.selectedFolder).toEqual(testFolder);  
  expect(result.current.folderHistory).toEqual([testFolder]);  
});
```

Figura 15. Test al store que maneja las carpetas seleccionadas. Fuente: Elaboración Propia.

```
describe('useDocumentFilter', () => {
  const mockDocuments = [
    { id: '1', name: 'Ficha Técnica', type: 'xlsx' },
    { id: '2', name: 'Contratos y Suplementos', type: 'docx' },
  ];

  test('should filter documents by search term', () => {
    const { result } = renderHook(() => useDocumentFilter());

    act(() => {
      result.current.setSearchTerm('Ficha Técnica');
    });

    const filtered = result.current.filterDocuments(mockDocuments);
    expect(filtered).toHaveLength(1);
    expect(filtered[0].name).toBe('Ficha Técnica');
  });

  test('should return all documents when search term is empty', () => {
    const { result } = renderHook(() => useDocumentFilter());

    const filtered = result.current.filterDocuments(mockDocuments);
    expect(filtered).toHaveLength(2);
  });
});
```

Figura 16. Test a la funcionalidad de buscar documentos. Fuente: Elaboración Propia.

```
describe('DocumentsView', () => {
  const mockDocuments = [
    { id: '1', name: 'Ficha técnica', type: 'xlsx' },
    { id: '2', name: 'Contratos y Suplementos', type: 'docx' },
  ];

  beforeEach(() => {
    (useDocumentationStore as jest.Mock).mockImplementation(() => ({
      documents: mockDocuments,
      isLoading: false,
      error: null,
    }));
  });

  test('should render documents in table view', () => {
    render(<DocumentsView />);
    expect(screen.getByText('Ficha Técnica')).toBeInTheDocument();
    expect(screen.getByText('Contratos y Suplementos')).toBeInTheDocument();
  });

  test('should toggle between grid and table view', () => {
    render(<DocumentsView />);
    const viewToggle = screen.getByRole('button', { name: '/toggle view/i' });
    fireEvent.click(viewToggle);
    expect(screen.getByTestId('grid-view')).toBeInTheDocument();
  });
});
```

Figura 17. Test al componente DocumentsView. Fuente: Elaboración Propia.

```

describe('handleDocumentPreview', () => {
  let mockWindow: any;

  beforeEach(() => {
    mockWindow = {
      open: jest.fn(() => ({
        document: { title: '' }
      })),
      URL: {
        createObjectURL: jest.fn(() => 'mock-blob-url'),
        revokeObjectURL: jest.fn()
      }
    };
    global.window = mockWindow;
    global.fetch = jest.fn().mockResolvedValue({
      blob: () => Promise.resolve(new Blob(['test'], { type: 'application/pdf' }))
    });
  });

  it('should preview a PDF document', async () => {
    const mockDocument = {
      name: 'test.pdf',
      url: 'http://example.com/test.pdf'
    };

    (isDocument as jest.Mock).mockReturnValue(true);

    await handleDocumentPreview(mockDocument as FolderOrDocument);

    expect(global.fetch).toHaveBeenCalledWith(mockDocument.url);
    expect(mockWindow.URL.createObjectURL).toHaveBeenCalled();
    expect(mockWindow.open).toHaveBeenCalledWith('mock-blob-url', '_blank');
  });
});

```

Figura 18. Test a la funcionalidad de visualizar documento. Fuente: Elaboración Propia.

```

describe('handleDocumentDownload', () => {
  it('should download document when item is a document', async () => {
    const mockBlob = new Blob(['test'], { type: 'text/plain' });
    const mockDocument = {
      name: 'test.txt',
      url: 'http://example.com/test.txt'
    };

    (isDocument as jest.Mock).mockReturnValue(true);
    (global.fetch as jest.Mock).mockResolvedValue({
      blob: () => Promise.resolve(mockBlob)
    });

    await handleDocumentDownload(mockDocument);

    expect(global.fetch).toHaveBeenCalledWith(mockDocument.url);
    expect(mockURL.createObjectURL).toHaveBeenCalledWith(mockBlob);
    expect(mockLink.download).toBe(mockDocument.name);
    expect(mockLink.click).toHaveBeenCalled();
    expect(mockLink.remove).toHaveBeenCalled();
    expect(mockURL.revokeObjectURL).toHaveBeenCalledWith('blob:mock-url');
  });
});

```

Figura 19. Test a la funcionalidad de descargar documento. Fuente: Elaboración Propia.

Pruebas de aceptación

Tabla 30. Prueba de Aceptación para Visualizar Documento

Caso de Prueba de Aceptación	
Código: HU4_P1	Historia de Usuario: 4
Nombre: Visualizar Documento	
Descripción: Verificar que el sistema permita visualizar documentos sin descargarlos.	
Condiciones de Ejecución: El usuario debe tener permisos para visualizar el documento.	
Pasos de Ejecución: 1. Iniciar sesión en el sistema. 2. Navegar a la sección "Programas". 3. Acceder a la sección "Proyectos". 4. Ir a la sección "Documentación". 5. Seleccionar un documento para visualizar. 6. Observar el documento en el visor.	
Resultado: Satisfactorio	

Tabla 31. Prueba de Aceptación para Descargar Documento

Caso de Prueba de Aceptación	
Código: HU5_P1	Historia de Usuario: 5
Name: Descargar Documento	

Descripción: Verificar que el sistema permita a los usuarios descargar documentos almacenados.
Condiciones de Ejecución: El usuario debe estar autenticado y tener permisos para descargar documentos.
Pasos de Ejecución: 1. Iniciar sesión en el sistema. 2. Navegar a la sección "Programas". 3. Acceder a la sección "Proyectos". 4. Ir a la sección "Documentación". 5. Seleccionar un documento para descargar. 6. Confirmar la descarga del documento.
Resultado: Satisfactorio

Tabla 32. Prueba de Aceptación para Ordenar Documentos

Caso de Prueba de Aceptación	
Código: HU6_P1	Historia de Usuario: 6
Name: Ordenar Documentos	
Descripción: Verificar que los usuarios puedan ordenar documentos almacenados según diferentes criterios.	
Condiciones de Ejecución: El usuario debe estar autenticado.	
Pasos de Ejecución: 1. Iniciar sesión en el sistema. 2. Navegar a la sección "Programas". 3. Acceder a la sección "Proyectos". 4. Ir a la sección "Documentación". 5. Seleccionar el criterio de ordenación (fecha, nombre, tipo). 6. Observar los documentos ordenados según el criterio seleccionado.	
Resultado: Satisfactorio	

Tabla 33. Prueba de Aceptación para Renombrar Documento

Caso de Prueba de Aceptación	
Código: HU7_P1	Historia de Usuario: 7
Nombre: Renombrar Documento	
Descripción: Esta funcionalidad permite a los usuarios cambiar el nombre de un documento existente de manera sencilla.	
Condiciones de Ejecución: El usuario debe estar autenticado y tener permisos para renombrar documentos.	
Pasos de Ejecución: 1. Iniciar sesión en el sistema. 2. Navegar a la sección "Programas". 3. Acceder a la sección "Proyectos". 4. Ir a la sección "Documentación". 5. Seleccionar un documento para renombrar. 6. Ingresar el nuevo nombre en el campo correspondiente. 7. Confirmar el cambio de nombre.	
Resultado: Satisfactorio	