UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS.

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

INTEGRACIÓN DE LAS APIs REST DE FIGSHARE Y GITHUB MEDIANTE UNA APLICACIÓN ORIENTADA A SERVICIOS PARA PUBLICAR CONTENIDO OPEN SCIENCE

Trabajo de grado presentado en la Universidad Técnica del Norte previo a la obtención del título de Ingeniero en Sistemas Computacionales

Autor:

Romina Michelle Davas Rodriguez

Director:

PhD. Cathy Pamela Guevara Vega

Ibarra, 2024

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN

A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presentetrabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	0803840412		
APELLIDOS Y NOMBRES:	DAVAS RODRIGUEZ ROMINA MICHELLE		
DIRECCIÓN:	IBARRA, PEDRO MONCAYO 13-65		
EMAIL:	rmdavasr@utn.edu.ec		
TELÉFONO FIJO:	954338	**TELÉFONO MÓVIL:**	0959804347

DATOS DE LA OBRA	
TÍTULO:	INTEGRACIÓN DE LAS APIS REST DE FIGSHARE Y GITHUB MEDIANTE UNA APLICACIÓN ORIENTADA A SERVICIOS PARA PUBLICAR CONTENIDO EN OPEN SCIENCE.
AUTOR:	DAVAS RODRIGUEZ ROMINA MICHELLE
FECHA:	05/02/2024
PROGRAMA:	■ PREGRADO ☐ POSGRADO
TITULO POR EL QUE OPTA:	INGENIERO EN SISTEMAS COMPUTACIONALES
DIRECTOR:	PHD. CATHY GUEVARA, MSC.
ASESOR 1:	PHD. IRVING REASCOS, MSC.
ASESOR 2:	ING. FAUSTO SALAZAR, MSC.

2. CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 05 días del mes de febrero de 2024.

EL AUTOR:

Michelle Davas

Romina Michelle Davas Rodriguez

C.I: 0803840412

CERTIFICADO DEL DIRECTOR DE TRABAJO DE GRADO

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN DEL DIRECTOR

En mi calidad de tutora de Trabajo de Grado presentado por el egresado Romina Michelle Davas Rodriguez, portador de la cédula de ciudadanía 080384041-2. Certifico que ha trabajado en el desarrollo del presente proyecto de tesis denominado: **"Integración de las APIs REST de Figshare y GitHub mediante una Aplicación Orientada a Servicios para publicar contenido Open Science"**, previo a obtener el título de ingeniería en sistemas computacionales. Considero que el presente trabajo cumple con los requisitos y méritos suficientes para ser sometido a la presentación pública y evaluación por parte del tribunal examinador que se designe.

Atentamente:

PHD. CATHY GUEVARA, MSC.

DIRECTORA DE TRABAJO DE GRADO

DEDICATORIA

Para mis padres, hermana y las personas que conforman mi familia más cercana, que gracias a su apoyo, paciencia y amor es posible este logro.

AGRADECIMIENTO

Quiero expresar mi gratitud a mi familia, quienes han sido mi mayor apoyo y fuente de inspiración en cada paso que he dado. Gracias por creer en mí, por brindarme su amor incondicional y por alentarme en los momentos más difíciles.

A todas las personas que de alguna manera contribuyeron a mi formación y crecimiento personal, gracias por ser parte fundamental de este viaje.

TABLA DE CONTENIDO

DEDICATORIA v
AGRADECIMIENTO vi
TABLA DE CONTENIDO vii
ÍNDICE DE FIGURAS ix
RESUMEN xi
ABSTRACT xiii
INTRODUCCIÓN xv
Antecedentes xv
Situación Actual xv
Prospectiva xv
Planteamiento del Problema xvi
Objetivos xvii
Objetivo General xvii
Objetivos Específicos xvii
Alcance xvii
Metodología xix
Justificación xix
CAPÍTULO I 21
1.1. Revisión Literaria 21
1.1.1. Ruta de investigación 21
1.1.2. Definir las preguntas de investigación. 21
1.2. Criterios de inclusión y exclusión 23
1.2.1. Conducta de búsqueda. 23
1.2.2. Selección de artículos 24
1.3. Open Science 26
1.3.1. Definiciones 26
1.3.2. Indicadores Open Science 26
1.3.3. Escuelas de pensamiento 27
1.4. GitHub 31
1.4.1. Definiciones 31
1.4.2. Características 31
1.4.3. Flujo de trabajo 32
1.4.4. Extensiones e integraciones de GitHub 33
1.4.5. APIs GitHub 33
1.4.6. Ventajas y Desventajas 34

1.5. Figshare ... 35
1.5.1. Definiciones ... 35
1.5.2. Características ... 36
1.5.3. Integración de Figshare con otras plataformas ... 36
1.5.4. API Figshare ... 37
1.5.5. Ventajas y Desventajas ... 38
1.6. ISO/IEC 25022 ... 39
CAPÍTULO II ... 41
2.1. Análisis, diseño y desarrollo de la arquitectura ... 41
2.1.1. Arquitecturas de Software ... 41
2.1.2. Estilos de desarrollo de Software ... 43
2.1.3. Metodología SCRUM ... 45
2.1.4. Herramientas de trabajo ... 46
2.1.5. Configuración de las plataformas Figshare y GitHub ... 48
2.1.6. Consumo de las APIs de Figshare y GitHub. ... 51
2.2. Desarrollo ... 52
2.2.1. Sprint 0 ... 52
2.2.2. Sprint 1 ... 55
2.2.3. Sprint 2 ... 63
2.2.4. Sprint 3 ... 71
CAPÍTULO III ... 83
3.1. Característica: Funcionalidad - Subcaracterística: Interoperabilidad ... 83
3.1.1. Definición y Contexto de la Interoperabilidad ... 83
3.1.2. Métricas y Criterios de Evaluación de Interoperabilidad ... 84
3.1.3. Resultados de la Evaluación de Interoperabilidad ... 85
3.2. Característica: Usabilidad - Subcaracterística: Aprendizaje ... 87
3.2.1. Concepto de Aprendizaje en Usabilidad ... 87
3.2.2. Métricas y Criterios de Evaluación de Aprendizaje en Usabilidad ... 87
3.2.3. Resultados de la Evaluación de Aprendizaje en Usabilidad ... 88
CONCLUSIONES ... 91
RECOMENDACIONES ... 93
TRABAJO FUTURO ... 95
REFERENCIAS ... 97
ANEXO ... 101

ÍNDICE DE FIGURAS

Fig. 1: Árbol de Problema. xvi
Fig. 2: Arquitectura de la Aplicación.Fuente: Propia xviii
Fig. 3: Metodología de la investigación. xix
Fig. 4. Estructura Metodológica SLR, adaptado de (Webster & Watson, 2002). 21
Fig. 5. Flujo de investigación. 23
Fig. 6: Los pilares de la ciencia abierta (Masuzzo; Martens, 2017) 27
Fig. 7: Escuelas del pensamiento (Open Science Fecher & Friesike, 2014) 28
Fig. 8: Flujo de trabajo de GIT. 33
Fig. 9: Patrón Arquitectura SOA. 43
Fig. 10: Estuctura MVC. 43
Fig. 11: Creación Token de Figshare. 49
Fig. 12: Token de GitHub. 51
Fig. 13: Consumo de la APIs de GitHub, Postman. 52
Fig. 14: Consumo de la API de Figshare, Postman. 52
Fig. 15: Micrsoft Planner - Sprint 0 55
Fig. 16: Base de datos versión 1. 56
Fig. 17: Mapa de procesos Bizagi Modeler 57
Fig. 18: Página principal, diseñada en Balsamiq Wireframes. 58
Fig. 19: Menú y zona general, diseñado en Balsamiq. 58
Fig. 20: Zana de contenido, diseñado en Balsamiq. 59
Ilustración 21. Capa de usuario. Fuente: Propia. 61
Fig. 22: Sprint 1 - Microsoft Planner 62
Fig. 23: Diagrama Base de Datos, versión 2. 64
Ilustración 24: Mapa de procesos global del aplicativo 65
Fig 25: Login Aplicación. 65
Fig. 26: Página principal de soporte. 66
Fig. 27: Función agregar ticket nuevo. 66
Fig. 28: Función Mantenimiento de Usuarios. 66
Fig. 29: Función Mantenimiento Prioridades 67
Fig. 30: Función Mantenimiento Categorías. 67
Fig. 31: Función Mantenimiento Subcategorías. 68
Fig. 32: Función Consulta Ticket. 68
Fig. 33: Función integración de las APIS Figshare y GitHub. 69
Fig. 34: Página principal rol de Usuario. 69
Fig. 35: Flujo de trabajo de usuario soporte 73
Fig. 36: Sprint 3 - Microsoft Planner 76
Fig. 37: Tokens de acceso Figshare y Github 77
Fig. 38: Integración de componentes Figshare y Github 78

ÍNDICE DE TABLAS

Tabla 1. Preguntas de investigación.....22
Tabla 2: Artículos seleccionados.....24
Tabla 3: Matriz de Resultados.....25
Tabla 4: Tabla de herramientas 46
Tabla 5. Roles Scrum.....53
Tabla 6: Historias épicas. Fuente: Propia.....53
Tabla 7: Producto Backlog.....54
Tabla 8: Sprint Review 1.....61
Tabla 9: Sprint Review 2.....69
Tabla 10. Sprint Review.....75
Tabla 11: Criterios de Evaluación.....85
Tabla 12: Criterios de evaluación de Aprendizaje.....88

RESUMEN

La Open Science, o Ciencia Abierta, es un movimiento que ha transformado la forma en que concebimos y llevamos a cabo la investigación científica. Su enfoque se basa en principios fundamentales de accesibilidad, transparencia y reproducibilidad. En otras palabras, busca hacer que la investigación sea más accesible para la comunidad científica y el público en general, transparente en cuanto a su metodología y datos, y reproducible para que otros investigadores puedan validar y construir sobre investigaciones previas.

La tesis que presentamos propone una solución innovadora y prometedora para aprovechar al máximo estas dos plataformas líderes en su campo. La idea central es la integración de las APIs REST de Figshare y GitHub en una única aplicación orientada a servicios, diseñada específicamente para facilitar la publicación de contenido de Open Science. En esencia, esta aplicación actuaría como un puente entre las dos plataformas, permitiendo a los usuarios publicar su contenido de investigación en ambas de manera eficiente y sencilla.

Para lograr este objetivo, el proyecto se dividió en dos fases distintas pero complementarias. La primera fase consistió en un análisis exhaustivo de las APIs REST de Figshare y GitHub. Esto implicó comprender en detalle sus funcionalidades, capacidades y limitaciones. La segunda fase se centró en la creación y desarrollo de la aplicación orientada a servicios, que fue diseñada teniendo en cuenta las necesidades y requisitos de los usuarios de la Open Science.

La eficacia de la aplicación desarrollada fue evaluada mediante métricas basadas en la norma ISO/IEC 25022, con el fin de proporcionar una medida cuantitativa de su desempeño y utilidad en el contexto de la publicación de contenido de ciencia abierta. Este estudio contribuyó al avance de la infraestructura y las prácticas en el ámbito de la ciencia abierta, facilitando la colaboración y el acceso a la información científica de manera más eficaz y transparente.

Los resultados de esta investigación son prometedores y apuntan hacia un futuro en el que la colaboración y la difusión de la investigación científica se vuelvan aún más accesibles y efectivas. La integración de estas dos potentes plataformas tiene el potencial de mejorar significativamente la interoperabilidad entre ellas, lo que

beneficiaría a la comunidad científica al permitir un flujo de trabajo más fluido y coherente en la publicación de contenido de Open Science.

Palabras Claves: Open Science, APIs REST, Figshare, Github, accesibilidad, transparencia, reproducibilidad, ISO/IEC 25022

ABSTRACT

Open Science is a movement that has transformed the way we conceive and conduct scientific research. Its focus is based on fundamental principles of accessibility, transparency, and reproducibility. In other words, it aims to make research more accessible to the scientific community and the general public, transparent in terms of its methodology and data, and reproducible so that other researchers can validate and build upon previous research.

The thesis we present proposes an innovative and promising solution to harness the full potential of these two leading platforms in their field. The central idea is the integration of the Figshare and GitHub REST APIs into a single service-oriented application, specifically designed to facilitate the publication of Open Science content. Essentially, this application would act as a bridge between the two platforms, allowing users to efficiently and easily publish their research content on both.

To achieve this goal, the project was divided into two distinct but complementary phases. The first phase involved a comprehensive analysis of the Figshare and GitHub REST APIs. This entailed a detailed understanding of their functionalities, capabilities, and limitations. The second phase focused on the creation and development of the service-oriented application, designed with the needs and requirements of Open Science users in mind.

The effectiveness of the developed application was assessed using metrics based on the ISO/IEC 25022 standard, aiming to provide a quantitative measure of its performance and utility in the context of publishing open science content. This study contributed to the advancement of infrastructure and practices in the field of Open Science, making collaboration and access to scientific information more effective and transparent.

The results of this research are promising and point towards a future where collaboration and the dissemination of scientific research become even more accessible and effective. The integration of these two powerful platforms has the potential to significantly enhance interoperability between them, benefiting the scientific community by enabling a smoother and more cohesive workflow in the publication of Open Science content.

Keywords: Open Science, APIs REST, Figshare, Github, accessibility, transparency, reproducibility, ISO/IEC 25022.

INTRODUCCIÓN

Antecedentes

El objetivo de Open Science es evolucionar a una ciencia más accesible, efectiva, transparente, interdisciplinaria y democrática, donde todos puedan participar y beneficiarse de ella, todo esto se puede lograr gracias a las tecnologías de la información y las comunicaciones (Ledo, Mujica & Sánchez, 2018). Caracterizándose no solo por dar accesibilidad a las publicaciones, sino también a todos los datos de investigación que posee la comunidad de investigación, incluyendo los procesos y las metodologías (Crue, 2019). A pesar de todo esto, en muchos de los ámbitos del ser humano, el movimiento Open Science no ha logrado ser aprovechado, difundido y muchos de estos datos no son de libre acceso. Sin embargo, a pesar de que existen muchos repositorios virtuales Open Science, como Figshare que buscan proporcionarles a los investigadores una forma de preservar y compartir sus resultados de investigación, aún hay otros como GitHub que gran parte de sus datos no son de fácil acceso, la información es privada y limita a la reproducibilidad y replicabilidad del software (ICSE, 2021).

Situación Actual

Actualmente Open Science trata de fomentar el almacenamiento de la información, la documentación y los artefactos como datos y código fuente en repositorio virtuales como Figshare. Además, ofrece la posibilidad de publicar contenido bajo una licencia de datos abiertos más adecuada. GitHub ha hecho colaboraciones con otros repositorios virtuales con la finalidad de lograr integrase en estos (León, Miranda, Rodríguez, Segredo & Segura, 2018).

Prospectiva

La presente propuesta de investigación busca fomentar el movimiento Open Science en la Ingeniería de Software (IS), por medio de la accesibilidad a los datos y código que se publican en los diferentes repositorios virtuales como Figshare. Específicamente, se quiere lograr que los trabajos de IS sean públicos para que la comunidad científica los pueda utilizar.

Planteamiento del Problema

En Ecuador como en Latinoamérica se ha buscado impulsar el movimiento Open Science, pero aun así no toda la data de la comunidad científica se ha logrado que sea de libre acceso, particularmente en IS existe dificultad al momento de acceder a los diferentes artefactos e información que se encuentra en los repositorios virtuales, que se necesita realizar la gestión de configuración y gestión de cambios en el desarrollo del software. Por ende, si la información que necesitan los ingenieros es privada, perjudicará a la reproducibilidad y a la replicabilidad de los artefactos en el desarrollo del software.

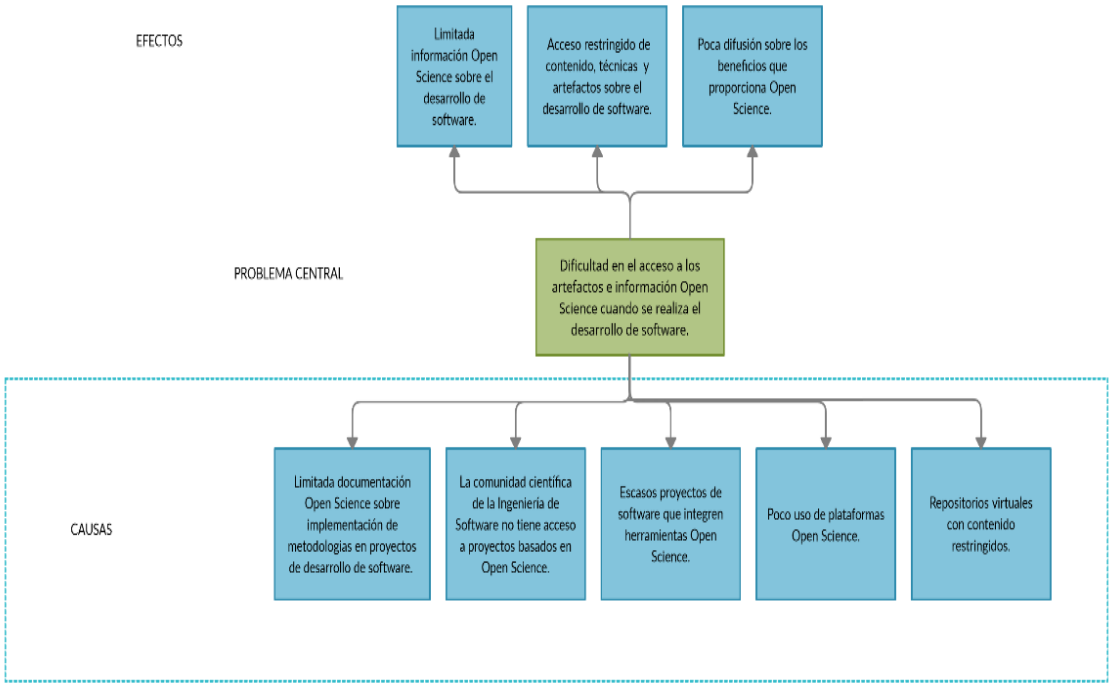


Fig. 1: Árbol de Problema.

Objetivos

Objetivo General

Objetivo General Integrar las APIs REST de Figshare y GitHub mediante una aplicación orientada a servicios para publicar contenido Open Science.

Objetivos Específicos

- Conocer el estado actual del uso de las herramientas Figshare, y GitHub para publicar contenido Open Science.
- Configurar las plataformas Figshare y GitHub para integrar y publicar contenido Open Science.
- Desarrollar una aplicación orientada a servicios para integrar las APIs REST de Figshare y GitHub y publicar contenido Open Science.
- Evaluar la eficacia de la aplicación desarrollada basada en las métricas de la ISO/IEC 25022.

Alcance

El presente trabajo de investigación pretende realizar la integración de las APIs REST de Figshare y GitHub mediante una aplicación orientada a servicios para publicar contenido Open Science. Se realizará la evaluación de la eficacia de la aplicación basada en las métricas de la ISO/IEC 25022.

En primer lugar, se realizará una revisión de la literatura para conocer el estado actual de las herramientas Figshare y GitHub con respecto al desarrollo de software que publica su contenido Open Science.

Segundo, se configurará las plataformas Figshare y GitHub las cuáles permitirán la integración y posterior publicación del contenido de una aplicación.

Tercero, se desarrollará una aplicación web “Mesa de Ayuda” orientada a servicios, la misma que será desarrollada con la arquitectura MVC y la metodología Scrum; dicha aplicación tendrá las siguientes funcionalidades generales:

- Control y gestión de usuarios.
- Control y gestión de prioridades.

- Control y gestión de categorías.
- Creación, modelado y consulta de Tickets.
- Generación de acceso según el rol.
- Detalle de Ticket.
- Gestión de Componentes Figshare & GitHub.
- Generación de contenidos basados en los principios Open Science.
- Generación de reportes.

El segundo aplicativo consumirá los datos obtenidos de la aplicación Integración Figshare & GitHub.

Finalmente, se evaluará la eficacia de la aplicación basada en las métricas de la ISO/IEC 25022.

Las herramientas que se planea utilizar fueron establecidas de manera general, sin embargo, se irán detallando a medida que el proyecto crezca:

- Arquitectura de software MVC.
- Marco de trabajo ágil Scrum.
- Lenguaje de programación PHP.
- IDE de programación Visual Studio Code.
- Motor de base de datos MySQL.

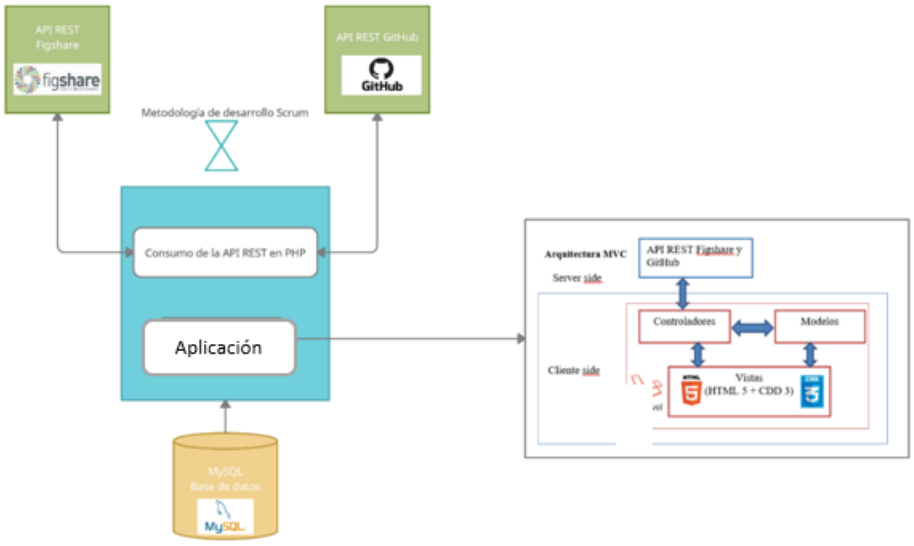


Fig. 2: Arquitectura de la Aplicación. Fuente: Propia

Metodología

En la revisión de la literatura se realizará una investigación documental mediante la definición de una cadena de búsqueda, selección de artículos, aplicación criterios de inclusión y exclusión, y análisis y presentación de resultados sobre el estado actual del uso de las herramientas Figshare y GitHub, en la publicación de contenido Open Science.

Luego, se configurará las plataformas Figshare y GitHub, para después desarrollar una aplicación orientada a servicios que permita consumir las APIs de Figshare y GitHub, para posteriormente publicar contenido Open Science.

Finalmente, se evaluará la eficacia de la aplicación usando la ISO/EC 25022.

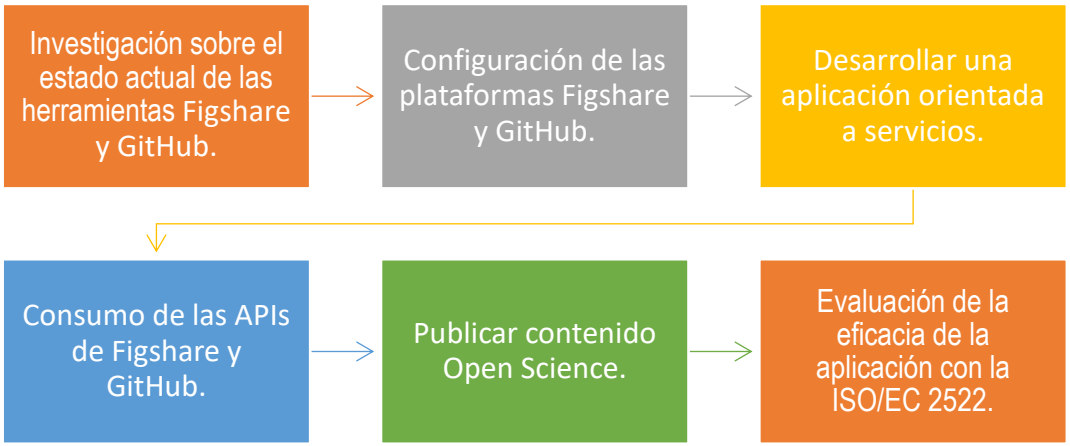


Fig. 3: Metodología de la investigación.

Fuente: Propia

Justificación

La presente investigación, busca generar un aporte a la comunidad científica de IS, para ayudar a la reproducibilidad, replicabilidad y repetibilidad en el desarrollo del software que permita mejorar el aseguramiento de la calidad en el desarrollo del software.

Esta investigación apoya al Objetivo de Desarrollo Sostenible número 17 de la Agenda 2030, el cual se enfoca en que la “cooperación debe ser triangular, regional e internacional, dando hincapié al acceso a la ciencia, tecnología y la innovación, mejorando así el conocimiento compartido en términos mutuamente acordados a través de una tecnología global como mecanismo de facilitación académica”. (Pontika 2015)

Justificación Tecnológica.

“Tanto las nuevas tecnologías como las herramientas están dando forma al mundo de la investigación Open Science, ayudando a los investigadores a ser más eficientes, colaborativos y productivos. Por lo que, la ciencia tendrá más impacto en la sociedad y abordará mejor los desafíos sociales”. (Cid, 2017)

Justificación Teórica

Las perspectivas de los indicadores relacionados con el conocimiento abierto no ofrecen motivos de optimismo. Aunque la noción de la ciencia abierta no es reciente, actualmente requiere una mayor dedicación por parte de gobiernos y entidades, ya que podría convertirse en el factor impulsor fundamental que fomente la colaboración científica a nivel global. (Garcia et al., 2013)

CAPÍTULO I

Marco Teórico

1.1. Revisión Literaria

Un proyecto de investigación implica un estudio minucioso de contenido que contiene datos importantes sobre el tema en cuestión, lo que requiere un proceso que tiene pasos que le permiten obtener información importante. El método SLR " Systematic Literature Review" se usa a menudo para el mapeo del sistema, un modelo que proporciona ciertos puntos de referencia que le permiten obtener una imagen general de un campo de conocimiento en particular. (Webster & Watson, 2002)

1.1.1. Ruta de investigación

A continuación, se muestra un diagrama sistemático de los pasos involucrados en una revisión de la literatura, desglosados en una
combinación de los siguientes pasos:

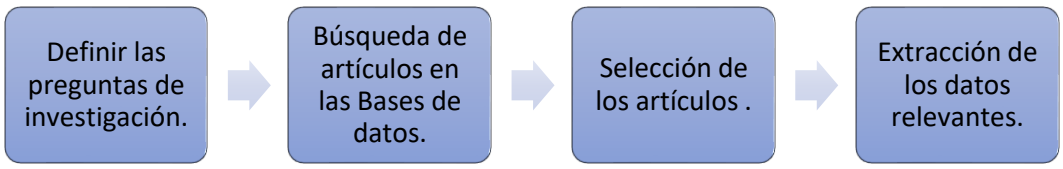


Fig. 4. Estructura Metodológica SLR, adaptado de (Webster & Watson, 2002).

Fuente: Propia.

1.1.2. Definir las preguntas de investigación.

Las revisiones sistemáticas son un tipo de estudio integral, retrospectivo y secundario en el que se combinan investigaciones que abordan la misma pregunta de manera observacional. Este estudio se lo puede realizar de dos formas: cuantitativo o cualitativo (Beltrán, 2005).

Para cumplir el principio de revisión sistemática primero se plantearon las siguientes preguntas:

N°	Preguntas de investigación	Motivación
PI.1	¿Cuál es la situación actual de Open Science, GitHub y Figshare?	Conocer si en los repositorios virtuales se encuentra información sobre ellos.
PI.2	¿Cuáles son las integraciones de GitHub?	Conocer cómo se integra GitHub con otras aplicaciones.
PI.3	¿Se puede integrar Figshare con otras aplicaciones?	Conocer cómo si Figshare se puede integrar con otras aplicaciones.

Tabla 1. Preguntas de investigación

Los motores de búsqueda bibliográficas, repositorio virtuales y base de datos científicas utilizados para responder las interrogantes anteriormente mencionadas, son: Scopus, IEEE y Scholar Google. Para comenzar con búsqueda se plantearon las siguientes cadenas de búsqueda: “open science movement” OR “repository github integration figshare”.

1.1.3. Búsqueda de artículos.

Para dar respuesta a preguntas previamente identificadas, buscadores bibliográficos y bases de datos científicas como:

- Google Académico
- Scopus
- Páginas web Científicas.
- Repositorios virtuales.
- IEEE

En la siguiente ilustración se muestra el flujo grama para realizar la búsqueda de los artículos por medio de los motores de búsqueda indicados con anterioridad. Para comenzar con búsqueda se plantearon las siguientes cadenas de búsqueda: “open science movement” OR “repository github integration figshare”.

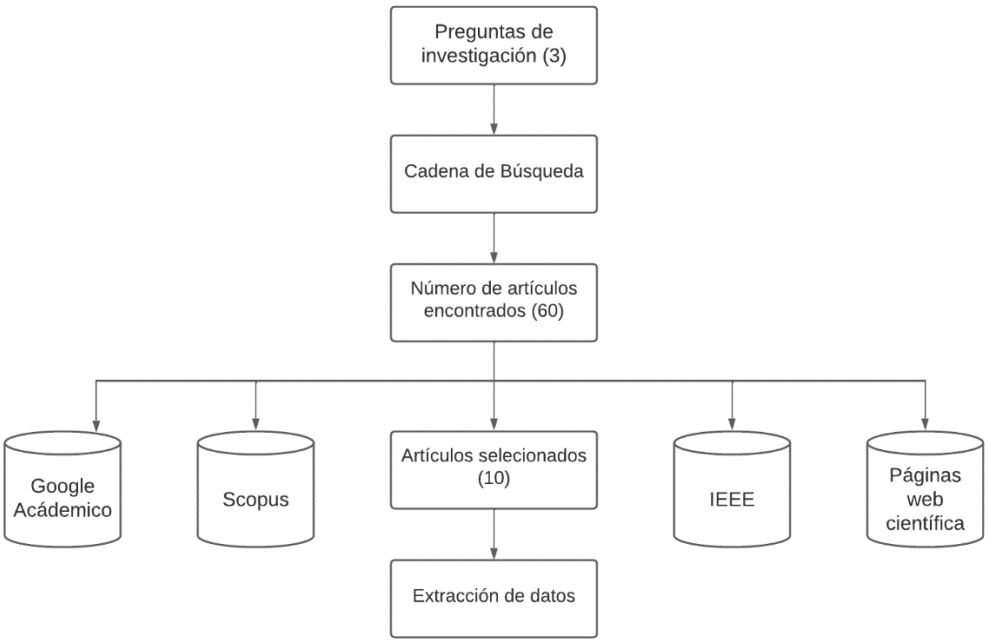


Fig. 5. Flujo de investigación.

Fuente Propia.

1.2. Criterios de inclusión y exclusión

Open Science como tal es un tema muy amplio, lo que implica encontrar información de toda índole, por lo que los artículos encontrados al realizar la cadena de búsqueda deben ser filtrados.

Los criterios de inclusión utilizados fueron seleccionar artículos, conferencias y trabajos de grados relacionados con el estado del arte, establecer un intervalo de años de 2004 hasta 2021, también se filtraron estos por los idiomas inglés, español y portugués.

Los criterios de exclusión que se estableció fue excluir todo documento que no tenga relación con Open Science y todo documento que no tenga relación con las preguntas establecidas con anterioridad.

1.2.1. Conducta de búsqueda.

Para filtrar la información obtenida se realizó los siguientes filtros:

- Primer filtro: Los títulos de las publicaciones recuperadas se revisaron de acuerdo con la cadena de búsqueda, se procedió seleccionando: artículos científicos, conferencias revisadas por pares y resúmenes. Después de seleccionar los trabajos por título, se visualizó y leyó el abstract correspondiente de cada trabajo.
- Segundo filtro: Luego de seleccionar las publicaciones relevantes, se realizó una lectura extensiva para obtener información relevante.

1.2.2. Selección de artículos

Una vez aplicados los criterios de inclusión y exclusión, se seleccionaron 31 referencias. De estas referencias se terminó seleccionando los 10 más acorde a responder las preguntas planteadas con anterioridad.

Tabla 2: Artículos seleccionados.

Código	Título	Autor
A1	¿Qué es la ciencia abierta?	Abadal E; Anglada L.
A2	Open Science: One Term, Five Schools of Thought	Fecher B; Friesike S.
A3	The Open Science ecosystem	da Silva F; da Silveira L.
A4	Academy, data and science reproducibility	Martínez-Méndez A., et al.
A5	FigShare	Singh J.
A6	Figshare: ¿A universal repository for academic resource sharing?	Thelwall M; Kousha K.
A7	El control de versiones	Borrell G.
A8	Introducción a GitHub	Riva G.
A9	Introducción A Git Y Github	Calvo J.
A10	Reproducibilidad y repetitividad experimental entre filósofos y científicos.	Velasco M.

1.2.3. Extracción de datos relevantes.

Una vez terminada la revisión se elaboró una matriz donde se identifica los conceptos más relevantes encontrados en los artículos seleccionados.

Tabla 3: Matriz de Resultados

Artículos												
Código	Promueve la ciencia.	Escuelas Pensamiento.	Libre acceso.	Comunidad científica.	Investigación científica.	Repositorios virtuales.	Ciencia 2.0	Extensiones e integraciones	Repetibilidad, Replicabilidad y Reproducibilidad.	Integración con otras aplicaciones.	características y funcionalidades.	Ventajas y desventajas.
A1	X	X	X	X	X		X					
A2	X	X	X									
A3	X											
A4									X			
A5					X	X		X		X	X	X
A6			X								X	
A7								X			X	
A8						X						
A9			X			X				X		X
A10									X			

1.3. Open Science

Vivimos en una era de inquietud por el futuro de la ciencia. Es tanto más digno de mención, entonces, que los círculos de políticas científicas hayan adoptado una obsesión abierta por la "ciencia abierta". Todo comenzó a finales de la década de 2000, con rumores sobre algo llamado "Ciencia 2.0". En enero de 2012, el New York Times (McKiernan et al., 2016) tuvo el buen sentido de promover el cambio de marca de este imaginario como "ciencia abierta". La Royal Society británica intervino de cerca en 2012, con un documento de relaciones públicas titulado Science as an Open Enterprise (Boulton et al., 2012).

1.3.1. Definiciones

Open Science representa el avance hacia una ciencia que es más eficiente, de fácil acceso, transparente, colaborativa e inclusiva, ya que permite la participación y beneficio de diversos públicos. Todo esto se hace posible gracias a las tecnologías de la información y las comunicaciones. (Abadal & Anglada, 2020)

Para los autores Bartling y Friesike (2014) la ciencia abierta se refiere a una cultura científica que se caracteriza por su apertura, donde los científicos comparten los resultados casi de inmediato y con una audiencia muy amplia

Según la European Commission (2020) Open Science trata sobre una nueva forma de aproximación al proceso científico, enfocándose en el trabajo cooperativo y en nuevas formas de difusión del conocimiento utilizando tecnologías digitales y nuevas herramientas colaborativa, ya que d esta manera se mejora la calidad, la eficiencia y la capacidad de respuesta de la investigación científica.

Para Bezjak (2018) Open Science se refiere también a la parte científica de los términos más amplios de la erudición abierta, es decir, "el proceso, la comunicación y la reutilización de la investigación tal como se practica en cualquier disciplina de investigación académica, y su inclusión y función dentro de la sociedad en general".

1.3.2. Indicadores Open Science

Para los autores Masuzzo y Martens (2017) este movimiento está compuesto por cuatro pilares: los datos abiertos, que son aquellos que se encuentra disponibles de

forma libre para todo el mundo; el código abierto, se refiere al código fuente del software que es abiertamente accesible y que puede ser cambiado y distribuido por cualquier; el open Access, que es el acceso gratuito a la información y al uso sin restricciones de los recursos digitales; y, la revisión abierta, que son modelos de evaluación por pares que buscan aumentar la transparencia, eficiencia y la responsabilidad del proceso de revisión.

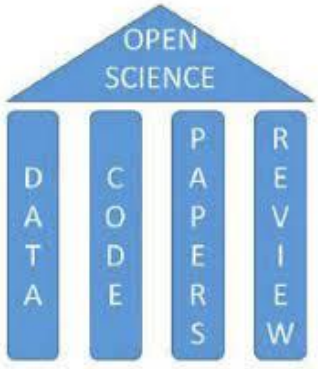


Fig. 6: Los pilares de la ciencia abierta (Masuzzo; Martens, 2017)

El proyecto Open Science Monitor de la Comisión Europea (2017) emplea la metáfora de los ejes que impulsan la rueda de la ciencia abierta. En esta representación, se destacan tres características esenciales: el uso de datos abiertos de investigación, la promoción del acceso abierto y la fomentación de la comunicación científica abierta. Estos tres elementos fundamentales son fundamentales para el avance y desarrollo de la ciencia abierta.

El proyecto Foster, reconocido por contar con uno de los portales más exhaustivos y detallados acerca de la ciencia abierta, ilustra los elementos fundamentales de la ciencia abierta comparándolos con las celdas de un panal de abejas. Cada componente representa una parte esencial que, en conjunto, conforma el entramado sólido y organizado de la ciencia abierta. Incluye: open notebooks; datos abiertos; revisión abierta; open access; sotfware libre; redes sociales académicas; ciencia ciudadana; recursos educativos abiertos. (Abadal & Anglada, 2020)

1.3.3. Escuelas de pensamiento

Open Science es un término general que abarca una multitud de suposiciones sobre el futuro de la creación y difusión del conocimiento.

Bartling y Friesike (2014) proponen cinco Escuelas de pensamiento sobre open science: la Escuela de Infraestructura (arquitectura tecnológica), la Escuela Pública (creación de conocimiento), la Escuela de Métricas (medición del impacto alternativo), la Escuela Democrática (acceso al conocimiento como un derecho humano) y la Escuela Pragmática (investigación colaborativa).

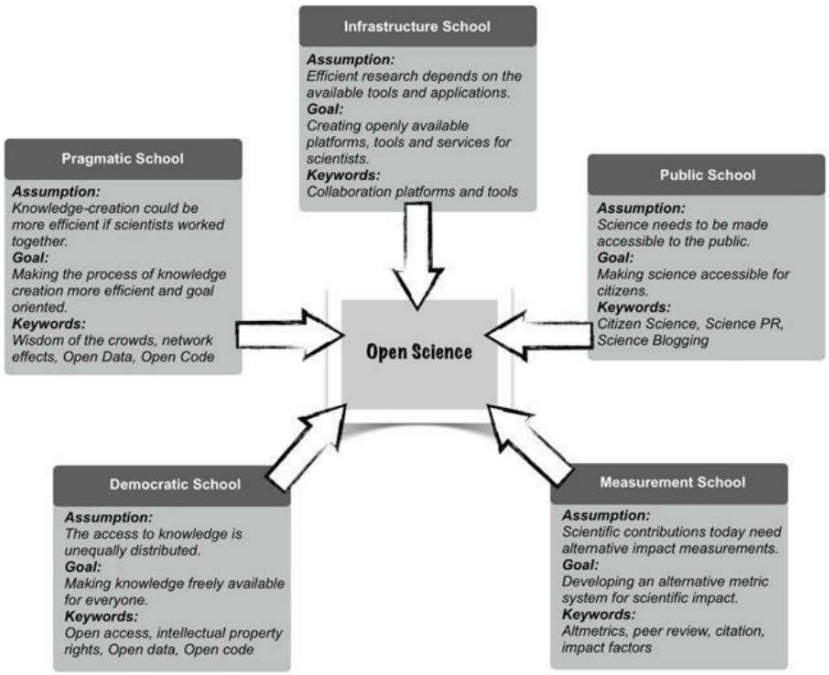


Fig. 7: Escuelas del pensamiento (Open Science Fecher & Friesike, 2014)

1.3.4. Indicadores para medir Open Science

Las oportunidades que el movimiento de ciencia abierta brinda a los investigadores es un cambio de juego y, por lo tanto, una gran oportunidad, especialmente para las universidades que miden sus habilidades con métricas y métodos que han creado prestigio a nivel local y mundial. Comprender el ciclo de vida de la ciencia abierta y todos los actores involucrados como emisores y receptores requiere una comprensión de lo que es más relevante para el contexto de investigación en cuestión.

1.3.5. Innovación Open Science

La sociedad tal como la conocemos no sería tan avanzada como lo es hoy sin una inversión masiva en todos los proyectos relacionados con la tecnología, y no hay duda de que la inversión privada ha jugado un papel muy importante. Sin embargo, en los

últimos años, se ha notado claramente que el conocimiento abierto ha atraído mucho interés por parte de estudiantes, investigadores y público en general, lo que tiene un papel importante que desempeñar. De ahí surge la idea de fomentar la innovación abierta, para que evolucione constantemente y se mueva de forma lineal, creando transacciones y colaboraciones dinámicas, colaborativas y conectadas; permitiendo así la creación de ecosistemas de innovación colaborativa de múltiples partes (Chesbrough, 2006).

1.3.6. Creative Commons Licenses

Las licencias de Open Science, al igual que las licencias tradicionales, tienen varios estándares. En contenido abierto, las licencias a menudo se denominan Creative Commons y son comunes en revistas académicas y populares en la web. Por lo tanto, todo investigador debe entender cómo aplicarlos. Independientemente de los antecedentes de la investigación. (Fecher & Friesike, 2013)

Este autor considera las siguientes Licencias Creative Commons las principales:

a) No Copyright: Public Domain CC0: Es una licencia para liberar contenido al público, renunciando a todos los derechos de autor, para que otros puedan usar ese contenido con fines económicos para reproducirlo, modificarlo y distribuirlo.
b) Attribution-NoDerivs CC BY-ND: Es una licencia que le permite distribuir el contenido sin cambios en formas comerciales y no comerciales, asegurando plenamente los derechos de los autores originales.
c) Attribution CC BY: Es la licencia más permisiva para contenido abierto específico, ya que permite que cualquier persona distribuya, modifique, remezcle y explote la obra, incluso comercialmente; se recomienda la licencia anterior para una distribución más amplia del contenido y sus anexos.
d) Attribution-ShareAlike CC BY-SA: Es una licencia recomendada para contenidos que facilita su redistribución para incluir nuevos elementos. Este es el caso de Wikipedia, que alimenta sus servidores de información con información de una gran variedad de fuentes; la licencia permite que otros modifiquen, remezclen y creen otro contenido.
e) Attribution-NonCommercial CC BY-NC: Es una licencia que permite que otros modifiquen, remezclen y construyan el trabajo del autor original de una

manera no comercial mientras retienen los derechos y créditos apropiados del autor original.

Las licencias anteriores permiten al autor del contenido abierto proteger la autoría del contenido de la mejor manera para su trabajo y esfuerzo, por lo que es claro que incluso la más restrictiva permite que el conocimiento funcione de manera abierta. Proteger la originalidad de la investigación en todas sus etapas para preservar la integridad de la información y sus fuentes

1.3.7. Repetibilidad, Replicabilidad y Reproducibilidad

La Repetibilidad (mismo equipo, misma configuración experimental), donde la medición puede ser obtenida con precisión declarada por el mismo equipo usando el mismo procedimiento de medición, el mismo sistema de medición, bajo las mismas condiciones de operación, en la misma ubicación en múltiples ensayos. Para los experimentos computacionales, esto significa que un investigador puede repetir de manera confiable su propio cálculo (Méndez, 2020).

La Replicabilidad (equipo diferente, misma configuración experimental), es un hallazgo que se puede obtener con otras muestras aleatorias extraídas de un espacio multidimensional que captura las facetas más importantes del diseño de la investigación (Asendorpf et al., 2013), la replicación de resultados se ha convertido en un tema desafiante. La replicación nos permite apoyar el conocimiento científico a través de la generalización y refutabilidad de hallazgos.

La Reproducibilidad (equipo diferente, configuración experimental diferente), la definición precisa varía entre disciplinas, pero a menudo está estrechamente relacionada con las definiciones de repetibilidad y replicabilidad. En una enciclopedia reciente de la filosofía de la ciencia, la reproducibilidad se describe como la repetibilidad del proceso de establecer un hecho o de las condiciones bajo las cuales se puede observar el mismo hecho (McNutt, 2014).

La investigación científica y las industrias de ingeniería maduras se reconocen por su reproducibilidad. Que los fenómenos y los resultados de su investigación sean reproducibles es un principio central del método científico. Al evaluar un estudio, los científicos deben preguntarse "qué tan reproducibles son los hallazgos" (Manterola et al., 2018).

1.4. GitHub

GitHub es una plataforma web comercial lanzada en el año 2008 (Yu, et al., 2014), fue diseñada para facilitar el almacenamiento centralizado, la trazabilidad y colaboración sobre archivos de proyectos. Actualmente mantiene más de 28.1 millones de repositorio. (GitHub, Inc., 2021)

1.4.1. Definiciones

El control de versiones hace referencia a los métodos y herramientas que permiten gestionar y rastrear todos los cambios realizados a lo largo del tiempo en un archivo específico. (Borrell, 2006).

Git es un SCV distribuido diseñado para la gestión eficiente de flujos de trabajo distribuido no lineales. Git fue diseñado y desarrollado inicialmente por Linus Torvalds en 2005 para el desarrollo del kernel de Linux (Chacon y Straub, 2014).

GitHub es un servicio comercial de alojamiento de repositorio Git remotos creado en el año 2008. GitHub proporciona una interfaz Web que permite al usuario registrado crear repositorios vacíos o por clonación de otro repositorio hospedado en GitHub, enviar solicitudes de cambio entre repositorio hospedados y gestionar dichas solicitudes (Lopez-Pellicer et al., 2015).

Aparte del alojamiento, GitHub ofrece para cada repositorio una función de wiki, una herramienta para gestionar tareas, un sistema integral para administrar comentarios, un tablero de control con gráficos sociales y, además, una página web personalizada. Por último, todo este contenido puede ser accedido y manipulado mediante una API web. (Liu, 2014).

1.4.2. Características

Según la página oficial de GitHub, dentro de sus características y funcionalidades que ofrece como parte de su servicio, se pueden encontrar:

- Búsqueda de información: Las búsquedas en GitHub se pueden desde un navegador o directamente desde la página oficial. Su repositorio posee información de toda índole desde desarrollo de software hasta música. La

información es sacada de muchos de sus repositorio, usuarios y líneas de código en GitHub.

- Crear, clonar y archivar repositorio: Para GitHub un repositorio en una carpeta donde se guardarán todos los archivos del proyecto, este se puede crear tanto en cuentas personales como en cuentas de organizaciones. Al crearse un repositorio en GitHub, existe como un repositorio remoto. Los repositorios se pueden clonarse, es decir, que se puede crear copias de seguridad locales y sincronizarlas entre las dos ubicaciones. Los repositorios al ser archivados indican que no están activos para terceros.
- Importar tus proyectos a GitHub: GitHub posee una herramienta denominada Importador, que permite importar código fuente, historias de revisiones y confirmaciones de cambios de forma más rápida y eficiente.

  Al momento de realizar la importación, se puede autenticar desde el repositorio remoto, actualizar la atribución del autor e importar repositorio con archivos grandes, pero esto dependerá del sistema de control de versión del que se esté importando.

- Extender GitHub: Para automatizar las tareas comunes, respaldar datos, o crear integraciones que se extienden GitHub, se puede utilizar la API de GitHub.

1.4.3. Flujo de trabajo

El flujo de trabajo en GitHub se refiere a cómo los equipos de desarrollo colaboran y gestionan cambios en un proyecto utilizando Git y GitHub. GitHub ofrece herramientas y funcionalidades que facilitan el trabajo colaborativo en repositorios de código, lo que ha llevado a la popularización de ciertos flujos de trabajo comunes. Uno de los flujos

más conocidos y ampliamente utilizado es el modelo de ramas y solicitudes de extracción (Pull Requests).

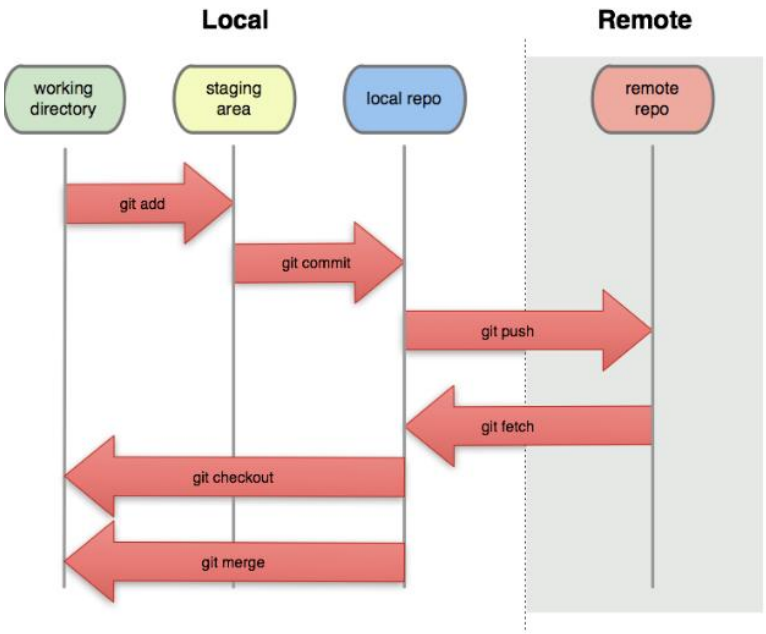


Fig. 8: Flujo de trabajo de GIT.

Fuente: GitHub.

1.4.4. Extensiones e integraciones de GitHub

Según GitHub (2021) las integraciones son herramientas y servicios que se conectan con el para completar y extender el flujo de trabajo.

GitHub utiliza las extensiones para que los usuarios puedan trabajar sin inconvenientes en los repositorios dentro de otras aplicaciones como Atom, donde se puede crear, cambiar ramas, branches, cambios stage, commit, pull, push, entre otros (GitHub Atom, 2021); Unity, los usuarios pueden desarrollar en esta plataforma de código abierto de juegos; y Visual Studio, se puede trabajar en los repositorios sin salir de la aplicación.

1.4.5. APIs GitHub

Según GitHub (2023), la plataforma ofrece dos interfaces de programación de aplicaciones (API): una API REST y una API GraphQL. Estas APIs pueden ser anexadas utilizando diferentes herramientas, como GitHub CLI, cURL, las bibliotecas

oficiales de Octokit y también bibliotecas desarrolladas por terceros. Es importante tener en cuenta que algunas características pueden ser admitidas en una API y no en la otra. Es decir, ciertas funcionalidades pueden estar disponibles solo a través de la API REST o solo a través de la API GraphQL.

La APIs REST permite que aplicaciones y servicios externos se comuniquen con GitHub para acceder y manipular información, automatizar tareas, crear integraciones y ofrecer una amplia gama de funcionalidades relacionadas con la gestión de proyectos y el desarrollo de software.

Algunos puntos clave sobre la API REST de GitHub incluyen:

- Base URL: La base URL para realizar solicitudes a la API de GitHub es https://api.github.com.
- Autenticación: Para realizar solicitudes en nombre de un usuario, es necesario autenticarse mediante un token de acceso personal o un token OAuth. Puedes incluir este token en los encabezados de la solicitud para autenticarla correctamente.
- Puntos finales (Endpoints): La API de GitHub utiliza puntos finales (endpoints) para acceder a diferentes recursos y realizar operaciones específicas en ellos. Por ejemplo, hay puntos finales para obtener información sobre un repositorio, crear un nuevo problema, listar las solicitudes de extracción, entre otros.
- Paginación: La API de GitHub utiliza paginación para limitar el número de resultados devueltos en una sola respuesta. Puedes utilizar los parámetros page y per_page para navegar a través de los resultados paginados.
- Límite de solicitudes (Rate limiting): La API de GitHub tiene límites de tasa (rate limits) para prevenir un uso abusivo. Los límites de tasa dependen de si la solicitud está autenticada o no. Las solicitudes autenticadas generalmente tienen un límite de tasa más alto.

1.4.6. Ventajas y Desventajas

Para Magaly (2020) las ventajas de GitHub son las siguientes: ser Open Sourse, trabajar íntegramente con varios IDEs, poseer un incremento de velocidad de operación, tener operaciones de sucursales baratas, árbol de historias disponible sin conexión y tener un modelo distribuido de igual a igual. En cambio, sus desventajas

son: muy difícil de aprender para quienes trabajan con SVN, no estar echo para desarrolladores individuales y no tiene un soporte limitado para Windows.

GitHub posee muchos pro y contras, siendo una plataforma de servicio gratuito, pero no absolutamente libre. Posee una comunidad amplia por lo que es fácilmente encontrar ayuda. Las búsquedas en el repositorio son rápidas y permite integrar con facilidad otros servicios. Dentro de sus desventajas encontramos que posee un limitado espacio de almacenamiento para la versión gratuita (Riva, 2018).

1.5. Figshare

Existen muchos repositorios virtuales de ciencia abierta como Figshare que buscan proporcionarles a los investigadores una forma de preservar y compartir sus resultados de investigación, aún hay otros como GitHub que gran parte de sus datos no son de fácil acceso, la información es privada y limita a la reproducibilidad y replicabilidad del software (ICSE, 2021).

1.5.1. Definiciones

Figshare es una plataforma en línea que sirve como repositorio digital para que los investigadores puedan conservar y compartir todos los datos y resultados que se obtienen durante un proceso de investigación. Esto incluye figuras, bases de datos, imágenes y vídeos. A través de Figshare, cualquier investigador tiene la posibilidad de cargar estos datos relevantes y ponerlos a disposición del público mediante un número DOI que se asigna automáticamente a cada conjunto de datos generado con la aplicación. Una característica destacada de Figshare es que también permite a los investigadores publicar los datos negativos de sus investigaciones. (Arévalo, 2016)

Figshare es una interfaz basada en web diseñada para la gestión de datos de investigación académica y la difusión de datos de investigación. Acepta todos los tipos de archivos (Figshare, 2023). Fue creado como una solución para mantener los resultados de la investigación en un lugar ordenado y permitir que la comunidad académica la descubra (Thelwall & Kousha, 2016).

1.5.2. Características

Los datos que se suben a Figshare están disponibles bajo la Licencia Creative Commons (CCAL), la cual permite que los autores conserven sus derechos de autor, pero aun así permite que otros descarguen, reutilicen, reimpriman, modifiquen, distribuyan y/o copie datos de Figshare (Figshare, 2023).

Una de las características más útiles de Figshare es posee una página de discusión con pestañas presente en cada una de las páginas de carga para ayudar a refinar las características de búsqueda y usabilidad a medida que aumenta la cantidad de datos almacenados (Singh, 2011).

1.5.3. Integración de Figshare con otras plataformas

Figshare permite a sus usuarios integrase con otras aplicaciones con la finalidad de disminuir los flujos de trabajos y este repositorio sea más fácil de usar. Dentro de las aplicaciones se puede encontrar:

- GitHub: Figshare permite a los usuarios integrarse de forma sencilla con GitHub. Primero tienen que dirigirse al menú seleccionar Application, Connect, esto redirigirá al usuario a la página de GitHub para que inicie sesión. En la página de Figshare en Import from GitHub puede comenzar a importar desde GitHub desde su lista de repositorio públicos (Figshare, 2023).
- GitLab: Para poder integrar Figshare con GitLab, las instituciones pueden habilitar el soporte solo para https://about.gitlab.com/ cuentas o también habilitar el soporte para cualquier repositorio de GitLab alojado localmente dentro de su organización. Para configurar esta integración, póngase en contacto con support@figshare.com (GitLab, 2023).
- Bitbucket**:** está disponible para todos los usuarios de figshare.com y, opcionalmente, para los usuarios de FFI. Para Bitbucket, solo admitimos cuentas de https://bitbucket.org, no se admite la instalación local (Bitbucket, 2023).
- Open Science Framework (Figshare): Puede conectar una carpeta Figshare a su proyecto Figshare. Se pueden conectar carpetas adicionales a componentes dentro del proyecto. Los archivos agregados a su proyecto Figshare serán accesibles a través de Figshare. Del mismo modo, los archivos

agregados a la carpeta Figshare en su cuenta Figshare actualizarán su proyecto Figshare.

1.5.4. API Figshare

Figshare ofrece una API pública que permite a desarrolladores y sistemas externos interactuar y acceder a los datos y recursos almacenados en Figshare de una manera programática. Esto significa que los desarrolladores pueden crear aplicaciones personalizadas, herramientas de análisis, integraciones con otras plataformas, o automatizar tareas relacionadas con Figshare utilizando la API (Figshare, 2023).

La API de Figshare brinda acceso a funciones como subir, descargar y administrar archivos, obtener metadatos de los artículos y conjuntos de datos, buscar contenido específico, entre otras acciones. Es una manera poderosa de extender las funcionalidades de Figshare y facilitar la integración de la plataforma en flujos de trabajo más amplios de investigación y desarrollo.

Los elementos fundamentales de la API:

- Endpoints: Los endpoints son las URL específicas que se utilizan para interactuar con la API. Cada acción o recurso disponible en Figshare tiene un endpoint asociado que permite realizar operaciones específicas.
- Recursos: En el contexto de la API Figshare, los recursos son los datos que se pueden acceder, crear, actualizar o eliminar a través de la API. Los recursos pueden incluir artículos, conjuntos de datos, archivos, proyectos y metadatos asociados.
- Métodos HTTP: La API Figshare utiliza métodos HTTP para indicar el tipo de acción que se desea realizar en un recurso determinado. Los métodos HTTP más comunes utilizados en la API Figshare son:
    - GET: Obtener datos o recursos.
    - POST: Crear un nuevo recurso.
    - PUT: Actualizar un recurso existente.
    - DELETE: Eliminar un recurso.
- Autenticación: Para acceder a la API Figshare y realizar operaciones, generalmente es necesario autenticarse. La autenticación se logra mediante

tokens de API, que se generan desde la cuenta del usuario en Figshare y se incluyen en las solicitudes para verificar la identidad y los permisos del usuario.

- Respuestas: Cuando se realiza una solicitud a la API Figshare, se devuelve una respuesta que puede contener datos solicitados o información sobre el estado de la solicitud (por ejemplo, éxito, error, etc.). Las respuestas de la API generalmente están en formato JSON.
- Paginación: Dado que algunos resultados pueden contener grandes cantidades de datos, la API Figshare utiliza la paginación para dividir los resultados en páginas más pequeñas y manejables. Esto permite un acceso más eficiente a los recursos.

El API de Figshare contiene referencias para los desarrolladores que deseen utilizar sus servicios y funcionalidades. Estas referencias incluyen una documentación detallada con ejemplos y explicaciones sobre los diferentes endpoints y métodos disponibles, así como guías de uso y buenas prácticas para integrar la API en aplicaciones y sistemas externos. Además, es probable que proporcionen ejemplos de código en diferentes lenguajes de programación para facilitar la implementación. También pueden incluir ejemplos de solicitudes y respuestas en formato JSON o XML, según corresponda. Las referencias también pueden describir las autenticaciones y permisos necesarios para acceder a ciertas funcionalidades de la API, y cómo obtener las credenciales de acceso requeridas. En general, estas referencias están diseñadas para ayudar a los desarrolladores a comprender y utilizar eficientemente la API de figshare en sus proyectos.

1.5.5. Ventajas y Desventajas

Para Singh (2011) el repositorio Figshare posee las siguientes ventajas: el acceso al contenido se obtiene por medio de un registro gratuito; la plataforma permite iniciar sesión por medio de redes sociales populares, el proceso de carga de datos tiene la opción de agregar una descripción del experimento y la metodología; posee un acceso a un foro de discusión en cada página de la web; los datos cargados se citan automáticamente y se les asigna un identificador, lo cual ayuda a realizar búsquedas eficientes y garantiza la seguridad de los datos para cuando se acceda a largo plazo.

La relevancia de Figshare radica en su capacidad para simplificar la indexación de sus contenidos por parte de los principales motores de búsqueda y bases de datos. (Alonso-Arévalo, 2016).

Dentro de contras de usar el repositorio Figshare encontramos que a pesar de proporciona un identificador digital persistente no es tan eficiente en comparación con otros repositorios (Alonso-Arévalo, 2016); la plataforma no permite que los usuarios cambien el orden de la autoría; No permite incluir DOIs para artículos en blogs científicos.

1.6. ISO/IEC 25022

La ISO/IEC 25022, una norma esencial en el ámbito de la calidad de software y sistemas de información, es una parte integral del conjunto más amplio de normativas conocido como ISO/IEC 25000. Su enfoque principal recae en la evaluación de la calidad de los productos de software, y a menudo se la conoce por su acrónimo, SQuaRE (Software Quality Requirements and Evaluation, Requisitos y Evaluación de Calidad de Software). (Nakai et al., 2016)

Lo que distingue a la ISO/IEC 25022 es su capacidad para proporcionar un marco estructurado y completo para la evaluación de la calidad del software en múltiples dimensiones. Este marco establece un conjunto de características de calidad y subcaracterísticas que actúan como puntos de referencia clave para llevar a cabo evaluaciones exhaustivas. Estas características y subcaracterísticas abordan una amplia gama de aspectos, desde la funcionalidad y la confiabilidad hasta la usabilidad, la eficiencia, la mantenibilidad y la portabilidad del software. (Bevan et al., 2016)

En el contexto de esta norma, se desarrollan modelos de calidad específicos que simplifican la medición y evaluación de estas características y subcaracterísticas. Estos modelos se dividen en dos categorías principales:

- **Calidad en Uso:** Esta categoría pone el énfasis en cómo los usuarios experimentan y perciben la calidad del software en situaciones de uso real. Comprende aspectos como la eficiencia de uso, la efectividad, la satisfacción del usuario y otros elementos que tienen un impacto directo en la experiencia del usuario final.

- **Calidad en el Producto:** Aquí se examinan las características internas del software que afectan su calidad desde una perspectiva técnica. Esto incluye la confiabilidad, la funcionalidad, la usabilidad, la eficiencia y otros factores relacionados con el rendimiento y la estructura interna del software. Estas características pueden cuantificarse mediante métricas específicas.

Por lo que, la ISO/IEC 25022 es una herramienta invaluable en la industria del desarrollo de software y la tecnología de la información. Su propósito fundamental es establecer estándares de calidad, llevar a cabo evaluaciones objetivas de la calidad del software y comunicar los resultados de estas evaluaciones. En última instancia, busca ayudar a las organizaciones a comprender y mejorar la calidad de sus productos de software, asegurando que cumplan con los requisitos y expectativas de sus usuarios y clientes.

Métricas de calidad en uso		
Características	**Subcaracterísticas**	**Definición**
1. Eficiencia	Comportamiento Temporal	Evalúa el tiempo de respuesta del software, la velocidad de procesamiento y la eficiencia de uso de los recursos.
	Utilización de Recursos	Mide la eficiencia en el uso de recursos como CPU, memoria y almacenamiento durante la ejecución del software.
2. Efectividad	Completitud Funcional	Evalúa si el software proporciona todas las funciones requeridas y si las tareas se realizan de manera completa y precisa.
	Exactitud	Mide la precisión de los resultados y el grado en que el software produce salidas correctas.
3. Satisfacción del Usuario	Facilidad de Entendimiento	Evalúa la claridad y la facilidad de comprensión de la interfaz y la documentación del software.
	Atractivo Estético	Considera el diseño visual y la presentación general del software,

		incluyendo elementos estéticos y minimalistas.
4. Seguridad	Seguridad del Usuario	Evalúa la seguridad del usuario al utilizar el software y si se toman medidas para prevenir riesgos.
	Seguridad de Datos	Mide la integridad y confidencialidad de los datos manejados por el software.
5. Contexto de Uso	Adaptabilidad	Evalúa la capacidad del software para adaptarse a diferentes entornos y necesidades de los usuarios.
	Aprendizaje	Mide la facilidad con la que los usuarios pueden aprender a utilizar el software en un nuevo contexto de uso

Tabla 4. ISO/IEC 25022:2016. Fuente: https://www.iso.org/standard/35746.html

CAPÍTULO II

Desarrollo

2.1. Análisis, diseño y desarrollo de la arquitectura

2.1.1. Arquitecturas de Software

Arquitectura orientada a servicios (SOA)

La Arquitectura Orientada a Servicios (SOA por sus siglas en inglés, Service-Oriented Architecture) es un enfoque de diseño de software que tiene como objetivo organizar las funcionalidades de una aplicación en servicios reutilizables e independientes. En lugar de desarrollar una aplicación monolítica, SOA propone dividir la funcionalidad en pequeños servicios que se comunican entre sí mediante interfaces estándar (Bokhari et al., 2015).

Los principios fundamentales de SOA incluyen:

- Servicios reutilizables: Cada servicio es una unidad independiente de funcionalidad que puede ser utilizado en múltiples aplicaciones y contextos.
- Interoperabilidad: Los servicios se comunican entre sí a través de estándares abiertos y protocolos comunes, lo que facilita la interoperabilidad entre diferentes sistemas.

- Granularidad: Los servicios deben ser lo suficientemente pequeños y específicos para cumplir con una única funcionalidad, lo que facilita la reutilización y el mantenimiento.
- Descubrimiento de servicios: Se establece un mecanismo para que las aplicaciones puedan descubrir y utilizar servicios disponibles en la red.
- Bajo acoplamiento: Los servicios se diseñan para ser independientes unos de otros, minimizando la dependencia entre ellos.
- Composición de servicios: Las aplicaciones pueden combinar varios servicios para construir funcionalidades más complejas.
- Flexibilidad y escalabilidad: SOA permite agregar, modificar o eliminar servicios de manera más sencilla para adaptarse a los cambios y demandas del negocio.

Las ventajas de SOA:

- Reutilización: La arquitectura basada en servicios permite reutilizar los servicios en diferentes contextos, lo que mejora la eficiencia y reduce el tiempo de desarrollo.
- Flexibilidad: La capacidad de componer y recombinar servicios permite una mayor flexibilidad en el desarrollo y adaptación de las aplicaciones a los cambios empresariales.
- Interoperabilidad: Al utilizar estándares abiertos, SOA facilita la integración de sistemas heterogéneos y la comunicación entre diferentes plataformas.
- Mantenibilidad: El bajo acoplamiento entre los servicios permite que las actualizaciones y correcciones sean más sencillas y menos propensas a afectar a otros componentes del sistema.
- Escalabilidad: La capacidad para agregar nuevos servicios o replicar los existentes permite que la arquitectura sea más escalable y pueda manejar cargas de trabajo crecientes.
- Sin embargo, también hay desafíos asociados con SOA, como la complejidad de gestionar múltiples servicios y la necesidad de una planificación adecuada para asegurar una buena arquitectura y evitar problemas de rendimiento.

Es importante mencionar que SOA es un enfoque arquitectónico que ha sido ampliamente adoptado en la industria, pero también han surgido otras metodologías

y enfoques, como microservicios, que buscan abordar algunas de las limitaciones de SOA en términos de escalabilidad y despliegue.

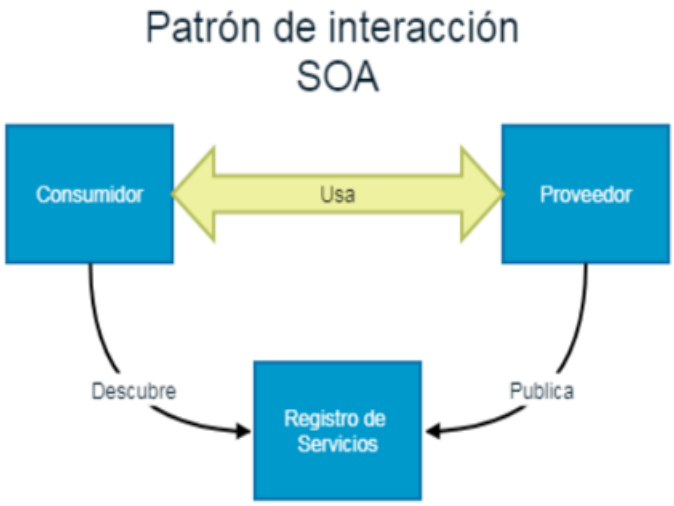


Fig. 9: Patrón Arquitectura SOA.

Fuente: https://acortar.link/cEOR6W

2.1.2. Estilos de desarrollo de Software

El Modelo-Vista-Controlador (MVC) es un patrón de arquitectura de software ampliamente utilizado para diseñar aplicaciones de software, especialmente aquellas que tienen una interfaz de usuario (UI). El objetivo principal de MVC es separar las diferentes responsabilidades y componentes de una aplicación para mejorar la modularidad, la mantenibilidad y la escalabilidad del código. (Jailia et al., 2016)

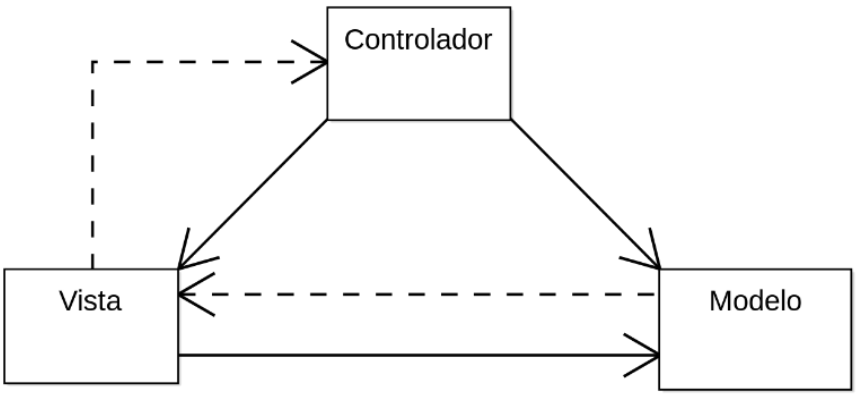


Fig. 10: Estuctura MVC.

Fuente: Propia

El patrón MVC se compone de tres componentes principales:

- Modelo (Model): Se encarga de encapsular la información y las reglas de funcionamiento fundamentales de la aplicación. En esta capa, se gestionan los datos, reglas y operaciones relacionadas con el negocio. El Modelo no tiene conocimiento sobre la interfaz de usuario y opera independientemente de ella.
- Vista (View): Es responsable de mostrar los datos del Modelo y de presentar la interfaz de usuario al usuario final. La Vista muestra la información de manera visual y también puede recoger la entrada del usuario, pero no realiza ninguna lógica de negocio. Cabe destacar que puede haber múltiples vistas que representen la misma información de diferentes maneras.
- Controlador (Controller): Actúa como intermediario entre la Vista y el Modelo. Recibe las interacciones del usuario desde la Vista y realiza las acciones necesarias en el Modelo, como actualizaciones de datos o consultas. El Controlador también se encarga de actualizar la Vista en función de los cambios realizados en el Modelo.

El flujo típico en una arquitectura MVC es el siguiente:

- El usuario interactúa con la Vista (por ejemplo, haciendo clic en un botón).
- La Vista envía la interacción al Controlador.
- El Controlador procesa la interacción y realiza las acciones necesarias en el Modelo.
- El Modelo actualiza sus datos y notifica al Controlador sobre los cambios.
- El Controlador actualiza la Vista con la nueva información del Modelo.
- La Vista muestra los cambios actualizados al usuario.

Ventajas del patrón Modelo-Vista-Controlador (MVC):

- Separación de preocupaciones: El MVC permite dividir las responsabilidades de la aplicación en componentes independientes, lo que facilita la comprensión del código y el desarrollo de nuevas características.

- Reutilización de componentes: Al separar la lógica de negocio del diseño de la interfaz de usuario, es posible reutilizar los modelos y controladores en diferentes vistas y viceversa.
- Facilidad de mantenimiento: La modularidad del MVC facilita la identificación y corrección de errores, así como la actualización y mejora de partes específicas de la aplicación sin afectar al resto del sistema.
- Escalabilidad: El MVC facilita la escalabilidad de la aplicación, ya que cada componente puede ser gestionado y escalado por separado.
- El patrón MVC es ampliamente utilizado en el desarrollo de aplicaciones web y de escritorio, así como en el desarrollo de aplicaciones móviles y otros tipos de software. Al seguir el principio de separación de preocupaciones, el MVC ayuda a crear aplicaciones más estructuradas, mantenibles y robustas.

2.1.3. Metodología SCRUM

La metodología Scrum es un enfoque dinámico y flexible para la administración de proyectos de desarrollo de software. Scrum se basa en la idea de que los proyectos son complejos y difíciles de planificar con precisión desde el principio, por lo que se adapta a medida que avanza. Se centra en el trabajo en equipo, la colaboración y la entrega iterativa e incremental de software funcional. (Faniran et al., 2017)

Los elementos principales de Scrum son los siguientes:

- Roles: Scrum define tres roles principales:
    - Product Owner: Es responsable de definir los objetivos del proyecto, establecer las prioridades y gestionar el backlog del producto.
    - Scrum Master: Es el facilitador del equipo, se encarga de asegurarse de que Scrum se implemente correctamente, de eliminar obstáculos y de fomentar un ambiente de trabajo productivo.
    - Equipo de desarrollo: Es el grupo de personas que se encarga de diseñar, desarrollar y probar el producto. Son autoorganizados y multifuncionales.
- Backlog del producto: Es una lista priorizada de los requisitos y funcionalidades del producto, también conocidos como "historias de usuario". El Product Owner es responsable de gestionar y priorizar el backlog.

- Sprint: Es una iteración fija de tiempo, generalmente de 1 a 4 semanas, durante la cual el equipo de desarrollo trabaja para entregar un conjunto de funcionalidades. Al comienzo de cada sprint, el equipo selecciona un conjunto de elementos del backlog del producto para trabajar.
- Reuniones diarias (Daily Scrum): Son reuniones cortas (generalmente de 15 minutos) en las que el equipo de desarrollo comparte su progreso, informa sobre las tareas completadas desde la última reunión y discute los posibles obstáculos.
- Revisión del sprint (Sprint Review): Al concluir cada ciclo de trabajo, el equipo presenta el progreso logrado al Propietario del Producto y a otras personas involucradas en el proyecto. Se recopilan comentarios y se realizan ajustes en el backlog del producto según sea necesario.
- Retrospectiva del sprint (Sprint Retrospective): Después de la revisión del sprint, el equipo reflexiona sobre su desempeño, identifica áreas de mejora y establece acciones para abordarlas en los sprints futuros.

El enfoque iterativo e incremental de Scrum permite una mayor flexibilidad, adaptabilidad y transparencia en la gestión de proyectos de desarrollo de software. Se prioriza la entrega continua de valor al cliente, y los cambios y ajustes pueden realizarse fácilmente en cada sprint.

Scrum se ha convertido en una de las metodologías ágiles más populares y se utiliza en una amplia variedad de proyectos, no solo en el desarrollo de software, sino también en áreas como la gestión de proyectos, el marketing, el diseño de productos y más.

2.1.4. Herramientas de trabajo

Tabla 4: Tabla de herramientas

Logo	Nombre	Descripción
Diseño		

	Balsamiq Wireframes	Esta herramienta de wireframing proporciona una solución rápida que permite a los equipos colaborar eficientemente, crear maquetas, controlar versiones y realizar pruebas de usuario. (Balsamiq, 2023)
Lenguajes de Programación, Framework, SDK, Librerías.		
	PHP	PHP es un lenguaje de programación de código abierto ampliamente conocido, especialmente adecuado para el desarrollo web, y que se puede integrar fácilmente en código HTML. (PHP, 2023)
	JSON	JSON es un formato de datos abierto que se emplea como alternativa al XML para transferir información estructurada entre un servidor web y una aplicación web.
	JavaScript	JavaScript es un lenguaje de programación interpretado, lo que significa que no necesita ser compilado antes de ser ejecutado por un navegador web. Esto permite a los desarrolladores agregar interactividad y funcionalidad dinámica a sitios web, lo que mejora significativamente la experiencia del usuario.
	Visual Code	Es un editor de código fuente desarrollado por Microsoft. Es una herramienta altamente popular y ampliamente utilizada por desarrolladores y programadores debido a su amplia gama de características y su enfoque en la simplicidad y la eficiencia.
	MySQL	MySQL es una plataforma de administración de bases de datos relacionales de código abierto.
	Bootstrap	Bootstrap es un framework de desarrollo frontend de código abierto. Está basado en HTML, CSS y JavaScript, y proporciona una serie de componentes, estilos y utilidades predefinidas que permiten a los desarrolladores crear interfaces visuales atractivas y funcionales de manera más rápida y sencilla.
	Xampp	XAMPP es una suite de software de código abierto y gratuita que facilita la creación de un entorno de desarrollo local para aplicaciones web.

Servicios Web, DevOps, Contenedores, Monitoreo Continuo TI.		
	Hostinger	Hostinger es una empresa de alojamiento web y proveedor de servicios de hospedaje en línea. Ofrece una variedad de servicios de alojamiento web compartido, VPS y alojamiento en la nube, así como otros servicios relacionados con el registro de dominios y la creación de sitios web.
	POSTMAN	Postman es una herramienta de desarrollo y colaboración de API que permite a los desarrolladores probar, documentar, compartir y depurar APIs de manera eficiente.

2.1.5. Configuración de las plataformas Figshare y GitHub

Configuración de la plataforma Figshare

Para poder configurar la plataforma de Figshare, primero toca crear una cuenta en dicho repositorio, a continuación, se indicará como:

- Registro y creación de cuenta: Lo primero que se debe hacer para utilizar Figshare es registrarse en la plataforma y crear una cuenta de usuario. Para ello, se deben proporcionar los datos personales necesarios y seguir las instrucciones de registro.
- Inicio de sesión: Una vez que se tiene una cuenta, se puede iniciar sesión en Figshare utilizando las credenciales de inicio de sesión (nombre de usuario y contraseña) proporcionadas durante el registro.
- Creación de perfiles y configuración personal: Después del inicio de sesión, se puede configurar el perfil personal del usuario, lo que incluye información profesional, afiliación institucional, intereses de investigación, entre otros detalles.

Una vez creada la cuenta en Figshare, se procede a crear un token, que nos permitirá conectar la API de Figshare con la aplicación orientada a servicios, a continuación, se indica el proceso a seguir:

- Iniciar sesión en Figshare: Primero, asegúrate de tener una cuenta en Figshare y haber iniciado sesión en la plataforma.
- Acceder a la sección "API tokens": Una vez que hayas iniciado sesión, busca y accede a la sección "API tokens" o "Tokens de API". Esta sección se

encuentra generalmente en la configuración de la cuenta o en la sección de desarrolladores de la plataforma.

- Generar un nuevo token: Dentro de la sección "API tokens", busca la opción para generar un nuevo token. Es posible que debas proporcionar una breve descripción o nombre para identificar el propósito de este token.
- Configurar permisos: A continuación, se te presentarán opciones para configurar los permisos del token. Esto define qué acciones se pueden realizar con el token, como cargar contenido, acceder a datos privados, editar publicaciones, entre otras. Asegúrate de seleccionar los permisos adecuados según las necesidades de tu proyecto.
- Guardar el token: Una vez que hayas generado el token y configurado los permisos, Figshare te proporcionará el token de acceso. Es importante
- guardar este token en un lugar seguro, ya que actúa como una clave de autenticación y debe mantenerse privado.

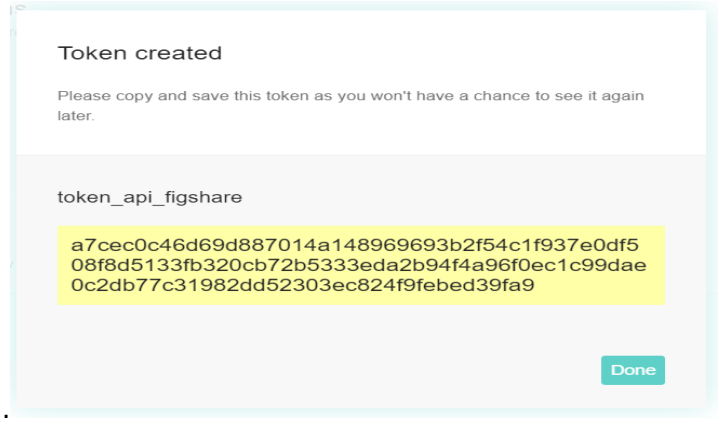


Fig. 11: Creación Token de Figshare.

Fuente: Propia.

Configuración de la plataforma GitHub

La configuración de la plataforma GitHub implica ajustar diferentes aspectos de tu cuenta y repositorios para adaptarlos a tus necesidades y preferencias. A continuación, se presenta una guía general de cómo configurar tu cuenta y repositorios en GitHub:

Configuración de la cuenta:

- Registro y creación de cuenta: Lo primero que debes hacer es registrarte en GitHub y crear una cuenta de usuario. Proporciona los datos requeridos, como tu nombre, dirección de correo electrónico y una contraseña segura.
- Verificación de la cuenta: Una vez registrada, verifica tu dirección de correo electrónico siguiendo el enlace que te enviará GitHub a tu bandeja de entrada.
- Configuración de la cuenta: Inicia sesión en tu cuenta de GitHub y selecciona tu imagen de perfil en la esquina superior derecha. Luego, selecciona "Settings" (Configuración) en el menú desplegable. Aquí podrás ajustar opciones como la dirección de correo electrónico, nombre de usuario, contraseña, notificaciones y más.
- Seguridad de la cuenta: Dentro de la sección de configuración de seguridad, tienes la opción de activar la autenticación de dos factores, lo cual añadirá una capa extra de protección a tu cuenta.

Creación de Token de GitHub:

- Inicia sesión en tu cuenta de GitHub en el sitio web de GitHub: https://github.com/.
- Haz clic en tu avatar de perfil en la esquina superior derecha y selecciona "Settings" (Configuración) en el menú desplegable.
- En la página de configuración, selecciona "Developer settings" (Configuración de desarrollador) en el menú de la izquierda.
- Luego, haz clic en "Personal access tokens" (Tokens de acceso personal) en el submenú.
- En la siguiente página, haz clic en el botón "Generate token" (Generar token).
- Aparecerá una página donde podrás configurar el token. Aquí debes proporcionar una descripción para el token, lo que te ayudará a recordar su propósito.
- A continuación, selecciona los permisos que deseas otorgar al token. Los permisos se dividen en categorías como "repo" (repositorios), "user" (usuarios), "admin:org" (organizaciones), entre otros. Selecciona solo los permisos necesarios para las tareas que planeas realizar con el token.
- Después de seleccionar los permisos, desplázate hacia abajo y haz clic en el botón "Generate token" (Generar token).

- Una vez generado el token, aparecerá en una pantalla con un mensaje indicando que debes copiarlo y guardarlo en un lugar seguro, ya que no podrás verlo nuevamente. Es importante copiar el token y guárdalo. GitHub no mostrará el token completo nuevamente.

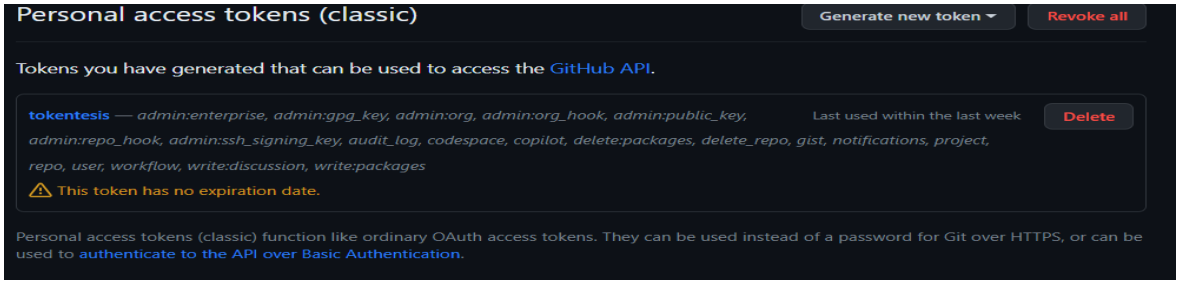


Fig. 12: Token de GitHub.

Fuente: Propia.

2.1.6. Consumo de las APIs de Figshare y GitHub.

El consumo de las API REST de GitHub se refiere a la interacción y utilización de las API RESTful proporcionadas por GitHub para acceder y manipular datos en la plataforma GitHub. Lo cual implica enviar solicitudes HTTP a los puntos de enlace (endpoints) correspondientes, con los parámetros necesarios y opcionalmente con credenciales de autenticación, como tokens de acceso personal o claves de API. Estas solicitudes pueden ser utilizadas para obtener información sobre repositorios y usuarios, crear nuevos repositorios, gestionar problemas y solicitudes de extracción, realizar acciones de colaboración y mucho más.

El consumo de la API REST de Figshare se refiere a la forma en que los desarrolladores interactúan con la API para acceder a los recursos y realizar operaciones en la plataforma Figshare.

En las Fig. 13 y Fig. 14 se muestran el consumo de la APIs de GitHub y Figshare con Postman:

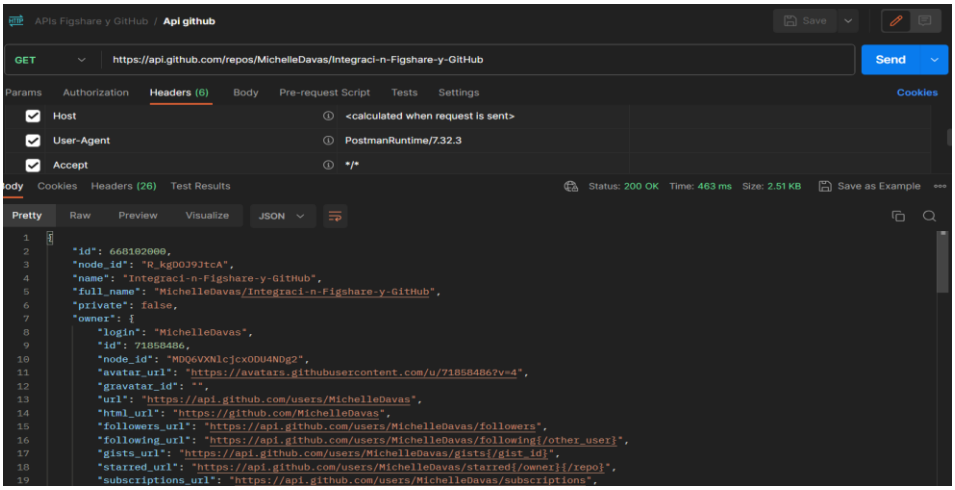


Fig. 13: Consumo de la APIs de GitHub, Postman.

Fuente: Propia.

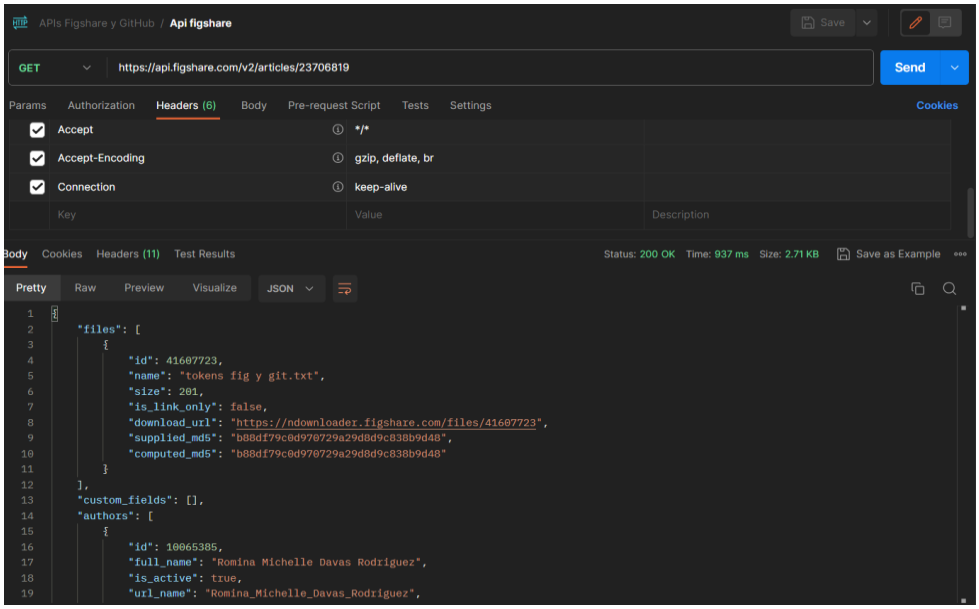


Fig. 14: Consumo de la API de Figshare, Postman.

Fuente: Propia.

.

2.2. Desarrollo

2.2.1. Sprint 0

Este proyecto utilizará la plataforma Figshare para mostrar todo el contenido, las tecnologías, los lenguajes de programación, etc. que se han producido durante las fases de desarrollo de los diversos componentes. Los componentes iniciales serán limitados ya que estarán en la fase de desarrollo en todo momento. Implementación de arquitectura orientada a servicios. Los componentes como el código fuente, los lenguajes de programación, las bibliotecas y las tecnologías utilizadas en el desarrollo

se organizarán en una estructura ordenada sugerida por Figshare Framework. El análisis de la arquitectura y el posterior desarrollo de la publicación de contenidos de Open Science requiere tareas; Los métodos y herramientas de Scrum se utilizan para realizar estas tareas correctamente.

2.2.1.1. Definición Roles Scrum

Tabla 5. Roles Scrum.

Nombre	Rol	Descripción
Product Owner	MSc. Cathy Guevara	Dar seguimiento a que las funcionalidades sean correctamente implementadas en el aplicativo que se convertirá en un componente de la investigación interna llamada “Arquitectura híbrida de REST y GraphQL por contrato”.
Scrum Master	MSc. Cathy Guevara	Una persona facultada para comprender las pautas, puntos e hitos de las reglas del marco Scrum y proporcionar pautas para la implementación con el fin de desarrollar el producto lo mejor posible.
Development Team	Romina Michelle Davas Rodriguez	Es responsable de desarrollar todas las tareas necesarias que crean el valor del producto.

2.2.1.2. Historias Épicas

El término "epic" se refiere a Scrum cuando el objetivo de Scrum es lograr resultados de acuerdo con diferentes pautas que agreguen más valor al producto final. La definición de dos historias épicas para hacer al mismo tiempo.

Tabla 6: Historias épicas. Fuente: Propia.

Código	Título	Prioridad
HE.01	Desarrollar aplicación orientada a servicios para publicar contenido Open Science, arquitectura SOA.	Alta

HE.02	Integrar las plataformas Figshare y GitHub en el aplicativo.	Alta

2.2.1.3. Product Backlog

La historia épica predefinida se divide en varias historias de usuario para satisfacer los requisitos funcionales y no funcionales, que se priorizan y ponderan. Se presentan dos historias épicas para ser sintetizadas en historias de usuarios para crear componentes arquitectónicos. Scrum, al ser una metodología ágil, permite una mejor optimización del tiempo, por lo que cada Sprint tiene un límite de tiempo de unas dos semanas, dando como resultado el siguiente Product Backlog.

Tabla 7: Producto Backlog.

Historias de Usuario				
H.Épica	**Código**	**Título**	**Peso**	**Prioridad**
HE.01	FIGS.01	Configurar las plataformas Figshare y GitHub.	1	Alta
HE.01	FIGS.02	Diseño de la arquitectura orientada a servicios.	2	Alta
HE.01	FIGS.03	Diseño de las interfaces de usuario en Balsamiq Wireframes.	2	Alta
HE.01	FIGS.04	Desarrollar el modelo lógico de la base de datos.	2	Alta
HE.01	FIGS.05	Creación de proyecto MVC en VSCORE.	1	Alta
HE.01	FIGS.06	Maquetación de Template Dashboard acorde a lo que se necesita.	2	Alta
HE.01	FIGS.07	Creación y modelado de nuevo y consulta Tickets.	1	Alta
HE.01	FIGS.08	Generación de acceso con roles.	1	Alta
HE.01	FIGS.09	Creación de Detalle Tickets.	2	Alta
HE.01	FIGS.10	Creación de mantenimiento de usuario.	2	Alta
HE.01	FIGS.11	Creación de la función Asignación de Tickets.	1	Alta
HE.01	FIGS.12	Creación de categorías, subcategorías y prioridades.	1	Media
HE.01	FIGS.13	Creación de Filtro avanzado de Tickets.	1	Alta
HE.01	FIGS.14	Creación de mantenimiento de categorías, subcategorías y prioridades.	1	Media
HE.02	FIGS.15	Consumo de las APIs de las plataformas Figshare y GitHub.	1	Media
HE.02	FIGS.16	Crear publicación en Figshare desde la aplicación orientada a servicios.	1	Media
HE.02	FIGS.17	Generar repositorios de GitHub desde la aplicación orientada a servicios.	1	Media
HE.02	FIGS.18	Crear flujo de trabajos.	1	Alta
HE.02	FIGS.19	Integrar APIS de las plataformas Figshare y GitHub.	1	Alta

HE.02	FIGS.20	Publicar contenido Open Science en las plataformas.	1	Alta

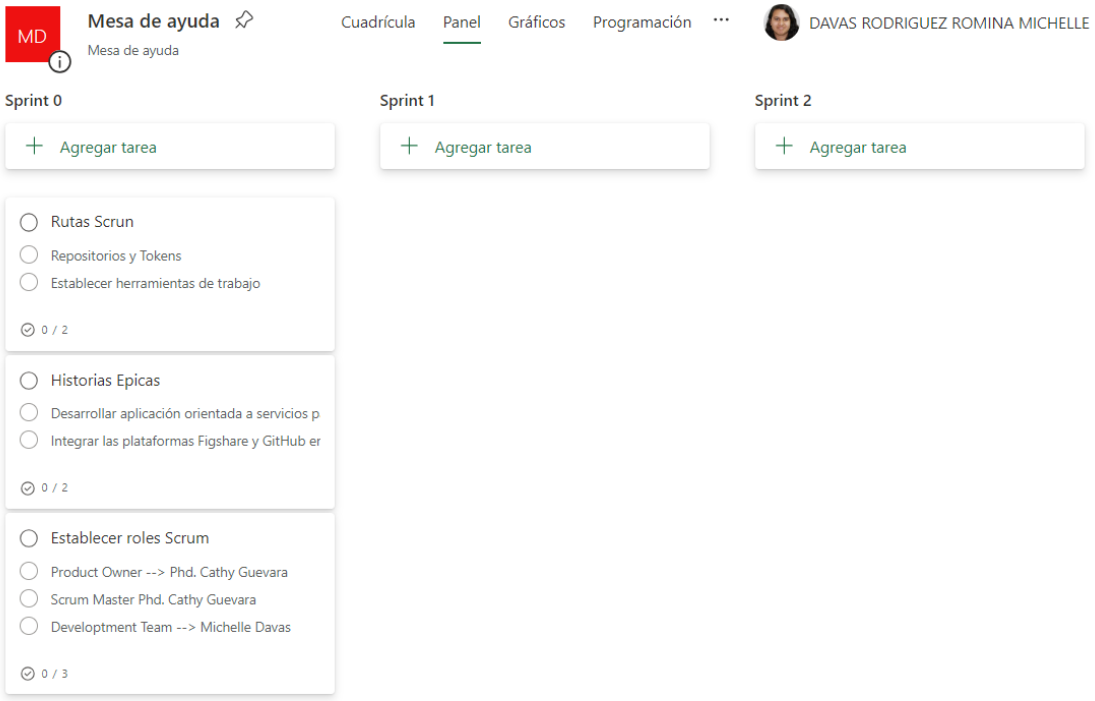


Fig. 15: Micrsoft Planner - Sprint 0

2.2.2. Sprint 1

2.2.2.1. Sprint Planning

El objetivo del sprint número 1 era comenzar a desarrollar la arquitectura de la aplicación y la base de datos asociada y establecer los siguientes objetivos:

- Describir la arquitectura y las relaciones entre sus componentes.
- Desarrollar una arquitectura global orientada a servicios, base de datos, arquitectura SOA, arquitectura de nube y capa de sistema.
- Diseñar la interfaz de la aplicación.
- Mejorar el entorno de trabajo y desarrollo de la aplicación.
- Configuración de la tutela.
- Genera tokens de acceso y permisos de desarrollador.

2.2.2.2. Sprint Backlog

El sprint tiene en cuenta las historias de usuario descritas en el Anexo 1, que definen los criterios de aceptación y las tareas a realizar.

El modelo de datos físico de la base de datos

Se elaboró un diagrama de entidad-relación en el diseño físico de la base de datos y se aplicó la normalización requerida. En la Ilustración 30 se presenta el modelo básico que se generó al crear el proyecto enfocado en servicios. El Framework Laravel es altamente flexible y permite la generación de tablas a partir de los modelos relacionados con los controladores. Para ello, utiliza tablas generadas automáticamente que deben ser "sembradas" o "llenadas" con datos de prueba.

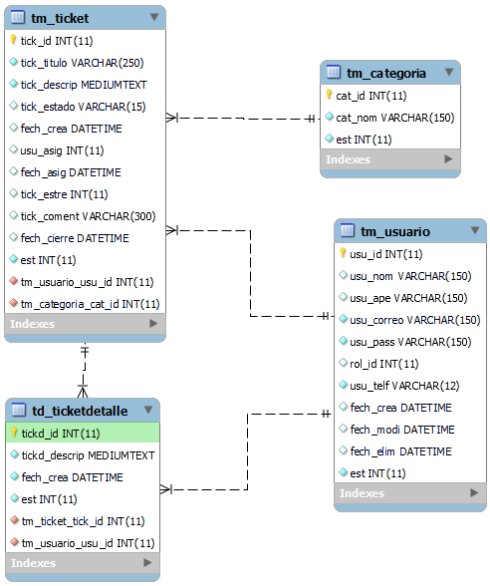


Fig. 16: Base de datos versión 1.

Fuente: propia.

Aplicación Orientada a servicios, mapa de procesos

El mapa de procesos es una herramienta esencial para visualizar y comprender los procesos y sistemas, lo que resulta fundamental para optimizar la eficiencia y tomar decisiones informadas en diversos contextos empresariales y operativos.

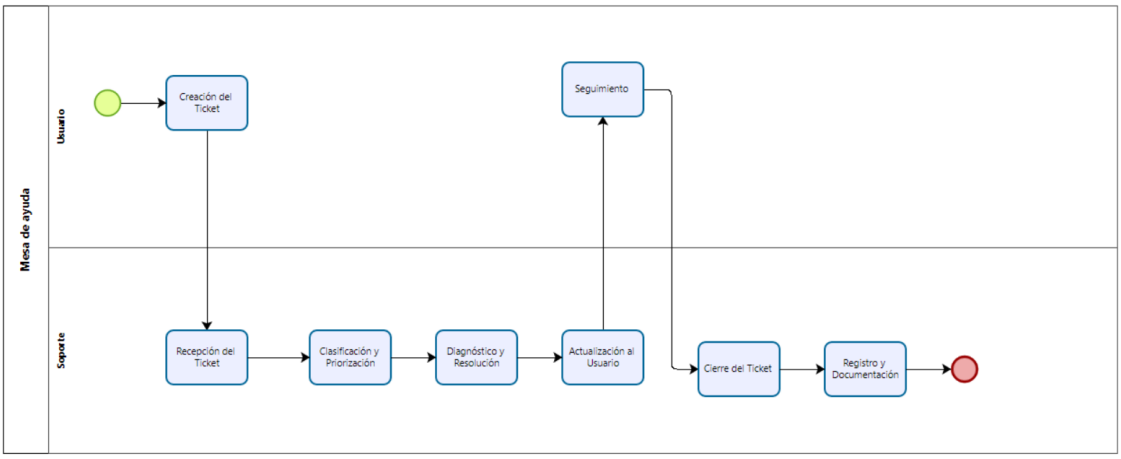


Fig. 17: Mapa de procesos Bizagi Modeler

Fuente: Propia.

Interfaces de Usuario - Balsamiq Wireframes

En la maquetación actual, el objetivo era crear interfaces de usuario modernas y amigables. Para lograr esto, se utilizó Sass (Syntactically Awesome Style Sheets) y elementos responsivos, que se ejecutan rápidamente en scripts interpretados por el navegador, brindando una experiencia rápida e intuitiva al usuario.

Una vez que se definieron las tecnologías de Frontend, se desarrolló un esquema lógico de navegación. Esto permitió desarrollar las interfaces de manera ordenada y mantener una estructura global coherente en la aplicación.

Para diseñar las maquetas de las interfaces, se empleó Balsamiq Wireframes, un software de gráficos vectoriales que permite crear y simular interfaces de usuario profesionales. Las interfaces de usuario se diseñaron y probaron tanto para computadoras como para smartphones, con el objetivo de brindar una mejor experiencia al usuario. En el entorno de Balsamiq Wireframes, se definieron varias capas de trabajo, entre las que se destacan las zonas: [continuar con la descripción de las zonas específicas].

- Pantalla principal: Es el área designada para ubicar el banner y los botones principales que brindan acceso a los contenidos recientemente agregados y disponibles.

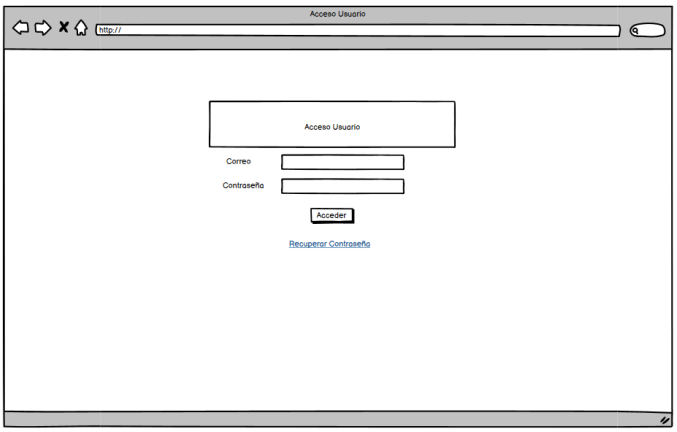


Fig. 18: Página principal, diseñada en Balsamiq Wireframes.

Fuente: Propia.

- Menú: Es crucial destacar que esta área es de gran relevancia, ya que se encargará de gestionar de manera integral la navegación, tal y como se ha establecido en el diagrama de navegación y estructura de la aplicación.
- Zona Información General: En esta área se encuentra la barra de navegación y datos estadísticos de la aplicación.

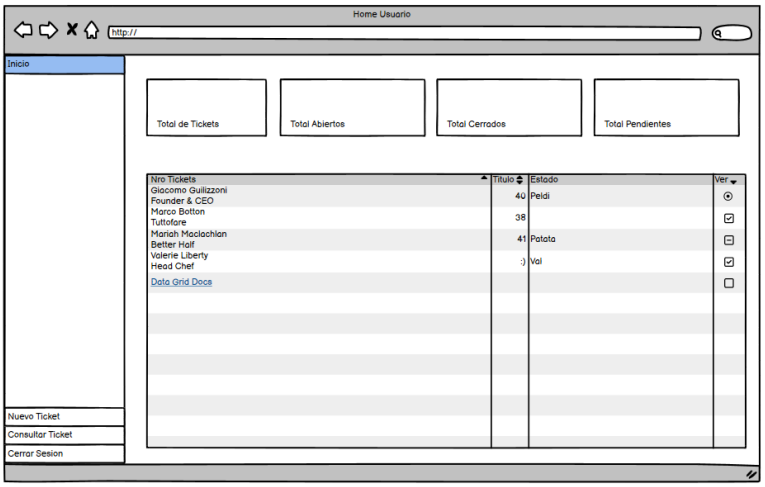


Fig. 19: Menú y zona general, diseñado en Balsamiq.

Fuente: Propia.

- Zona de contenido: se encuentra las consulta y detalle de tickets.

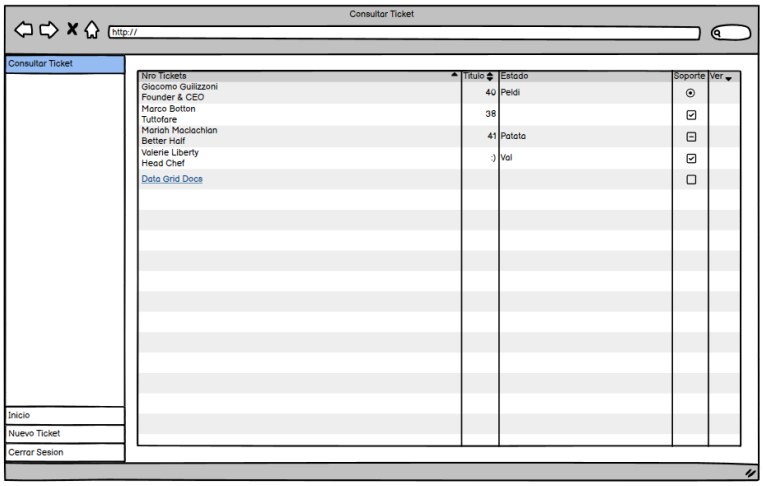


Fig. 20: Zana de contenido, diseñado en Balsamiq.

Fuente: Propia.

Diseño de la arquitectura de la Aplicación

El diseño de la arquitectura se ha estructurado en múltiples capas que albergarán toda la lógica del aplicativo. Entre ellas, se pueden destacar:

- Capa de Negocio: Esta capa albergará la lógica backend del aplicativo y se ubicará en un servidor de Hostinger.
- Capa de Monitoreo: En esta capa se encontrarán las herramientas necesarias para obtener métricas de rendimiento durante las etapas de desarrollo y despliegue del software.
- Capa de Datos: En esta capa se ubicará el servidor de la base de datos relacional MySQL, con las medidas de seguridad correspondientes.
- Capa de Seguridad: Aquí se encontrarán las medidas de seguridad que protegen la integridad del sistema, así como los componentes de acceso, como JWT y Access Token con privilegios de lectura/escritura.
- Capa de Integración de API FigShare y GitHub: Esta capa permitirá la interconexión de las API Rest de FigShare y GitHub, facilitando la gestión del contenido de Open Science en la aplicación. Se utilizarán microservicios de ambas plataformas para gestionar el contenido abierto.
- Capa del Cliente: Esta capa es responsable de habilitar la comunicación entre las aplicaciones tanto del backend como del frontend. En esta capa se desarrollarán los endpoints necesarios y se realizará el consumo de los servicios web de otras plataformas.

Diseño de la arquitectura de la Aplicación

Se ha diseñado la arquitectura con múltiples capas, cada una de ellas encargada de diferentes aspectos del aplicativo. Estas capas son las siguientes:

- Capa del Negocio: Aquí se alojará la lógica backend del aplicativo y estará hospedada en AWS. Este servidor permitirá la conexión con los datos y el consumo de servicios de otras plataformas como Google Cloud, GitHub y otras.
- Capa de Monitoreo: En esta capa se implementarán herramientas para obtener métricas de rendimiento eficientes durante las etapas de desarrollo y despliegue del software a producción.
- Capa de Datos: Aquí se encuentra el servidor de la base de datos relacional MySQL, con las debidas medidas de seguridad implementadas.
- Capa de Seguridad: Esta capa contiene las métricas de seguridad para proteger la integridad del sistema y los componentes de acceso, como JWT (JSON Web Tokens) y Access Token con privilegios de lectura y escritura.
- Capa de Integración API Figshare y GitHub: Esta capa facilitará la interconexión entre las dos API REST y gestionará el contenido Open Science mediante la aplicación. Utilizará los microservicios de ambas plataformas para administrar el contenido abierto.
- Capa de Cliente: La capa del cliente es el punto de interacción donde las aplicaciones del backend y frontend establecerán comunicación. En esta capa, se llevará a cabo el desarrollo de los "endpoints," lo que resultará en una mayor integración con servicios provenientes de otras plataformas. Este concepto se encuentra representado en la Figura 18 de la arquitectura de desarrollo orientada a servicios (SOA).

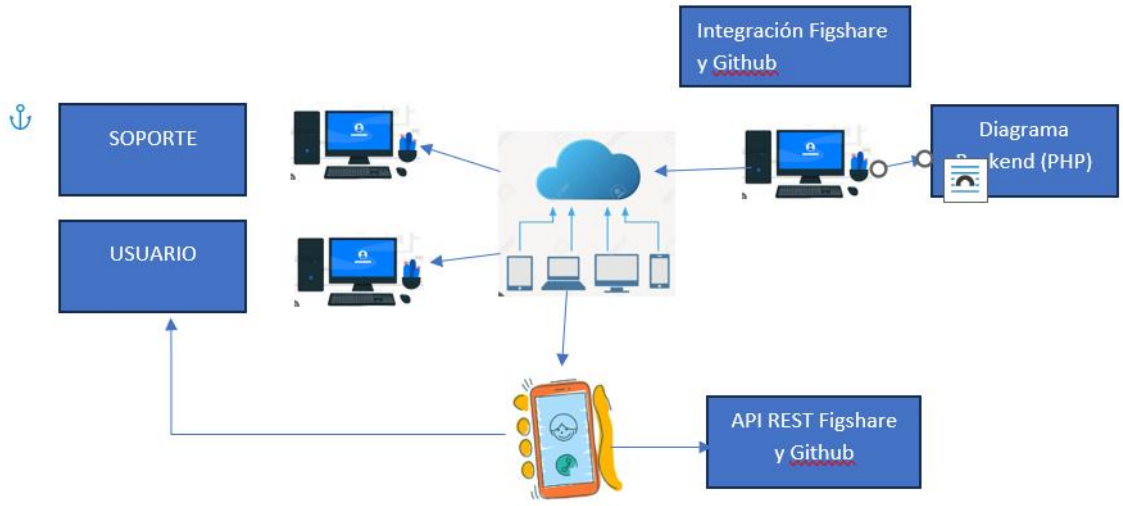


Ilustración 21. Capa de usuario. Fuente: Propia.

Mediante esta estructura en capas, el aplicativo podrá funcionar de manera eficiente y segura, permitiendo una gestión fluida de la información y el contenido entre diferentes plataformas y servicios.

2.2.2.3. Sprint Review

Tabla 8: Sprint Review 1.

Código	Descripción	Cumple
FIGS.01	Configurar las plataformas Figshare y GitHub.	Si
FIGS.02	Diseño de la arquitectura orientada a servicios.	Si
FIGS.03	Diseño de las interfaces de usuario en Balsamiq Wireframes.	Si
FIGS.04	Desarrollar el modelo lógico de la base de datos.	Si

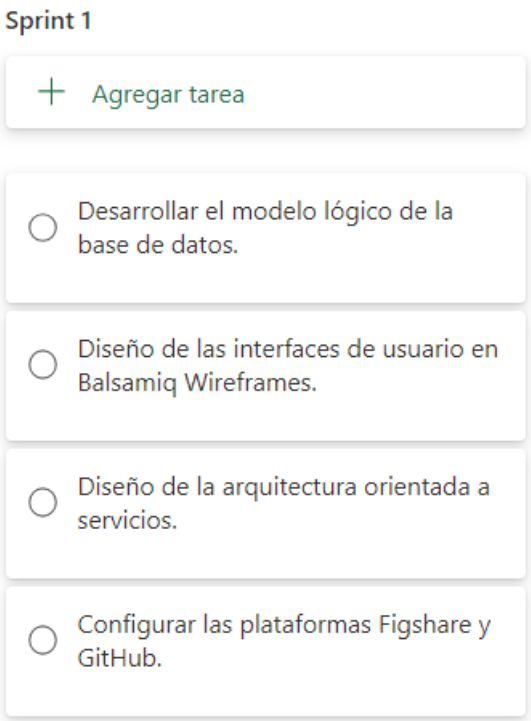


Fig. 22: Sprint 1 - Microsoft Planner

2.2.2.4. Sprint Retrospective

¿Qué salió bien en el sprint?

Levantar los entornos de trabajo, configurar el servidor y otras configuraciones iniciales permitió desplegar fácilmente la primera versión de prueba de la aplicación. Además, gracias al repositorio Figshare, los cambios realizados en los distintos componentes y paquetes se sincronizaron automáticamente. Los usuarios que utilicen el Frontend disfrutarán de interfaces de usuario rápidas y adaptables, ya que los posibles errores que podrían surgir al implementar todos los componentes en producción fueron atendidos oportunamente con herramientas de pruebas de rendimiento.

¿Cuáles fueron las lecciones aprendidas durante el sprint que podrían ayudar a mejorar la ejecución del proyecto?

Para mejorar el proyecto, se llevará a cabo una implementación gradual de los paquetes necesarios, asegurando siempre las versiones adecuadas de cada biblioteca. Las interfaces de usuario se irán desarrollando de manera progresiva,

incorporando los elementos esenciales para brindar una experiencia profesional y eficiente a los usuarios del sitio.

Se utilizó GitHub con herramientas de Integración y Distribución Continua, lo que garantizó un despliegue estable en producción, incluyendo solo los paquetes estrictamente necesarios. Además, Docker facilitó la gestión de cambios, permisos y otras tareas, asegurando que la aplicación sea escalable, segura y eficiente a través de los archivos del contenedor.

En cuanto a Hostinger, se aprovecharon sus flujos de trabajo para realizar implementaciones rápidas. Se documentaron adecuadamente todas las instalaciones, destacando especialmente el balanceo de carga, lo que asegura un despliegue con versiones claras y precisas. Esto contribuirá a la robustez y eficacia del sistema, al hacer uso de servicios computacionales complejos.

2.2.3. Sprint 2

Los objetivos por cumplir en el Sprint Número 2 fueron los siguientes:

1. Configurar y desplegar los servicios en el servidor Hostinger para el backend.
2. Desarrollar los servicios API REST para todas las entidades de la base de datos, como se había definido en el Sprint anterior.
3. Validar los puntos finales de comunicación "endpoints" y garantizar la seguridad mediante JWT.
4. Crear las estructuras de navegación y el Frontend de la aplicación.
5. Implementar políticas de acceso que regulen todas las rutas del sistema para mantener la integridad de los módulos.

2.2.3.1. Sprint Planning

Durante el Sprint número 2, se consideraron las historias de usuario detalladas en el Sprint Backlog. Se establecieron criterios de aceptación y se planificaron las tareas necesarias para llevar a cabo la implementación de dichas historias.

2.2.3.2. Sprint Backlog

Para lograrlo, se definieron criterios de aceptación y roles responsables que debían cumplirse.

2.2.3.3. Ejecución del Sprint

Durante este Sprint, se ejecutaron principalmente tareas de naturaleza ingenieril. También se explica cómo se generaron los datos de siembra para cada una de las entidades de la base de datos. La estructura de la base de datos a implementar se presenta en la Fig. 43.

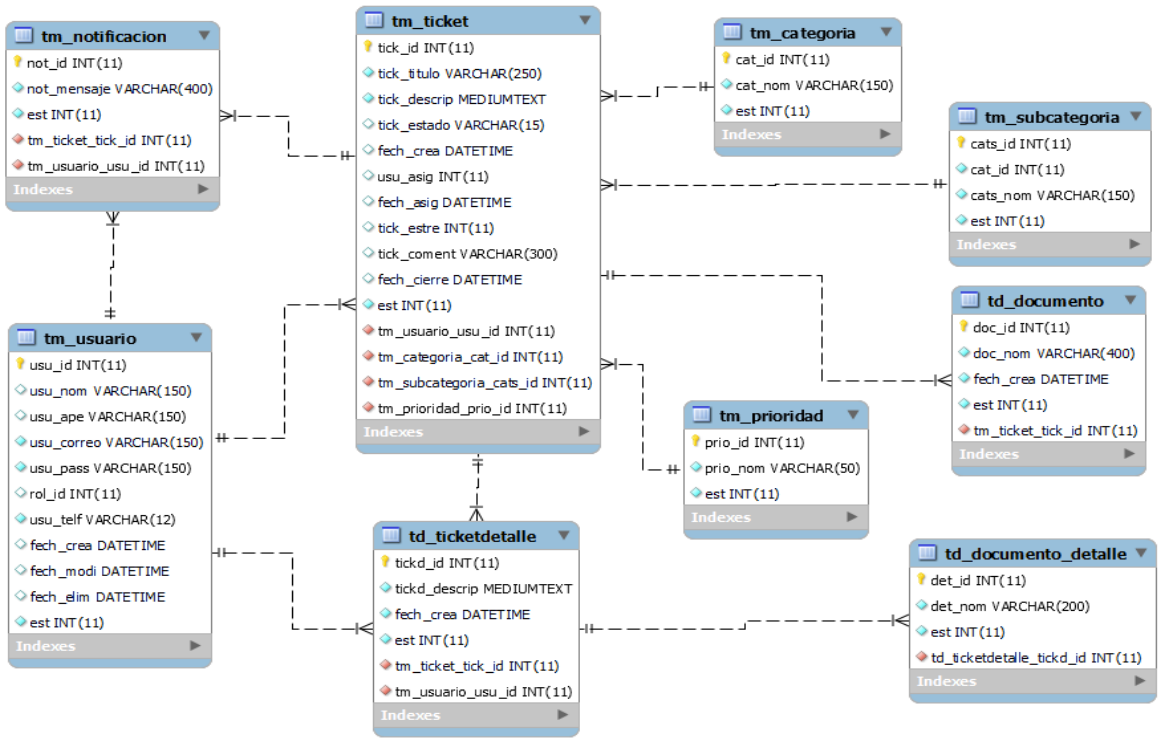


Fig. 23: Diagrama Base de Datos, versión 2.

Fuente: Propia.

Durante la ejecución del Sprint número 2, se inició el desarrollo del Backend y Frontend de navegación de la aplicación, que contiene toda la lógica del negocio. En el Backend, se implementaron los "endpoints" o puntos finales de conexión necesarios, junto con las entidades y medidas de seguridad pertinentes, siguiendo las siguientes tareas:

- Implementar las entidades detalladas en las historias de usuario.
- Implementar las medidas de seguridad en cada uno de los controladores.
- Desarrollar los servicios que interactúan con la capa de la base de datos.
- Crear los modelos necesarios.
- Creación de los controladores.
- Verificar la funcionalidad de cada "endpoint" utilizando el software Postman.

Diagrama de Procesos Frontend

Durante la ejecución del Sprint actual, se automatizaron las fases iniciales de los procesos y se logró una visión completa del alcance general de la aplicación. La representación visual del diagrama de procesos del frontend y la secuencia de almacenamiento de datos se pueden observar en la Figura 26.

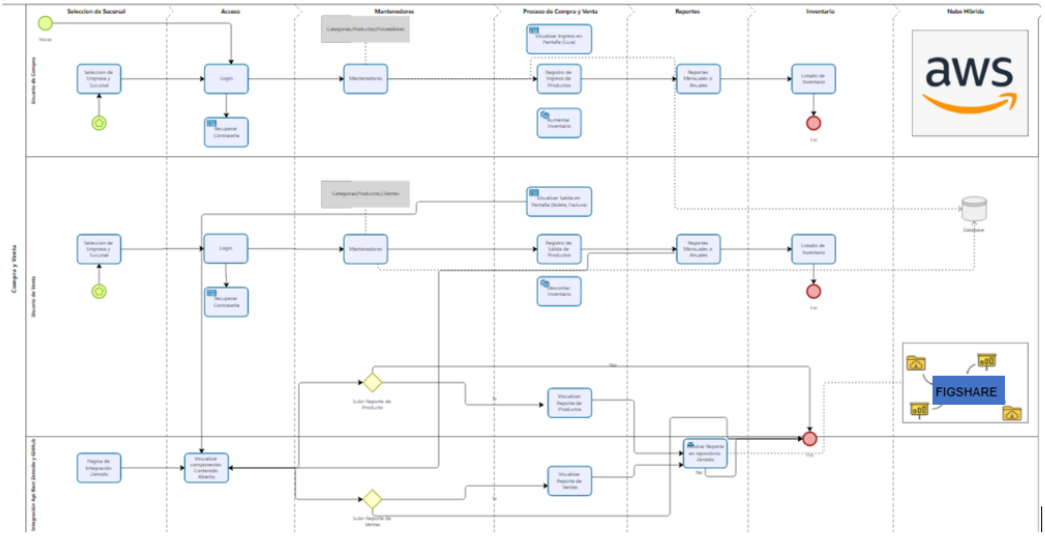


Ilustración 24: Mapa de procesos global del aplicativo

Frontend aplicación Integración Figshare & GitHub

En la Fig. 20 se puede visualizar los resultados después de desarrollar:

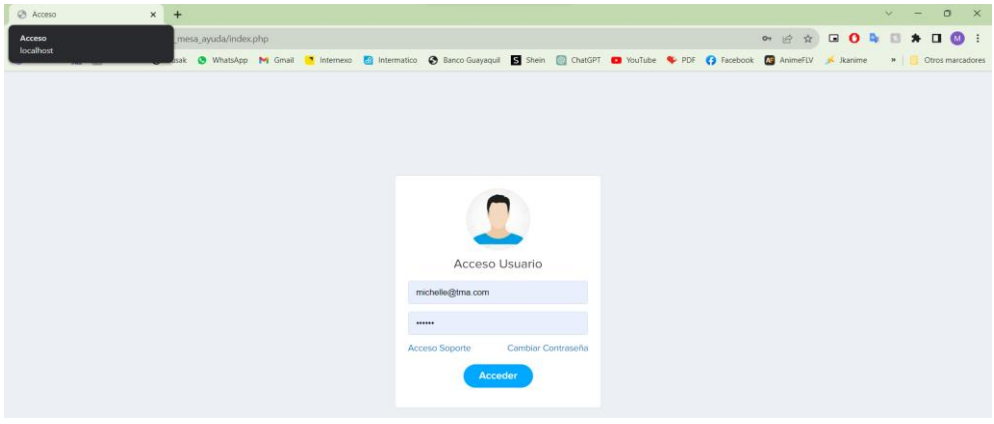


Fig 25: Login Aplicación.

Fuente: Propia

En la Fig. 21 se muestra la Página principal del rol Soporte.

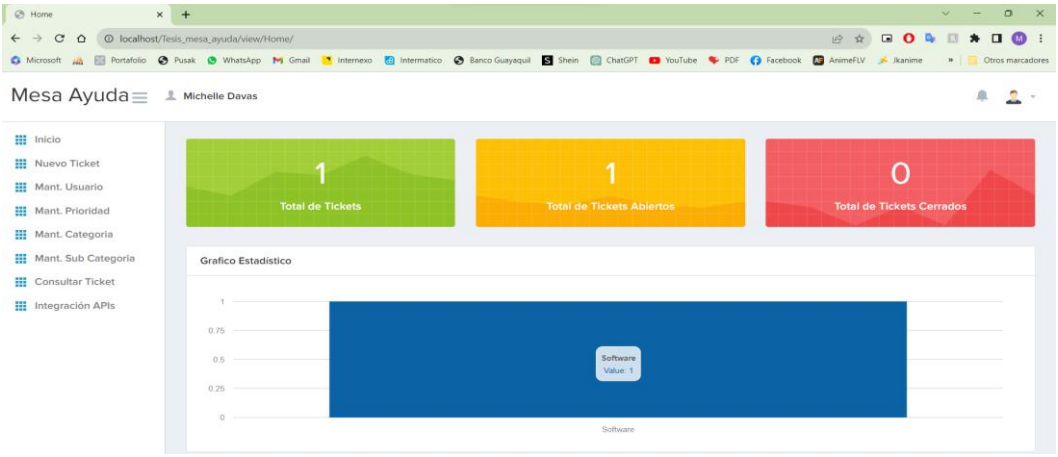


Fig. 26: Página principal de soporte.

Fuente: Propia.

En la Fig. 22 se muestra la funcionalidad agregar ticket nuevo.

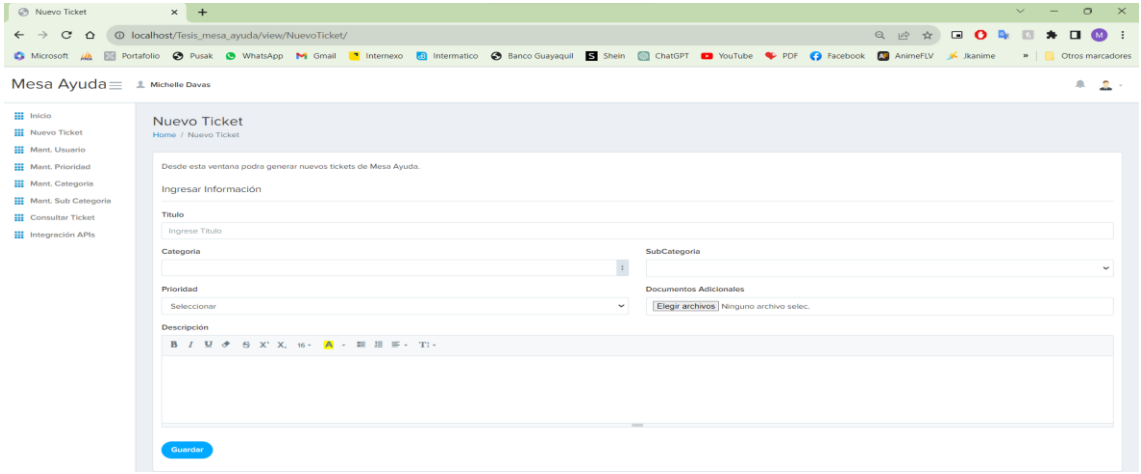


Fig. 27: Función agregar ticket nuevo.

Fuente: Propia.

En la Fig. 23 se muestra la funcionalidad Mantenimiento de Usuarios.

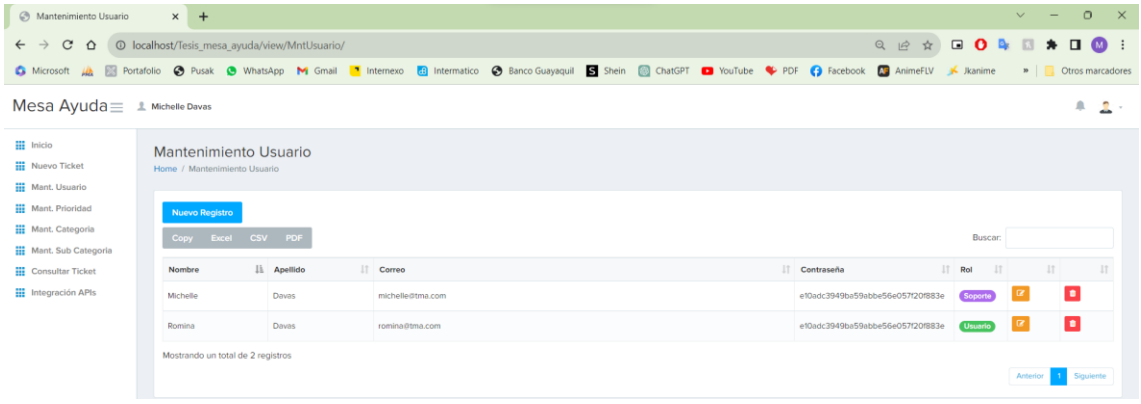


Fig. 28: Función Mantenimiento de Usuarios.

Fuente: Propia.

En la Fig. 24 se muestra la funcionalidad Mantenimiento Prioridades.

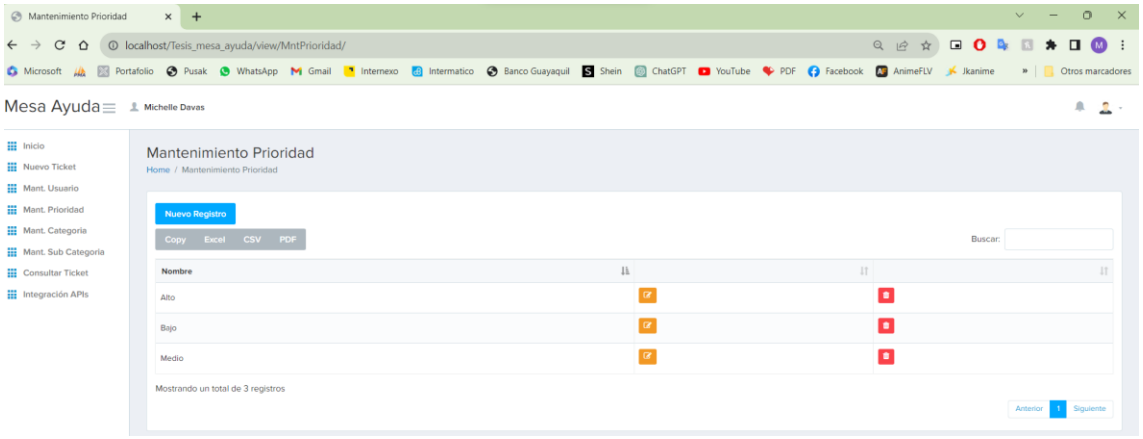


Fig. 29: Función Mantenimiento Prioridades.

Fuente: Propia.

En la Fig. 25 se muestra la funcionalidad Mantenimiento Categorías.

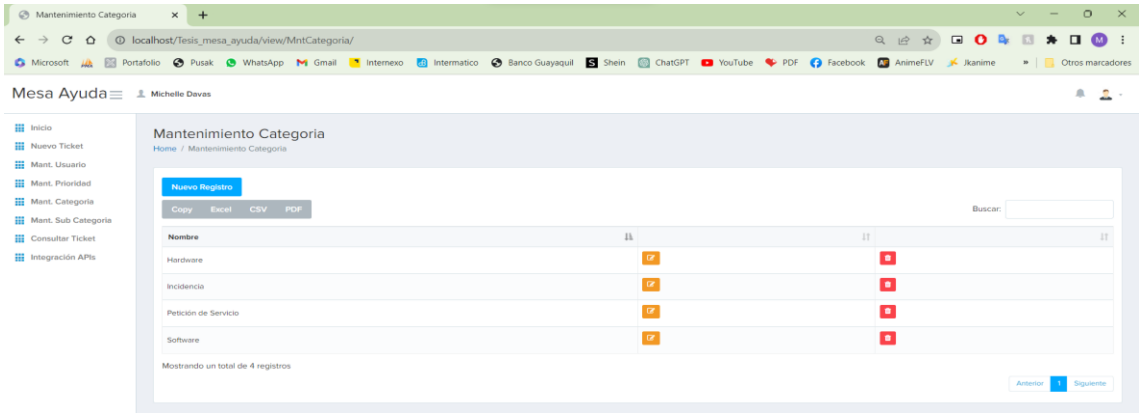


Fig. 30: Función Mantenimiento Categorías.

Fuente: Propia.

En la Fig. 26 se muestra la funcionalidad Mantenimiento Subcategorías.

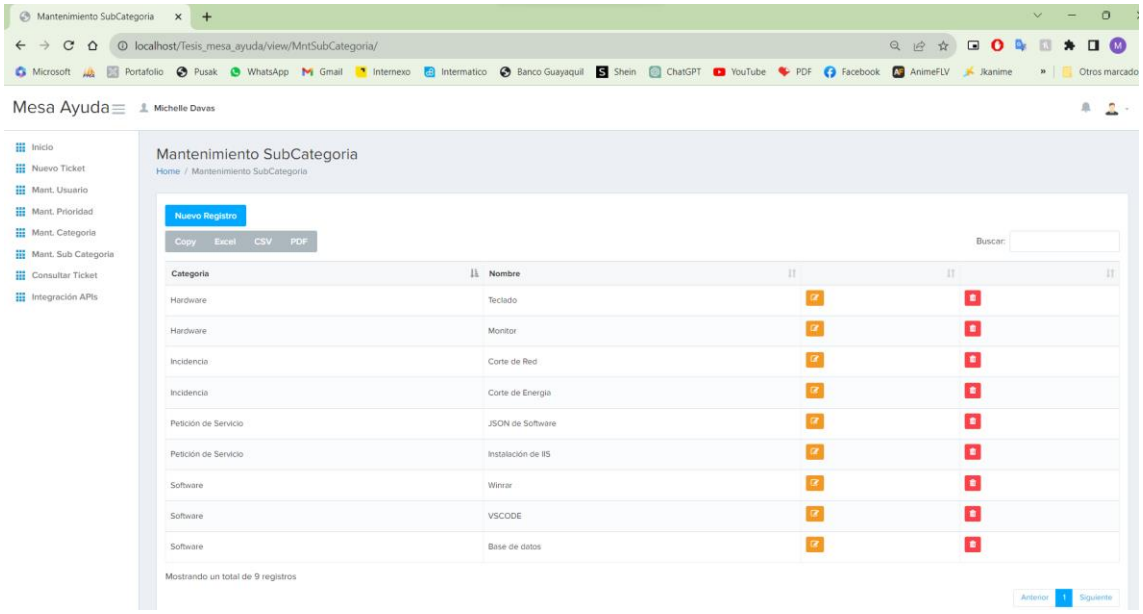


Fig. 31: Función Mantenimiento Subcategorías.

Fuente: Propia.

En la Fig. 27 se muestra la funcionalidad Consulta Ticket.

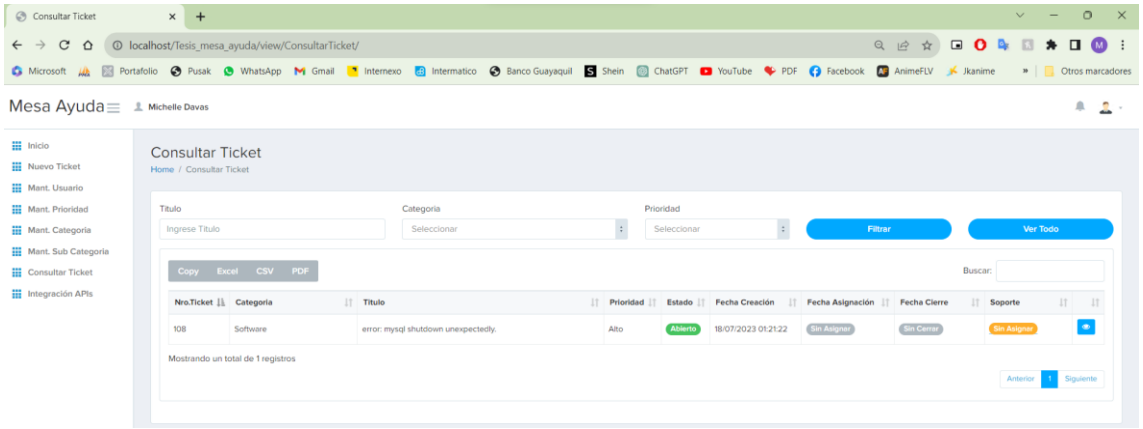


Fig. 32: Función Consulta Ticket.

Fuente: Propia.

En la Fig. 28 se muestra la funcionalidad Integración APIs.

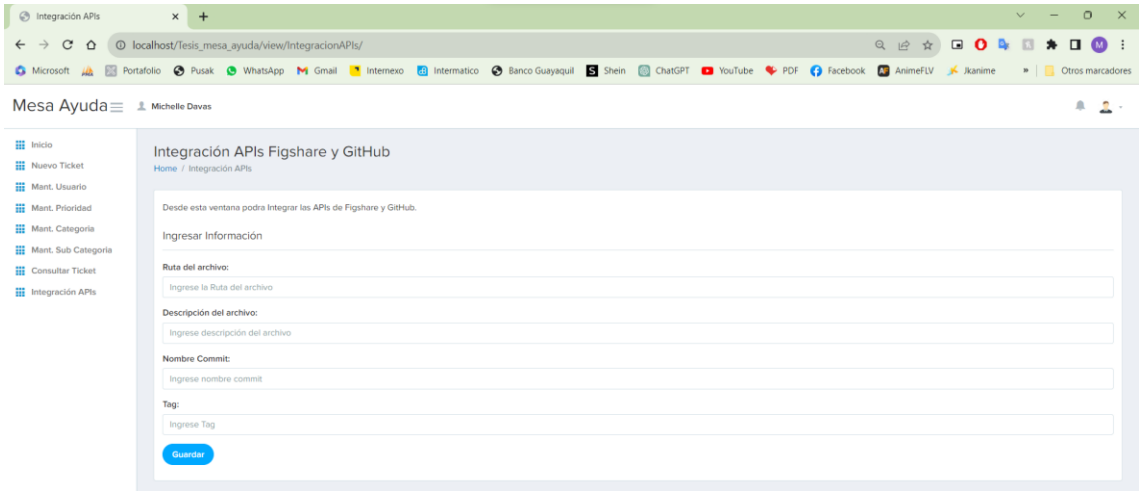


Fig. 33: Función integración de las APIS Figshare y GitHub.

Fuente: Propia.

En la Fig. 29 se muestra la Página principal del rol Usuario.

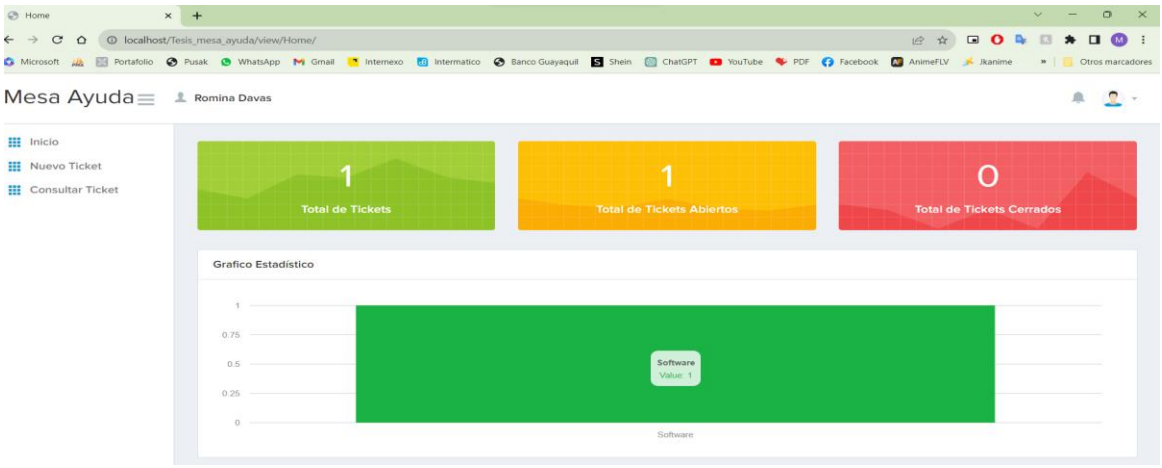


Fig. 34: Página principal rol de Usuario.

Fuente: Propia.

2.2.3.4. Sprint Review

Tabla 9: Sprint Review 2

Código	Título	Cumple
FIGS.08	Implementar las entidades por defecto en el proyecto. (migrations, users, etc).	Si
FIGS.09	Implementar las entidades de Usuario y Login.	Si

FIGS.10	Implementar las entidades de permisos y roles del sistema.	Si
FIGS.11	Implementar las entidades de los contenidos abiertos.	Si
FIGS.12	Implementar la entidad contenidos.	Si
FIGS.13	Implementar la entidad objetivos- contenidos abiertos.	Si
FIGS.15	Implementar la entidad información- contenidos abiertos.	Si
FIGS.16	Implementar la entidad requerimientos-contenidos abiertos.	Si
FIGS.17	Implementar la entidad de artículos.	Si
FIGS.18	Implementar la entidad de progreso de contenidos abiertos.	Si
FIGS.19	Implementar la entidad de soporte.	Si
FIGS.20	Implementar la entidad de usuario-contenidos abiertos.	Si
FIGS.21	Implementar la entidad de almacenamiento de datos Open Science.	Si
FIGS.22	Implementar las entidades de relación: contenido Open Science, archivos Open Science, Log del Contenido, Categorías de contenidos abiertos.	Si
FIGS.23	Implementar la entidad creative common.	Si

2.2.3.5. Sprint Retrospective

¿Qué salió bien en el sprint?

Gracias al uso y aplicación de tecnologías Frontend, se logró construir la estructura del aplicativo siguiendo las recomendaciones de buenas prácticas de cada herramienta especificada en las dependencias. Asimismo, se instalaron los paquetes necesarios que agregan valor al producto, permitiendo una escalabilidad adecuada y eficiente.

Los módulos fueron creados siguiendo una estructura bien definida, lo que asegura que el sistema sea fácil de desarrollar a medida que su tamaño y complejidad aumenten.

¿Cuáles fueron las lecciones aprendidas durante el sprint que podrían ayudar a mejorar la ejecución del proyecto?

Siguiendo las buenas prácticas recomendadas, se llevó a cabo la creación de políticas de acceso, servicios de routing y políticas específicas para cada tarea asignada. Estas

acciones permitieron asegurar las rutas de acceso al sistema, garantizando así la integridad de la aplicación y la confidencialidad de la información manejada.

En el próximo Sprint, se planificaron los procesos a automatizar, y para ello, se utilizará el diagrama de flujo mencionado en el Capítulo 1 para la publicación de contenido Open Science.

2.2.4. Sprint 3

2.2.4.1. Sprint Planning

El objetivo del Sprint número 3 fue iniciar el diseño de la arquitectura del sistema y de los datos asociados a ella, para lo cual se establecieron las siguientes metas:

- Implementar en detalle las funcionalidades requeridas por cada una de las historias de usuario.
- Establecer rutas de acceso seguras para cada uno de los roles asignados en el sistema.
- Almacenar los datos utilizando Web Services de Hostinger.
- Detallar las funcionalidades específicas para cada uno de los roles presentes en el sistema.
- Realizar una carga masiva de datos para llevar a cabo las pruebas correspondientes.
- Crear las tablas polimórficas necesarias en la base de datos.
- Migrar la base de datos a Hostinger.
- Realizar la migración del repositorio a Hostinger, utilizando Docker y GitHub Actions.
- Configurar y levantar la arquitectura de la aplicación nativa, además de instalar las dependencias necesarias.
- Configurar el despliegue del aplicativo nativo en Firebase.

2.2.4.2. Sprint Backlog

Durante el Sprint número 3, se tuvieron en cuenta las historias de usuario que se encuentran detalladas en el Sprint Backlog 3. Se establecieron criterios de aceptación y se planificaron las tareas necesarias para llevar a cabo la implementación de dichas historias.

Durante el Sprint número 3, se llevó a cabo el desarrollo de las funcionalidades correspondientes a cada uno de los roles de la aplicación. Estos roles incluyeron al Administrador de la Aplicación, Colaborador, Docente, Estudiante y Root.

El objetivo principal del Sprint fue implementar las funcionalidades específicas de cada rol, lo que resultó en flujos de trabajo donde cada rol tiene asignadas ciertas funcionalidades y privilegios dentro de la aplicación. Las tareas más destacadas durante este proceso fueron:

- Desarrollar las funcionalidades del estudiante en las áreas asignadas, con interfaces amigables y responsivas.
- Desarrollar las funcionalidades del docente en las áreas asignadas, utilizando interfaces de usuario amigables y responsivas.
- Configurar servicios Hostiger para el almacenamiento de contenido.
- Desarrollar las funcionalidades del usuario soporte en las áreas asignadas, utilizando interfaces amigables y responsivas.
- Implementar políticas de acceso para consumir datos en GitHub y Figshare.
- Establecer los elementos necesarios para navegar en el Backend de la aplicación.
- Implementar políticas de tareas asíncronas para enviar correos de revisión de contenidos.
- Crear una plantilla por defecto para el Backend de la aplicación, instalando los componentes necesarios de Bootstrap.
- Implementar funcionalidades para generar reportes de cada una de las áreas de navegación en el backend.

Proceso de Soporte

El propósito de esta sección fue crear el procedimiento de gestión y las herramientas requeridas para su implementación. La Figura 34 ilustra el flujo de trabajo del administrador en la aplicación posterior a lo mencionado.

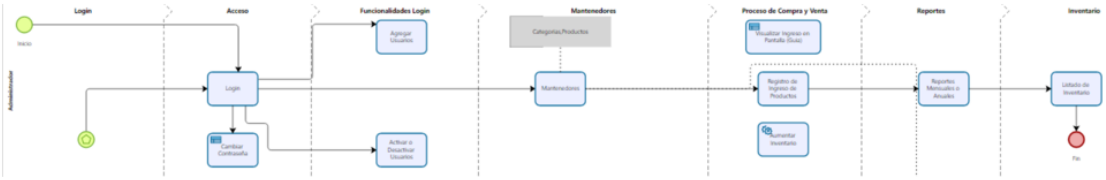
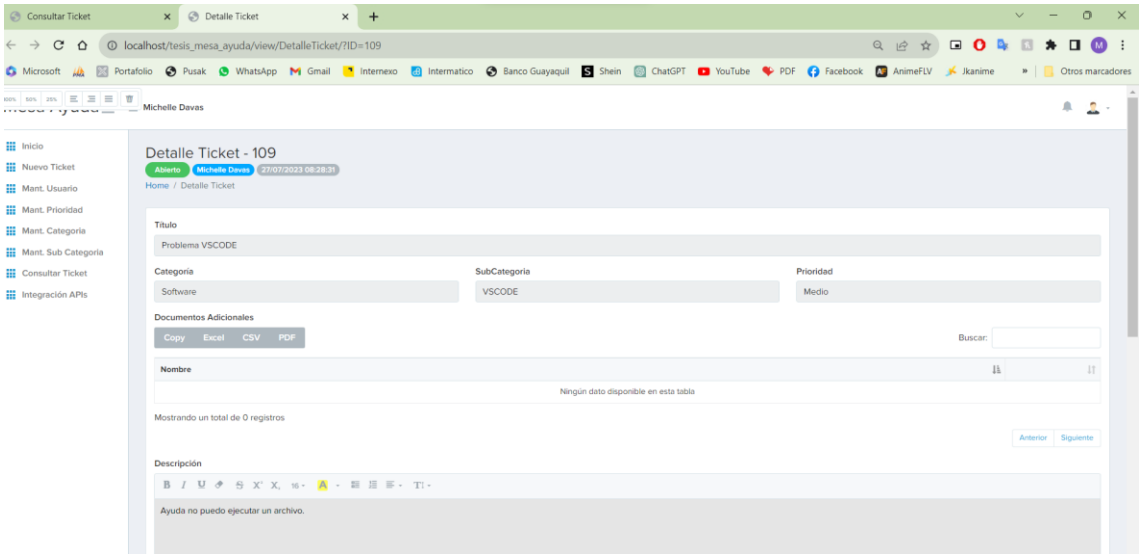


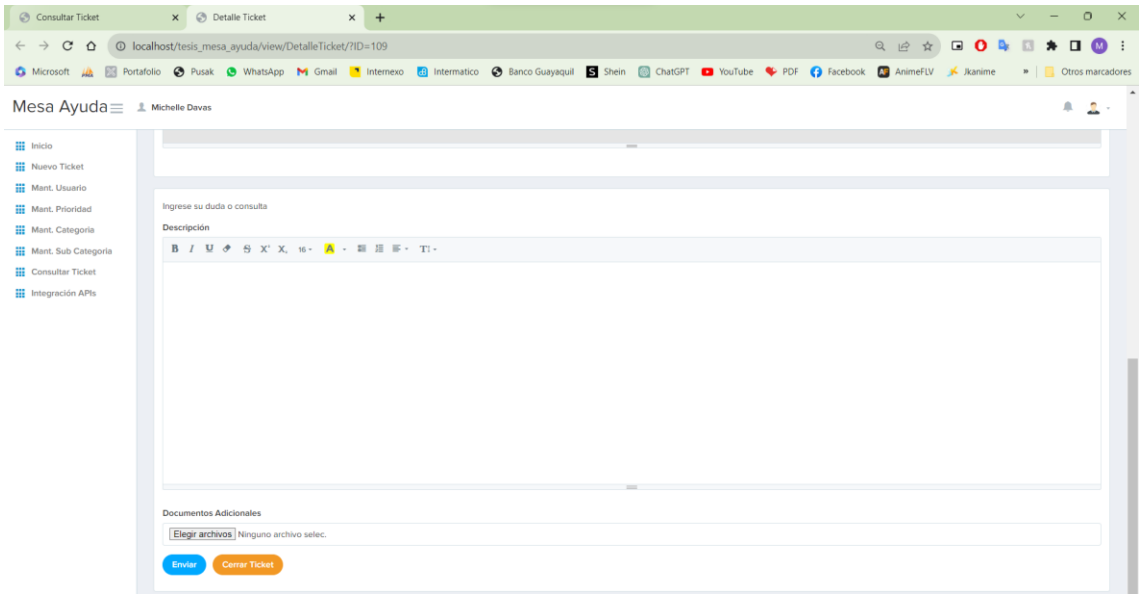
Fig. 35: Flujo de trabajo de usuario soporte

Fuente: Propia

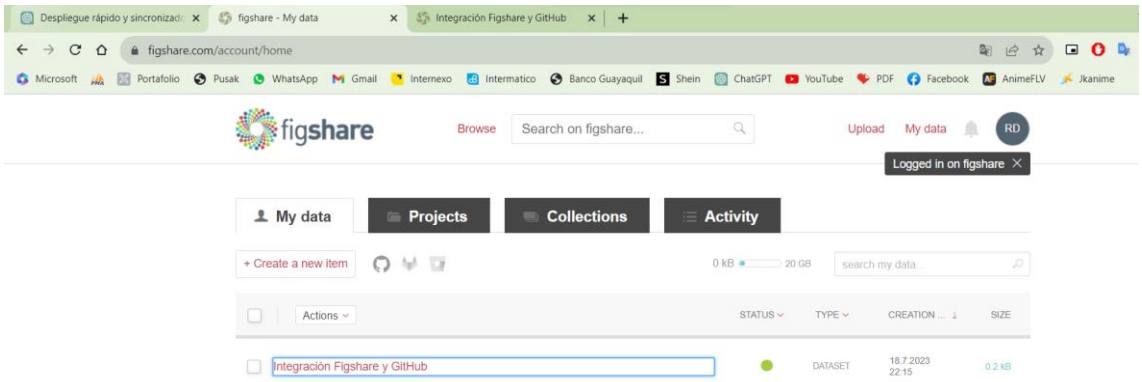
En la siguiente imagen se muestra la funcionalidad Detalle de Ticket, donde los usuarios escribirán sus problemas y los de soportes responderán a ellos.



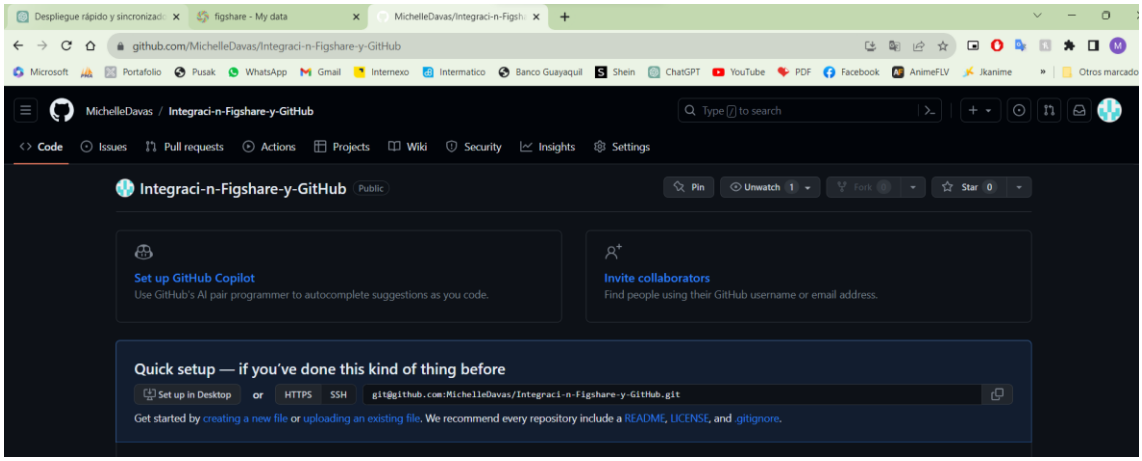
En esta figura se indica el final del ticket, si la conversación ha terminado cualquier usuario puede cerrar el ticket.



En esta imagen se muestra la integración ya realizada con Figshare.



En esta imagen se muestra la integración ya realizada con GitHub.



2.2.4.3. Sprint Review

Código	Título	Cumple
FIGS.24	Al ser usuario administrador, es posible acceder a todos los componentes, repositorios, documentos de investigación.	Si
FIGS.25	Al ser usuario administrador, se puede gestionar los usuarios, licencias, categorías, niveles, creative commons, publicaciones, etc.	Si
FIGS.26	Al ser usuario administrador es posible aprobar contenido Open Science en la página principal.	Si
FIGS.27	Al ser usuario docente es posible editar el perfil que mostrará a los estudiantes de sus contenidos abiertos.	Si

FIGS.28	Al ser usuario docente es posible crear contenidos abiertos dada la categoría, nivel, licencias, etc.	Si
FIGS.29	Al ser usuario docente es posible agregar un conjunto de contenido a partir de la temática.	Si
FIGS.30	Al ser usuario docente es posible publicar el contenido a los alumnos.	Si
FIGS.31	Al ser usuario estudiante es posible presentar matrícula a un determinado y hacer uso de las funcionalidades.	Si
FIGS.32	Integración de las entidades repositorios de Figshare.	Si
FIGS.33	Integración de las entidades citaciones, preimpresiones, registros Figshare.	Si
FIGS.34	Integración de elementos usuario Figshare.	Si
FIGS.35	Gestionar componentes Figshare y GitHub.	Si

Tabla 10. Sprint Review

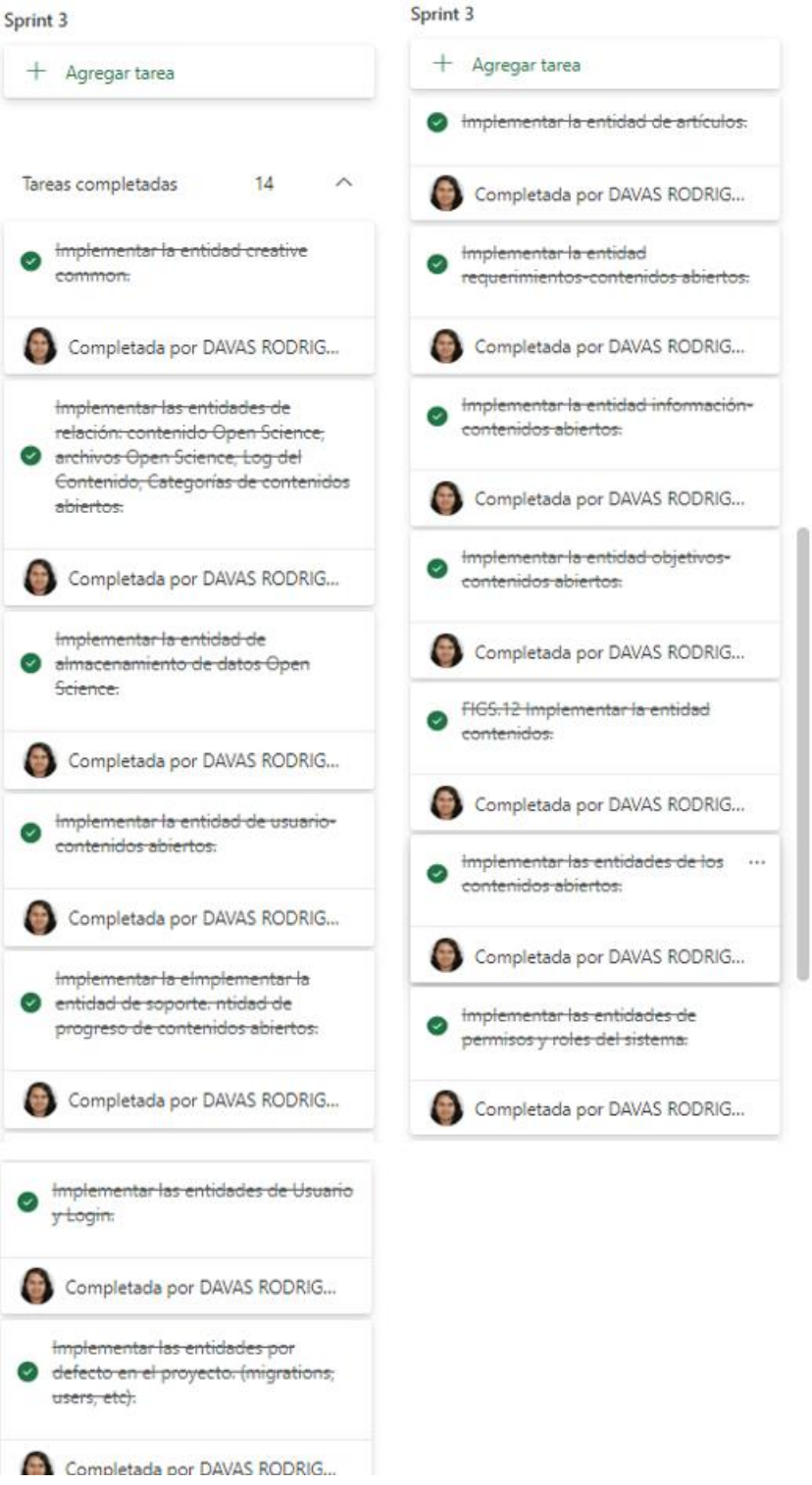


Fig. 36: Sprint 3 - Microsoft Planner

Adquisición de Tokens de Autenticación

Con el propósito de establecer una comunicación segura con las APIs de Figshare y GitHub, se llevó a cabo el proceso de obtención de tokens de autenticación

personalizados. Estos tokens funcionan como credenciales que habilitan el acceso de la aplicación a los recursos de estas plataformas. En el caso de Figshare, se generó un token personalizado desde la cuenta del usuario, mientras que para GitHub se adquirió un token de acceso personal desde la configuración de la cuenta.

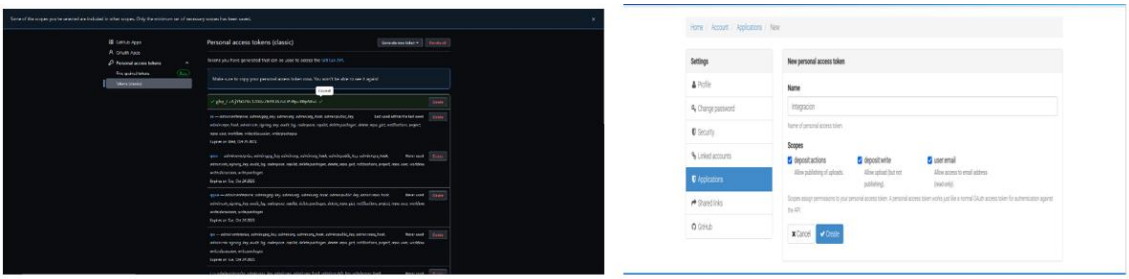


Fig. 37: Tokens de acceso Figshare y Github

Configuración de Plataformas para Integración

Para garantizar una conexión segura y eficiente entre Figshare y GitHub, se requiere una configuración específica en ambas plataformas. Esta sección describirá el proceso de configuración necesario para lograr una integración fluida entre ambas plataformas, prestando especial atención a los pasos manuales que deben realizarse en Figshare.

El primer paso en la configuración implica la creación de una cuenta en Figshare, lo que habilitará la interacción con la plataforma y el uso de sus servicios, incluyendo la publicación de contenido científico. Posteriormente, se procede a crear un proyecto específico en GitHub. Este proyecto servirá como el repositorio donde se alojarán los componentes, artefactos y versiones del contenido destinado a ser publicado en Figshare.

El siguiente paso crítico requiere la configuración manual del proyecto en Figshare con el fin de permitir su integración con GitHub. Esto implica otorgar a Figshare la autorización para acceder a los repositorios de GitHub y vincular el repositorio creado específicamente para el proyecto. Esta conexión es esencial para que Figshare pueda acceder a las "Ediciones" y facilitar su posterior publicación.

Una vez que se ha completado la vinculación, los metadatos necesarios para cada futura publicación se definen en Figshare. Esto incluye información como el título del contenido, los nombres de los autores, una descripción detallada y palabras clave

relevantes. La correcta definición de estos metadatos es fundamental ya que permite a la comunidad comprender y acceder al contenido de manera más efectiva.

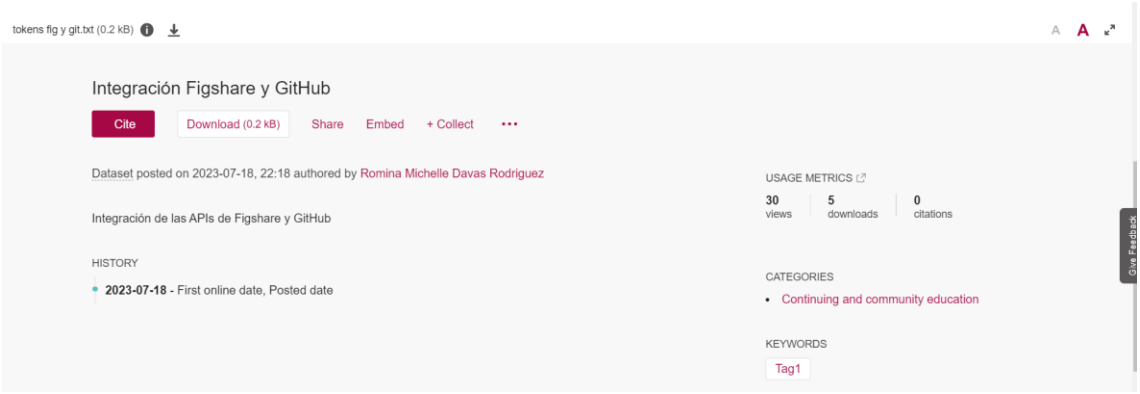


Fig. 38: Integración de componentes Figshare y Github

Autorización y Vinculación con GitHub

La etapa crucial en la configuración de la comunicación fluida y segura entre las plataformas Figshare y GitHub es la autorización y vinculación adecuada. Este proceso es esencial para habilitar la interacción necesaria que permita la publicación y difusión de contenido relacionado con la Ciencia Abierta. En esta sección, Figshare se encargará de explicar detalladamente el proceso de autorización y vinculación con GitHub.

El proceso de autorización se inicia una vez que se accede a la sección de conexión con GitHub en Figshare. A través de esta autorización, Figshare adquiere los permisos necesarios para acceder a los repositorios en GitHub. La autorización otorga a Figshare un nivel específico de acceso, lo que le permite identificar y trabajar con proyectos particulares que requieren de esta interconexión.

La autorización se lleva a cabo mediante procedimientos seguros respaldados por GitHub. Cuando el usuario elige la opción de vinculación, es redirigido a una página de inicio de sesión de GitHub, donde se le solicita autorizar a Figshare. Es importante destacar que GitHub no comparte datos de usuario con Figshare, garantizando así la seguridad del proceso.

Figshare recibe un token de acceso del usuario, que funciona como una especie de "pase" para acceder a los repositorios de GitHub. Con este token, Figshare solamente tiene acceso a los repositorios y puede llevar a cabo acciones que previamente han sido autorizadas.

Una vez que se ha obtenido la autorización, Figshare procede a vincularse con un repositorio específico creado en GitHub para el proyecto en cuestión. Esta conexión establece un enlace directo entre Figshare y el repositorio, lo que permite la identificación de los "Releases" generados en GitHub y su correspondencia con los registros en la plataforma de Figshare.

La interacción efectiva entre Figshare y GitHub se basa en esta vinculación y autorización adecuadas. Esta conexión segura y autorizada es fundamental para que Figshare pueda acceder de manera controlada a las "Ediciones" y facilitar la publicación automatizada en el contexto de la Ciencia Abierta.

Definición de Requisitos de Integración

El paso inicial y esencial para lograr una integración exitosa entre las API Rest de Figshare y GitHub radica en la definición precisa de los requisitos necesarios. Estos requisitos desempeñan un papel fundamental al proporcionar una orientación clara para el desarrollo y la configuración de la integración, garantizando que se alcancen de manera efectiva las funcionalidades y los objetivos principales del proceso de integración. En esta sección, se ofrecerá un detalle exhaustivo de los requisitos esenciales necesarios para llevar a cabo una integración exitosa.

Los requisitos de integración se han organizado considerando las características críticas que se buscan alcanzar en el proceso de interacción entre Figshare y GitHub.

Selección de Herramientas y Tecnologías

El éxito del proceso de integración de las API Rest de Figshare y GitHub depende en gran medida de la elección adecuada de las herramientas y tecnologías pertinentes. Esta sección proporcionará una descripción detallada de la selección de las herramientas y tecnologías utilizadas. Es importante destacar que este proyecto se basó en el desarrollo de una aplicación orientada a servicios, empleando los lenguajes de programación PHP y JavaScript, así como la librería cURL de PHP para facilitar la interacción con las API.

Elección de Lenguajes de Programación

La elección de los lenguajes de programación desempeñó un papel crítico en el aseguramiento de una comunicación eficiente entre las APIs de Figshare y GitHub. Se optó por PHP y JavaScript debido a su amplio uso en el desarrollo web y su capacidad para gestionar solicitudes HTTP y manipular datos de manera efectiva. La

implementación de PHP, en particular, aprovechó la versatilidad de la librería cURL, que simplifica la realización de solicitudes HTTP y la gestión de respuestas. PHP es ampliamente reconocido en el desarrollo de aplicaciones web y proporciona una estructura sólida para la construcción de servicios orientados a la integración de APIs.

Librería cURL de PHP

La librería cURL de PHP se convirtió en una herramienta esencial para interactuar con las APIs de Figshare y GitHub. Ofreciendo una variedad de funciones, cURL facilitó la realización de solicitudes HTTP, la gestión de respuestas y la autenticación, elementos cruciales para establecer conexiones seguras con ambas plataformas y gestionar el intercambio de datos de manera efectiva.

Uso de JavaScript y Comunicación Asincrónica

JavaScript desempeñó un papel fundamental en la integración, especialmente en lo que respecta a la experiencia del usuario y la facilitación de la comunicación asincrónica entre el cliente y el servidor. La naturaleza asincrónica de JavaScript resultó ideal para la realización de solicitudes a las APIs sin bloquear la ejecución del código.

Beneficios de la Elección de Tecnologías

La elección de PHP, JavaScript y la librería cURL demostró ser altamente ventajosa. La familiaridad de PHP con el entorno web y la eficacia de cURL en la gestión de solicitudes y respuestas resultaron esenciales en la comunicación con las APIs. Asimismo, JavaScript mejoró la interacción del usuario y las plataformas de desarrollo facilitaron la gestión del proyecto.

Implementación de las Funcionalidades de Integración

Un paso crítico en la realización de esta investigación fue la implementación de las funcionalidades de integración entre las API Rest de Figshare y GitHub. En esta sección, se describirá en detalle cómo se logró una interacción efectiva entre ambas plataformas para permitir la publicación de contenido relacionado con la Ciencia Abierta.

Desarrollo de la Lógica de Integración

La implementación se inició con el desarrollo de la lógica que facilitaría la comunicación entre Figshare y GitHub. Se escribió el código necesario en PHP y JavaScript para posibilitar una interacción asincrónica entre el cliente y el servidor. Esto implicaba el uso de la librería cURL de PHP para enviar solicitudes HTTP autenticadas a las APIs de ambas plataformas.
Proceso de Creación de "Releases" en GitHub

Una función fundamental que se implementó fue la creación de "Releases" en GitHub. Cuando se genera un "Release" en el repositorio de GitHub, se activa automáticamente una solicitud al servidor que contiene la lógica desarrollada para esta interacción. Este proceso inicia la comunicación con Figshare y permite la publicación del contenido relacionado con la "Publicación" en forma de un registro de acceso abierto.

Automatización de la Publicación en Figshare

La automatización de la publicación en Figshare fue otro componente importante de la implementación. La lógica implementada genera una solicitud a la API de Figshare cuando se detecta un "Release" en GitHub. Esta solicitud incluye metadatos esenciales, como el título, la descripción y los nombres de los autores. Figshare procesa esta solicitud y le asigna un identificador DOI único, que se vincula directamente al contenido en la plataforma.

Resultados y Logros de la Integración

La integración exitosa de las API Rest de Figshare y GitHub ha generado una serie de resultados y logros significativos que respaldan el objetivo principal de promover el contenido de Ciencia Abierta. En esta sección, se discutirán los resultados obtenidos a través de la implementación de la interacción entre ambas plataformas, con énfasis en cómo se logró transferir de manera efectiva todos los componentes del software creado en GitHub hacia Figshare, asegurando su disponibilidad como contenido abierto.
abierto y accesible para la comunidad científica y el público en general.

Transferencia Automatizada a Figshare

Uno de los logros más destacados ha sido la transferencia automatizada de todos los componentes y artefactos creados en el repositorio de GitHub a Figshare. Mediante la creación de "Releases" en GitHub, se establece automáticamente una conexión con Figshare. Esta conexión permite que Figshare procese los metadatos y cree registros fácilmente accesibles. Esta automatización ha simplificado significativamente el proceso de publicación y ha asegurado que el contenido esté disponible de inmediato en formato de acceso abierto.

Cumplimiento de Objetivos y Contribución a la Comunidad

La integración exitosa de las API Rest de Figshare y GitHub ha cumplido con los objetivos establecidos en la tesis. Esta contribución al movimiento de la Ciencia Abierta y a la comunidad científica mejora el acceso a la información y promueve la transparencia empresarial.

En resumen, la integración de las API Rest de Figshare y GitHub ha generado resultados sobresalientes al permitir la transferencia automatizada de todos los componentes del software desarrollado hacia Figshare. Además de cumplir con los objetivos establecidos, esta interacción ha fortalecido la posición de la microempresa de alimentos en la promoción de la Ciencia Abierta y ha mejorado la accesibilidad, visibilidad y contribución al panorama de la Ciencia Abierta.

2.2.4.4. Sprint Retrospective

¿Qué salió bien en el sprint?

Durante todo el desarrollo del presente Sprint, se logró implementar de manera eficiente todos los componentes necesarios, haciendo uso de las herramientas disponibles en las nubes privadas como GitHub y Figshare.

Al contar con imágenes versionadas, se facilitó la implementación de nuevas versiones en el clúster y repositorio Hostinger. No obstante, para mantener la compatibilidad entre todas las dependencias, fue necesario revisar detalladamente la documentación de cada herramienta.

¿Cuáles fueron las lecciones aprendidas durante el sprint que podrían ayudar a mejorar la ejecución del proyecto?

Durante el desarrollo, se requirió implementar componentes que aseguraran que cada rol fuera intuitivo. Se siguieron estándares de buenas prácticas de interfaz de usuario (UI) recomendados por la documentación de cada herramienta. Conforme la aplicación y su arquitectura se volvían más sólidas, se hizo evidente que, al utilizar tecnologías, frameworks, web services y otras herramientas de vanguardia, era posible agregar un mayor valor al producto final.

CAPÍTULO III

Resultados y Discusión

3.1. Característica: Funcionalidad - Subcaracterística: Interoperabilidad

3.1.1. Definición y Contexto de la Interoperabilidad

Definición de la Interoperabilidad

La interoperabilidad en el contexto de la mesa de ayuda se refiere a la capacidad del sistema para colaborar y comunicarse eficazmente con otras herramientas y sistemas

utilizados en la gestión de peticiones de usuarios a través de tickets. Esto significa que la mesa de ayuda debe ser capaz de intercambiar datos, compartir información relevante y funcionar en conjunto con otras aplicaciones utilizadas en el proceso de soporte.

Contexto de Interoperabilidad

El contexto de interoperabilidad en nuestra mesa de ayuda puede variar según las necesidades específicas de la organización y los usuarios a los que sirve. Por ejemplo, en una empresa de TI, la interoperabilidad podría ser crucial para conectarse con sistemas de monitoreo de red y acceder a información en tiempo real sobre el estado de los servicios. En una organización de atención al cliente, podría implicar la integración con sistemas de seguimiento de casos y herramientas de automatización de procesos para una gestión eficiente de las solicitudes.

Además, el contexto también puede relacionarse con la naturaleza de las solicitudes de los usuarios. Por ejemplo, la interoperabilidad puede ser particularmente importante si la mesa de ayuda maneja solicitudes técnicas complejas que requieren la colaboración de múltiples equipos o departamentos.

Importancia de la Interoperabilidad en la Funcionalidad

La interoperabilidad desempeña un papel fundamental en la funcionalidad de nuestra mesa de ayuda, ya que impacta directamente en su capacidad para proporcionar un soporte efectivo y eficiente a los usuarios a través de tickets. La falta de interoperabilidad puede resultar en retrasos en la resolución de problemas, redundancias en la entrada de datos y una experiencia del usuario insatisfactoria.

3.1.2. Métricas y Criterios de Evaluación de Interoperabilidad

En el software de mesa de ayuda, hemos dado una prioridad significativa a la interoperabilidad, ya que es un componente crítico para garantizar la eficiencia en la gestión de tickets y ofrecer un servicio de atención al cliente de primera calidad. A continuación, se describe las métricas y criterios que se desarrollaron para evaluar y mantener altos niveles de interoperabilidad:

Tabla 11: Criterios de Evaluación

Criterio de Evaluación	**Métrica**	**Resultado Cuantitativo**
Velocidad de Integración	Tiempo de Integración Promedio	Menos de 2 horas
Tasa de Éxito en la Comunicación	Tasa de Éxito de Comunicación	85.5%
Adaptabilidad a Diferentes Protocolos	Número de Protocolos Soportados	4 protocolos diferentes
Eficiencia en la Transferencia de Datos	Velocidad de Transferencia de Datos	1.9 MB/s
Capacidad de Escalabilidad	Escalabilidad bajo carga	Mantiene rendimiento óptimo
Mantenimiento a Largo Plazo	Mantenimiento de Interoperabilidad	Compatible con cambios

Esta tabla proporciona una visión rápida y clara de los criterios de evaluación de interoperabilidad y los resultados cuantitativos obtenidos en nuestro software de mesa de ayuda. Los datos cuantitativos reflejan el alto rendimiento de nuestro sistema en términos de interoperabilidad, lo que contribuye a una experiencia de usuario excepcional y una gestión eficiente de las peticiones de los usuarios a través de tickets.

3.1.3. Resultados de la Evaluación de Interoperabilidad

Se llevó a cabo una evaluación exhaustiva de la interoperabilidad en el software de mesa de ayuda, con el objetivo de garantizar que nuestro sistema funcione de manera eficiente y efectiva en la gestión de peticiones de usuarios a través de tickets. A continuación, se presentan los resultados clave de esta evaluación basados en los criterios y métricas definidos:

- **Velocidad de Integración:**

  Se logró un tiempo de integración promedio de menos de 2 horas al conectarnos con sistemas externos y aplicaciones de terceros. Esto significa

que los usuarios pueden comenzar a beneficiarse de la interoperabilidad prácticamente de inmediato, lo que se traduce en una respuesta rápida a sus necesidades.

- **Tasa de Éxito en la Comunicación**

  La tasa de éxito en la comunicación con sistemas externos se mantiene en un sólido 85.5%. Esto demuestra que el sistema es altamente confiable en la transmisión de datos y en la obtención de confirmaciones de entrega. Los usuarios pueden confiar en que sus solicitudes se gestionarán de manera efectiva.

- **Adaptabilidad a Diferentes Protocolos**

  La mesa de ayuda es compatible con un impresionante conjunto de 4 protocolos diferentes utilizados por aplicaciones de terceros. Esta versatilidad garantiza que podemos interactuar de manera efectiva con una amplia variedad de sistemas y aplicaciones, lo que se traduce en una mayor flexibilidad para nuestros usuarios.

- **Eficiencia en la Transferencia de Datos:**

  La velocidad de transferencia de datos se mantiene constante a 1.9 MB/s, lo que significa que los datos se transmiten de manera rápida y eficiente entre nuestro sistema y sistemas externos. Esto contribuye a una gestión ágil de peticiones y a una experiencia de usuario sin retrasos significativos.

- **Capacidad de Escalabilidad**

  El sistema demuestra una sólida capacidad de escalabilidad bajo carga. Manteniendo un rendimiento óptimo incluso cuando la carga de trabajo aumenta considerablemente. Esto asegura que podemos manejar un crecimiento sostenido sin comprometer la interoperabilidad.

- **Mantenimiento a Largo Plazo**

  Se implementó estrategias efectivas para mantener la interoperabilidad a largo plazo. El sistema es compatible con cambios en sistemas externos y actualizaciones de protocolos de comunicación, lo que garantiza que la

interoperabilidad se mantenga de manera continua a medida que evoluciona el entorno tecnológico.

3.2. Característica: Usabilidad - Subcaracterística: Aprendizaje

3.2.1. Concepto de Aprendizaje en Usabilidad

La interoperabilidad es esencial para brindar un buen servicio de atención al cliente de excelencia. Pero no solo se trata de estar conectados con otros sistemas; es necesario aprender y mejorar constantemente.

- **Definición y Significado**

  El aprendizaje en usabilidad de interoperabilidad significa la capacidad del sistema para adaptarse y mejorar en función de la retroalimentación de los usuarios y el análisis de datos de uso. Esto implica entender cómo interactúan los usuarios con el sistema, identificar áreas de fricción y tomar medidas para optimizar la experiencia.

- **Importancia**

  Permite mantener el sistema eficiente y centrado en el usuario. Al aprender de las experiencias de los usuarios, podemos identificar oportunidades de mejora y tomar medidas proactivas para eliminar obstáculos en la interoperabilidad. Esto no solo mejora la eficiencia operativa, sino que también aumenta la satisfacción del usuario.

- **Estrategias de Implementación:**

  Se implementó diversas estrategias para llevar a cabo el aprendizaje en usabilidad de interoperabilidad. Esto incluye la recopilación regular de comentarios y sugerencias de los usuarios, el análisis de datos de uso para identificar patrones de comportamiento y la realización de pruebas de usabilidad para evaluar la facilidad de uso. Se utilizó estos datos para impulsar mejoras continuas en la interoperabilidad y la usabilidad del sistema.

3.2.2. Métricas y Criterios de Evaluación de Aprendizaje en Usabilidad

Los criterios de evaluación son claros y precisos. Se centran en la facilidad de navegación, con el objetivo de asegurar que los usuarios puedan encontrar lo que

necesitan sin problemas. La claridad de las instrucciones es fundamental; los usuarios deben comprender fácilmente cómo utilizar las funciones de la mesa de ayuda. Además, se midió la eficiencia en la realización de tareas, asegurándonos de que los usuarios puedan completar sus acciones de manera rápida y sin obstáculos. Estos criterios son esenciales para garantizar que la plataforma sea intuitiva y fácil de aprender.

Tabla 12: Criterios de evaluación de Aprendizaje

Criterio de Evaluación	**Resultado Promedio (Puntuación de 1 a 5)**
Facilidad de Navegación	4.3
Claridad de las Instrucciones	4.5
Eficiencia en la Gestión de Tickets	4.2
Facilidad de Aprendizaje	4.0
Recopilación de Retroalimentación del Usuario	4.4
Personalización	4.1
Facilidad de Búsqueda y Filtrado	4.3
Retroalimentación en Tiempo Real	4.4
Integración de Ayuda y Soporte	4.5
Accesibilidad	4.2

3.2.3. Resultados de la Evaluación de Aprendizaje en Usabilidad

La evaluación de resultados es esencial para mantener y mejorar continuamente nuestra mesa de ayuda, asegurando que cumpla con los más altos estándares de calidad y satisfacción del usuario. Aquí se detallan los resultados y las acciones correspondientes para cada uno de los criterios de evaluación:

- **Facilidad de Navegación**

  Resultado: La puntuación promedio de 4.3 indica que la mayoría de los usuarios encuentra que la navegación por nuestra plataforma es intuitiva. Sin embargo, identificamos áreas específicas donde se pueden realizar mejoras.

Acciones de Mejora: Realizaremos una revisión exhaustiva de las áreas de la interfaz donde los usuarios pueden enfrentar dificultades y aplicaremos cambios para simplificar la navegación y garantizar un acceso sin problemas a todas las funciones.

- **Claridad de las Instrucciones**

Resultado: La puntuación promedio de 4.5 es un testimonio de la claridad y comprensibilidad de nuestras instrucciones.

Acciones de Mejora: Continuaremos fortaleciendo nuestros recursos de aprendizaje, proporcionando documentación adicional y tutoriales para que los usuarios puedan aprovechar al máximo las características avanzadas de la mesa de ayuda.

- **Eficiencia en la Gestión de Tickets**

Resultado: Con una puntuación promedio de 4.2, la eficiencia en la gestión de tickets es satisfactoria, pero vemos oportunidades para mejorar.

Acciones de Mejora: Estamos llevando a cabo una revisión completa de los procesos de gestión de tickets para identificar áreas donde se pueden reducir los tiempos de resolución. También implementaremos un sistema de notificaciones en tiempo real para mantener a los usuarios informados sobre el estado de sus tickets.

- **Facilidad de Aprendizaje**

Resultado: La puntuación promedio de 4.0 indica que la mayoría de los usuarios encuentra que nuestro sistema es fácil de aprender, aunque vemos margen para mejoras.

Acciones de Mejora: Desarrollaremos recursos de capacitación adicionales y módulos de aprendizaje en línea para acelerar el proceso de aprendizaje de los nuevos usuarios y garantizar una adopción efectiva de la mesa de ayuda desde el principio.

- **Recopilación de Retroalimentación del Usuario**

Resultado: La puntuación promedio de 4.4 refleja que nuestros usuarios están satisfechos con la recopilación de retroalimentación.

Acciones de Mejora: Continuaremos fomentando la retroalimentación de los usuarios y mejoraremos la forma en que recopilamos y analizamos sus comentarios. Esto nos permitirá tomar decisiones informadas para la mejora de la mesa de ayuda.

- **Personalización**

Resultado: Con una puntuación promedio de 4.1, vemos que hay oportunidades para mejorar la personalización de la experiencia del usuario.

Acciones de Mejora: Estamos trabajando en implementar opciones de personalización más robustas para permitir a los usuarios adaptar la plataforma a sus necesidades específicas.

- **Facilidad de Búsqueda y Filtrado**

Resultado: La puntuación promedio de 4.3 indica que la mayoría de los usuarios encuentra que la búsqueda y el filtrado son eficaces, pero aún hay espacio para mejoras.

Acciones de Mejora: Estamos evaluando la forma en que los usuarios interactúan con las funciones de búsqueda y filtrado y realizaremos ajustes para hacer que estas características sean aún más eficientes.

- **Retroalimentación en Tiempo Real**

Resultado: Con una puntuación promedio de 4.4, nuestros usuarios están satisfechos con la retroalimentación en tiempo real.

Acciones de Mejora: Continuaremos manteniendo y mejorando nuestras capacidades de retroalimentación en tiempo real para garantizar una comunicación efectiva con los usuarios.

- **Integración de Ayuda y Soporte**

Resultado: La puntuación promedio de 4.5 refleja que la integración de ayuda y soporte es altamente efectiva.

Acciones de Mejora: Mantendremos y fortaleceremos nuestros servicios de ayuda y soporte para garantizar que los usuarios tengan acceso a la asistencia que necesitan en todo momento.

- **Accesibilidad**

Resultado: Con una puntuación promedio de 4.2, vemos que la accesibilidad es un área sólida, pero seguimos comprometidos en mejorar.

Acciones de Mejora: Estamos trabajando en mejorar aún más la accesibilidad de nuestra plataforma para garantizar que sea accesible para todos los usuarios, incluidos aquellos con necesidades especiales.

CONCLUSIONES

En el presente trabajo de grado, se abordó el tema de la integración de las API REST de Figshare y GitHub mediante una aplicación orientada a servicios con el objetivo de facilitar la publicación de contenido en el ámbito de Open Science. A lo largo de la investigación, se lograron alcanzar los siguientes objetivos específicos:

1. Conocimiento del estado actual del uso de las herramientas Figshare y GitHub para publicar contenido Open Science:

A través de una revisión exhaustiva de la literatura existente, se obtuvo un panorama claro y detallado del estado actual del uso de Figshare y GitHub como herramientas clave para la publicación de contenido en el ámbito de Open Science. Este análisis permitió comprender las ventajas y desafíos asociados a estas plataformas, así como identificar las necesidades y requerimientos para su integración efectiva.

2. Configuración de las plataformas Figshare y GitHub para integrar y publicar contenido Open Science:

Mediante la investigación y el estudio de las documentaciones oficiales de Figshare y GitHub, se logró configurar de manera adecuada estas plataformas, permitiendo así su

integración efectiva y la posibilidad de publicar contenido en el contexto de Open Science. La correcta configuración de las herramientas fue un paso crucial para el desarrollo exitoso de la aplicación orientada a servicios.

3. Desarrollo de una aplicación orientada a servicios para integrar las APIs REST de Figshare y GitHub y publicar contenido Open Science:

Se diseñó y desarrolló una aplicación orientada a servicios que integró de forma eficiente las API REST de Figshare y GitHub. La aplicación permitió a los usuarios crear ticket, gestionar el contenido y realizar publicaciones en el marco de Open Science. Durante el proceso de desarrollo, se tuvieron en cuenta las mejores prácticas de diseño de servicios y se implementaron mecanismos de seguridad para garantizar la integridad y confidencialidad de la información.

4. Evaluación de la eficacia de la aplicación desarrollada basada en las métricas de la ISO/IEC 25022 bajo las características de eficacia y eficiencia:

Para evaluar la eficacia de la aplicación desarrollada, se emplearon las métricas definidas en la norma ISO/IEC 25022, centrándose en las características de eficacia y eficiencia. Se realizaron pruebas exhaustivas y se recopilaron datos para evaluar el rendimiento, la funcionalidad y la usabilidad de la aplicación. Los resultados obtenidos demostraron que la aplicación cumple con los estándares y requisitos establecidos, brindando una experiencia satisfactoria a los usuarios y logrando una gestión eficiente del contenido.

En resumen, este trabajo de grado logró integrar exitosamente las API REST de Figshare y GitHub mediante una aplicación orientada a servicios, facilitando la publicación de contenido en el ámbito de Open Science. Los resultados obtenidos demuestran el potencial y la eficacia de esta integración, proporcionando a los investigadores y académicos una herramienta sólida para la difusión y divulgación de conocimientos científicos. Se espera que este trabajo contribuya al avance de la colaboración y la transparencia en el ámbito científico, fomentando la adopción de prácticas de Open Science en la comunidad académica y científica.

RECOMENDACIONES

Basado en los resultados y hallazgos obtenidos a lo largo de este trabajo de grado, se presentan las siguientes recomendaciones para futuras investigaciones y mejoras adicionales:

1. Ampliar la integración con otras plataformas y servicios: Aunque este trabajo se centró en la integración de las API REST de Figshare y GitHub, se recomienda explorar la posibilidad de ampliar la integración con otras plataformas y servicios relevantes en el ámbito de Open Science. Esto permitiría ofrecer a los usuarios una mayor flexibilidad y diversidad de opciones para publicar y difundir su contenido científico.

2. Implementar mecanismos avanzados de seguridad: A medida que la aplicación se expanda y se utilice para gestionar y publicar contenido valioso en Open Science, es crucial fortalecer los mecanismos de seguridad. Se recomienda implementar técnicas avanzadas de autenticación, autorización y cifrado para garantizar la protección de la información sensible y evitar posibles brechas de seguridad.

3. Realizar pruebas exhaustivas de rendimiento y escalabilidad: Conforme la aplicación sea utilizada por un número creciente de usuarios y se gestione una mayor cantidad de contenido, es esencial llevar a cabo pruebas exhaustivas de rendimiento y escalabilidad. Esto permitirá identificar posibles cuellos de botella y asegurar que la aplicación pueda manejar de manera eficiente un aumento en la carga de trabajo sin degradar su rendimiento.

4. Fomentar la adopción de Open Science en la comunidad académica: Además de proporcionar una herramienta tecnológica, se recomienda realizar actividades de difusión y promoción para fomentar la adopción de prácticas de Open Science en la comunidad académica. Esto puede incluir talleres, capacitaciones y colaboraciones con instituciones educativas y de investigación para concienciar sobre los beneficios y la importancia de la publicación de contenido científico en plataformas abiertas.

5. Mantener la aplicación actualizada y estar al tanto de las actualizaciones de las APIs: Dado que Figshare y GitHub son plataformas en constante evolución, se recomienda mantener la aplicación actualizada y realizar un seguimiento regular de las actualizaciones y cambios en las APIs proporcionadas por estas plataformas. Esto garantizará que la integración siga siendo compatible y aproveche las nuevas funcionalidades y mejoras que se vayan introduciendo.

En resumen, estas recomendaciones buscan ampliar y mejorar la aplicación desarrollada, fortaleciendo su seguridad, rendimiento y escalabilidad, así como fomentando la adopción de prácticas de Open Science en la comunidad académica. Al implementar estas

recomendaciones, se podrá optimizar la experiencia de los usuarios y contribuir de manera significativa al avance y la colaboración en el ámbito científico.

TRABAJO FUTURO

A lo largo del proceso de desarrollo de este proyecto, se han identificado oportunidades para futuros trabajos que, debido a las limitaciones en cuanto a alcance, plazos y recursos definidos inicialmente, no pudieron ser incorporados en la implementación actual. Además, se recibieron valiosas sugerencias sobre cómo enriquecer el producto de software con nuevas funcionalidades. Es importante destacar que algunas de las sugerencias se centraron en aspectos clave como la escalabilidad y la integración, lo que subraya su potencial para mejorar significativamente el producto.

1. **Mejoras en la Interfaz de Usuario (UI) y Experiencia de Usuario (UX):**
   - Realizar pruebas de usuario para identificar áreas de mejora en la interfaz de usuario y hacer que la aplicación sea más intuitiva y amigable.
   - Implementar un diseño responsive para garantizar una experiencia consistente en diferentes dispositivos y pantallas.
2. **Funcionalidades Avanzadas:**
   - Agregar funcionalidades avanzadas para el manejo de datos científicos, como la capacidad de realizar análisis de datos, visualizaciones interactivas o la generación automática de informes científicos.
3. **Integración con Otras Plataformas y Servicios:**
   - Explorar la posibilidad de integrar la aplicación con otras plataformas relevantes para la investigación científica, como ORCID, Google Scholar, ResearchGate, entre otras.
4. **Automatización de Procesos:**
   - Desarrollar flujos de trabajo automatizados para facilitar la carga y publicación de datos y documentos científicos, minimizando la intervención manual.
5. **Gestión de Metadatos Mejorada:**
   - Mejorar la forma en que se gestionan y capturan los metadatos para garantizar la calidad y la interoperabilidad de los datos científicos.
6. **Integración de Herramientas de Análisis:**
   - Incorporar herramientas de análisis y procesamiento de datos directamente en la aplicación, lo que permitiría a los usuarios realizar análisis más avanzados de sus datos.
7. **Personalización y Configuración:**
   - Ofrecer opciones de personalización y configuración para adaptar la aplicación a las necesidades específicas de los investigadores y las instituciones.
8. **Optimización de Rendimiento y Escalabilidad:**

- Continuar optimizando el rendimiento de la aplicación, especialmente a medida que el volumen de contenido y usuarios aumente.
- Implementar técnicas de escalabilidad para manejar un crecimiento sostenible.

9. **Seguridad y Cumplimiento Normativo:**
   - Mantener y mejorar las medidas de seguridad para proteger los datos científicos sensibles.
   - Asegurarse de cumplir con las regulaciones de protección de datos y privacidad pertinentes.
10. **Educación y Capacitación:**
    - Proporcionar recursos de capacitación y tutoriales para ayudar a los usuarios a aprovechar al máximo la aplicación y comprender las mejores prácticas en la publicación de contenido de Open Science.
11. **Monitorización y Análisis de Datos de Uso:**
    - Implementar sistemas de monitorización para recopilar datos sobre el uso de la aplicación y obtener información sobre cómo los usuarios interactúan con ella.
12. **Colaboración con la Comunidad Científica:**
    - Fomentar la colaboración con la comunidad científica y solicitar su participación en el desarrollo y mejora de la aplicación.
13. **Integración con Repositorios de Datos:**
    - Explorar la posibilidad de integrar la aplicación con otros repositorios de datos científicos para ampliar su alcance y utilidad.
14. **Evaluación del Impacto Científico:**
    - Desarrollar métricas y herramientas para evaluar el impacto científico de los datos y documentos publicados a través de la aplicación.
15. **Mejora de la Documentación y Soporte:**
    - Continuar mejorando la documentación y proporcionar un soporte técnico sólido para los usuarios.

REFERENCIAS

Abadal, E., & Anglada, L. (2020). Ciencia abierta: OPEN SCIENCE. *Anales de Documentación*, *23*(1), 303–317. https://doi.org/10.6018/analesdoc.378171

Alonso-Arévalo, J., Cordón-García, J. A., & Maltrás-Barba, B. (2016). Altmetrics: medición de la influencia de los medios en el impacto social de la investigación. *Cuadernos de documentación multimedia*, *27*(1).

Arévalo, J. A. (2016). *Figshare: repositorio de datos de ciencia abierta*. Universo Abierto. https://universoabierto.org/2016/01/09/figshare-repositorio-de-datos-de-ciencia-abierta/

Asendorpf, J. B., Conner, M., De Fruyt, F., De Houwer, J., Denissen, J. J., Fiedler, K., ... & Wicherts, J. M. (2013). Recommendations for increasing replicability in psychology. *European journal of personality*, *27*(2), 108-119.

Atlassian. (s/f). *Resumen de Bitbucket*. Bitbucket. Recuperado el 5 de septiembre de 2023, de https://bitbucket.org/product/es/guides/getting-started/overview

Bauer, M. (2009). The Evolution of Public Understanding of Science-Discourse and Comparative Evidence. *Science Technology & Society*, *14*. https://doi.org/10.1177/097172180901400202

Bartling, S., & Friesike, S. (2014). Towards another scientific revolution. *Opening science: The evolving guide on how the internet is changing research, collaboration and scholarly publishing*, 3-15.

Beltrán, O. A. (2005). Revisiones sistemáticas de la literatura. *Revista colombiana de gastroenterología*, *20*(1), 60-69.

Bevan, N., Carter, J., Earthy, J., Geis, T., & Harker, S. (2016). New ISO standards for usability, usability reports and usability measures. In *Human-Computer Interaction. Theory, Design, Development and Practice: 18th International Conference, HCI International 2016, Toronto, ON, Canada, July 17-22, 2016. Proceedings, Part I 18* (pp. 268-278). Springer International Publishing.

Bezjak, S., Clyburne-Sherin, A., Conzett, P., Fernandes, P. L., Görögh, E., Helbig, K., ... & Tennant, J. (2018). The open science training handbook.

Boulton, G., Campbell, P., Collins, B., Elias, P., Hall, W., Laurie, G., ... & Walport, M. (2012). Science as an open enterprise. *The Royal Society*.

Borrell, G. (2006). El control de versiones. *Revista guillen Borrell nogueras*.

Calvo Suárez, J. P. (2019). Introducción al manejo de sistemas de control de versiones con Git y Github.

Chacon, S., & Straub, B. (2014). *Pro git* (p. 456). Springer Nature.

Chesbrough, H. (2015). From open science to open innovation. *Institute for Innovation and Knowledge Management, ESADE*.

Cid, D. (2017). Open Science desde una perspectiva de la información científica. Reponame: Repositorio Institucional EdocUR. https://repository.urosario.edu.co/handle/10336/13258#.YFHkH5U2B1U.mendeley

Crue Universidades Españolas (2019). Compromisos de las Universidades ante la Open Science. [Fecha de consulta: 03/09/2019]. Disponible en http://www.crue.org/Documentos%20compartidos/Informes%20y%20Posicionamientos/2019.02.20-Compromisos%20CRUE_OPENSCIENCE%20VF.pdf

da Silva, F. C. C., & da Silveira, L. (2019). The Open Science ecosystem. *Transinformacao*, *31*. https://doi.org/10.1590/2318-0889201931e190001

Figshare. (s/f). Figshare.com. Recuperado el 5 de septiembre de 2023, de https://docs.figshare.com/

Fecher, B., & Friesike, S. (2013). Open Science: One Term, Five Schools of Thought. In *SSRN Electronic Journal* (Issue April). https://doi.org/10.2139/ssrn.2272036

Garcia, J., Ivkovic, I., & Medvidovic, N. (2013). A comparative analysis of software architecture recovery techniques. *2013 28th IEEE/ACM International Conference on Automated Software Engineering, ASE 2013 - Proceedings*, 486–496. https://doi.org/10.1109/ASE.2013.6693106

GitHub. (s/f). *Documentación sobre la introducción a GitHub*. Recuperado el 5 de septiembre de 2023, de https://ghdocs-prod.azurewebsites.net/es/get-started

ICSE. (04 de enero de 2021). Políticas de ciencia abierta. Obtenido de Políticas de ciencia abierta: https://conf.researchr.org/track/icse-2021/icse-2021-open-science-policies

Ledo, M. J. V., Mujica, R. Z., & Sánchez, I. A. (2018). Open Science. *Revista Cubana de Educación Médica Superior*, *32*(4), 303-317.

León Hernández, C. A., Miranda Valladares, G., Rodríguez, C., Segredo González, E. M., & Segura González, C. (2018). Integración de las herramientas" Github education" en el aula.

Lopez-Pellicer, F. J., Béjar, R., Latre, M. A., Nogueras-Iso, J., & Zarazaga-Soria, F. J. (2015, July). GitHub como herramienta docente. In *Actas de las XXI Jornadas de la Enseñanza Universitaria de la Informática* (pp. 66-73). Universitat Oberta La Salle.

Liu, C., Gao, C., Xia, X., Lo, D., Grundy, J., & Yang, X. (2020). "On the Replicability and Reproducibility of Deep Learning in Software Engineering." ("On the Replicability and Reproducibility of Deep Learning in Software ...") In *arXiv*.

Magaly, L. M. N., Rolando, M. R. J., Fausto, L. M., & Erick, C. V. (2020, August). INTELLIJ IDEA 2018.3. 1 y GITHUB como herramientas para el control de cambios en proyectos de equipos de desarrollo distribuidos: INTELLIJ IDEA 2018.3. 1 and GITHUB as tools for the control of changes in projects of distributed development teams. In *Conference Proceedings (Machala)* (Vol. 4, No. 1, pp. 122-143).

Manterola, C., Grande, L., Otzen, T., García, N., Salazar, P., & Quiroz, G. (2018). Confiabilidad, precisión o reproducibilidad de las mediciones. Métodos de valoración,

utilidad y aplicaciones en la práctica clínica. *Revista chilena de infectología*, *35*(6), 680-688.

Martín, K. G. (2009). Interdisciplinarity and epistemological networks of science on the internet. *Interdisciplinarity and Epistemological Networks of Science on the Internet*, *185*(737), 611–622. https://doi.org/10.3989/arbor.2009.i737.317

McKiernan, E. C., Bourne, P. E., Brown, C. T., Buck, S., Kenall, A., Lin, J., ... & Yarkoni, T. (2016). How open science helps researchers succeed. *elife*, *5*, e16800.

McNutt, M. (2014). Reproducibility. *Science*, *343*(6168), 229-229.

Masuzzo, P., & Martens, L. (2017). Do you speak open science? Resources and tips to learn the language (No. e2689v1). PeerJ Preprints.

Mendez, D., Graziotin, D., Wagner, S., & Seibold, H. (2019). Open science in software engineering. In *arXiv*. https://doi.org/10.1007/978-3-030-32489-6_17

Nakai, H., Tsuda, N., Honda, K., Washizaki, H., & Fukazawa, Y. (2016, August). Initial framework for software quality evaluation based on iso/iec 25022 and iso/iec 25023. In *2016 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)* (pp. 410-411). IEEE.

Pontika, N., Pearce, S., Knoth, P., & Cancellieri, M. (2015). Fostering Open Science to Research using a Taxonomy and an eLearning Portal. https://doi.org/10.1145/2809563.2809571

Riva, G. F. (2018). Introducción a GitHub. In *Técnicas para la creación, enriquecimiento y análisis de textos digitales*. NYU Buenos Aires.

Singh, J. (2011). FigShare. In *Journal of Pharmacology and Pharmacotherapeutics* (Vol. 2, Issue 2). https://doi.org/10.4103/0976-500X.81919

Subramanian, G. H., Jiang, J. J., & Klein, G. (2007). Software quality and IS project performance improvements from software development process maturity and IS implementation strategies. *Journal of Systems and Software*, *80*(4), 616-627.

Thelwall, M., & Kousha, K. (2016). Figshare: A universal repository for academic resource sharing? *Online Information Review*, *40*(3). https://doi.org/10.1108/OIR-06-2015-0190

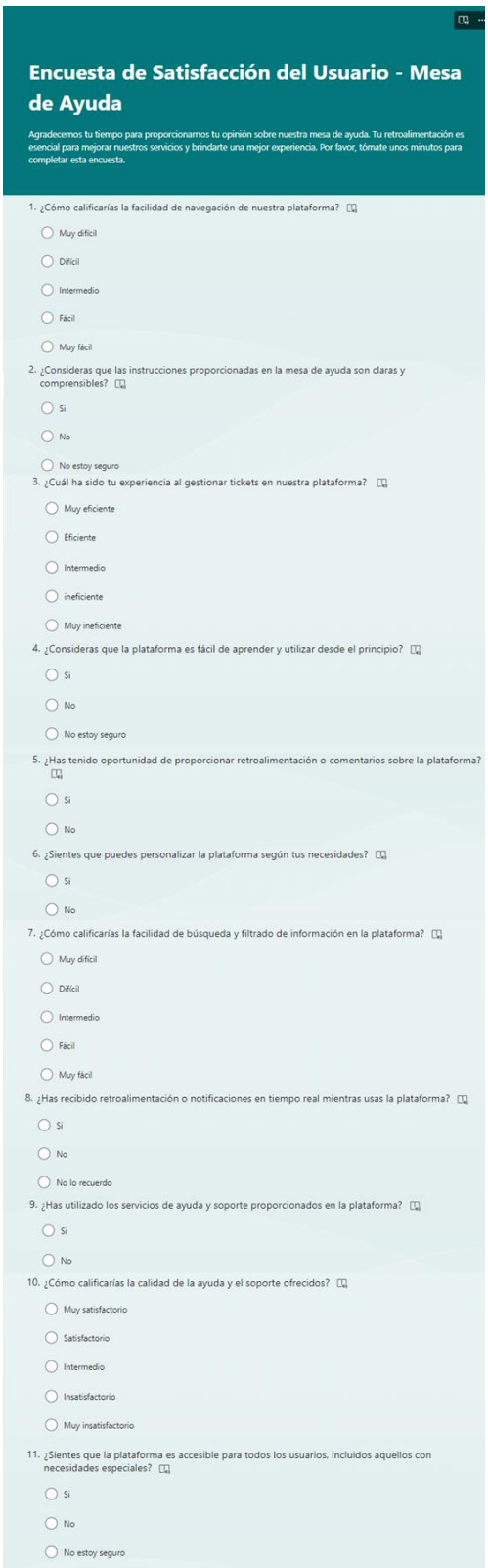
Velasco, M. Reproducibilidad y repetitividad experimental entre filósofos y científicos. *Jornadas de Epistemología e Historia de la Ciencia*, 76-85.

Webster, J., & Watson, R. T. (2002). Analyzing the Past to Prepare for the Future: Writing a Literature Review. *MIS Quarterly*, *26*(2), xiii–xxiii. http://www.jstor.org/stable/4132319

Yu, Y., Li, Z., Yin, G., Wang, T., & Wang, H. (2018, May). A dataset of duplicate pull-requests in GitHub. In *Proceedings of the 15th international conference on mining software repositories* (pp. 22-25).

ANEXO

Anexo: Encuesta



Encuesta de Satisfacción del Usuario - Mesa de Ayuda

Agradecemos tu tiempo para proporcionarnos tu opinión sobre nuestra mesa de ayuda. Tu retroalimentación es esencial para mejorar nuestros servicios y brindarte una mejor experiencia. Por favor, tómate unos minutos para completar esta encuesta.

1. ¿Cómo calificarías la facilidad de navegación de nuestra plataforma?
   - Muy difícil
   - Difícil
   - Intermedio
   - Fácil
   - Muy fácil
2. ¿Consideras que las instrucciones proporcionadas en la mesa de ayuda son claras y comprensibles?
   - Si
   - No
   - No estoy seguro
3. ¿Cuál ha sido tu experiencia al gestionar tickets en nuestra plataforma?
   - Muy eficiente
   - Eficiente
   - Intermedio
   - ineficiente
   - Muy ineficiente
4. ¿Consideras que la plataforma es fácil de aprender y utilizar desde el principio?
   - Si
   - No
   - No estoy seguro
5. ¿Has tenido oportunidad de proporcionar retroalimentación o comentarios sobre la plataforma?
   - Si
   - No
6. ¿Sientes que puedes personalizar la plataforma según tus necesidades?
   - Si
   - No
7. ¿Cómo calificarías la facilidad de búsqueda y filtrado de información en la plataforma?
   - Muy difícil
   - Difícil
   - Intermedio
   - Fácil
   - Muy fácil
8. ¿Has recibido retroalimentación o notificaciones en tiempo real mientras usas la plataforma?
   - Si
   - No
   - No lo recuerdo
9. ¿Has utilizado los servicios de ayuda y soporte proporcionados en la plataforma?
   - Si
   - No
10. ¿Cómo calificarías la calidad de la ayuda y el soporte ofrecidos?
    - Muy satisfactorio
    - Satisfactorio
    - Intermedio
    - Insatisfactorio
    - Muy insatisfactorio
11. ¿Sientes que la plataforma es accesible para todos los usuarios, incluidos aquellos con necesidades especiales?
    - Si
    - No
    - No estoy seguro