UNIVERSIDAD TÉCNICA DEL NORTE

Facultad de Ingeniería en Ciencias Aplicadas

Carrera de Ingeniería en Sistemas Computacionales

Integración de las APIs REST de OSF 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:

Santiago Andres Moreta Chuqui

Directora:

MSc. Cathy Pamela Guevara Vega

Ibarra – Ecuador

2021

AUTORIZACIÓN DE USO Y PUBLICACIÓN

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 presente trabajo 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:	1004280135		
APELLIDOS Y NOMBRES:	MORETA CHUQUI SANTIAGO ANDRES		
DIRECCIÓN:	Quito - Ecuador		
EMAIL:	andresm98@protonmail.com		
TELÉFONO FIJO:		**TELÉFONO MÓVIL:**	0967316479

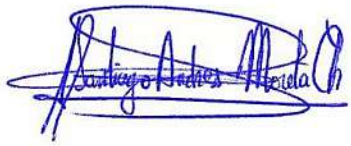
DATOS DE LA OBRA	
TÍTULO:	Integración de las APIs REST de OSF y GitHub mediante una Aplicación Orientada a Servicios para publicar contenido Open Science.
AUTOR (ES):	MORETA CHUQUI SANTIAGO ANDRES
FECHA: DD/MM/AAAA	22/11/2021
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA:	■ **PREGRADO** ☐ **POSGRADO**
TÍTULO POR EL QUE OPTA:	INGENIERO EN SISTEMAS COMPUTACIONALES
ASESOR /DIRECTOR:	MSc. Cathy Pamela Guevara Vega

CONSTANCIAS

El autor (es) manifiesta (n) 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 (son) el (los) titular (es) de los derechos patrimoniales, por lo que asume (n) la responsabilidad sobre el contenido de la misma y saldrá (n) en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 22 días del mes de noviembre de 2021.

EL AUTOR:



Santiago Andres Moreta Chuqui

CC: 1004280135

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 Santiago Andres Moreta Chuqui, portador de la cédula de ciudadanía 100428013-5. Certifico que ha trabajado en el desarrollo del presente proyecto de tesis denominado: "**Integración de las APIs REST de OSF 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:

CATHY PAMELA GUEVARA VEGA

Firmado digitalmente por CATHY PAMELA GUEVARA VEGA
Fecha: 2021.11.22 13:13:38 -05'00'

MSc. Cathy Guevara

DIRECTORA DE TRABAJO DE GRADO

DEDICATORIA

Para mis padres, especialmente a mi madre. Gracias por darme una voz de aliento cuando tantas veces ahí sentado en aquel letargo anímico de mis pensamientos creí todo perdido.

AGRADECIMIENTO

A mis padres por el apoyo que me han sabido brindar, ya que han sido los únicos y fieles testigos de todos aquellos contratiempos que he tenido que soslayar hasta del día de hoy.

A las personas que confiaron en mi la labor de llevar a cabo una determinada tarea ya que de no haber sido así me habría sido imposible aprender a pescar por mí mismo desde chico. De igual modo a las contadas personas que compartieron incondicionalmente conmigo un libro, trozo de código, canción, consejo, manual técnico u otro recuerdo que seguiré guardando gratamente.

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

En los últimos años se ha hecho evidente la necesidad de adoptar las políticas de movimientos sin fines de lucro que buscan lograr que los componentes de una determinada investigación científica no sean centralizados. Por lo general los grandes conglomerados invierten enormes recursos para desarrollar nuevas técnicas, tecnologías, metodologías u otro elemento, sin embargo, pocas personas tienen acceso a dicha información.

La Unión Europea durante los últimos años le ha dado gran importancia al tema e inclusive va más allá, pues en palabras de sus representantes el hecho de que el conocimiento científico sea centralizado genera desigualdades muy graves y es aquello que impide a los países en vías de desarrollo integrarse a nuevos modelos de generar conocimiento; es entonces cuando la Unión Europea apuesta por un movimiento llamado Open Science que permite mitigar los problemas subyacentes de aquella centralización.

El presente trabajo consta de tres capítulos y busca elaborar una arquitectura de software que permita integrar los servicios de dos herramientas relacionadas a los términos Open Source y Open Science. En dicha arquitectura se pretende aplicar técnicas DevOps y tecnologías modernas que permitan acceder a elementos de investigaciones científicas que se encuentran en colecciones de Open Science Framework y en repositorios de código en GitHub.

Previo al desarrollo de la arquitectura se llevó a cabo una revisión sistemática de literatura SLR para conocer la información relevante acerca de métodos de publicación de contenido abierto y las herramientas que pueden ayudar a lograr tal fin. En el desarrollo de los componentes de la aplicación se usaron varias tecnologías que ayudaron a que la arquitectura sea eficiente.

Finalmente se realizó el análisis del producto de software a ser evaluado, mediante los requisitos según el estándar IEEE 830 y las métricas de la calidad en uso según la ISO/IEC 25022. También se monitorearon los componentes y servicios de la arquitectura para verificar su correcto funcionamiento, entre los elementos que fueron evaluados están: Clúster AWS, Contenedor Docker, Aplicación Nativa, Base de Datos MySQL y otros componentes de la arquitectura final.

Palabras Claves: Open Science, Open Source, DevOps, Revisión Sistemática de Literatura, IEEE 830, ISO/IEC 25022, Integración de Servicios, Amazon Web Services.

ABSTRACT

In recent years it has become evident that there is a need to adopt the policies of non-profit movements that seek to ensure that the components of a given scientific research are not centralized. In general, large conglomerates invest enormous resources to develop new techniques, technologies, methodologies or other elements, but few people have access to such information.

The European Union in recent years has given great importance to the issue and even goes further, because in the words of its representatives, the fact that scientific knowledge is centralized generates very serious inequalities and prevents developing countries from integrating new models of generating knowledge; it is then when the European Union bets on a movement called Open Science that allows mitigating the underlying problems of that centralization.

This work consists of three chapters and seeks to elaborate a software architecture that allows the service integration of two tools related to the terms Open Source and Open Science. In this architecture it aims to apply DevOps techniques and modern technologies that allow access to scientific research elements found in the Open Science Framework collections and GitHub code repositories.

Prior to the development of the architecture, a Systematic Literature Review SLR was carried out to learn the most relevant information about open content publishing methods and the tools that can help achieve this goal. In the development of the application components, several technologies were used to make the resulting architecture efficient.

Finally, the analysis of the software product to be evaluated was performed, using the requirements according to the IEEE 830 standard and the quality in use metrics according to ISO/IEC 25022. The components and services of the architecture were also monitored to verify their correct operation, among the elements that were evaluated are: AWS Cluster, Docker Container, Native App, MySQL Database and other components of the final architecture.

Keywords: Open Science, Open Source, DevOps, Systematic Literature Review, IEEE 830, ISO/IEC 25022, Service Integration, Amazon Web Services.

INTRODUCCIÓN

Antecedentes

El conocimiento científico a lo largo de los años se ha logrado obtener a partir de métodos científicos, técnicas, estudios que han necesitado grandes contingentes de inversión; las naciones y las organizaciones tanto del sector público y privado dirigen grandes recursos para lograr los mejores progresos que permitan fomentar el avance tecnológico; pero el acceso a dicha información se ha vuelto restringido lo que impide que estudiantes, docentes, investigadores universitarios y público en general puedan tener acceso a conocimientos relacionados a investigaciones de gran impacto científico. El fundamento está sustentado en que la ciencia sea compartida desde el punto clásico del comunismo, entendiendo que el conocimiento debe ser impartido no desde el punto de vista político sino como la propiedad común de los bienes (Merton, 1968).

Open Science es un movimiento que pretende cambiar dicho paradigma, haciendo que el acceso al conocimiento científico logre ser abierto durante todas las etapas de la investigación científica: diseño, recolección de data, revisión, publicación, refinamiento, etc.; mediante el uso de herramientas como OSF “Open Science Framework” que incentivan el objetivo de lograr el conocimiento abierto en la Ingeniería de Software (IS).

COS es una organización tecnológica sin fines de lucro, la cual tiene como objetivo principal: “aumentar la apertura, la integridad y la reproducibilidad de la investigación científica”, dicha organización tiene a disposición herramientas como OSF misma que permite realizar el seguimiento en proyectos independientemente de la rama de estudio para conocer los resultados obtenidos en un determinado producto de investigación.

El cambio surge también para lograr implementar y dar a conocer nuevos conocimientos relacionados a proyectos de software que logren evadir las barreras de los recursos económicos, contingente experimental, material humano e infraestructura; dando así la oportunidad a conocer cómo se realiza un determinado método científico para lograr replicarlo y mejorarlo (David, 2008). Por lo tanto, no significa que se van a quebrantar los métodos científicos o tecnológicos; sin embargo, la ciencia abierta, tiene el carácter cultural y social de un nuevo modelo de hacer y entender la investigación científica.

Situación Actual

Actualmente la dificultad que surge al requerir acceso al conocimiento en el área de Ingeniería de Software se genera en elementos tales como artefactos tecnológicos, información y documentación necesaria que son de crucial importancia al momento de realizar el uso o

cambio de técnicas en un determinado software; lo que genera conflictos como el lograr que un determinado proyecto logre ser transmitido a la comunidad de manera continua a medida que va incrementando su volumen de código fuente.

Open Science a lo largo de los últimos años ha logrado tener grandes impactos de manera inmediata en varias áreas del conocimiento, especialmente en áreas investigativas ya que fue posible desarrollar grandes proyectos que antes habrían sido imposibles lograr y sobre todo incentivó la comunicación entre científicos investigadores de todo el mundo (Fry et al., 2009).

Lo mismo se pretende implementar en desarrollos de software que se encuentran en repositorios privativos, cerrados y que no permiten facilitar la colaboración, dicho obstáculo se puede mitigar por medio de herramientas como OSF aplicándolas a repositorios alojados en GitHub. La principal razón es que las organizaciones pequeñas, estudiantes, universidades no se pueden permitir un gran contingente material y humano, para dar seguimiento a un proyecto que se encuentre alojado a un determinado repositorio (Olbrich et al., 2009).

Prospectiva

La presente propuesta de investigación plantea fortalecer el movimiento Open Science por medio de la accesibilidad a repositorios GitHub con la herramienta OSF, lo que pretende fomentar la libre difusión y replicación de proyectos alojados en repositorios virtuales que están amparados a derechos de autor. Reconociendo la problemática que surge al tener técnicas y métodos tecnológicos restringidos que ayuden a mejorar conocimientos aplicados a las tecnologías de vanguardia inmersas en el desarrollo de un producto de software.

Planteamiento del Problema

A lo largo de los años en Latinoamérica se ha tratado de fomentar la ciencia abierta incentivando a los investigadores en la Ingeniería de Software a dar a conocer sus conocimientos en fuentes de datos públicas, pero lastimosamente no se ha llegado a tener acceso a contenido de gran impacto al ser en gran parte repositorios privativos, también por el poco uso que se les da a herramientas Open Science.

Para lograr definir el diagrama de planteamiento de problemas se llevó a cabo el uso de la (Matriz Vester), tal como se muestra a continuación.

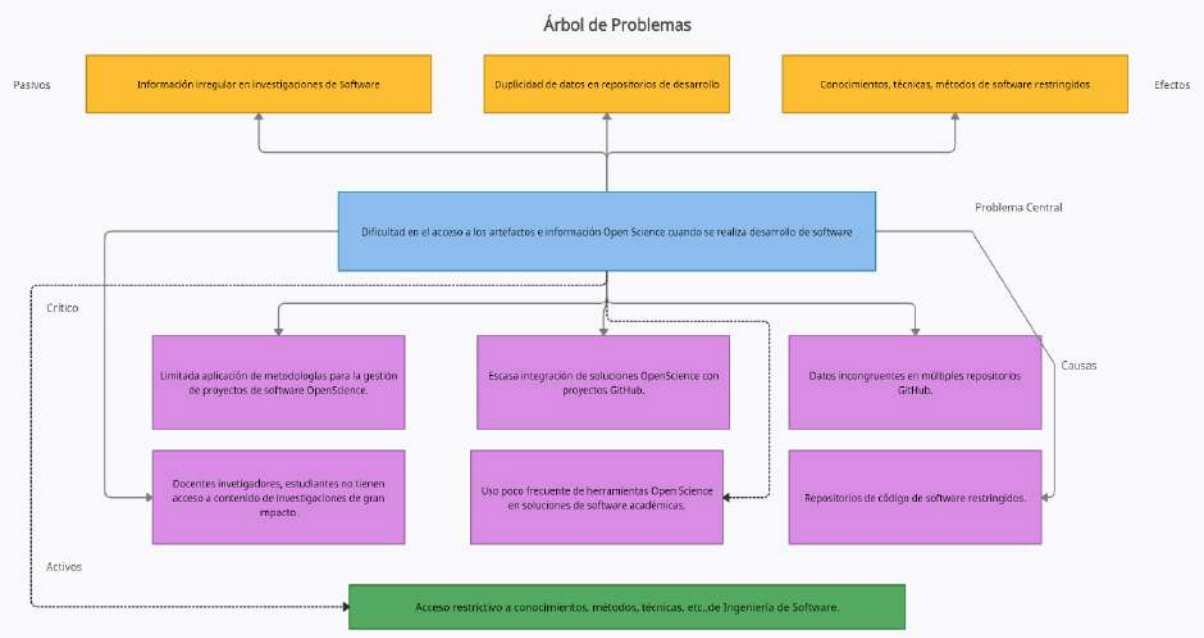


Fig. 1. Árbol de Problemas

Fuente: Propia.

Objetivos

Objetivo General

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

Objetivos Específicos

- Conocer el estado actual del uso de las herramientas OSF y GitHub para publicar contenido Open Science.
- Configurar las plataformas OSF y GitHub para integrar y publicar contenido Open Science.
- Desarrollar los aplicativos mediante una arquitectura orientada a servicios para integrar las APIs REST de OSF - GitHub y publicar contenido Open Science.
- Evaluar la arquitectura mediante las métricas de la ISO/IEC 25022.

Alcance

El presente proyecto tiene contemplado realizar la Integración de las APIs REST de las plataformas OSF y GitHub para publicar contenido Open Science mediante la aplicación de una arquitectura orientada a servicios. Se va a evaluar la eficacia de dos aplicativos mediante el uso de las métricas de la ISO/IEC 25022 las cuales permitirán conocer resultados de la integración de los aplicativos en todas las etapas del desarrollo.

En primer lugar, se realizará una revisión de la literatura que permita conocer el estado actual de las herramientas OSF y GitHub con respecto al desarrollo de Software que publica su contenido Open Science. Segundo, se procederá a configurar las plataformas las cuáles permitirán la integración y la posterior publicación del contenido del proyecto. Tercero, se procederá a desarrollar la arquitectura híbrida que abarcará dos aplicativos: una aplicación web “Integration OSF & GitHub” 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.
- Gestión de Componentes OSF & GitHub.
- Generación de contenidos e-Learning basados en los principios Open Science.
- Gestión de Procesos Académicos.
- Gestión de Procesos de Colaboración.
- Gestión de Procesos de Seguridad.
- Visualización de Reportes.

El segundo aplicativo consumirá los datos obtenidos de la aplicación Integration OSF & GitHub. Finalmente, se procederá a evaluar el software con las métricas de la ISO/IEC 25022 para conocer el alcance de usabilidad que se ha logrado al implementar el proyecto con herramientas Open Science.

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

- Arquitectura de Software MVC.
- Marco de trabajo ágil Scrum.
- Lenguaje de programación PHP.
- SDK Flutter.
- IDE de programación Visual Studio - GitHub Codespaces.
- Motor de base de datos MySQL.
- GitKraken.
- Node, Vue, React.
- Librerías UX.

Al ser un proyecto de investigación el sistema estará desplegado con data aleatoria con el fin de obtener resultados acerca de la implementación de la herramienta Open Science.

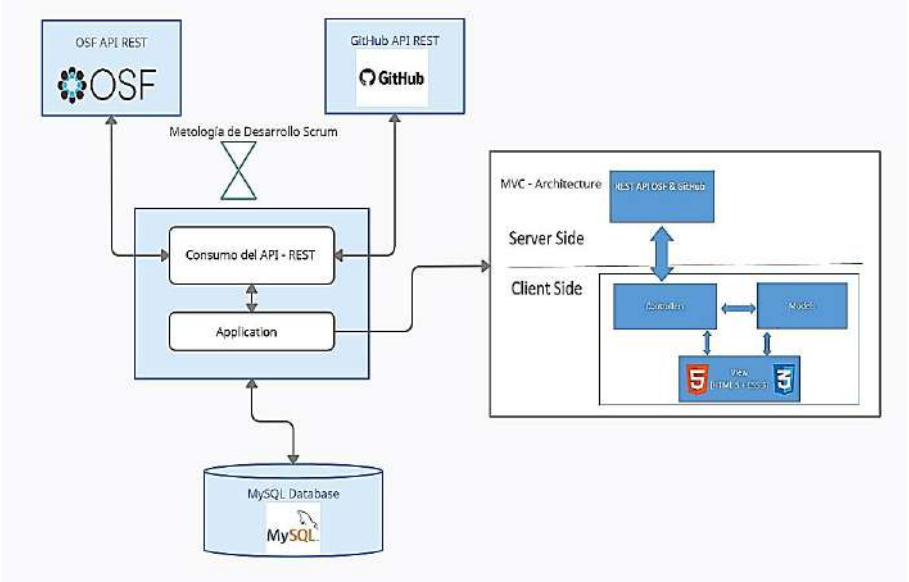


Fig. 2. Bosquejo Arquitectura Inicial del Sistema, versión preliminar.

Fuente: Propia.

Metodología

En la revisión de la literatura se realizará una revisión de la literatura mediante la definición de una cadena de búsqueda, selección de artículos, aplicación de criterios de inclusión y exclusión, y análisis y presentación de resultados sobre el estado actual del uso de herramientas OSF y GitHub en la publicación de contenido Open Science. Luego se configurarán las plataformas OSF y GitHub, para después desarrollar los aplicativos basados en la arquitectura orientada a servicios que permita consumir las APIs de OSF y GitHub, para posteriormente publicar código abierto y metodologías aplicadas en OSF. Finalmente, para realizar la evaluación de la eficacia de la arquitectura se aplicará el uso de la ISO/IEC 25022.

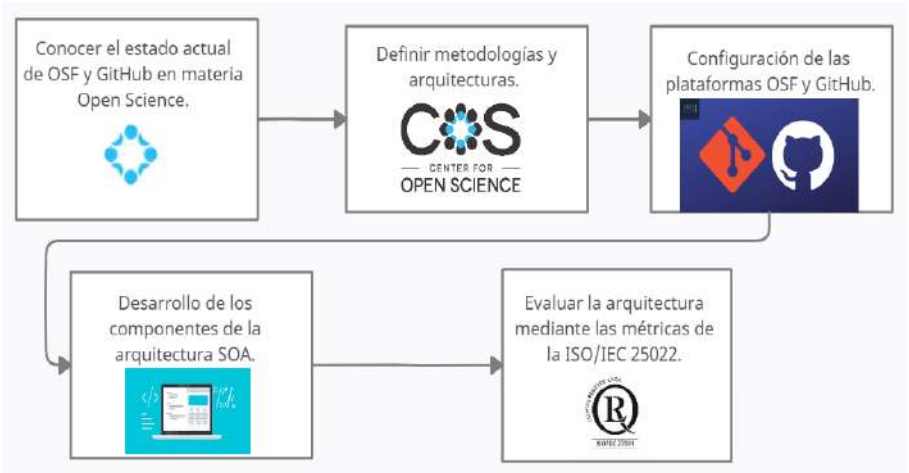


Fig. 3. Metodología de Investigación e Implementación

Fuente: Propia.

Justificación

El objetivo Nro.17 que contempla la Agenda de 2030 para el desarrollo sostenible tiene como meta conseguir alianzas para lograr sus objetivos, ya que los ODS se pueden lograr con agrupaciones mundiales sólidas y de cooperación. Específicamente en el punto 17.6 se destaca que la "**Cooperación debe ser triangular, regional e internacional, dando hincapié al acceso a la ciencia, la 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 et al., 2015).

Según (Mintel, 2019), en las principales líneas de investigación tic, específicamente la política 5.6 apunta a promover la investigación, la formación, la capacitación, la innovación y la transferencia intelectual tomando en cuenta la propiedad intelectual para impulsar el cambio de la matriz productiva, manifestando así la importancia que conlleva el conocimiento abierto y accesible para fortalecer métodos de crecimiento a nivel académico, social y estructural.

El presente proyecto tiene la intención de fomentar e incentivar el movimiento Open Science en la Ingeniería del Software aplicado en el desarrollo de una aplicación orientada a servicios, contribuyendo también a los estudiantes y docentes de la carrera de Ingeniería de Software al fomentar la investigación basada en nuevos paradigmas para generar conocimiento y soluciones basadas en tecnologías emergentes.

Justificación Tecnológica

La necesidad de realizar tareas Open Science en todas las etapas del desarrollo de la sociedad son de gran importancia, especialmente en los tiempos que transcurren, ya que será posible promover especialmente en el ámbito de la tecnología la necesidad de implementar nuevas metodologías de crear conocimientos relacionados a elementos de las ciencias de la computación (David, 2008).

Justificación Teórica

Los indicadores son poco alentadores cuando de conocimiento abierto se trata, aunque la idea de Open Science no es nueva, en la actualidad es cuando más atención requiere por parte de los gobiernos y organizaciones ya que puede ser el próximo motor que ayude a dinamizar la colaboración científica a nivel global (Garcia et al., 2013).

CAPÍTULO 1

MARCO TEÓRICO

1.1. Revisión Sistemática de Literatura

Un proyecto de investigación implica una revisión minuciosa de los contenidos que poseen los datos importantes del tema a tratar y para ello es necesario realizar un proceso el cual tiene etapas que permitirán obtener información importante. La metodología SLR “Systematic Literature Review” suele ser aplicada generalmente para mapeos sistemáticos, dicho modelo brinda ciertos hitos los cuales permiten abstraer una perspectiva general de un determinado campo del conocimiento (Webster & Watson, 2002).

1.1.1. Ruta de investigación

A continuación, se muestra el diagrama de actividades que se realizaran para la revisión sistemática de literatura, la mista que se divide en las siguientes etapas, ver Fig. 4.

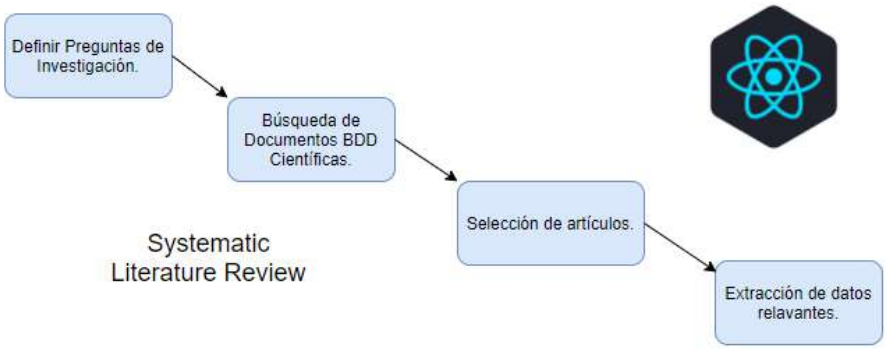


Fig. 4. Estructura Metodológica SLR, adaptado de (Webster & Watson, 2002).

Fuente: Propia.

1.1.2. Definición de preguntas de investigación

Para lograr filtros eficientes de búsqueda es necesario establecer preguntas que brinden una línea de investigación en la cual se basarán los hitos necesarios para completar el proceso, tal como se muestra en la siguiente tabla:

Tabla 1. Preguntas investigación para definición de búsqueda.

Número Pregunta	Contexto	Causas
PREG.INV.001	¿Cuál es el objetivo de publicar contenido Open Science y los métodos aplicados para hacerlo?	Estar al tanto de los objetivos de publicar contenido Open Science y los métodos usados.

PREG.INV.002	¿Qué puede aportar Open Science a la Educación?	Conocer los motivos por los que nace la necesidad de generar contenido Open Science en la educación.
PREG.INV.003	¿Cómo implementar varios recursos de plataformas Open Science mediante tecnologías API REST?	Conocer las tecnologías de vanguardia orientadas a compartir recursos entre plataformas de contenido abierto.

1.1.3. Búsqueda de documentos

Para lograr responder las preguntas establecidas con anterioridad, deben ser usados los motores de búsqueda bibliográfica y bases de datos científicas, tales como:

- Science Direct
- Scopus
- Researchgate
- Scholar Google
- Web of Science
- Peerage of Science

En la Fig. 5, se muestra el flujograma para realizar la revisión de la literatura mediante los motores de búsqueda establecidos anteriormente, es necesario generar la siguiente cadena de búsqueda: (“open science publications” OR “service oriented architecture” OR “publish open science content” OR “replicability on open science content”).

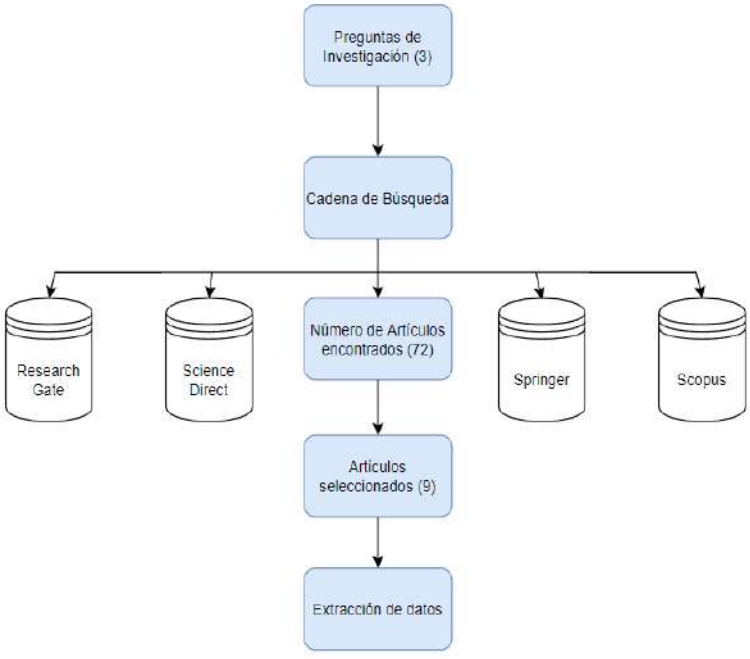


Fig. 5. Flujo de investigación.

Fuente: Propia.

1.2. Criterios de inclusión y exclusión

Open Science al ser un tema que abarca diferentes disciplinas y áreas del conocimiento debe ser filtrado a los métodos aplicados a la publicación, revisión, replicabilidad y reproducibilidad de un determinado contenido.

Los criterios de inclusión fueron establecidos por estudios relacionados al estado del arte, artículos científicos y trabajos de tesis, mismos que fueron filtrados por idiomas en este caso inglés y español. Los documentos están estrechamente relacionados a la Ingeniería de Software, asimismo se filtraron contenidos publicados entre los años 1999 hasta 2020, ya que en dicho intervalo fue el auge de Open Science y la necesidad de aplicarlo a todas las áreas del conocimiento (Abadal & Lluís, 2020).

Los criterios de exclusión para el presente trabajo fueron generados a partir de supresión de trabajos como encuestas, artículos ausentes de creative commons, catalogación y otros documentos que no posean una línea o pregunta de investigación establecida.

1.2.1. Conducta de búsqueda

Para los filtros de búsqueda de la información se llevó a cabo los siguientes pasos:

Primer Filtro:

- Fueron revisados los títulos de las publicaciones resultantes de acuerdo con la cadena de búsqueda, para ello se procedió a escoger: artículos científicos, conferencias evaluadas por pares y trabajos de titulación.
- Al seleccionar los documentos según su título, se procedió a revisar y leer el respectivo abstract de cada uno de los documentos.

Segundo Filtro:

- Después de seleccionar las publicaciones adecuadas se procedió a realizar una lectura completa para abstraer la información relevante.

1.2.2. Selección de artículos

Todas las publicaciones que se muestran en la Tabla 2, están relacionadas con todas las áreas de las ciencias de la computación.

Tabla 2. Artículos seleccionados para la revisión literaria.

Código	Título	Autor
A1	A Science Driven Production Cyberinfrastructure—the Open Science Grid	(Altunay et al., 2010)
A2	Towards Open Science: The myExperiment approach	(De Roure et al., 2010)
A3	The Grid: A New Infrastructure for 21st Century Science	(Foster, 2003)

A4	Digital Science: Cyberinfrastructure, e-Science and Citizen Science	(Pacheco et al., 2018)
A5	Decoupling the Scholarly Journal	(Priem & Hemminger, 2012)
A6	The Evolving Guide on How the Internet is Changing Research, Collaboration and Scholarly Publishing	(Bartling & Friesike, 2014)
A7	The Evolution of Public Understanding of Science-Discourse and Comparative Evidence	(Bauer, 2009)
A8	Science as a public enterprise: The case for open data	(Boulton et al., 2011)
A9	Research Data in Current Research Information Systems	(Schöpfel et al., 2017)

1.2.3. Extracción de datos relevantes

Se realizó una matriz de conceptos que ayudaron a identificar los puntos más importantes para publicar contenido Open Science, en todo el proceso de investigación, publicación, replicación. También fueron explorados aquellos términos, metodologías, técnicas, actividades, conceptos para llevar a cabo dicho proceso, para ello en la Tabla 3 se muestra la respectiva matriz de resultados.

Tabla 3. Matriz de Resultados.

Artículos												
Código	Open Science Grid	Distributed Computing	Data intensitive computing	Resource Description Framework	Computer-in-the-loop-instrumentation	Digital Science	Open Journal Systems	Public Understanding of Science	Software Architecture	Collaborative Knowledge Production	Althmetrics	Open Data
A1	X	X	X									
A2				X					X			
A3	X				X							
A4						X	X					
A5							X					
A6	X	X	X		X	X	X	X	X	X	X	X
A7										X		
A8	X						X					
A9	X											X

1.3. Open Science

1.3.1. Definición

La ciencia abierta es un cambio de modelo en la manera de generar conocimiento para hacerlo accesible a personas o instituciones, de tal modo que tiene una visión de acceso abierto “open access”, dicho modelo hace posible tener acceso abierto a todas aquellas etapas que intervienen en la investigación científica: diseño, data, revisión, corrección y publicación.

La iniciativa principalmente tiene gran alcance en países de la Unión Europea en donde se han logado evidenciar grandes avances en cuanto a compartir conocimiento de manera abierta se refiere, el objetivo de la ciencia abierta esta materializado en el desarrollo de una nube científica europea y un mayor acceso a los datos científicos generados a partir de “Horizon 2020 projects”, dicha comisión ha tomado medidas históricas para abrirse al mundo al afirmar que busca generar estrechos lazos con países de la antigua unión soviética, así como acuerdos internacionales con China y países de Sudamérica (Moedas, 2020).

El modelo que en la actualidad ha tomado más fuerza y no es nuevo, por lo que para lograr determinar cuáles son los elementos fundamentales que subyacen en la ciencia abierta es necesario obtener información de aquellas fuentes de prestigio ya sea Scopus, IEEE, Springer, etc. Como resultado, es posible determinar que la definición y ramas que se desglosan de la misma han llevado varios años siendo modificadas; en un principio la ciencia abierta comprendió cuatro elementos básicos: acceso abierto, datos abiertos, software libre y reproducibilidad.

1.3.2. Estado del Arte

Open Access actualmente es aceptado como una estrategia para la publicación científica, por lo que investigadores en todo el mundo han optado por publicar el producto de sus investigaciones en plataformas Open Access, lo que significa que el estado del arte es impresionante desde el punto de vista de actores internacionales de la ciencia, la política y la ciencia.

En el año 2018 en Europa se crea High Level Experts Group – Open Science Policy Platform, mismo que estableció los ochos peldaños sobre los que se sustenta Open Science los cuales son: reconocimiento e incentivos, métricas con nuevos indicadores para la evaluación investigativa, comunicación científica a futuro, European Open Science Clod (EOSC), data “findable, accesible, interoperable and reusable”, integridad, capacitación y formación, ciencia ciudadana (LERU, 2018).

La Unión Europea es el actor principal en generar programas de financiación e innovación que busca que los artículos producidos con financiamiento Horizon 2020, sean accesibles en línea por los editores y que todos aquellos gastos serán objeto de reembolso a los investigadores. Todo aquel producto, artefacto y técnica aplicada a una investigación científica deberá ser publicado en un repositorio público, dicho proceso deberá seguir una serie de pasos los cuales tomarán alrededor de 12 meses para ciertas disciplinas, ver Fig. 6.

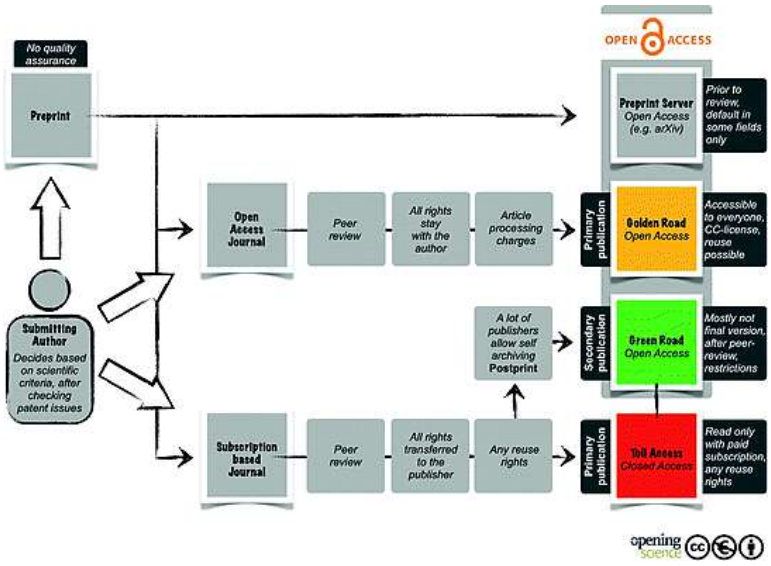


Fig. 6. Open Access process adaptado de (Sitek & Bertelmann, 2014).

Fuente: https://bit.ly/3oL3Ez6.

El proceso de publicación Open Access requiere llevar a cabo varias tareas síncronas y asíncronas por lo que es necesario comprender que implica cada una de ellas para determinar la secuencia de los procesos en una visión en conjunto:

- Preprint: Se refiere al contenido, mismo que al no estar publicado completamente, puede estar abierto a varias interpretaciones por lo que no se garantiza la calidad del contenido que contenga una determinada investigación.
- Toll Access – Publicación Primaria: El contenido desplegado está basado en las políticas de una determinada plataforma, usualmente los investigadores permiten una vista previa de aquellos resultados obtenidos durante la investigación y gran parte de los contenidos se encuentran ligados a suscripciones de pago.
- Postprint: Los contenidos obtenidos durante la investigación son archivados por quienes los publican en plataformas para una revisión.
- Green Road – Publicación Secundaria: El contenido es sometido a revisión por pares con ciertas restricciones, cabe recalcar que no es una versión final de la investigación.

- Golden Road – Publicación Primaria: El contenido es publicado en una revista científica y por ende en una plataforma de acceso abierto después de llevar a cabo la revisión por pares, establecer los derechos del autor inicial y aplicando los respectivos cambios al contenido con una licencia CC-license.
- Preprint Server: El contenido es finalmente desplegado de manera global a un servidor Open Science el cual contará con las políticas necesarias para mantener los derechos de autor, fomentando así la replicabilidad, reutilización de los artefactos de una determinada investigación.

Los pasos anteriormente detallados buscan que el conocimiento no sea centralizado de tal manera que todas las personas tengan acceso a el. Según (Peter Murray-Rust), el contenido abierto salva vidas; por ello que el programa Horizon 2020 busca que los resultados fruto de la investigación sean de acceso completamente abierto. Con dichos procesos y etapas mencionadas para la publicación de contenido abierto se espera que todos aquellos conocimientos generados por la investigación generen valor en áreas tales como la energía, medio ambiente, salud, tecnologías de la información y la comunicación.

El contenido abierto no tan solo pretende ser un nuevo paradigma para generar conocimiento, sino que quiere ir más allá superando fronteras tales como las revistas tradicionales que niegan el acceso a instituciones, investigadores o personas que no pueden permitirse pagar precios elevados; el actual paradigma propone un nuevo modelo de negocio para la publicación de contenido abierto en revistas de acceso gratuito.

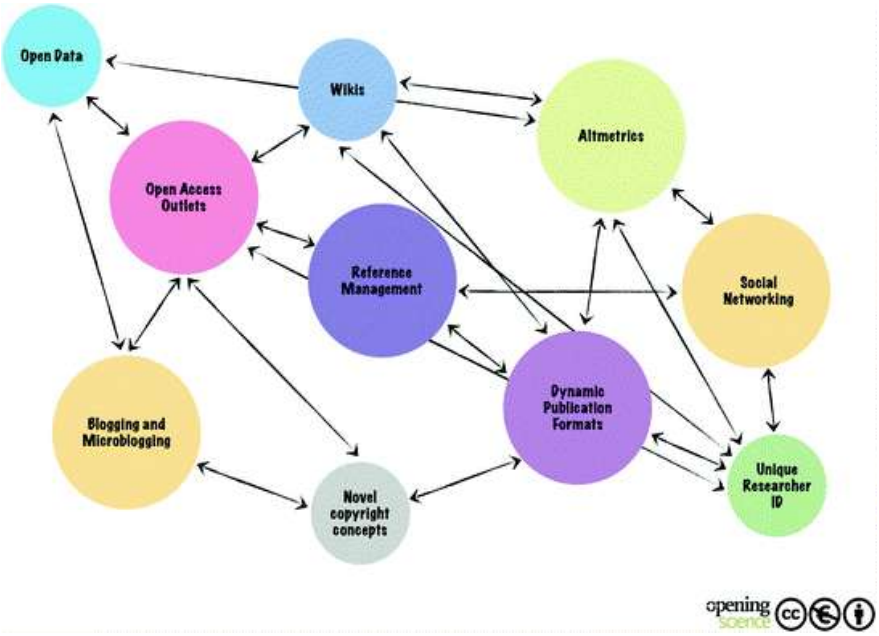


Fig. 7. Segunda Revolución Científica adaptado de (Fecher & Friesike, 2014).

Fuente: https://bit.ly/3cpbMj6.

También llamada la Segunda Revolución Científica, se sustenta en varios aspectos y herramientas tecnológicas novedosas, mismas que tienen sentido si se crean nuevas estrategias como lo son sistemas de tendencias que pueden lograr que investigadores se logren identificar en cualquier lugar del mundo dentro de una red global.

El acceso abierto permite que no solamente artículos sean publicados con los estándares mencionados, ya que los contenidos generados pueden ser publicados por sus autores en sitios alternativos e incluso en otros servidores, por lo general se puede dar uso de servidores institucionales, los cuales cuenten con las políticas necesarias para publicar contenidos, materiales de trabajo, técnicas aplicadas y otros elementos (Harnad, 2007).

1.3.3. Aprendizaje Abierto

Open Science al ser un movimiento que busca que el conocimiento científico sea abierto, tiene como determinación que los datos y su diseminación sea accesible a todos los niveles de la sociedad. Generando de dicha manera un aprendizaje abierto en el que se detalle de manera minuciosa todas las etapas que ha tenido que seguir un determinado trabajo para lograr la distribución, replicabilidad y reutilización de los componentes.

En la actualidad las razones para generar conocimiento son apremiantes pues la necesidad de información de valor y contrastada se hace necesaria, el internet y todas sus herramientas han logrado llevar conocimiento, noticias, etc., a lugares en donde antes la censura previa impedía que llegase el conocimiento, ahora eso forma parte del pasado pero como lo demuestra la historia, los grandes cambios generalmente traen consigo ciertos inconvenientes y uno de los más apremiantes son las conocidas “fake news” (Peters, Isabella. Frodeman, Robert. Wilsdon, James. Bar-Ilan, Judit. Lex, Elisabeth. Wouters, 2017).

Se ha percibido cierta dicotomía contrastada entre las universidades y la sociedad, ciertas noticias falsas llegan a tener un impacto considerable ante la opinión pública por ello nace la necesidad de que las noticias sean contrastadas desde puntos de vista sustentados en hechos comprobables por investigadores de universidades dando así noticias relevantes y que generen valor a la sociedad. Es necesario tener en cuenta que el movimiento ha tenido una gran acogida durante los últimos años especialmente por las políticas implantadas por la Unión Europea en todas las áreas investigativas que sean de interés general para países de la zona euro y sus aliados, así también para que puedan ser de gran ayuda en países en vías de desarrollo (MacCallum, 2018).

1.3.4. Indicadores para medir Open Science

Las oportunidades que genera para los investigadores el movimiento Open Science representa cambios significantes y por ende grandes oportunidades, especialmente para las

universidades las cuales miden sus habilidades de acuerdo con métricas, métodos los cuales generan que perciban prestigio a nivel local y mundial.

Para lograr comprender el ciclo de vida de Open Science y todos los actores que intervienen tanto como emisores y receptores, es necesario percibir lo más conveniente de acuerdo con el contexto investigativo que será abordado.

Open Science Monitor fue desarrollado por el RAND Europe, Deloitte y Digital Science & Research Solutions, incluyendo Altmetric.com y Figshare, ver Fig. 8.

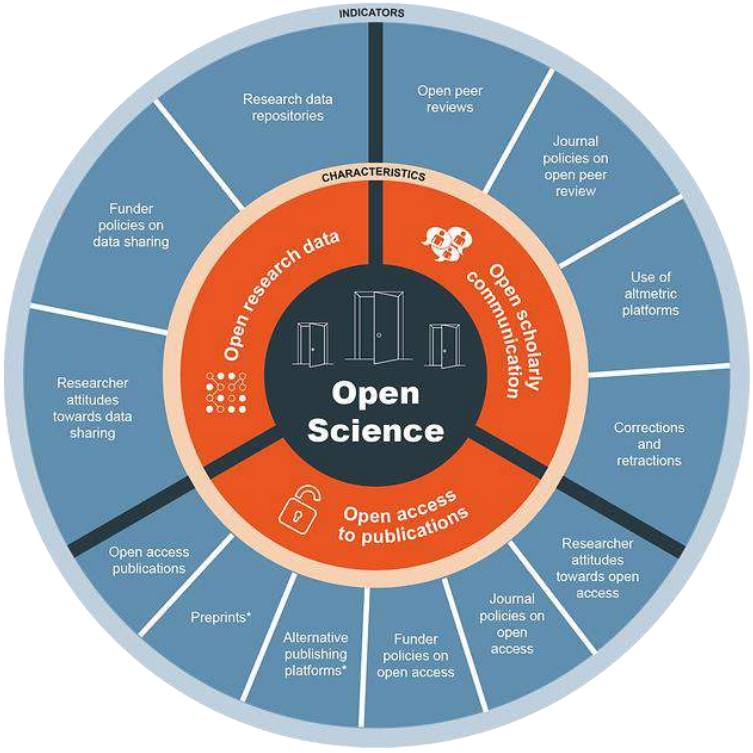


Fig. 8. Open Science Monitor – European Commision.

Fuente: https://bit.ly/3FvfIvb.

Según la (Commision, 2020), las principales funciones para medir Open Science son las siguientes:

1. Comunicación Académica Abierta (Open Scholarly Communication): Se encarga de generar una cultura de comunicación abierta.
    - Revisión Abiertas por Pares (Open Peer Reviews): Etapa en la que se realiza la revisión de un determinado contenido por dos o más personas relacionadas al enfoque investigativo para validar el contenido.
    - Políticas de Revistas sobre revisión Abierta por pares (Journal Policies on Open Peer Review): A diferencia de la primera, en dicha etapa se realiza la revisión de contenido en revistas de índole científica e investigativa.

- Uso de Plataformas Altmétricas (Use of Altmetric Platforms): Se refiere a métricas alternativas para evaluar el contenido open science.
- Corrección y Retractaciones (Correction and Retractions): En caso de cometer errores por pequeños que sean deben ser reportados para evitar que personas e instituciones que comparten el mismo tema sigan caminos de investigación erróneos.

2. Acceso Abierto a Publicaciones (Open Access to Publications): Se refiere a que todo contenido de cualquier área del conocimiento debe ser publicado en repositorios digitales de Acceso Abierto.
   - Actitudes de los Investigadores hacia el conocimiento abierto (Researcher attitudes towards open access): Es muy importante evaluar y hacer seguimiento a las actitudes de los investigadores referente al contenido abierto.
   - Políticas de las Revistas sobre Acceso Abierto (Journal Policies on Open Access): Generar políticas equitativas que permitan generar contenidos científicos.
   - Políticas del Financiamiento sobre Acceso Abierto (Funder policies on open access): Busca generar políticas que fomenten una sociedad académica dirigida hacia la publicación de contenido abierto.
   - Plataformas de Publicación Alternas (Alternative Publishing Platforms): Evaluar las métricas de alcance e impacto de un determinado contenido que ha sido publicado en plataformas no convencionales.
   - Preimpresiones (Preprints): Encargado de monitorear las preimpresiones las cuales están sujetas a derechos de autor.
   - Publicaciones Acceso Abierto (Publication Open Access): Encargado de monitorear las publicaciones de acceso abierto manteniendo Creative Commons.
3. Datos de Investigación Abiertos (Open Research Data): Se refiere a que los datos deberían estar abiertos en todos sus formatos, para lograr mejores filtros de búsqueda.
   - Repositorios de datos de Investigación (Research data repositories): Buscan establecer repositorios de datos investigativos.
   - Políticas de los Financiadores de datos compartidos (Funder policies on data sharing): Pretende generar políticas de índole económica que permita llevar a cabo el financiamiento de datos.
   - Actitudes de los Investigadores hacia los datos compartidos (Researcher attitudes towards data sharing): Permite generar indicadores que muestren las actitudes de los investigadores hacia los datos de investigación abiertos.

Innovación Open Science

La sociedad tal como la conocemos no habría progresado como lo podemos constatar hoy en día de no haber sido por las grandes inversiones que se han hecho a todos los proyectos que se relacionan a la tecnología, no cabe duda de que la inversión privada ha tenido un rol muy importante. Sin embargo, en los últimos años se denota claramente que el conocimiento abierto genera gran interés en gran parte de estudiantes, investigadores y personas en general; con la llegada del nuevo siglo también llegó el auge del internet y las telecomunicaciones que han jugado un rol importante en todo aspecto cotidiano. A partir de lo anteriormente mencionado surge la idea de impulsar la Innovación Abierta por ello se encuentra constantemente evolucionando y moviéndose de manera lineal, generando transacciones y colaboraciones dinámicas, colaborativas y conectadas; permitiendo así establecer ecosistemas de innovación multi colaborativos (Chesbrough, 2006).

1.3.5. Conceptualización Open Science

El modelo establecido por Open Science el cual se sustenta en la investigación e innovación participativa ha logrado tener un éxito rotundo cuando se trata de unir estudiantes, investigadores y la sociedad, con el fin de abordar problemas reales a nivel local e internacional, así mismo para generar resultados positivos en la creación colaborativa de soluciones del mundo real, contribuyendo de dicha manera a la excelencia en materia de investigación y aceptabilidad de los resultados de la innovación (Commission, 2017).

Después de una minuciosa revisión bibliográfica de todas las fuentes posibles. (Fecher & Friesike, 2014) llegan a la conclusión que del término Open Science subyacen cinco conceptualizaciones o Escuelas del Conocimiento:

a) Public School (Escuela Pública): La Escuela Pública hace hincapié en la necesidad de que la ciencia sea publica, de la misma apuntan a dos corrientes: La Accesibilidad al proceso de investigación (Producción) y la comprensión del proceso de investigación (Producto); permitiendo así que su audiencia aumente considerablemente al explotar las herramientas disponibles como lo son redes sociales y las tecnologías Web 2.0 (Fecher & Friesike, 2014).
b) Democratic School: La Escuela Democrática se centra en la importancia que se le debe dar al acceso a los productos obtenidos después de todos los procesos adyacentes a una investigación científica, los defensores de dicha escuela no solo buscan que el acceso sea libre a los productos y datos científicos, van más allá tomando en cuenta la materia prima empleada, representaciones digitales, material de formato multimedia y materiales digitales. Lo que denota que todo

producto de investigación debe ser accesible de manera gratuita e igualitaria, más aún cuando son productos provenientes de inversión pública.

c) Pragmatic School: La Escuela Pragmática considera que la ciencia abierta es un proceso que puede mejorarse y por ende optimizarse para que la divulgación científica logre ser eficiente en todos los campos del conocimiento. Es aquí cuando el término Web 2.0 viene a formar parte de un campo aún en auge, tomando en cuenta que los problemas de la era moderna se han vuelto tan complejos que requieren formar conexiones de esfuerzos para lograr solucionarlos (M. Nielsen, 2013).

d) Infrastructure School: La Escuela de Infraestructura es aquella que permite prácticas de investigación que tienen origen en la internet en base a las necesidades de organizaciones públicas o privadas, generalmente dicha escuela procura mejorar infraestructuras tecnológicas en su gran parte herramientas de software y aplicaciones. (Nentwich, 2003), fue quien estableció el término cyberscience (ciberciencia) para describir las tendencias de aplicar tecnologías de la información y las comunicaciones a la resolución de problemas que surgen en la ciencia de la investigación.

e) Measurement School: En la Escuela de Medición la importancia del conocimiento se genera de manera que se pueda catalogar y medir, para ello genera o promueve estándares alternativos para determinar el impacto científico que han tenido las investigaciones. Es decir, se mide el promedio de número de citas de un artículo en una revista lo que tiene gran influencia en establecer una reputación al investigador encargado, generando así financiamiento y oportunidades profesionales (Fecher & Friesike, 2014). A medida que la era tecnológica fomenta la mejora de procesos surgen nuevos desafíos en todas las áreas del conocimiento, mismos que necesitan ser superados como, por ejemplo, la revisión por pares que requiere mucho tiempo.

   Asimismo, dicha escuela busca generar un impacto alternativo y más rápido de medición mismo que incluye otras formas de publicación como por ejemplo Altmetrics que son una métrica alternativa para analizar e informar a los académicos sobre el impacto de algún artículo publicado (CH Yeong; Abdullah BJJ., 2012).

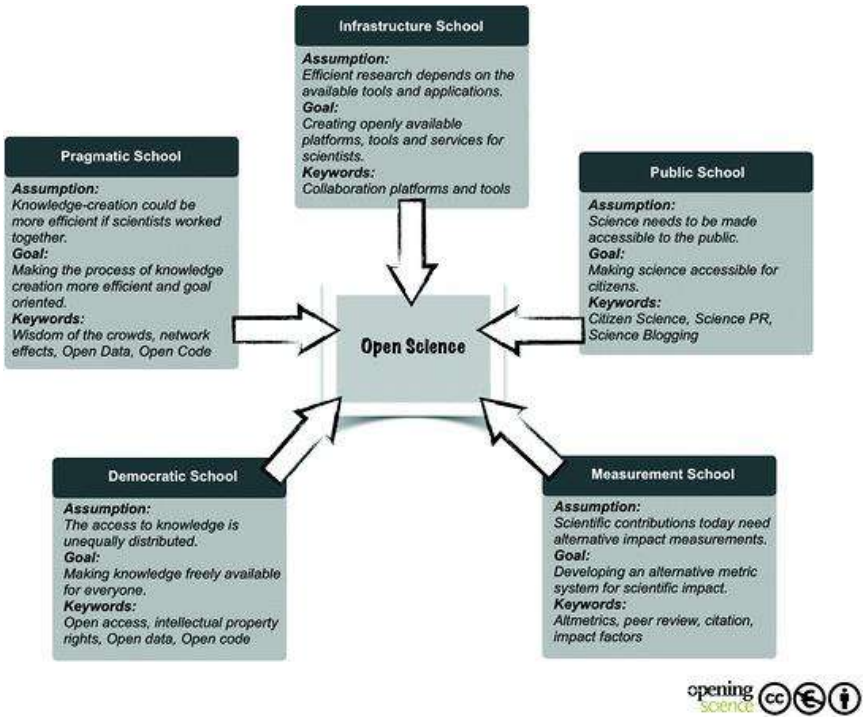


Fig. 9. Five Scholls Of Thought adaptado de (Fecher & Friesike, 2014).

Fuente: https://bit.ly/3nr5IwN.

Creative Commons Licenses

Las licencias Open Science al igual que las licencias convencionales tienen varios estándares, en contenido abierto las licencias habitualmente son llamados creative commons los cuales son usados habitualmente en revistas científicas y también son populares en la red, por ello es necesario que todo investigador comprenda como aplicarlas independientemente del contexto investigativo (Friesike, 2014).

Según el autor, entre las principales Licencias Creative Commons se encuentran:

a) No Copyright: Public Domain CC0: Es la licencia con la cual una determinada persona ha publicado contenido para el público en general, misma que ha renunciado a todos los derechos de autor, por lo que otras personas pueden usar el contenido para realizar copias, modificaciones y distribuirlo con fines de índole económica.
b) Attribution-NoDerivs CC BY-ND: Es la licencia que permite la distribución del contenido sin modificación alguna de manera comercial y no comercial, salvaguardando en totalidad los derechos del autor original.
c) Attribution CC BY: Es la licencia que mayores permisos generan en un determinado contenido abierto, ya que permite que todos la puedan distribuir, modificar, mezclar y aprovechar el trabajo, inclusive de manera comercial; se recomienda usar dicha licencia para una mayor difusión del contenido y sus complementos.

d) Attribution-ShareAlike CC BY-SA: Es la licencia que se recomienda para contenidos los cuales beneficiarían su difusión al incorporar nuevos elementos, es el caso de Wikipedia la cual nutre sus servidores de información de todas las fuentes posibles; la licencia permite a los demás modificar, mezclar y construir otros contenidos.
e) Attribution-NonCommercial CC BY-NC: Es la licencia que permite a otros modificar, mezclar y construir sobre el trabajo del autor inicial de forma no comercial, manteniendo los derechos y créditos correspondientes del autor inicial.
f) Attribution-NonCommercial-ShareAlike CC BY-NC-SA: Es la licencia que permite a otros modificar, mezclar y construir en base al trabajo del autor inicial sin fines comerciales, siempre y cuando le den a este los derechos a nuevos contenidos derivados del original que se basen en la misma licencia.
g) Attribution-NonCommercial-NoDerivs CC BY-NC-ND: Es la licencia más restrictiva de todas las mencionadas hasta el momento, ya que sólo permite a otros descargar contenidos y compartirlos en la comunidad siempre y cuando le den crédito al autor inicial, tampoco es posible hacer cambios al contenido de cualquier manera ni usarlo de manera comercial.

Las licencias mencionadas permiten al autor del contenido abierto precautelar la autoría del contenido teniendo en cuenta cual de todas se adapta de mejor manera a su trabajo y esfuerzo, por tal razón es evidente que inclusive la más restrictiva de todas permite que el conocimiento pueda abrirse camino de manera abierta, dichas licencias se establecen con el fin de salvaguardar la originalidad de las investigaciones en todas sus etapas para mantener la integridad de la información y su origen.

1.4. Estado del arte OSF y GitHub

Según (Golder & Macy, 2012), usar enfoques innovadores ayuda a superar las capacidades de la investigación tradicional ya que permiten realizar observaciones de cambios que se llevan a cabo en un determinado sistema complejo a medida que se desarrolla y permiten conocer aquellas predicciones sobre el comportamiento de dicho sistema. Las herramientas que se encuentran disponibles en la red permiten manipular almacenamiento masivo, altos volúmenes de procesamiento y el acceso independiente que tienen cada una de ellas; lo que permite generar nuevas formas de investigación que dependen de un eficiente funcionamiento del procesamiento de datos.

Las ciencias de la computación y las ramas que la conforman juegan un rol fundamental que permite generar mayor valor a una investigación, para el presente trabajo se requiere conocer el estado actual de las plataformas OSF y GitHub; ya que son dos herramientas robustas y frecuentemente aplicadas a térmicos muy conocidos que son Open Source y Open Science.

1.4.1. Estado actual OSF

Una de las plataformas que será utilizada para la integración en el presente proyecto es OSF Open Science Framework, por ello es necesario evaluar el estado y componentes que se encuentran al servicio del público en general; si bien es cierto OSF es la herramienta para desarrolladores, investigadores, estudiantes, etc., misma que forma parte de las herramientas de COS “Center of Open Science”, ver Fig. 10.

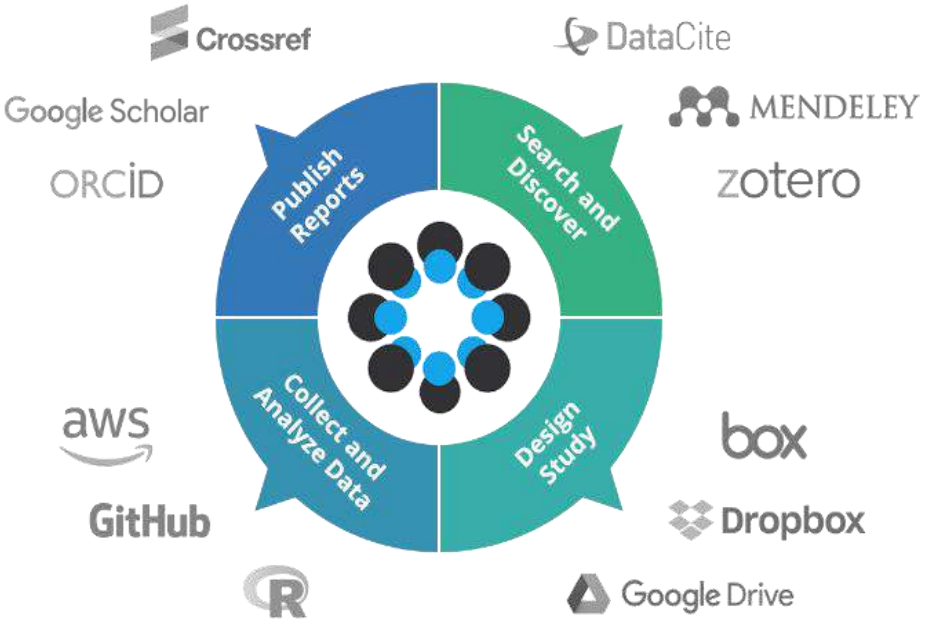


Fig. 10. COS Workflow.

Fuente: https://bit.ly/2WuNlfP.

COS es una organización tecnológica sin fines de lucro, fundada en el año 2013. Según sus miembros promueve la apertura, integridad y reproducibilidad de la investigación en todas las fases en las que se encuentre. Con ello busca cambiar los modelos establecidos en cuanto a divulgación científica para que sea abierta y disponible para todos.

(Bartling, 2014a) describe los métodos aplicados en la actualidad que resultan novedosos para la publicación y colaboración del conocimiento académico son de gran utilidad, herramientas tales como Google Docs, Google Cloud, Dropbox con sus nubes de almacenamiento fomentan que recursos sean gestionados de una manera eficiente y ahorrando así cientos de horas de trabajo a los involucrados en un determinado equipo de investigación. Por tal razón OSF cuenta con un flujo de trabajo con varias herramientas o complementos los cuales permiten llevar de mejor manera todo el conjunto de componentes, datos, documentos, etc., de un determinado repositorio.

1.4.2. OSF (Open Science Framework)

OSF contiene una variedad de servicios los cuales son de gran utilidad para aquellos interesados en conocer las etapas del proceso investigativo, dicho Framework ha establecido varios términos que son de gran importancia para publicar un repositorio investigativo sin violar ningún derecho de autor, norma internacional o estatuto legal. Términos que se describen a continuación:

a) Proyecto: Es un conjunto de elementos que son manipulados por una organización individual o colaborativa, la cual publica el contenido investigativo de componentes, herramientas, manuales y otros elementos.
b) Preimpresión: Es un documento que se ha compartido públicamente y sus metadatos asociados. No se pueden eliminar, pero se pueden retirar dejando consigo evidencias en metadatos que pueden ser usados por otros investigadores.
c) Registro: Es una versión congelada y no editable, la cual contiene una marca de tiempo del proyecto y cada uno de sus componentes, al igual que una preimpresión se puede retirar su registro, pero dejará un metadato de evidencia.
d) Contenido: Según los términos de uso del Framework, el contenido abarca: datos, texto, imágenes, código de software, ejecutables de software, conjunto de datos, información, material multimedia y otros elementos.
e) Colección: Es un conjunto de datos alojado en la nube del Framework. Contienen preimpresiones OSF, registros OSF, proyectos OSF y otras herramientas desarrolladas por una determinada organización o usuario.
f) Usuario: Es el individuo que tienen la facultad de aprobar, agregar, editar o eliminar el contenido en un proyecto de manera responsable.
g) Administrador: El administrador de un proyecto o componente posee todos los derechos y facultades para controlar todos aquellos aspectos de un proyecto, así mismo puede controlar los datos y contenidos de cada uno de los usuarios contribuyentes para publicarlo sin infringir ninguna ley de derecho intelectual.
h) API Pública: Es la interfaz de programación de aplicaciones que facilita la comunicación entre los servidores OSF y otras plataformas. Con dicha herramienta es posible acceder a todos los servicios de las API públicas de: GitHub, Google Scholar, Mendeley, Dropbox y otras plataformas.

API OSF

OSF tiene entre sus varias herramientas una API pública que permite gestionar de mejor manera los repositorios de cada uno de sus usuarios, por ello no es difícil perder el hilo de un

determinado trabajo de investigación desde el punto de partida, ya que por cada cambio se genera un creative common que guarda el usuario, fecha, hora y determinado componente mejorado. Dicha herramienta es usada para gestionar repositorios y archivos producto de estudios, materiales, datos, manuscritos o cualquier tipo de instrumento asociado a una determinada investigación, la API cumple con la función de medir el impacto al monitorear el tráfico de proyectos y archivos que se hacen públicos (OSF.IO, 2021).

Los principales componentes con los que cuenta la API son:

a) Autenticación: Para dar uso de la API es necesario generar un token con privilegios a todos los servicios que se deseen utilizar, es importante recalcar que el token puede ser usado desde una terminal Python la cual genera varios beneficios entre los que se destacan manejar grandes volúmenes de datos para Machine Learning, Inteligencia Artificial y otros.
b) Control de Versiones: OSF cuenta con varias versiones de la API que pueden ser usadas de acuerdo con las necesidades del usuario, para ello se debe detallar en el encabezado de solicitud con las opciones necesarias.
c) Filtración: Las colecciones de datos deben ser filtradas mediante parámetros de consultas de formulario, es decir en cadenas.
d) Conjuntos de campos dispersos: Es un método para restringir los campos de atributos que tengan relación entre sí y generar filtros a campos específicos de un determinado nodo del repositorio.
e) Estructura de errores: Si la solicitud realizada por el usuario genera conflictos, la API dará como resultado un error en dónde se detallará el motivo y código de error.
f) Entidades: Es un recurso único que se ha obtenido a partir del ID de un determinado elemento del repositorio a través de la API.
g) Colecciones de entidades: Son los criterios de valoración de una determinada entidad, la cual retorna una lista de entidades y una estructura de datos de paginación.

La presente API de OSF ofrece un conjunto de elementos que permite manipular los datos de un repositorio haciendo uso de las referencias asignadas por el Framework, dichas referencias son versátiles en su uso, ya que permiten realizar operaciones dando uso a JSON Web Token, por ello queda claro que para acceder a dichas referencias el usuario deberá generar un token en la plataforma OSF. Según la plataforma COS, el API OSF para desarrolladores contiene las siguientes referencias (OSF.IO, 2021).

a) Complementos: Objetos que están relacionados a colecciones que deben ser configurados por el usuario.
b) Raíz: Es la ruta que COS usa para permitir el acceso a usuarios, proyectos, componentes, registros y archivos mediante OSF.
c) Citas: Objetos que están relacionados a los estilos estandarizados para renderizado de citas.
d) Comentarios**:** Solicitud de respuesta a un determinado mensaje en un repositorio.
e) Colecciones**:** Objetos que están relacionados a colecciones. Pueden ser públicas o privadas, de acuerdo con el criterio del usuario.
f) Archivos**:** Objetos que están relacionados a la representación de un archivo solicitado, siempre y cuando el fichero esté disponible en el repositorio.
g) Licencias**:** Objetos que están relacionados a las licencias de acuerdo con el nombre y permisos que se le atribuyen a un determinado contenido.
h) Registros**:** Objetos que están relacionados a registros con los respectivos detalles, dicho registro es permanente e inmutable de acuerdo con el nodo al que se asignó.
i) Instituciones**:** Objetos que están relacionados a la institución a la que se encuentra sujeto el repositorio y los recursos.
j) Metaesquemas**:** Objetos que están orientados a describir preguntas complementarias de un registro.
k) Nodos**:** Objetos que contienen proyectos o componentes de un determinado repositorio.
l) Preimpresiones**:** Objetos que están relacionados a los proveedores de preimpresiones con fecha de creación y actualización**.**
m) Proveedor de Preimpresión**:** Objetos que están relacionados a los proveedores de cada preimpresión de un determinado contenido abierto.
n) Inscripciones**:** Objetos que están relacionados al registro que tiene acceso el usuario. Las inscripciones no pueden ser eliminadas, pero si pueden ser retiradas.
o) Taxonomías**:** Objetos que están relacionados a las taxonomías de las disciplinas Bepress.
p) Usuarios**:** Objetos que están relacionados a los usuarios de un determinado repositorio.

Mediante la aplicación de las propiedades anteriormente descritas es posible gestionar repositorios haciendo uso de los accesos a la API OSF, tareas tales como gestión de complementos, usuarios, licencias, nodos y otros elementos resultan de gran ayuda para llevar a cabo proyectos de investigación relacionados al desarrollo de software. Pues su

versatilidad permite la integración de repositorios de almacenamiento externos, herramientas de desarrollo, plataformas de versionamiento y de hosting.

1.4.3. Estado actual GitHub

La plataforma GitHub en los últimos años se ha vuelto tan importante para el campo de desarrollo de software que para nadie es ajena su variedad herramientas y ventajas a la hora de fabricar software mediante control de versiones. La plataforma presta servicios en la nube y repositorios libres, que resultan una gran ventaja para gestionar repositorios de manera remota y llevando a cabo buenas prácticas (GitHub, 2021).

El movimiento Open Science ha tenido un gran éxito tras demostrar que puede transformar significativamente las tareas asociadas a la ciencia, el propósito de fomentar el término “ciencia” es cambiar por completo su significado para ello hay que tomar en cuenta que las prácticas de comunicación varían entre diferentes disciplinas, pero en la actualidad no existe una taxonomía completa que permita definir por completo el término y sus diferencias dentro de la comunidad Open Science (Sidler, 2014).

Learning GitHub

GitHub ofrece una gran variedad de herramientas, entre las que se encuentran aquellas que fomentan la educación, para ello la plataforma ofrece servicios específicos tales como:

- GitHub Research: El paquete ofrece herramientas y servicios que permiten la colaboración en trabajos de investigación alrededor de todo el mundo.
- GitHub Classroom: El paquete ofrece herramientas, materiales educativos, obsequios y otros elementos a estudiantes. Es aplicado para distribuir código y brindar comentarios a los estudiantes que disfrutan discutir tendencias tecnológicas actuales aplicadas a la educación.
- Paquete Estudiantil: El paquete en cuestión requiere que el usuario se encuentre asociado a una institución acreditada para usar los servicios de GitHub, entre las herramientas se encuentran: Namecheap, Microsoft Azure, JetBrains Licence, Bootstrap Studio, Education Host, GitHub Campus Experts y otras herramientas usadas en toda la etapa de desarrollo de software.
- Aula GitHub: Es el espacio en donde los estudiantes y docentes llevan a cabo una interacción durante un determinado curso. Un aula GitHub permite crear registros de alumnos para luego realizar el seguimiento de tareas relacionadas a un tema relacionado a desarrollo de software.

1.4.4. GitHub (Open Source Repositories)

API GITHUB

GitHub cuenta con una API REST que permite manipular determinados repositorios que pueden ser gestionados de manera remota, llevando así a cabo una Integración de datos con diferentes plataformas para publicar código.

La API REST de GitHub, permite que la manipulación de elementos de repositorios sea sencilla de manipular desde cualquier otra plataforma dando uso al denominado Personal Access Token (PAT) que deberá ser solicitado por el usuario creador del repositorio. Para generar la autentificación necesaria; la API cuenta con las siguientes referencias que son de ayuda para administrar repositorios.

a) Acciones: Administración de acciones de la API REST que está disponible para los usuarios autenticados, OAuth Apps y GitHub Apps. Dichas acciones cuentan con artefactos que permiten descargar, borrar y obtener información.
b) Actividades: Se refiere al conjunto de eventos, fuentes, notificaciones, alcance y observaciones de un determinado repositorio.
c) Aplicaciones: Las aplicaciones permiten obtener información de alto nivel sobre alguna aplicación GitHub App, también permite generar información específica acerca de las instalaciones en otras plataformas mediante la autenticación con un token OAuth.
d) Bases de datos Git: La API REST de la base de datos de Git provee servicios de lectura y escritura para todos aquellos objetos que sea necesario procesar en un determinado repositorio. En la Fig. 11 se muestra el procesamiento de los objetos de la base de datos Git.

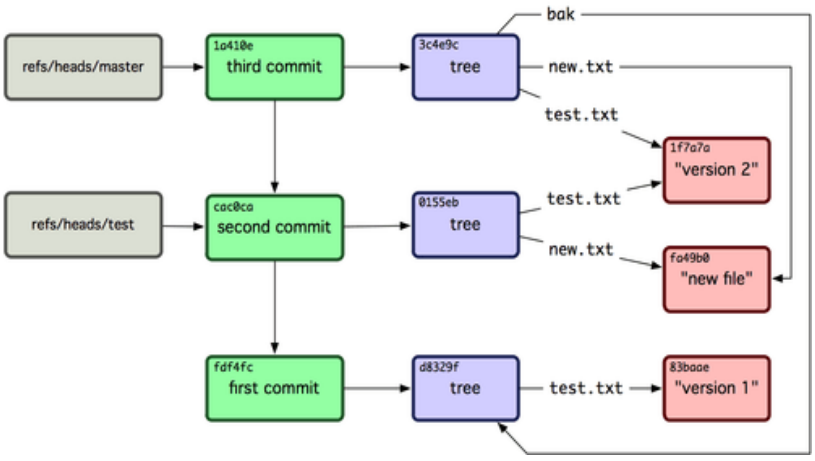


Fig. 11. Base de Datos GitHub API, recuperado de Docs GitHub.

Fuente: https://bit.ly/3FtQPzV.

e) Proyectos: Se refiere al conjunto de colaboradores, columnas y conjuntos de los que hace uso la GitHub para acceder a los repositorios de una determinada organización.
f) Repositorios: Un repositorio es un conjunto de elementos que abarca colaboradores, comentarios, confirmaciones, comunidades, organizaciones, revisiones y otros elementos que ayudan a incrementar valor a un determinado proyecto.
g) Usuarios: Mediante el token de autenticación el usuario con los permisos necesarios puede gestionar las tareas específicas en todos los elementos de la cuenta GitHub.

Codespaces

Es una tecnología implementada en GitHub que permite gestionar código directamente en la nube dando uso a contenedores Docker y máquinas virtuales las cuales ejecutan el código fuente en tiempo real. Por ende, facilitan el versionamiento y la colaboración hasta las fases de despliegue en producción de un determinado proyecto de software. El usuario puede acceder al código mediante el registro de imágenes docker privadas, ver Fig. 12.

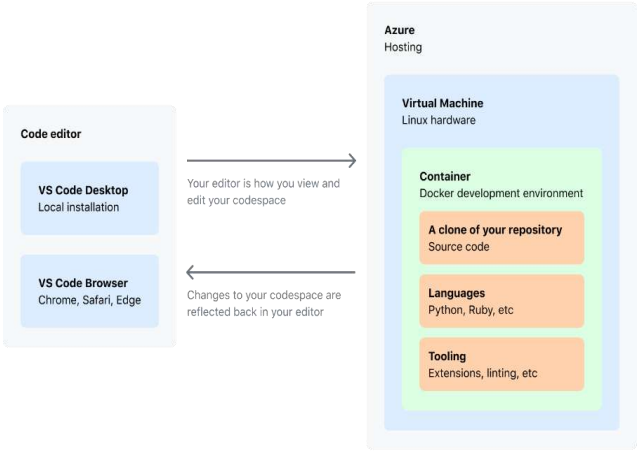


Fig. 12. Arquitectura Codespaces, Recuperado de Docs GitHub.

Fuente: https://bit.ly/3yoY5Jn.

Codespaces permite que el código fuente sea ejecutable por cualquier persona haciendo uso de un navegador web y que de ese modo tenga un mayor alcance al momento de incrementar valor a los repositorios. También provee a los usuarios una imagen exacta del repositorio mediante un entorno docker que cuenta con herramientas definidas y una pila de tiempo de ejecución que hace referencia directamente al código fuente.

GitHub Actions

Según (Bartling, 2014b), las publicaciones científicas son la pieza angular de la ciencia, pero en el camino existen varios retos que superar y uno de ellos es el tecnológico. Sin embargo, la integración de nuevas tecnologías puede facilitar el proceso al evitar cambios de flujo de

trabajo bruscos. Plataformas modernas tales como GitHub y COS tienen herramientas robustas que hacen que la colaboración y el flujo de trabajo sean eficientes.

En la Fig. 13, se muestra como GitHub permite automatizar, personalizar y ejecutar flujos de trabajo de desarrollo de software directamente en repositorios con GitHub Actions de la mano con CI/CD que en materia DevOps se refiere a la Entrega e Integración continuas.

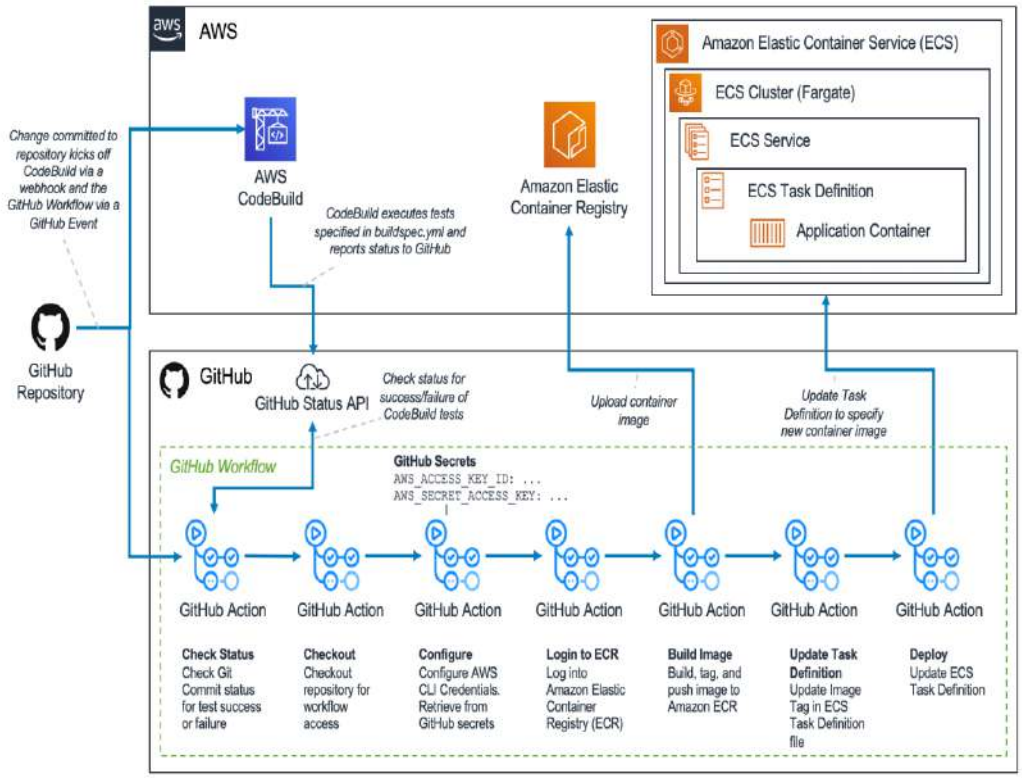


Fig. 13. GitHub Actions flujo de trabajo CI/CD en AWS.

Fuente: https://amzn.to/3tkM4nJ.

DevOps es un conjunto de técnicas de desarrollo de software ágiles y eficientes aplicadas durante todo el ciclo de vida de software. DevOps permite mitigar aquellas problemáticas desde un punto de vista operacional a las que se enfrentan las organizaciones en cuanto a sus actividades de TI, ingeniería de calidad y seguridad (Anastasov, 2021).

- Integración continua: Proceso que permite fusionar cambios de funcionalidades, métodos, clases, etc., en una aplicación de manera ágil, segura y eficiente.
- Entrega continua: Proceso que le sigue a CI, ya que permite automatizar la entrega de los cambios realizados en un determinado repositorio de código.
- Despliegue continuo: Proceso que garantiza y optimiza que los cambios alojados en un determinado repositorio de código son desplegados a uno o más ambientes de pruebas o producción.

1.5. ISO/IEC 25022:2016

La ISO/IEC 25022 es el estándar establecido por la ISO “International Organization for Standardization” que contiene las características, mediciones y métricas de calidad en uso que demuestran una dependencia directa con la calidad del producto. Por lo que, si una de ellas obtiene valores óptimos, entonces se puede determinar que la calidad del software es eficiente. (Nakai et al., 2016)

Tabla 4. ISO/IEC 25022:2016.

Fuente: https://www.iso.org/standard/35746.html.

Métricas de calidad en uso		
Características	**Subcaracterísticas**	**Métricas**
Efectividad	Efectividad	• Completitud de la tarea. • Efectividad de la tarea. • Frecuencia de error.
Eficiencia	Eficiencia	• Tiempo de la tarea. • Tiempo relativo de la tarea. • Eficiencia de la tarea. • Eficiencia relativa de la tarea. • Productividad Económica. • Porcentaje productivo. • Número relativo de las acciones del usuario.
Satisfacción	Utilidad	• Nivel de satisfacción. • Uso discrecional de las funciones. • Porcentaje de quejas de los clientes.
Libertad de Riesgo	Libertad de riesgo económico	• Retorno de la inversión (ROI). • Tiempo para lograr el (ROI). • Rendimiento relativo de negocios. • Balanced Score Card. • Tiempo de entrega. • Ganancias para cada cliente. • Errores con consecuencias económicas. • Corrupción del software.
	Libertad de riesgo de salud y seguridad	• Frecuencia de problemas en la salud y seguridad del usuario. • Impacto de la salud y la seguridad del usuario. • Seguridad de las personas afectadas por el uso del sistema.
	Libertad de riesgo ambiental	• Impacto Ambiental.
Cobertura del Contexto	Completitud de contexto	• Completitud del contexto.
	Flexibilidad	• Función flexible del diseño.

La calidad en uso está estrechamente relacionada con el comportamiento del sistema mientras este efectúa la ejecución de un determinado proceso. Por lo tanto, el valor obtenido de una prueba de aceptación buscará cuantificar la usabilidad a través de las subcaracterísticas cuando un usuario haya culminado un determinado objetivo (Estdale & Georgiadou, 2018).

Heurísticas de Usabilidad de Jakob Nielsen

(J. Nielsen, 1994) considerado el padre de la usabilidad web establece heurísticas o principios que deben ser cumplidos para que el desarrollo de un determinado software logre obtener

productos que tengan un mayor nivel de aceptación en los usuarios. A continuación, se detallan dichas heurísticas de usabilidad:

1. Visibilidad del estado del sistema: Un sistema (web, aplicación o cualquier otro producto que necesite interactuar con un dispositivo electrónico) debe mantener informado al usuario de lo que está ocurriendo, pues de no ser así cometerá errores con una mayor tasa de frecuencia.
2. Relación entre el sistema y el mundo real: El diseño debe hablar el idioma de los usuarios. Por ello hay que usar terminologías del entorno local y no jerga técnica, y de ser necesario deberá ir acompañada de aclaraciones.
3. Control y libertad de Usuario: El usuario debe ser capaz de salir de cualquier percance o error de manera rápida, y sin contratiempos.
4. Coherencia y estándares: No sobrecargar la coherencia cognitiva es importante, no es necesario usar términos que obliguen al usuario a aprender algo nuevo con frecuencia.
5. Prevención de errores: Los mensajes de error son importantes, pero es mejor evitarlos o de ser necesario se debe descartar las condicionales innecesarias que puedan dar un error al ejecutar una determinada tarea.
6. Reconocimiento antes de recordar: Es necesario hacer visibles las acciones y opciones del sistema, así se puede evitar que el usuario tenga que recordar información de diversas secciones.
7. Flexibilidad y eficiencia de uso: Es importante crear atajos o enlaces que permitan acceder de manera rápida a la información, de tal modo la curva de aprendizaje se vuelve eficiente tanto para usuarios expertos y novatos.
8. Estética y diseño minimalista: Las secciones de contenido no deben tener información innecesaria, ya que aglomerar cierta información irrelevante puede contrastar a la importante.
9. Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de errores: Los mensajes de error deben tener un lenguaje claro e instrucciones precisas de aquello que está ocurriendo y como debe solucionarse.
10. Ayuda y documentación: El producto debe ser diseñado para que el usuario no necesite ayuda, pero en ciertos casos se debe proveer información cuando se trate de procesos complejos.

CAPÍTULO 2

DESARROLLO

2.1. Análisis, diseño y desarrollo de la arquitectura

2.1.1. Arquitecturas de Software

- Arquitectura orientada a servicios (SOA)

SOA es un estilo de arquitectura de tecnología de la información (TI) que sustenta su funcionamiento en piezas de software o componentes que proporcionan servicios individuales, en la que el acoplamiento flexible y la encapsulación es fundamental para su correcto funcionamiento. Sin embargo, SOA no incluye un estilo arquitectónico en la nube por lo que es necesario unirla con herramientas de computación en la nube para obtener ventajas, tales como costos e independencia de ubicación (Bokhari et al., 2015).

Una arquitectura orientada a servicios debe cumplir con los siguientes requisitos para ser considerada como tal:

1. Flexibilidad: Debe ser flexible tal que permite la reutilización de sus recursos en otras instancias o servicios basados en diversas plataformas.
2. Versatilidad: Debe se versátil tal que haga posible que los servicios puedan ser consumidos por clientes y procesos distintos.
3. Escalabilidad: Permiten la escalabilidad continua de tal modo que es redituable el realizar nuevos cambios en cuanto al seguimiento continuo de todos los componentes que conforman la estructura arquitectónica.
4. Granularidad: Debe tener un nivel alto de granularidad en todos los complementos que lo conforman buscando coherencia en todos sus componentes.
5. Disponibilidad: Debe garantizar alta disponibilidad de los servicios.
6. Rendimiento: Al ser una arquitectura en la cual varios servicios funcionan independientemente el uno del otro. Es decir, cada servicio debe tener el rendimiento esperado, sin ambigüedades de uso u otro factor técnico.
7. Segura: Debe generar una comunicación segura de los usuarios y todos los componentes que la conforman, evitando que personas sin acceso manipulen la información.

En SOA existen cuatro componentes básicos que son: proveedor de servicios, corredor de servicios, registro de servicios y consumidor de los servicios. Dicho eso se definen tres roles encargados de manejar cada componente: proveedor, bróker y solicitante, ver Fig. 14.

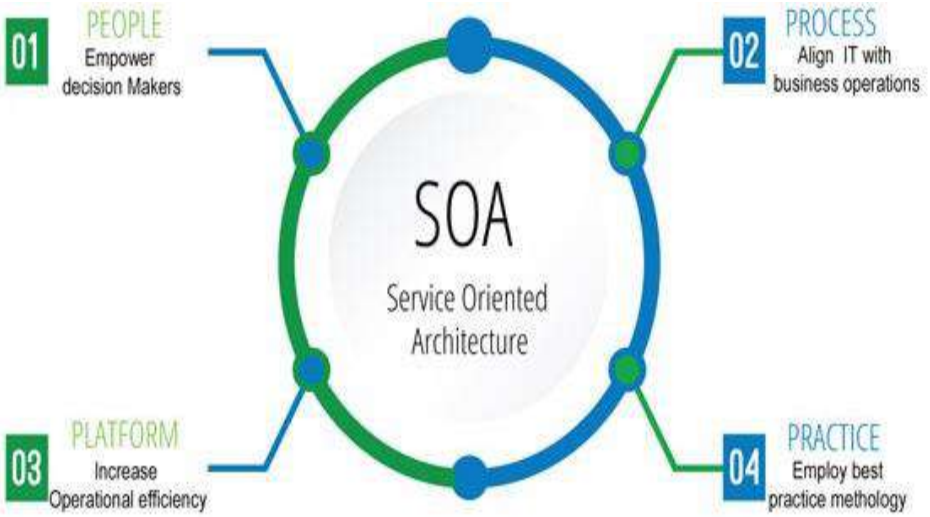


Fig. 14. Patrones de Arquitectura SOA.

Fuente: https://bit.ly/3jn4RLp.

Reutilizar Integraciones

La diferencia entre una arquitectura orientada a servicios y una arquitectura orientada a microservicios se reduce al alcance, pues una arquitectura SOA tiene un alcance empresarial mientras que una arquitectura orientada a microservicios tiene un alcance de automatización por servicio. Se hace evidente que si se pretende reutilizar integraciones de entornos grandes y diversos; SOA es la arquitectura ideal para aplicaciones heterogéneas. (IBM, 2021)

ESB

Un ESB o Enterprise Service Bus en español Bus de Servicio Empresarial, es un patrón arquitectónico centralizado el cual posee la facultad de realizar integraciones entre aplicaciones. Se encarga de realizar transformaciones de modelos de datos, maneja conectividad, enrutamiento de mensajes, conversión de protocolos de comunicación y potencialmente administra la composición de diversas solicitudes a servicios externos.

- Arquitectura híbrida en la nube

La nuble híbrida es una arquitectura de tecnología de la información (TI) que está compuesta de cierto grado de gestión, organización y portabilidad de las cargas de trabajo en dos o más entornos de desarrollo y de producción que abarcan: proveedores de IaaS (Infraestructura as a Service), servicios de cómputo como AWS o Azure, almacenamiento y bases de datos (Urbina et al., 2016).

Para que una arquitectura sea considerada híbrida debe cumplir con los siguientes enunciados:

1. Existencia de al menos una nube privada y una pública.
2. Flujo de información de dos o más nubes privadas.
3. Flujo de información de dos o más nubes públicas.
4. Entorno virtual o sin sistema operativo conectado al menos a una nube pública o privada.

Según (Dixit & chhabra, 2015), una arquitectura híbrida en la nube representa modelos de despliegue de los servicios que se encuentran en:

1. Nube Pública: Es aquella que utiliza servicios en la nube publicados y usados en la red, dicha nube presenta problemas de seguridad, tales como usuarios no autorizados que pueden acceder a información compartida en la nube.
2. Nube Privada: Es aquella que está dedicada a una determinada organización que hace uso de su propia infraestructura, que es de uso exclusivo para la organización evitando así problemas de seguridad.
3. Nube Híbrida: Es la unión de dos o más nubes (privada, pública o compartida) que pertenecen a organizaciones únicas, pero que son conectadas a la vez por tecnologías estandarizadas que permiten la portabilidad de los datos y aplicación.

2.1.2. Estilos de desarrollo de Software

- Modelo Vista Controlador (MVC)

MVC es un estilo de arquitectura de software que permite construir software; es adaptable a cualquier tipo de entorno de programación, lenguaje o Framework de Desarrollo. MVC fue discutido y posteriormente aplicado en la década de los 70 en Smalltalk-76 (Jailia et al., 2016).

MVC tiene tres componentes principales que son, ver Fig. 15:

a) Modelo: Es la capa que se encarga de abarcar los datos del software misma que contiene la implicación y definición de la lógica del negocio, dicho elemento puede ser uno o varios objetos que permitirán la comunicación con los datos.
b) Vista: Es la capa de cara al usuario final, misma que debe ser intuitiva al mostrar los atributos de los objetos, se encarga de mostrar los datos que el usuario requiera manipular manteniendo la integridad de la información.
c) Controlador: Es la capa que permite la interacción de los objetos del Modelo y los atributos de la Vista, dicho elemento permite un flujo constante de información mientras la aplicación se ejecuta; ya que al cambiar algo el en modelo, la fuente de datos se ve afectada arrojando inmediatamente los cambios en la vista.

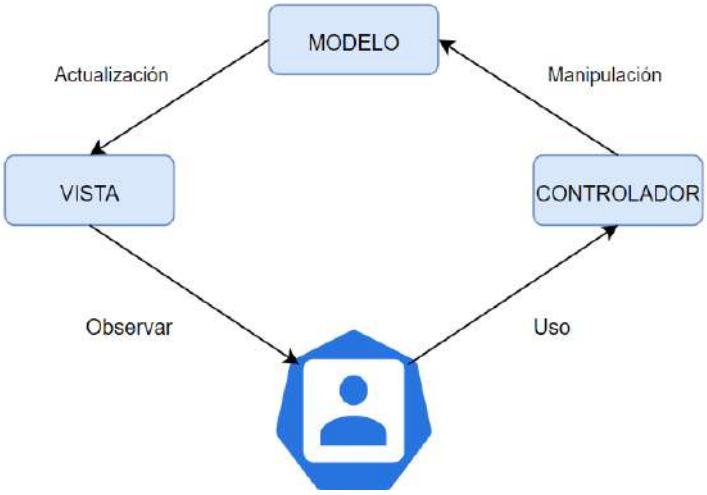


Fig. 15. Model - View - Controller Architecture.

Fuente: Propia.

- Flutter Arquitectura ScopedModel

ScopedModel es el estilo arquitectónico para aplicaciones nativas por excelencia, pues a diferencia de otras arquitecturas establecidas, el estilo permite dos métodos de implementación mediante un inicio rápido y un punto de partida que sea sencillo de seguir, ya que cada Vista de la aplicación tendrá un Modelo raíz.

Según (Mackier, 2019), la arquitectura mencionada tiene dos métodos de implementación, en ambos modelos los datos interactúan con los servicios que se encargan de realizar todas las tareas solicitadas; el método aplicado a la presente arquitectura global SOA es:

a) One AppModel with FeatureModel mixins: Es el método mediante el cual un Modelo extiende Mixins (reutilización de clases en múltiples jerarquías de código fuente), que también tiene el estilo de un Modelo con la diferencia que éstos abarcan piezas lógicas relacionadas para acceder de manera global, ver Fig. 16.

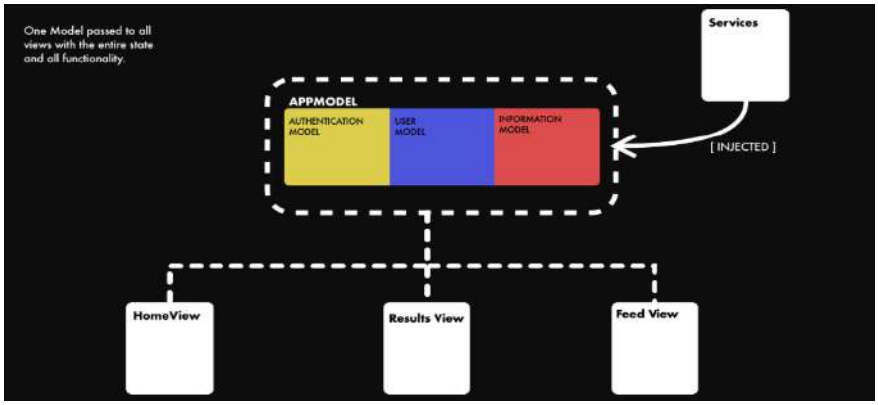


Fig. 16. One AppModel with FeatureModel mixins.

Fuente: https://bit.ly/2XY64AF.

2.1.3. Metodología SCRUM

Scrum es un marco de trabajo que permite desarrollar productos de una manera ágil y obtener el mejor resultado posible, es aplicado para planificar la automatización de un determinado proceso. En la actualidad ha sido adoptado y adaptado por organizaciones que distribuyen soluciones de software a nivel nacional e internacional (Faniran et al., 2017).

Scrum está compuesto por las siguientes actividades que permiten cumplir flujos de trabajo que se adaptan a las circunstancias, requerimientos y alcance de un determinado producto:

a) Backlog: Es el elemento que se encarga de la revisión de todas las tareas que deberá cumplir el equipo de trabajo.
b) Sprint: Es el parámetro de tiempo en el que el equipo de trabajo debe realizar las tareas encomendadas para generar valor al producto.
c) Planificación Sprint: Son las reuniones que se realizan para delimitar y establecer lo que se hará en el Sprint.
d) Daily Standup: Reunión rápida en la cual se controla aquello que se realizó, de tal modo que sea posible solventar posibles contratiempos y errores.
e) Revisión Sprint: Reunión en la que se comprueba que los Sprints hayan sido culminados correctamente, usualmente se lo realiza al comprobar con un software demo.
f) Retrospectiva del Sprint: Para dar por terminado el Sprint se realiza una reunión en la cual se genera una retrospectiva de aquello que ha salido bien y lo que es necesario mejorar en el producto.

Scrum está compuesto por varios artefactos que ayudan a llevar a cabo el proyecto de la mejor manera:

a) Product Backlog: Se refiere a todas características, correcciones, mejoras, requisitos y requerimientos que se deben realizar para cada uno de los elementos entregables del producto.
b) Sprint Backlog: Son aquellas tareas que se realizarán en el Sprint y que son seleccionadas desde el Product Backlog. Las tareas incluyen a detalle las actividades necesarias para lograr los objetivos del Sprint, además son adaptables y se pueden modificar durante el desarrollo del producto.
c) Incremento: Es el resultado que se obtiene al final de cada Sprint, representa la información de los Sprints pasados y las tareas del Product Backlog que han sido realizados.

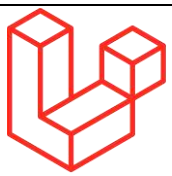
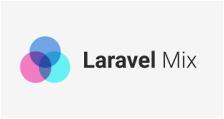
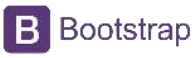
Scrum está compuesto por los siguientes cargos:

a) Product Owner: Es la persona con experiencia y solvencia suficiente como para solucionar contratiempos que pueden ocurrir al llevar a cabo la automatización de procesos. También se encarga de gestionar el personal de manera eficiente por medio de reuniones.
b) Scrum Master: Es la persona que conoce, gestiona, prioriza el proceso y todas las personas que están involucradas en él. Optimiza los tiempos para que el equipo de trabajo logre cumplir cada Sprint.
c) Development Team: Es la persona o personas que están involucrados estrechamente en el desarrollo del proyecto, aportan valor al producto al cumplir las tareas encomendadas en cada Sprint.

2.1.4. Herramientas de trabajo

Tabla 5. Matriz de Herramientas.

Logo	Nombre	Descripción
Diseño, Multimedia, Edición.		
	digiKam	digiKam es una herramienta de software para la gestión de fotografías, logos y demás objetos usados en el desarrollo web en la capa Frontend (DigiKam, 2021).
	Adobe Xd	Adobe Xd es una herramienta de software la cual permite generar y emplear el uso de diseños de interfaces de usuario en el desarrollo de aplicaciones informáticas (Adobe, 2021).
Lenguajes de Programación, Framework, SDK, Librerías.		
	Vue.js	Vue.js es un Framework de JavaScript progresivo, el cual es usado para construir interfaces y SPA (Single Page Apps) de manera muy sencilla y eficiente. Provee grandes ventajas cuando de integración con otras herramientas de desarrollo se trata (VueJs, 2021).
	React	React es una biblioteca de JavaScript que permite hacer interfaces de usuario dinámicas e implementarlas de manera eficiente a medida que un proyecto incrementa su volumen. React permite crear vistas simples para cada estado de una aplicación, ya que se encarga de actualizar y renderizar componentes reutilizables.
	JavaScript	JavaScript es un lenguaje de programación interpretado en tiempo real, mismo que permite generar respuestas y peticiones de la mano de APIs y DOM. Generalmente es aplicado tanto del lado de cliente y el servidor

	Laravel	Es el Framework que basa su sintaxis en PHP, contiene instrucciones sencillas de implementar y posee una gran variedad de herramientas que permiten la integración de paquetes de software para generar aplicaciones web modernas y complejas (Laravel, 2021).
	LaravelMix	LaravelMix es un paquete de módulos que prepara un ambiente ideal para la ejecución de JavaScript Moderno ya que activa un preprocesador de CSS y es una herramienta ideal para trabajar de la mano con Livewire, Vue.
	TypeScript	Es un lenguaje de programación de código abierto que contiene un conjunto de herramientas para asegurar la mutabilidad de código fuente, y brindar una estructura eficiente y mantenible en un proyecto de software (*TypeScript Documentation*, 2021).
	YAML	Es un lenguaje de serialización de datos que se usa para ficheros de configuración, es popular porque es legible y sencillo de entender. Generalmente es usado para crear procesos de automatización e implementación de recursos en Docker y Kubernetes.
	Node.js	Node.js es un entorno de ejecución de JavaScript orientado a ejecutar eventos asíncronos y está diseñado para crear aplicaciones escalables (*NodeJS Documentation*, 2021).
	Flutter	Flutter es un SDK que permite la creación de aplicaciones nativas mediante el uso de widgets que se adaptan a entornos iOS y Android. El presente SDK proporciona a los desarrolladores vistas de estilo reactivo que evitan problemas de rendimiento derivados del uso de otros lenguajes (*Flutter Docs*, 2021).
	Dart	Dart es un lenguaje optimizado para desarrollar aplicaciones veloces en cualquier plataforma y su objetivo es generar herramientas para desarrollar software multiplataforma de manera productiva y flexible (*Dart Overview*, 2021).
	MySQL	MySQL es una de las tecnologías más reconocidas en el ecosistema moderno de Big Data, suele ser llamada la base de datos más popular. Actualmente disfruta de un uso generalizado y efectivo en la industria de las ciencias computacionales.
	PromQL	PromQL, abreviatura de Prometheus Querying Language, es el método principal para realizar consultas de métricas dentro de Prometheus. Puede mostrar el rendimiento de una expresión como un gráfico o exportarlo mediante una API.
	Node Package Manager	Es el sistema para la gestión de paquetes del entorno de ejecución Node.js, dicha herramienta proporciona una consola de comandos que permite interactuar con proyectos durante toda la etapa de desarrollo.
	Bootstrap	Es una biblioteca de componentes el cual abarca diseño web, plantillas web y estilos de código abierto aplicados al desarrollo de software.

Servicios Web, DevOps, Contenedores, Monitoreo Continuo TI.		
	AWS	Amazon Web Services (AWS) es la plataforma en la nube más adoptada y completa en el mundo, que ofrece más de 200 servicios integrales de centros de datos a nivel global. Dia a día sus servicios son adoptados por millones de usuarios y las empresas emergentes que crecen más rápido. Las compañías más grandes y los organismos gubernamentales, usan AWS para reducir los costos, aumentar su agilidad e innovar de forma más rápida (*Amazon Web Services - Documentation*, 2021).
	Firebase	Firebase es una plataforma de desarrollo que provee la integración de varios servicios en la nube de Google. De la mano con Flutter integra widgets sofisticados de interfaz de usuario, además permite a los desarrolladores desplegar aplicaciones de manera rápida y sencilla ya que Firebase se encarga de replicar datos y enviar notificaciones de eventos.
	Docker	Docker es un proyecto de código abierto que permite la creación y uso de contenedores Linux en máquinas virtuales extremadamente livianas y escalables. Asimismo, elimina tareas de configuración repetitivas haciendo posible que el desarrollo de software sea rápido, fácil y portable. Docker contiene UI, CLI y API que permiten ser integradas en todo el ciclo de vida de entrega de aplicaciones (*Docker Documentation*, 2021).
	Grafana	Grafana es una solución de código abierto para ejecutar análisis de datos, extraer métricas que dan sentido a datos de rendimiento y para monitorear aplicaciones con la ayuda de paneles de control modernos. Grafana es usado por PayPal, eBay, Intel y otras grandes empresas; pues ayuda a rastrear el comportamiento del usuario y la aplicación, analizar y monitorear datos complejos (Grafana, 2021).
	Prometheus	Prometheus es uno de los muchos proyectos de código abierto gestionados por la Cloud Native Computing Foundation (CNCF). Es un software de monitoreo que se integra con una amplia gama de sistemas de forma nativa o mediante el uso de complementos. También posee funciones poderosas para monitorear métricas y proporcionar alertas que pueden ayudar a gestionar contenedores, por ejemplo, Kubernetes o Docker.
	GitKraken	Es una herramienta usada para la gestión de repositorios en Git, dicha herramienta proporciona una interfaz de usuario amigable e intuitiva para realizar un control de versiones eficiente mediante el uso de CLI, API y Paneles de Control (*GitKraken Documentation*, 2021).
	Postman	Es una herramienta usada para el desarrollo, pruebas, verificación de API (Aplication Programming Interface), dicha herramienta ofrece una elegante interfaz de usuario con la que realizar solicitudes HTML, sin la necesidad de escribir un código para testear servicios web api.

2.1.5. Configuración de las plataformas OSF y GitHub

El presente proyecto hace uso de tecnologías estables y frecuentemente aplicadas en entornos de desarrollo escalables; teniendo en cuenta lo anteriormente mencionado se establecieron los siguientes componentes que conforman la arquitectura global:

- Aplicación Orientada a Servicios (SOA).
- Aplicación Nativa (ScopedModel).
- Repositorios OSF con OAuth Token y acceso a propiedades de desarrollador (Servicios en la nube pública OSF).
- Clúster en Amazon con Linux Server, Docker, Load Balancer y otros servicios.
- Google Cloud, nube privada que será usada para almacenar los datos y componentes producto de la investigación.
- Contenedor Docker para Codespaces en la nube privada de GitHub.
- Repositorios en GitHub con Webhooks, Actions, Access Token y otras configuraciones para establecer la capa en la nube.
- Nubes publicas Mendeley, S3 Storage y Protocols.io.

2.1.5.1. Configuración del repositorio OSF

Configuraciones para desarrollador en OSF

La API REST de OSF tiene varias políticas de seguridad establecidas y es necesario crear un acceso con un token con los permisos necesarios para manejar los repositorios y todos los componentes que se le asocian, tal como se muestra a continuación, ver Fig. 17.

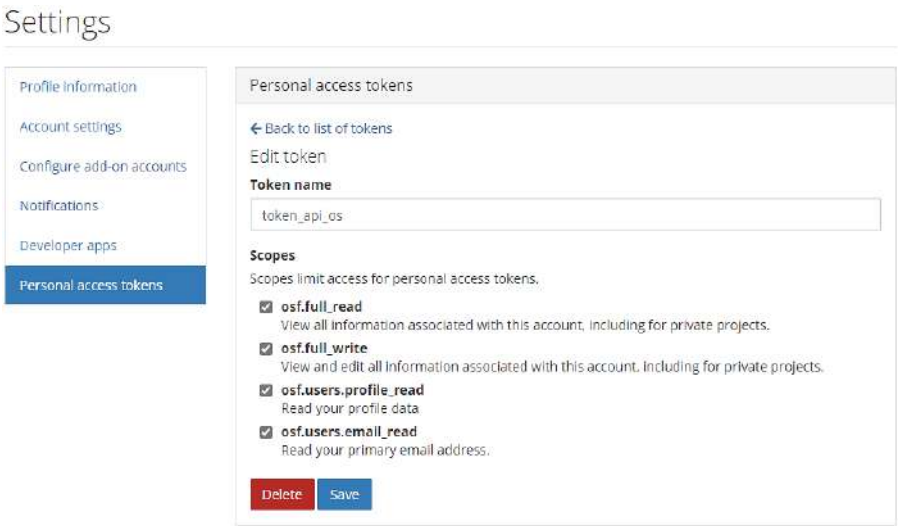


Fig. 17. Personal Access Token OSF.

Fuente: Propia.

Repositorio – Integration GitHub & OSF

En la Fig. 18 se muestra la creación del repositorio OSF en donde se alojarán los contenidos abiertos con el nombre inicial “Integration GitHub & OSF”. La licencia que se aplicó al repositorio fue CC-BY Attribution 4.0 International, misma que se recomienda para proyectos Open Science que tendrán varios contenidos que serán de acceso privado y otros que no lo serán hasta que el proyecto haya concluido por completo.

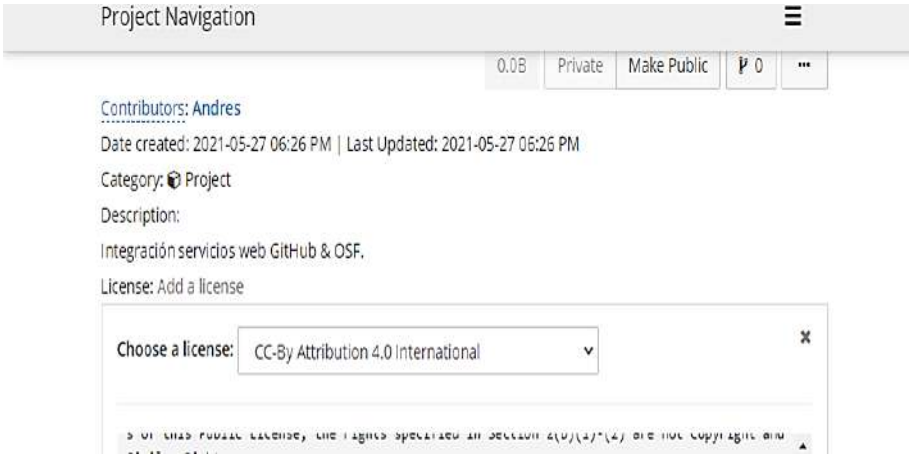


Fig. 18. Creación del Repositorio OSF, asignación Creative Common.

Fuente: Propia.

2.1.5.2. Configuración de repositorios GitHub

Configuraciones para desarrollador en GitHub

En la Fig. 19, se muestra la configuración para desarrolladores que establece OAuth Apps y Personal Access Tokens. Los permisos que le fueron asignados al token son de lectura y escritura en organizaciones, repositorios, configuraciones de perfil y otros.

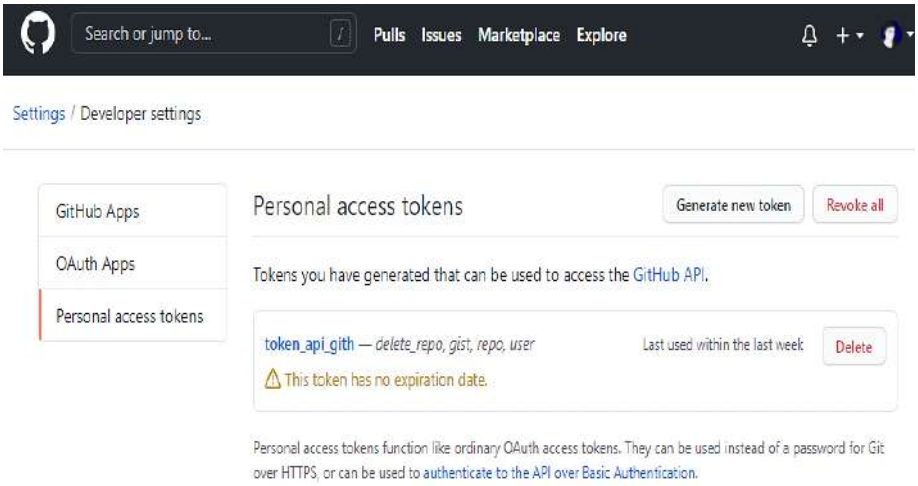


Fig. 19. Configuración de Personal Access Token en GitHub.

Fuente: Propia.

Para administrar el acceso a GitHub desde la aplicación se configuró OAuth Apps que proporciona un ID que contiene un identificador único en GitHub y un Client Secret. Los datos obtenidos a través de la autenticación de GitHub permitirán sincronizar los componentes de la arquitectura y los aplicativos desplegados en producción, permitiendo así un tráfico seguro y eficiente de la información mediante un OAuth App con el nombre de “Integration GitHub & OSF”, como se muestra en la Fig. 20.

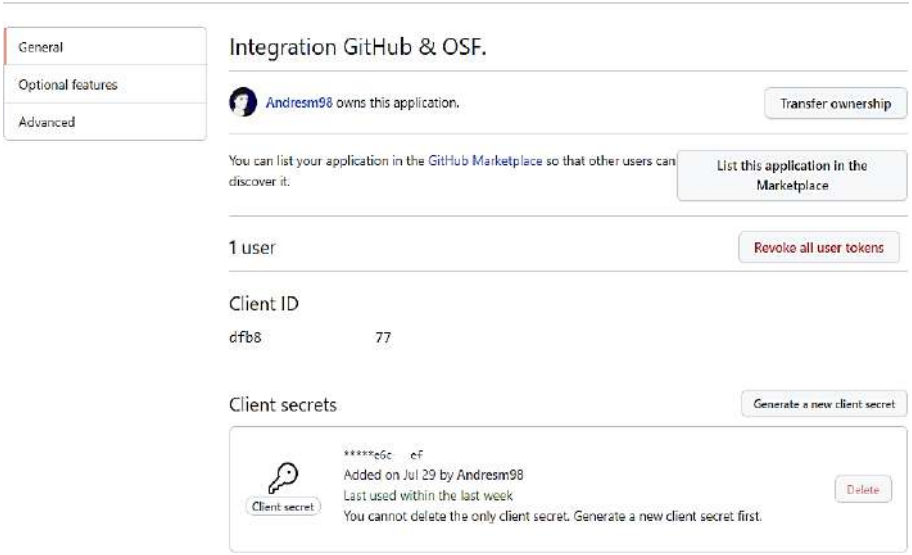


Fig. 20. Configuración OAuth Apps en GitHub.

Fuente: Propia.

En la Fig. 21, se puede visualizar el consumo de las APIs GitHub y OSF mediante Postman.

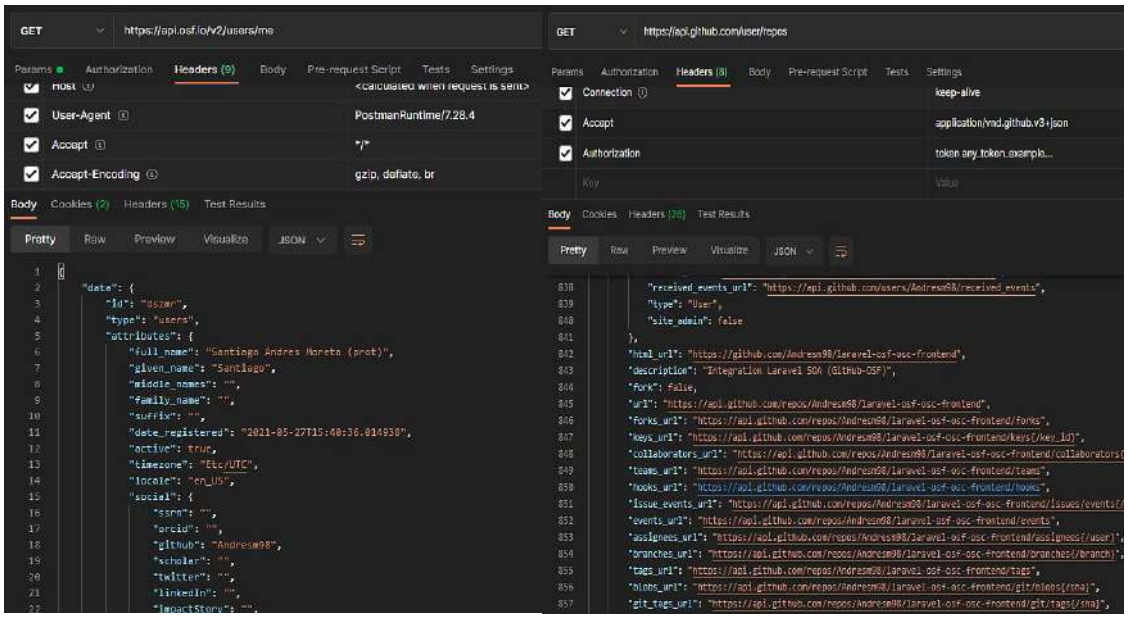


Fig. 21. Consumo API REST de GitHub y OSF, Postman.

Fuente: Propia.

Repositorios GitHub

Repositorio – Integration Laravel SOA

En la Fig. 22, se muestra el ambiente de desarrollo basado en la arquitectura SOA, se creó un repositorio con el nombre “laravel-osf-osc-frontend” en la plataforma GitHub.

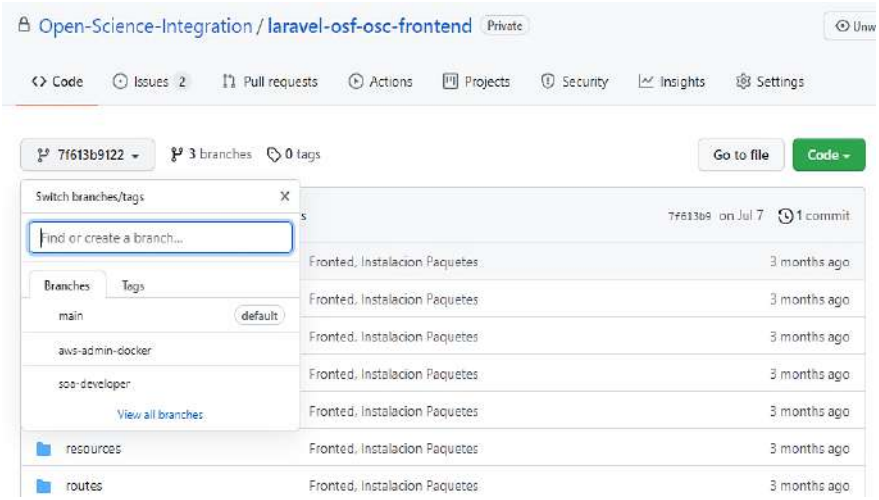


Fig. 22. GitHub Repositorio Laravel Etapa Inicial, repositorio Nro.1.

Fuente: Propia.

En la Fig. 23, se muestra las credenciales para llevar a cabo tareas CI/CD con las herramientas de GitHub, cabe mencionar que se levantaron servicios previamente en AWS, misma en la que se encuentra ejecutándose una máquina virtual basada en Linux que se hará el despliegue a producción del aplicativo haciendo uso de las Acciones del repositorio.

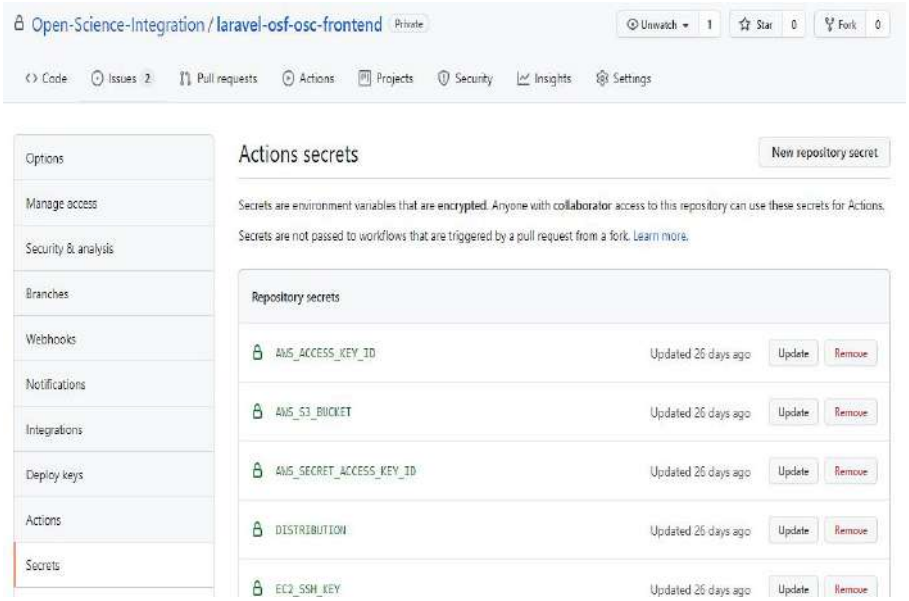


Fig. 23. Configuración en GitHub bajo CI/CD, repositorio Nro.1.

Fuente: Propia.

Para que los eventos sean sincronizados correctamente, se generaron Webhooks de los servicios propios de OSF con ello será posible hacer las peticiones a los servicios necesarios, ver Fig. 24.

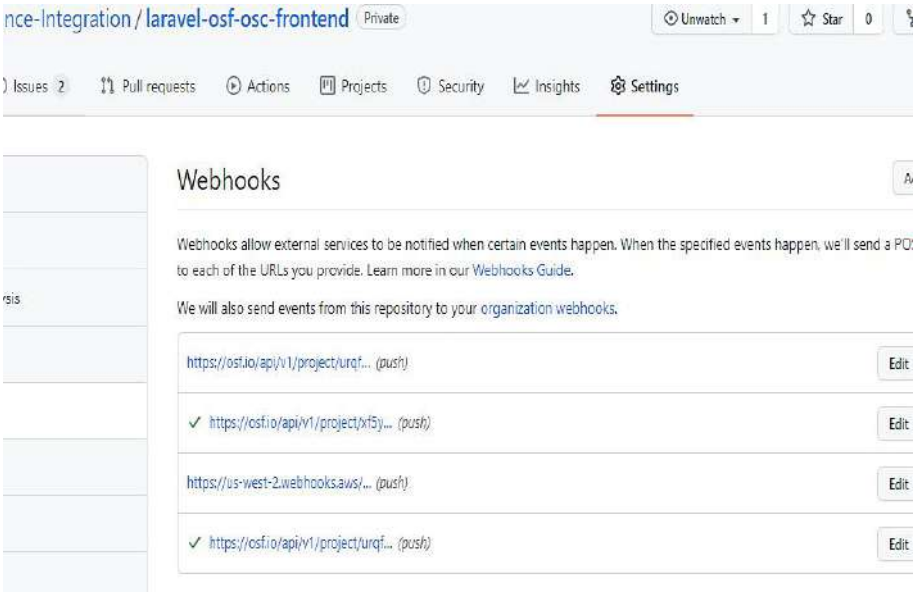


Fig. 24. Configuración de Webhooks repositorio Nro. 1.

Fuente: Propia.

Repositorio – Integration Flutter Native App

El repositorio con la aplicación nativa en Flutter fue desplegado de la misma manera que el repositorio Nro.1 y se le asignó el nombre “flutter_osf_osc_platform_api”, se realizaron las configuraciones previamente mencionadas en la plataforma Firebase, ver Fig. 25.

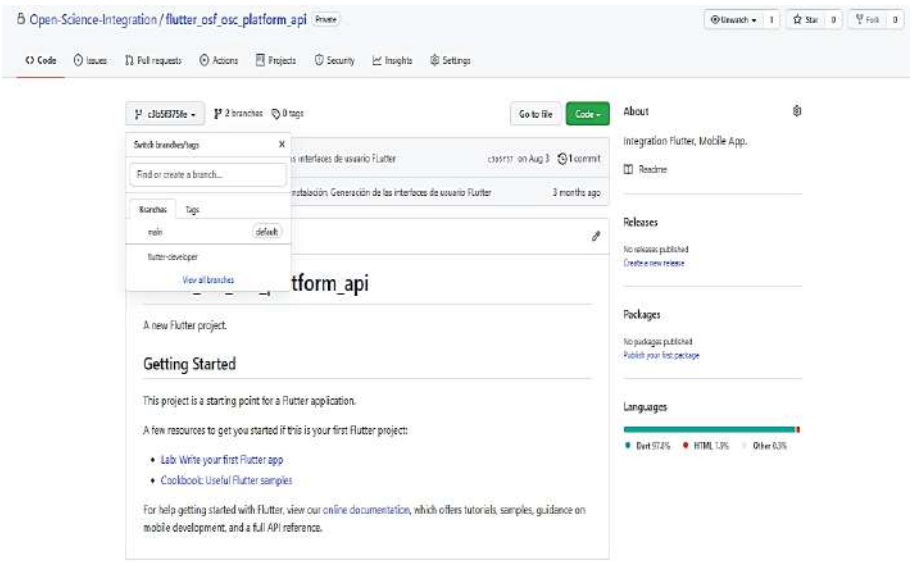


Fig. 25. GitHub Repositorio Flutter Etapa Inicial, repositorio Nro.2.

Fuente: Propia.

Repositorio – Integration React App

El repositorio con la aplicación para almetrics fue desplegado de la misma manera que los repositorios anteriores y se le asignó el nombre “react-osf-osc-ai-altmetrics”, se agregaron dos orígenes: GitHub (versionamiento) y Heroku (despliegue para pruebas), ver Fig. 26.

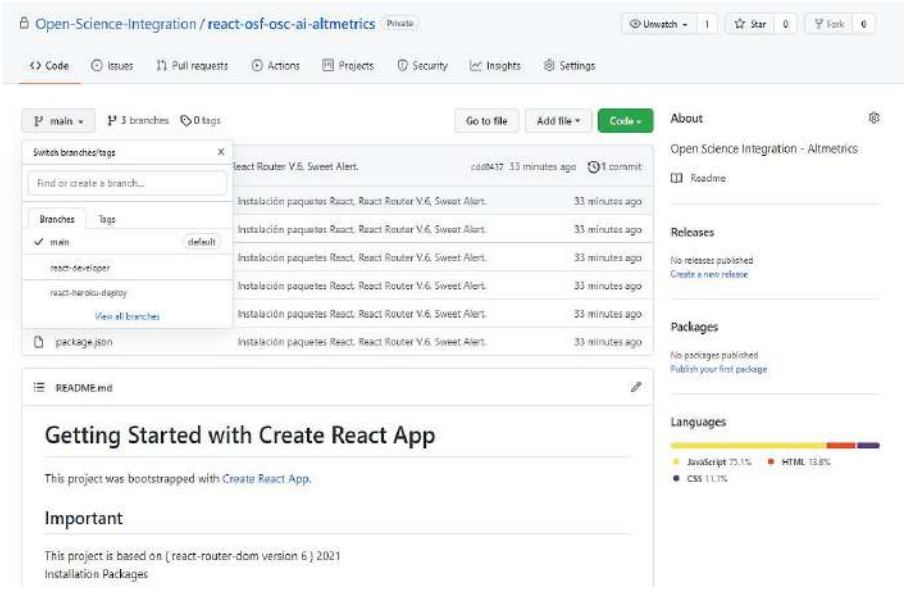


Fig. 26. GitHub Repositorio React Etapa Inicial, repositorio Nro.3.

Fuente: Propia.

Organización GitHub

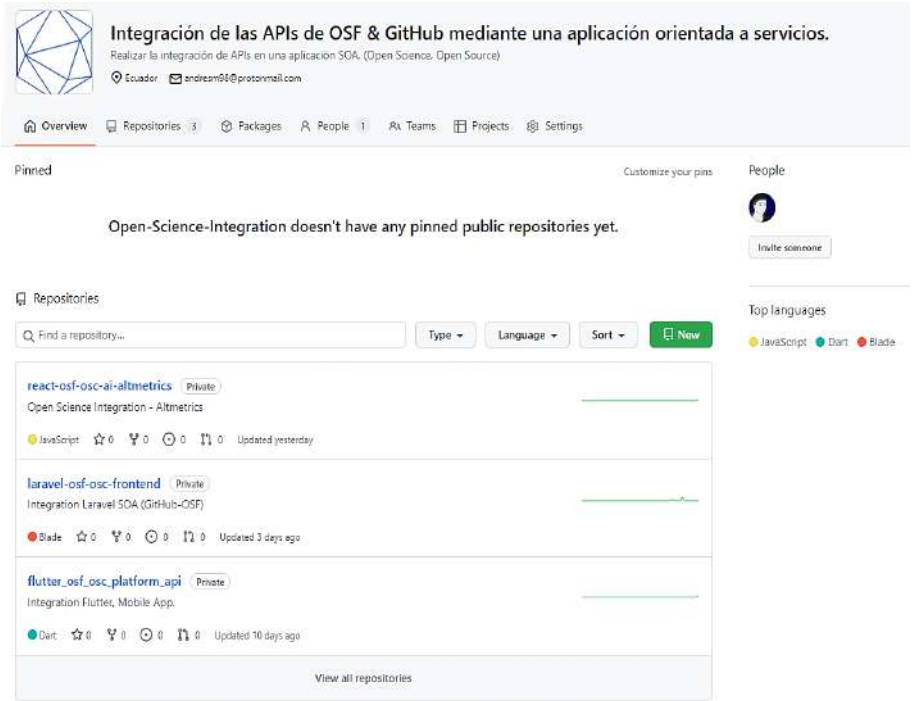


Fig. 27. Organización GitHub, Open Science Integration.

Fuente: Propia.

2.2. Desarrollo

2.2.1. Sprint 0

En el presente proyecto se hará uso de la plataforma OSF en donde se desplegarán todos los contenidos que se vayan generando en toda la etapa de desarrollo de cada uno de los componentes, técnicas, tecnologías, lenguajes de programación, etc., los componentes al inicio serán restringidos ya que se encontrarán en etapa de desarrollo durante todo el proceso de implementación de la arquitectura orientada a servicios. Componentes tales como el código fuente, lenguajes de programación, librerías y técnicas empleadas en el desarrollo serán ordenadas en estructuras ordenadas como lo recomienda el Framework OSF.

El análisis y posterior desarrollo de la arquitectura para publicar contenido Open Science necesita llevar a cabo tareas; para que dichas tareas sean ejecutadas correctamente se aplicó la metodología Scrum y el uso herramientas que fueron detalladas en la Tabla 5.

2.2.1.1. Definición Roles Scrum

Tabla 6. 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 “Gobierno de arquitecturas híbridas REST y GraphQL mediante contratos”.
Scrum Master	MSc. Cathy Guevara	Persona que tiene la facultad de entender y dar las pautas a poner en práctica en las reglas, puntos e hitos del marco de trabajo Scrum, para que de tal forma el producto sea desarrollado de la mejor manera.
Development Team	Santiago Andres Moreta	Es el encargado de desarrollar todas las tareas necesarias que logren incrementar el valor del producto.

2.2.1.2. Historias Épicas

El término “epic” es aplicado a Scrum cuando se pretende conseguir resultados basados en diversas directrices que generen mayor valor al producto final. En la Tabla 7 se muestra la definición de dos historias épicas que deben ser cumplidas simultáneamente.

Tabla 7. Historias Épicas.

Código	Título	Prioridad
HE.01	Desarrollar aplicación orientada a servicios para publicar contenido Open Science, arquitectura SOA.	Alta
HE.02	Integrar las plataformas OSF y GitHub en el aplicativo. Desarrollar aplicación nativa en Flutter con la arquitectura ScopedModel.	Alta

2.2.1.3. Product Backlog

Las historias épicas definidas anteriormente fueron divididas en varias historias de usuario que permitan cumplir con los requerimientos funcionales y no funcionales, para ello se definió la prioridad y el peso. En la Tabla 7, se muestran dos historias épicas que deben ser sintetizadas en historias de usuario para construir los componentes de la arquitectura.

Scrum al ser una metodología ágil permite optimizar tiempos de mejor manera, por ello para cada Sprint se establecieron tres semanas aproximadamente, quedando como resultado el siguiente Product Backlog, ver Tabla 8.

Tabla 8. Product Backlog.

Historias de Usuario				
H.Épica	**Código**	**Título**	**Peso**	**Prioridad**
HE.01	OSI.01	Preparar el ambiente de trabajo OSF y GitHub.	1	Alta
HE.01	OSI.02	Desarrollar el modelo lógico de la base de datos.	2	Alta
HE.01	OSI.03	Diseño de las interfaces de usuario en Adobe XD.	2	Alta
HE.01	OSI.04	Diseño de la arquitectura orientada a servicios.	2	Alta
HE.01	OSI.05	Levantar el ambiente de desarrollo para el aplicativo, generar GitHub Actions CI/CD.	1	Alta
HE.01	OSI.06	Levantar el ambiente de desarrollo en el servidor Amazon Web Services con Docker en EC2.	2	Alta
HE.01	OSI.07	Implementar las librerías propias de Laravel.	1	Alta
HE.01	OSI.08	Implementar las entidades por defecto en el proyecto. (migrations, users, etc).	1	Alta
HE.01	OSI.09	Implementar las entidades de Usuario y Login.	2	Alta
HE.01	OSI.10	Instalar las dependencias Laravel y Vue, iniciar las interfaces de usuario.	2	Alta
HE.01	OSI.11	Implementar las entidades de permisos y roles del sistema.	1	Alta
HE.01	OSI.12	Implementar las entidades de los contenidos abiertos.	1	Media
HE.01	OSI.13	Implementar la entidad contenidos.	1	Media
HE.01	OSI.14	Implementar la entidad objetivos-contenidos abiertos.	1	Media
HE.01	OSI.15	Implementar la entidad información-contenidos abiertos.	1	Media
HE.01	OSI.16	Implementar la entidad requerimientos-contenidos abiertos.	1	Media
HE.01	OSI.17	Implementar la entidad de artículos.	1	Media
HE.01	OSI.18	Implementar la entidad de progreso de contenidos abiertos.	1	Alta
HE.01	OSI.19	Implementar la entidad de profesor.	1	Alta
HE.01	OSI.20	Implementar la entidad de estudiantes-contenidos abiertos.	1	Alta
HE.01	OSI.21	Implementar la entidad de almacenamiento de datos Open Science.	1	Alta
HE.01	OSI.22	Implementar las entidades de relación: contenido Open Science, archivos Open Science, Log del Contenido, Categorías del Contenido Abierto.	2	Alta
HE.01	OSI.23	Implementar la entidad creative common.	2	Alta
HE.02	OSI.24	Al ser usuario administrador, es posible acceder a todos los componentes, repositorios, documentos de investigación.	2	Media
HE.02	OSI.25	Al ser usuario administrador, se puede gestionar los usuarios, licencias, categorías, niveles, creative commons, publicaciones, etc.	2	Media
HE.02	OSI.26	Al ser usuario administrador es posible aprobar contenido Open Science en la página principal.	2	Alta
HE.02	OSI.27	Al ser usuario docente es posible editar el perfil que mostrará a los estudiantes de sus contenidos abiertos.	2	Alta
HE.02	OSI.28	Al ser usuario docente es posible crear contenidos abiertos dada la categoría, nivel, licencias, etc.	2	Alta

HE.02	OSI.29	Al ser usuario docente es posible agregar un conjunto de contenido a partir de la temática.	2	Alta
HE.02	OSI.30	Al ser usuario docente es posible publicar el contenido a los alumnos.	2	Alta
HE.02	OSI.31	Al ser usuario estudiante es posible presentar matrícula a un determinado y hacer uso de las funcionalidades.	2	Alta
HE.02	OSI.32	Los datos que se consumen a través del API Backend, OSF y GitHub se reflejan automáticamente en el contenedor AWS.	1	Alta
HE.02	OSI.33	La aplicación orientada a servicios es subida al clúster de aplicaciones AWS con el script Docker y GitHub Actions.	1	Media
HE.02	OSI.34	Se instalan los paquetes necesarios para el aplicativo mediante el fichero yml.	1	Baja
HE.02	OSI.35	Integración de las entidades colecciones de OSF.	4	Media
HE.02	OSI.36	Integración de las entidades nodos de OSF.	4	Alta
HE.02	OSI.37	Integración de las entidades repositorios de OSF.	3	Alta
HE.02	OSI.38	Integración de las entidades citaciones, preimpresiones, registros OSF.	2	Alta
HE.02	OSI.39	Integración de elementos usuario OSF.	2	Media
HE.02	OSI.40	Gestionar componentes OSF y GitHub.	4	Alta

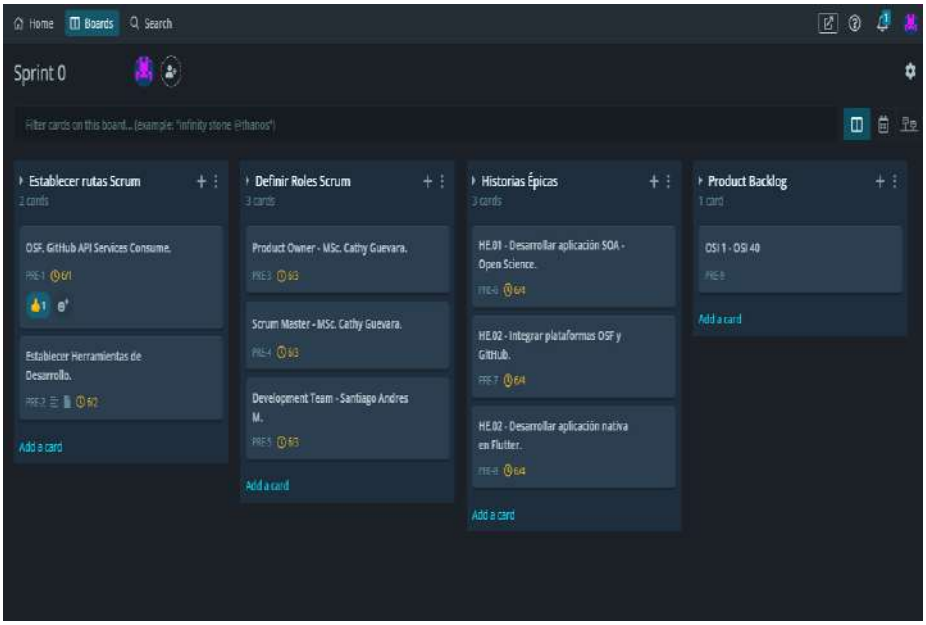


Fig. 28. GitKraken Board - Sprint 0.

Fuente: Propia.

2.2.2. Sprint 1

2.2.2.1. Sprint Planning

El propósito del Sprint número 1 fue dar inicio al diseño de la arquitectura de la aplicación y la base de datos que serán circunstanciales a la misma, se establecieron los siguientes objetivos:

- Describir la arquitectura y como sus componentes estarán relacionados.

- Diseñar la arquitectura global orientada a servicios, base de datos, arquitectura SOA, arquitectura en la nube y capas del sistema.
- Maquetar las interfaces del aplicativo (AdobeXd).
- Levantar los ambientes de trabajo y entornos de desarrollo para la arquitectura.
- Versionamiento continuo de los repositorios.
- Generar tokens de accesos y privilegios de desarrollador.

2.2.2.2. Sprint Backlog

Para el Sprint número 1, se tomaron en cuenta las historias de usuario que están descritas en el Anexo Nro. 1 en donde se definieron criterios de aceptación y tareas a realizar.

2.2.2.3. Ejecución del Sprint

Ambientes Orientados a Servicios

En cada uno de los ambientes del proyecto, tanto para OSF y GitHub fueron habilitados los servicios necesarios para realizar la integración de los repositorios. Los datos se encuentran sincronizados en el proyecto hasta su etapa final, se hizo hincapié en los principios Open Science, los cuales establecen que toda la información relacionada a un proyecto de investigación deberá estar en el repositorio OSF, ver Fig. 29.

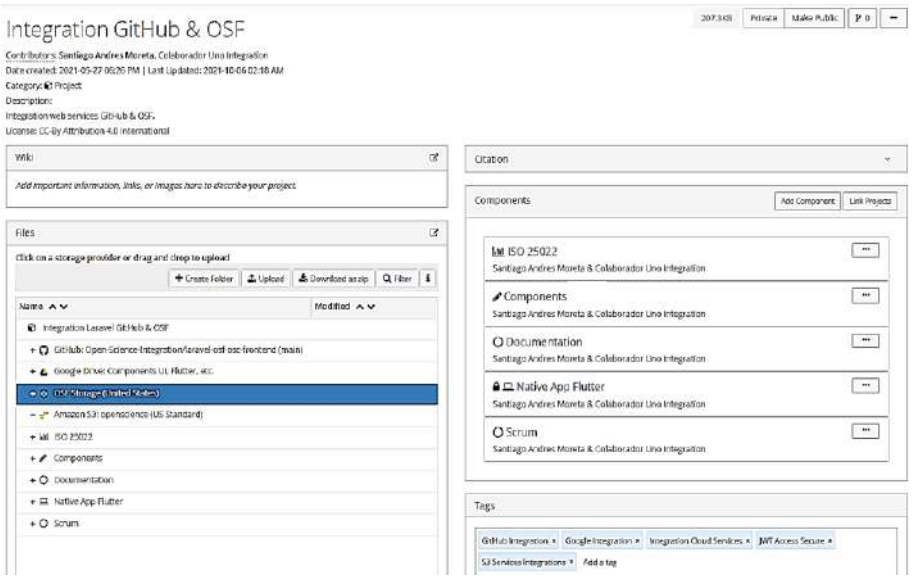


Fig. 29. Componentes del proyecto en el repositorio OSF.

Fuente: Propia.

SOA, módulos del sistema

Los módulos del presente sistema cumplen con una funcionalidad concreta para que la aplicación funcione con OSF y GitHub.

El módulo de administración académica tiene toda la lógica del negocio, como por ejemplo gestión de contenidos, estudiantes, docentes, contenidos abiertos, etc., con ello será posible reutilizar el código en todas las etapas del desarrollo.

En el proyecto se utilizaron las funciones de todos los submódulos que serán parte del sistema, mismos que serán detallados en los Sprints a medida que el proyecto crezca. El Backend será generado en tecnologías API REST, que a posterior serán consumidas en las vistas necesarias con las seguridades pertinentes.

El modelo de datos físico de la base de datos

En el diseño físico de la base de datos se realizó un diagrama de entidad relación, se aplicó la normalización necesaria. En la Fig. 30 se muestra el modelo básico que se generó al crear el proyecto orientado a servicios, el Framework Laravel es muy versátil y permite generar tablas a partir de los modelos relacionados a los controladores, para ello se hace uso de tablas generadas automáticamente mismas que deben ser “sembradas” término usado para llenar automáticamente las tablas con datos de prueba.

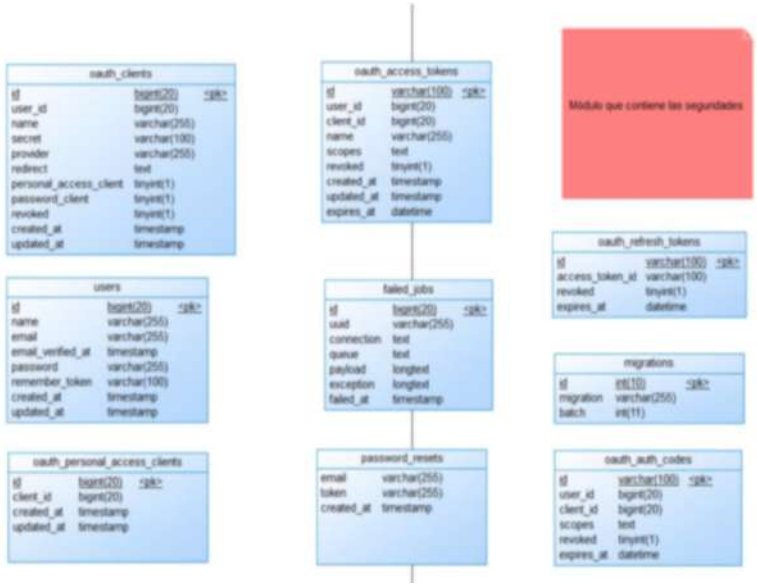


Fig. 30. Diagrama Base de Datos, versión 1.

Fuente: Propia.

El Framework Laravel contiene una propiedad llamada Eloquent ORM que permite realizar las relaciones directamente en el código para que las solicitudes necesarias sean sencillas manejar entre los datos y las relaciones de las tablas establecidas en el modelo de la base de datos. Tablas tales como users, se crearon automáticamente y fueron modificadas posteriormente para generar los roles de acceso.

Aplicación Orientada a Servicios, estructura de navegación

Las heurísticas de J.Nielsen recomiendan establecer coherencia y estándares para lograr una aplicación web eficiente. Por tal motivo se realizó el diagrama global de navegación en correlación a todos los roles requeridos y aquellos privilegios de los que gozan, ver Fig. 31.

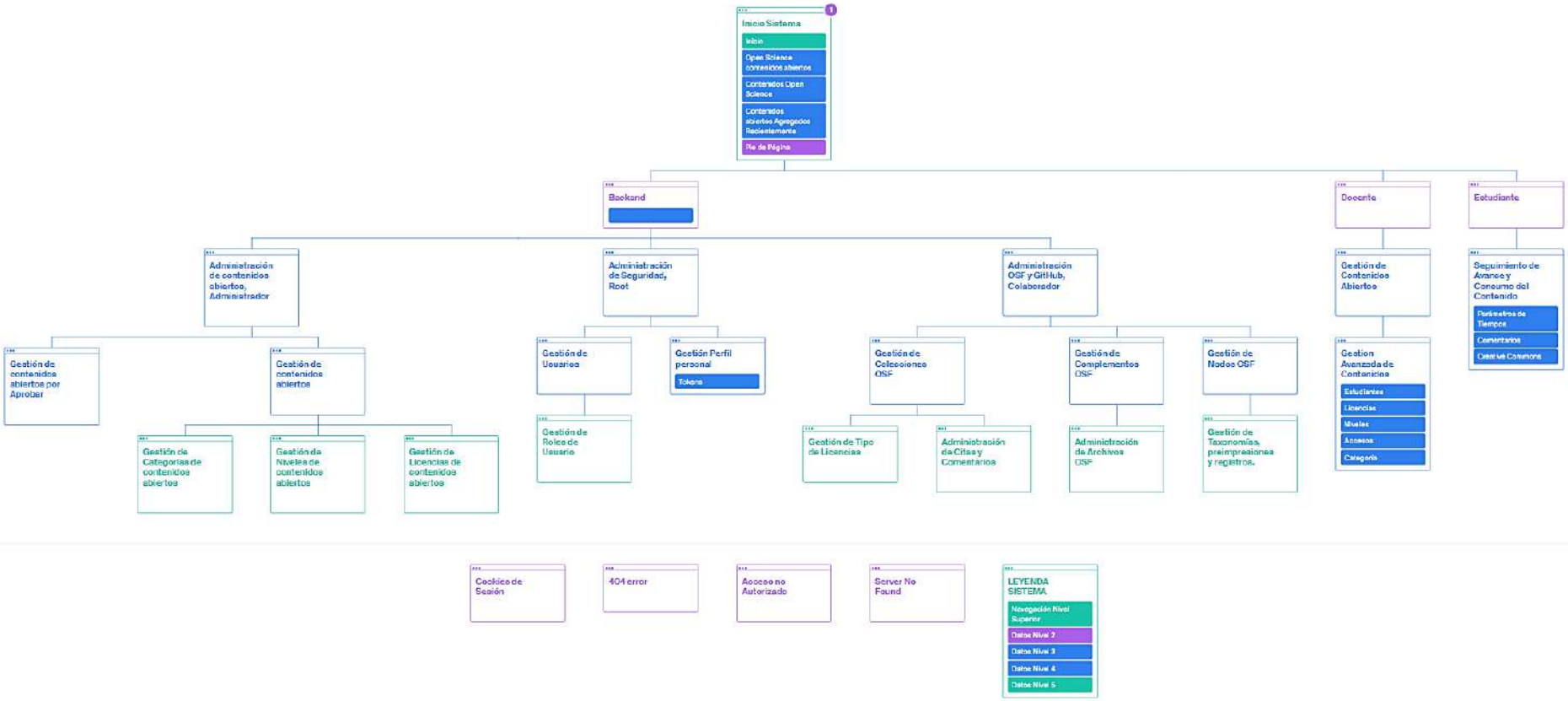


Fig. 31. Diagrama de navegación por niveles, aplicación web.

Fuente: Propia.

Interfaces de Usuario UI - Adobe Xd

En la presente maquetación se pretendía generar interfaces de usuario modernas y amigables para el usuario, por ello se hizo uso de Sass (Syntactically Awesome Style Sheets) y elementos responsivos, los cuales al ser ejecutados directamente en scripts interpretados por el navegador resultan ser veloces e intuitivos para el usuario.

Una vez definidas las tecnologías Frontend, se creó un esquema lógico de navegación, ya que al realizar dicha tarea será posible desarrollar las interfaces de manera ordenada y manteniendo una estructura global de la aplicación.

Para generar las maquetas de las interfaces se hizo uso de Adobe Xd el cual es un software de gráficos vectoriales que permite el diseño y simulación de interfaces de usuario profesionales. Las interfaces de usuario fueron diseñadas y puestas a prueba tanto para computadores y smartphones, para que el usuario pueda gozar de una mejor experiencia en el sitio, fueron definidas varias capas de trabajo en el entorno Adobe Xd, entre las que destacan las zonas:

- **Hero:** Es la zona en donde irán el banner y los botones principales de acceso a contenidos abiertos recientemente agregados, ver Fig. 32.

Fig. 32. Diseño Web UX, Zona Hero.

Fuente: Propia.

- **Menú:** Es la zona más importante, ya que manejará toda la lógica de acceso tal cual como se definió en el diagrama de navegación y estructura del aplicativo.
- **Zona Información General:** Es la zona en donde se colocaron los principales objetivos que busca Open Science y una de sus ramas, ver Fig. 33.

Fig. 33. Diseño Web UX, Zona Información General.

Fuente: Propia.

- **Zona Contenidos:** Se encuentran todos los contenidos abiertos recientemente agregados con los datos del instructor, título, estudiantes, etc. También se generó una barra de búsqueda para hacer más amena la navegación para el usuario, ver Fig. 34.

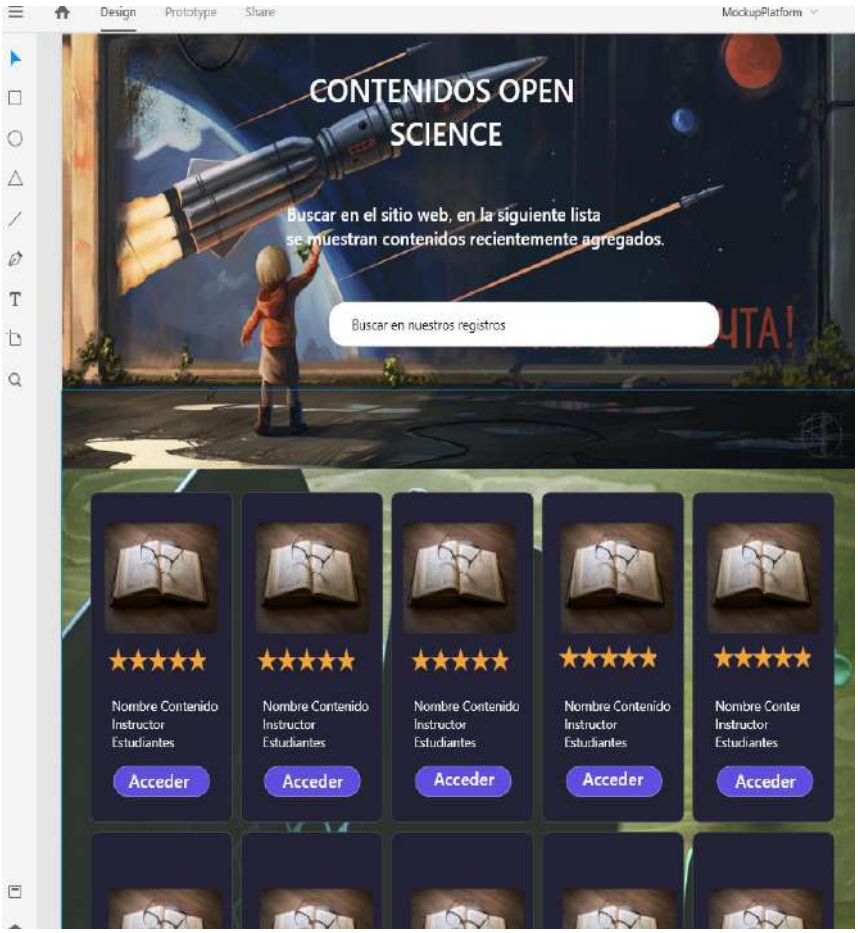


Fig. 34. Diseño Web UX, Zona Contenidos.

Fuente: Propia.

Así mismo se generaron las otras vistas y bosquejos del producto final, cabe recalcar que entre las grandes ventajas que proporciona la herramienta Adobe Xd es que se dispone de una zona de pruebas para un entorno de simulación en la que permite visualizar tal cuál como se verían las interfaces después de ser desarrolladas, se realizaron las pruebas necesarias de UX, entre las que destacan:

- El sitio será completamente responsivo.
- El sitio contiene los contenidos correspondientes a cada una de las zonas establecidas en el diagrama de navegación.
- El sitio será frecuentemente expuesto a pruebas de rendimiento lo que hará que el sitio sea veloz y tenga una taza de respuesta óptima.
- Los complementos tales como rellenar formularios, navegación y otros, serán establecidos con componentes que ofrece LaravelMix, ya que permiten optimizar la ejecución en pequeños espacios de código.

Diseño de la arquitectura de la Aplicación

El diseño de la arquitectura fue desarrollado con varias capas las cuales contendrán toda la lógica del aplicativo, entre las que destacan:

- **Capa del Negocio:** En la capa de negocio se encuentra la lógica Backend, dicha lógica será alojada en AWS, este servidor permitirá conectar los datos y el consumo de los servicios de otras plataformas como, por ejemplo: Google Cloud, GitHub, y otros.
- **Capa de Monitoreo:** En la capa de monitoreo se encuentran las herramientas que permitirán obtener métricas de rendimiento eficientes durante las etapas de desarrollo y despliegue a producción del software.
- **Capa de Datos:** En la capa de datos se encuentra el servidor de la base de datos relacional MySQL, dicho servidor tiene las seguridades pertinentes.
- **Capa de Seguridad:** En la capa de seguridad se encuentran las métricas de seguridad que protegen la integridad del sistema y también los componentes de acceso como JWT y Access Token con privilegios de lectura/escritura.
- **Capa de Integración API OSF y GitHub:** Es la capa que permitirá interconectar las dos API REST y gestionar el contenido Open Science mediante la aplicación. La presente capa hará uso de los servicios de las dos plataformas para gestionar el contenido abierto.

- **Capa del Cliente:** Es la capa que permitirá que los aplicativos tanto del Backend y Frontend puedan comunicarse entre sí, en dicha capa serán desarrollados aquellos “endpoints” y por ende el consumo se servicios de otras plataformas.

Arquitectura SOA desarrollo capa del cliente, ver Fig. 35.

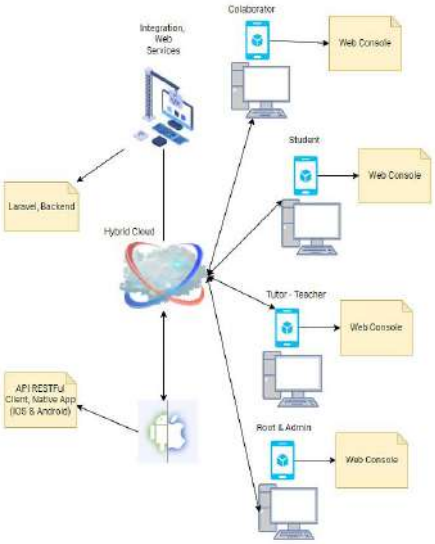


Fig. 35. Capa Cliente.

Fuente: Propia.

En la Fig. 36, se puede visualizar la unión de todas las capas que forman parte de la arquitectura en la nube y como se relacionan entre sí, cabe mencionar que dichas capas dependen de servicios remotos, que fueron configurados previamente.

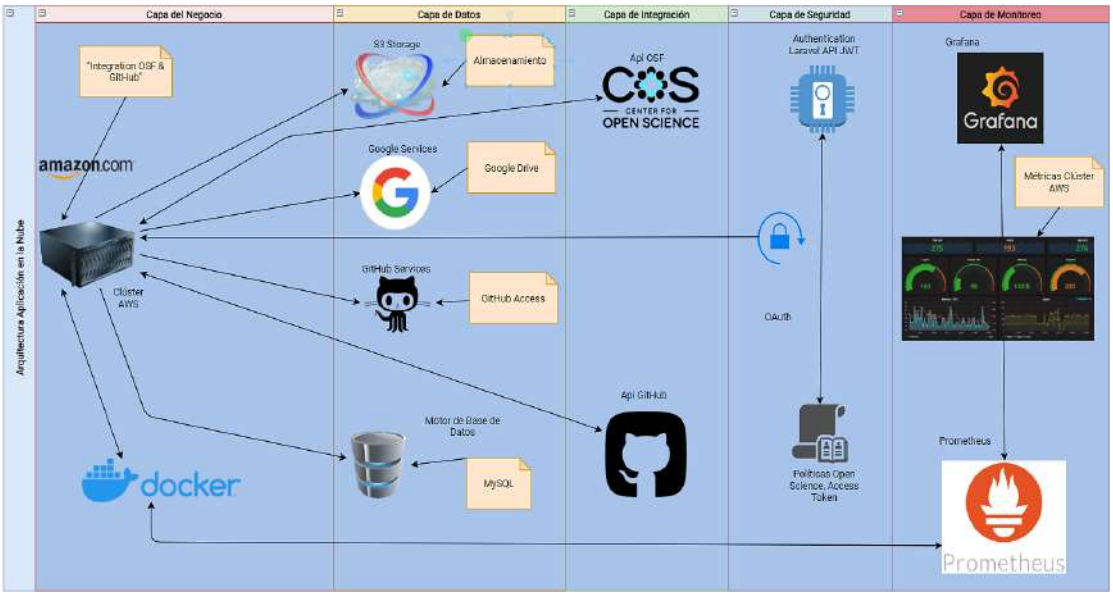


Fig. 36. Arquitectura, nube híbrida.

Fuente: Propia.

Levantar los repositorios de versionamiento

Para los ambientes de versionamiento, se generó el repositorio respectivo en AWS. Los directorios necesarios se crean automáticamente, para ello se agregó la carpeta de configuración de AWS, dichos ficheros ejecutan el flujo CI/CD en el servidor automáticamente.

Levantar los ambientes de producción en el servidor

En la consola de administración de AWS se generó la instancia necesaria para ser ejecutado el sistema desde el repositorio GitHub, durante el proceso de desarrollo del proyecto resultará sencillo desplegar los cambios de inmediato a producción. AWS posee varias herramientas para desarrollo y diversas funcionalidades de computación en la nube, en el presente proyecto se hará uso de varias de ellas. En la Fig. 37, se muestra el clúster después de configurar las tareas, repositorios, instancias de ejecución, balanceo de carga, S3, reglas de routing, políticas de usuario, VPC y otros servicios.

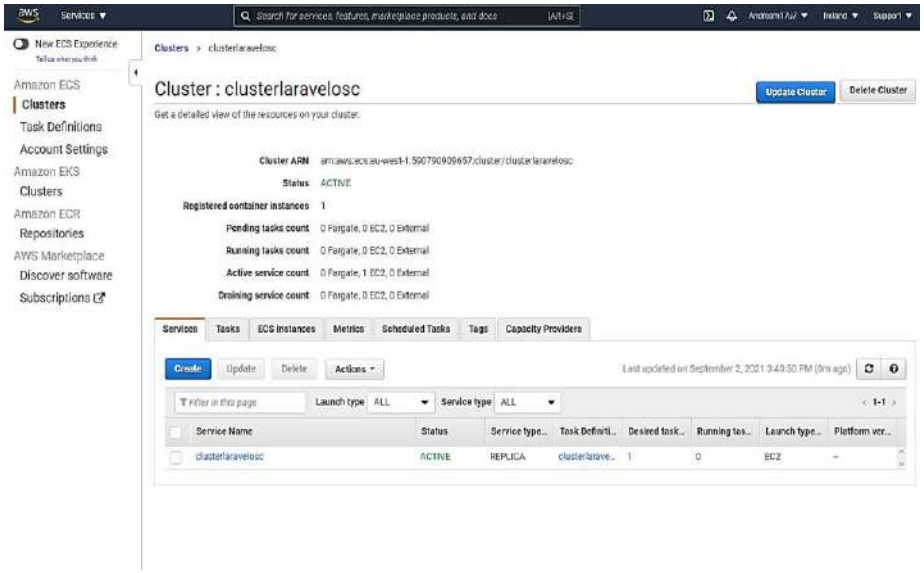


Fig. 37. Levantar ambientes en AWS, clúster-repositorio-tareas-servicios.

Fuente: Propia.

Creación de contenedor de despliegue Docker en GitHub

En Ingeniería de Software el término CI/CD es frecuentemente aplicado, ya que al generar un ambiente controlado en docker es posible realizar el despliegue e integración continua de aplicaciones sin mayores complicaciones, por ello en el proyecto se implementaron varios componentes que ayudaron a desplegar el sistema de una manera eficiente en AWS. Todos los componentes se actualizarán en un repositorio sincronizado en S3 de AWS con el de la rama principal de GitHub que contiene el fichero de extensión (.yml) y Docker creará el ambiente controlado mediante un contenedor, ver Fig. 38.

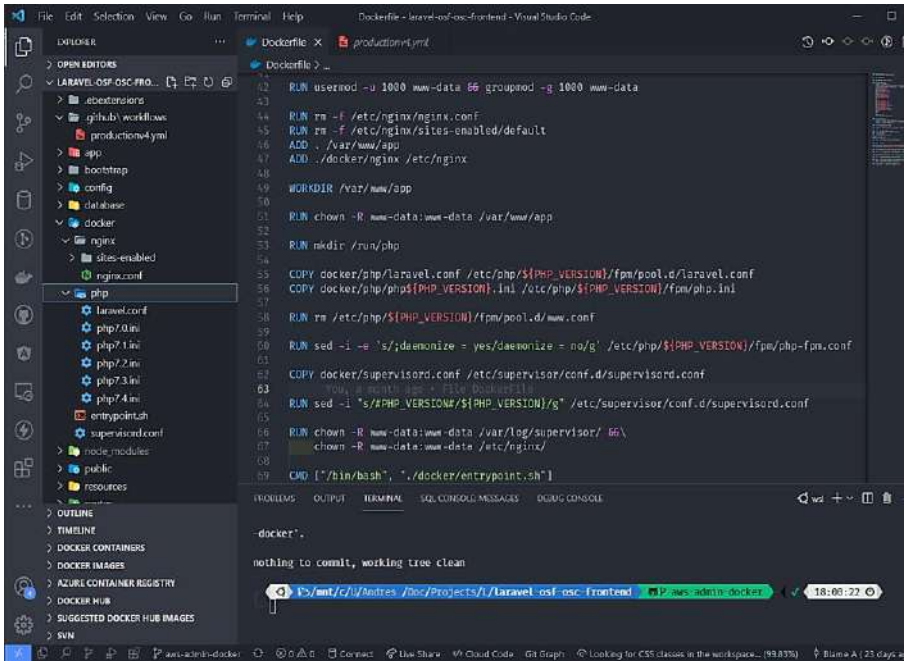


Fig. 38. Creación de contenedor en Docker.

Fuente: Propia.

Ejecución de tareas en GitHub Actions

En la Fig. 39, se muestra el despliegue a producción CI/CD que se llevó a cabo de manera exitosa después de crear la imagen de docker que contiene el código de aplicación, bibliotecas, herramientas, dependencias y otros archivos necesarios para ejecutar una aplicación con las características del presente proyecto.

Ruta Docker: */.github/workflows/docker_github_image.yml.*

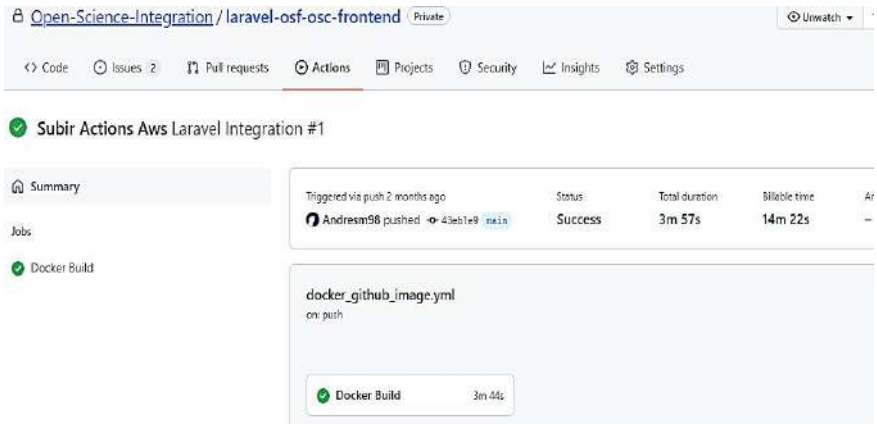


Fig. 39. Ejecución de Workflows en GitHub.

Fuente: Propia.

Despliegue en producción del aplicativo de arquitectura SOA en Laravel

En la Fig. 40, se puede visualizar el resultado de los flujos de trabajo iniciales, la aplicación en su versión de prueba fue desplegada a producción de manera exitosa y sin ningún error,

cabe recalcar que el ambiente controlado dejará una imagen de registro en el clúster AWS misma que ocupará tareas, balanceo de carga y un contenedor.

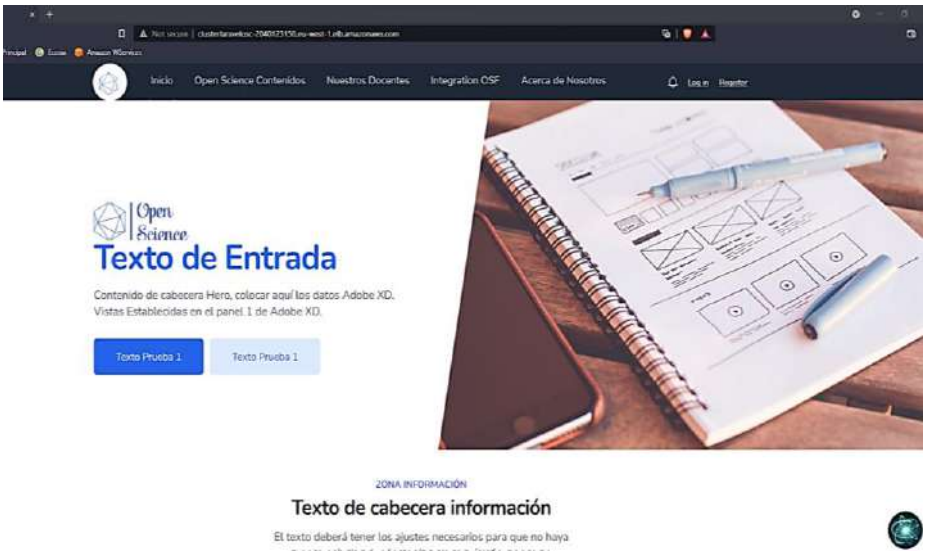


Fig. 40. Despliegue en producción Integration GitHub & OSF versión inicial.

Fuente: Propia.

Dependencias Aplicación y JWT

Las dependencias iniciales de la aplicación se generaron automáticamente en el fichero “package.json” después de ser instaladas las herramientas, entre las que destacan: Typescript, Laravel Mix, Vue, Sweet – Alert, JWT Auth, Axios y otras.

También se generaron las estructuras “routing” con las rutas de acceso para cada uno de los servicios que presta la aplicación haciendo uso de middleware, ver Fig. 41.

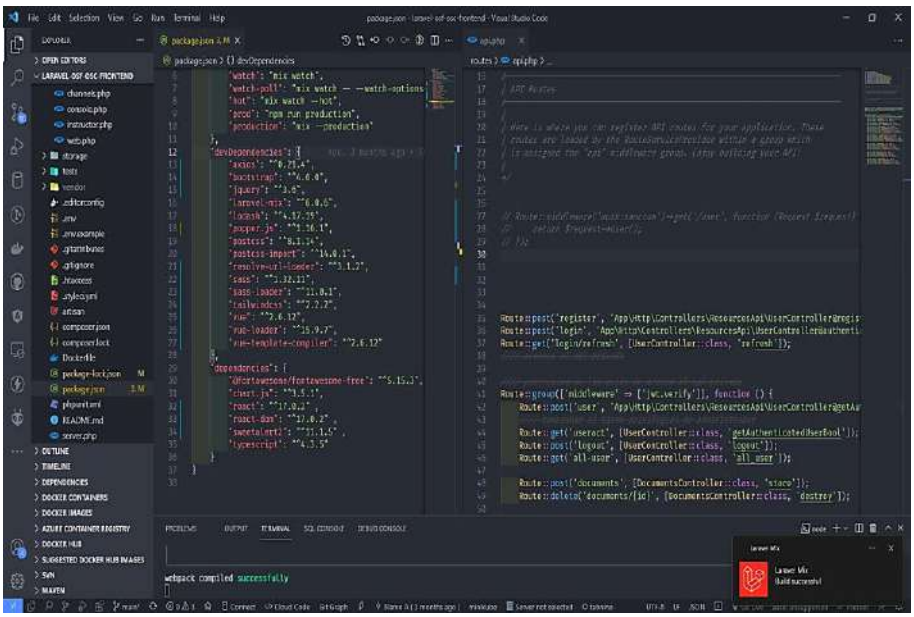


Fig. 41. Dependencias Iniciales.

Fuente: Propia.

2.2.2.4. Sprint Review

Las historias de usuario llevadas a cabo se detallan en la siguiente tabla.

Tabla 9. Sprint Review 1.

Código	Descripción	Cumple
OSI.01	Preparar el ambiente de trabajo OSF y GitHub.	Si
OSI.02	Desarrollar el modelo lógico de la base de datos.	Si
OSI.03	Diseño de las interfaces de usuario en Adobe XD.	Si
OSI.04	Diseño de la arquitectura orientada a servicios.	Si
OSI.05	Levantar el ambiente de desarrollo para el aplicativo.	Si
OSI.06	Levantar el ambiente de desarrollo en el servidor Amazon Web Services.	Si
OSI.07	Implementar las librerías propias de Laravel.	Si

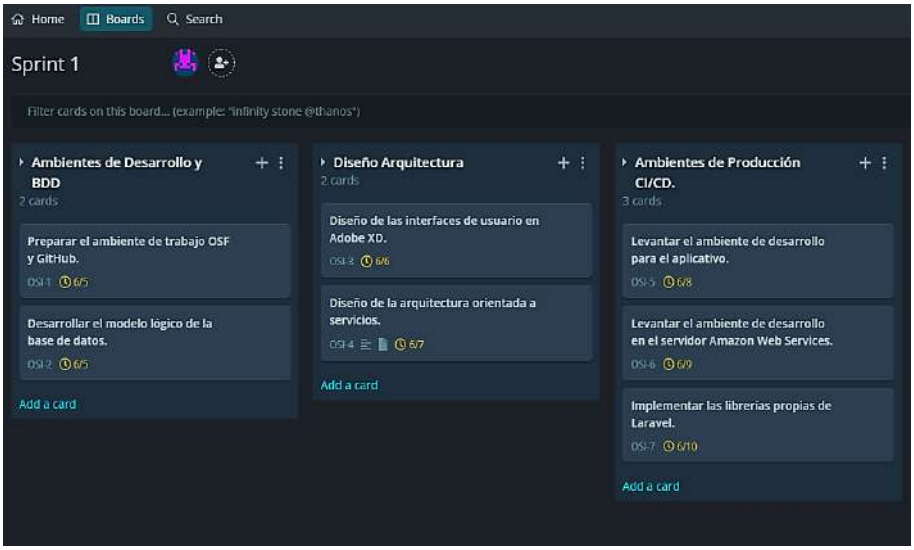


Fig. 42. GitKraken Board - Sprint 1.

Fuente: Propia.

2.2.2.5. Sprint Retrospective

¿Qué salió bien en el sprint?

Los ambientes de trabajo, instancias del servidor y otros servicios iniciales fueron correctamente instalados; de igual manera en el repositorio OSF se sincronizaron automáticamente los cambios en todos los componentes y paquetes. Los usuarios que accedan al Frontend tendrán acceso a interfaces de usuarios responsivas y veloces, los errores que podrían suscitarse al desplegar todos los componentes en producción fueron resueltos oportunamente con herramientas de pruebas de rendimiento.

¿Qué se aprendió en el sprint para mejorar la forma de llevar a cabo el proyecto?

Para mejorar el proyecto, será necesario implementar gradualmente los paquetes necesarios y precautelando las versiones de cada una de las librerías. Las interfaces de usuario se irán

generando gradualmente con los elementos necesarios para que el sitio ofrezca una experiencia de usuario con interfaces profesionales y eficientes.

Laravel Mix permite la integración de elementos Vue y otras herramientas de manera sencilla, por lo que resulta práctico realizar una aplicación escalable y al realizar tal acción será posible manipular componentes, consumir datos, etc.

Las herramientas de GitHub son realmente robustas, se usaron herramientas de Integración y Distribución Continua que dan por resultado un despliegue en producción estable y con los paquetes estrictamente necesarios, del mismo modo Docker permitió hacer cambios, permisos y otras tareas que harán que la aplicación sea escalable, segura y eficiente mediante el uso de los ficheros del contenedor. Es importante mencionar que se hicieron configuraciones en Docker para que el aplicativo React se sincronice automáticamente en el servidor y sea desplegado como una dependencia.

AWS posee flujos de trabajo que pueden ser aplicados de manera veloz, al hacer uso de servicios computacionales complejos se fue realizando la documentación necesaria de cada una de las instalaciones, entre una de ellas y la más importante es el contenedor y repositorio los cuales permitirán que el despliegue tenga un versionamiento claro y preciso.

2.2.3. Sprint 2

2.2.3.1. Sprint Planning

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

- Levantar los servicios y desplegarlos en el servidor AWS (Backend).
- Crear los servicios API REST de todas las entidades de la base de datos, especificados en el Sprint anterior.
- Validar los puntos finales de comunicación “endpoins” y generar las seguridades necesarias con JWT.
- Crear las estructuras de navegación y Frontend de la aplicación.
- Generar políticas de acceso para todas las rutas del sistema y mantener así la integridad de los módulos.
- Conectar los componentes necesarios: Google OAuth, GitHub OAuth, AWS e implementar los servicios necesarios para el redireccionamiento de autenticación.
- Levantar políticas de seguridad en AWS para la integración de servicios de nubes privadas.
- Establecer las políticas de acceso en los ficheros JSON de los contenedores de almacenamiento S3.

2.2.3.2. Sprint Backlog

En el Sprint número 2, se tomaron en cuenta las historias de usuario que están descritas en el Anexo Nro. 3. Para ello se definieron criterios de aceptación y tareas a realizar.

2.2.3.3. Ejecución del Sprint

En el presente Sprint se llevaron a cabo tareas en su gran parte de índole ingenieril, por lo que para ello se ha generado la documentación necesaria en el Anexo Nro. 4. En el mencionado apartado se encuentran los métodos, técnicas, conexión de datos, servicios instalados y como fueron creados los datos de siembra para cada una de las entidades de la base de datos. Es pertinente mencionar que entidades como colecciones fueron creadas para que se almacene la información necesaria que se consume directamente de COS, la base de datos a ser implementada se muestra en la Fig. 43.

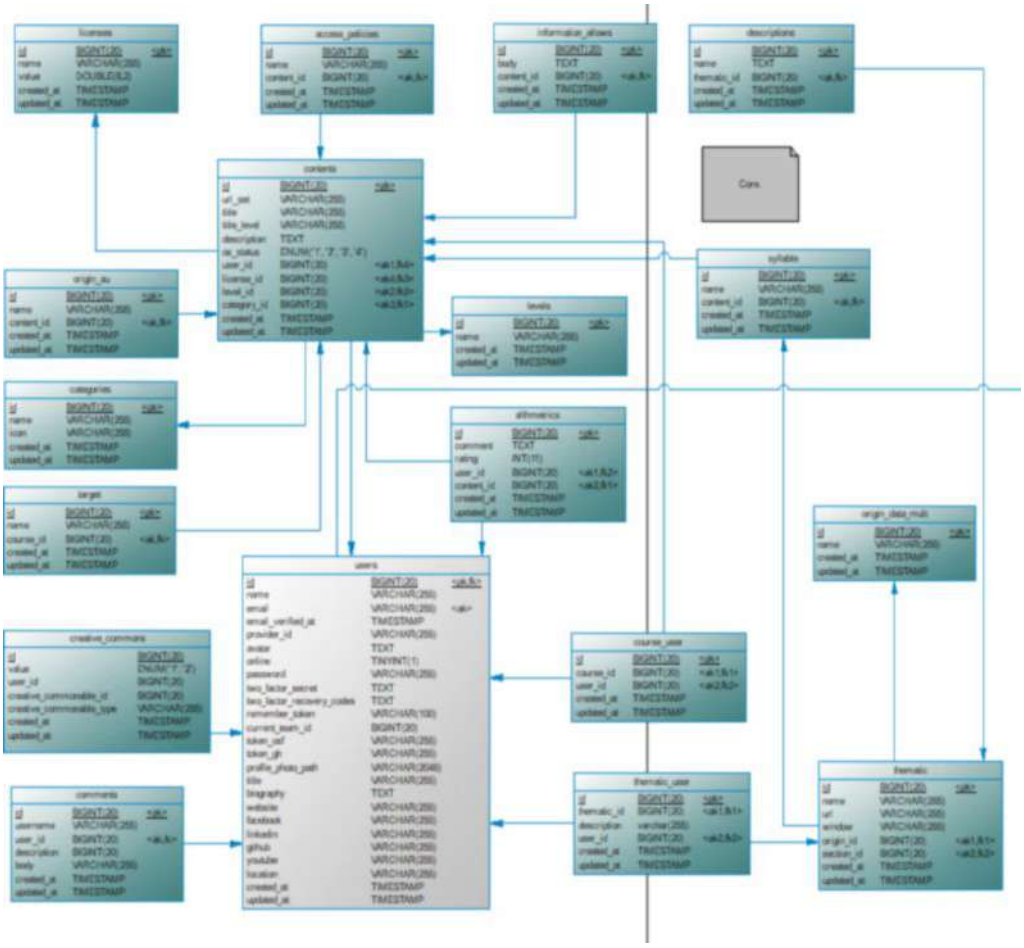


Fig. 43. Diagrama Base de Datos, versión 2.

Fuente: Propia.

Para la ejecución del Sprint número 2 fue necesario dar inicio al desarrollo del Backend y Frontend de la aplicación que contiene toda la lógica del negocio, en el Backend están desarrollados los “endpoints” o puntos finales de conexión requeridos, con las entidades y seguridades pertinentes teniendo en cuenta las siguientes tareas:

- Implementar las entidades que se encuentran detalladas en las historias de usuario.
- Creación de los controladores.
- Implementar las seguridades de cada uno de los controladores.
- Implementar los servicios que interactúan en la capa de la base de datos.
- Crear los modelos necesarios.
- Migración masiva de datos de prueba.
- Comprobar la funcionalidad de cada uno de los “endpoint” con el software Postman.

Así mismo, se realizaron los esquemas de navegación mediante una estructura lógica y ordenada con ello el usuario tendrá una experiencia de usuario (UX) amigable mediante interfaces de usuario (UI) intuitivas, dinámicas y responsivas. Teniendo esto en consideración se establecieron las siguientes tareas:

- Implementación de servicios Google y GitHub, mediante las herramientas de desarrollador y OAuth.
- Instalación de paquetes de conexión a servicios externos.
- Crear capas de servicios seguras.
- Establecer la estructura de navegación de la aplicación.
- Desarrollar interfaces responsivas, para ello se aplicaron herramientas Live Server.
- Depurar cada una de las interfaces con las herramientas de desarrollador de un navegador con ello se validarán objetos css, javascript, cache, etc.

Mapa de Procesos Frontend

Durante la ejecución del presente Sprint se automatizaron procesos los cuales en sus primeras etapas fueron plasmados en un mapa de procesos y se obtuvo una visión general del alcance general que abarcará la aplicación. En la Fig. 44, se puede visualizar la creación del mapa de procesos del Frontend y también la secuencia de almacenamiento de los datos.

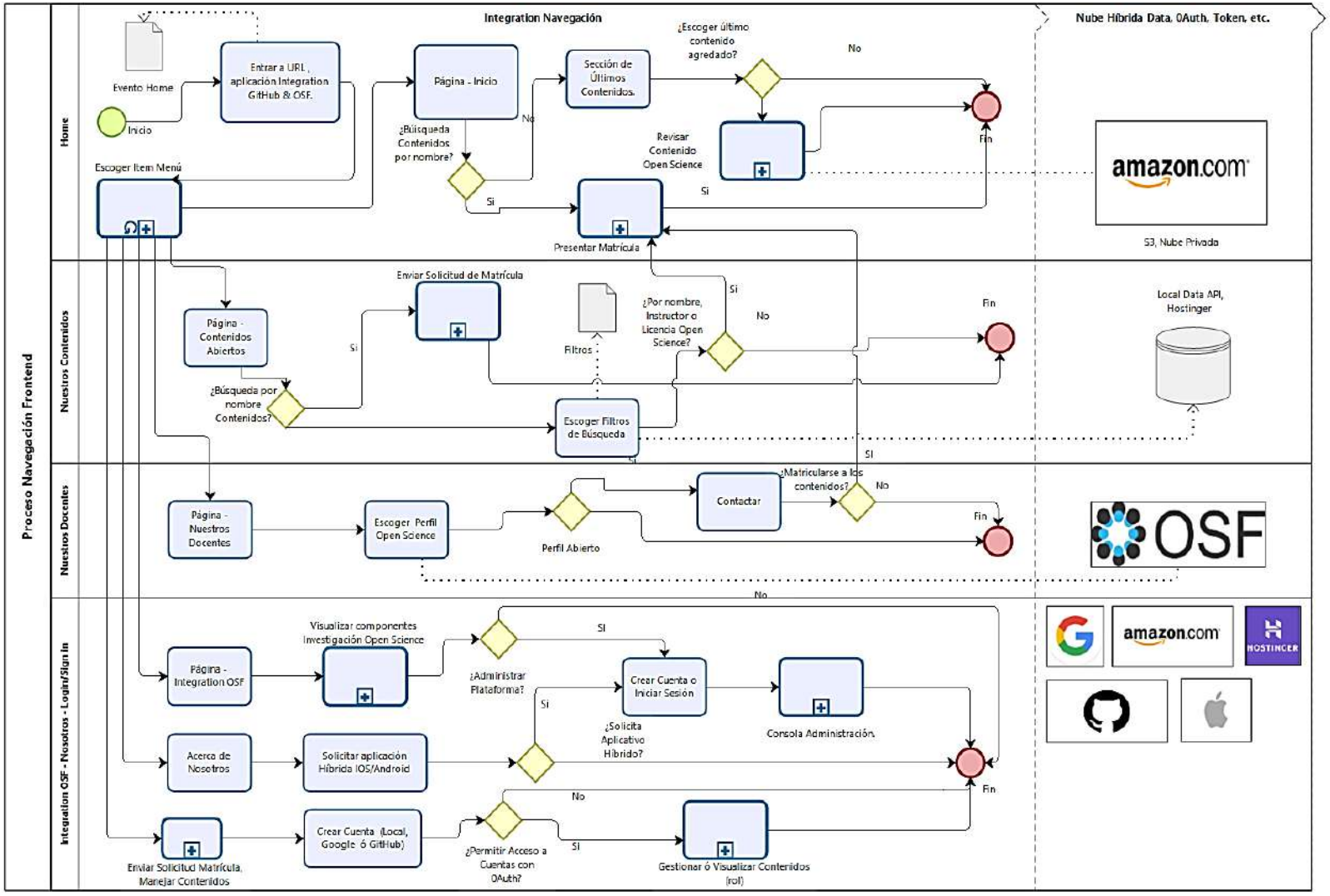


Fig. 44. Mapa de Procesos Global de la aplicación, Frontend.

Fuente: Propia.

Frontend aplicación Integration OSF & GitHub

En la Fig. 45, se puede visualizar los resultados después de desarrollar la denominada “Zona Hero” que será el primer elemento que observará el usuario al ingresar a la aplicación.

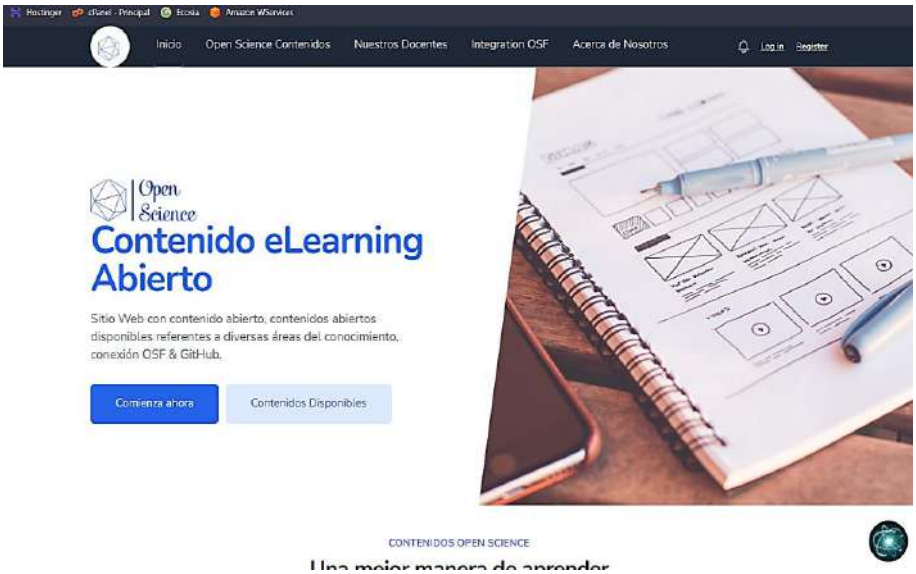


Fig. 45. Página Principal, Zona Hero – ejecución.

Fuente: Propia.

En la Fig. 46, se puede visualizar la denominada Zona de Búsqueda de Contenidos que permitirá al usuario generar filtros personalizados, cabe recalcar que para ello se aplicaron componentes reutilizables para que sea posible instanciarlos en cualquier parte del Frontend.

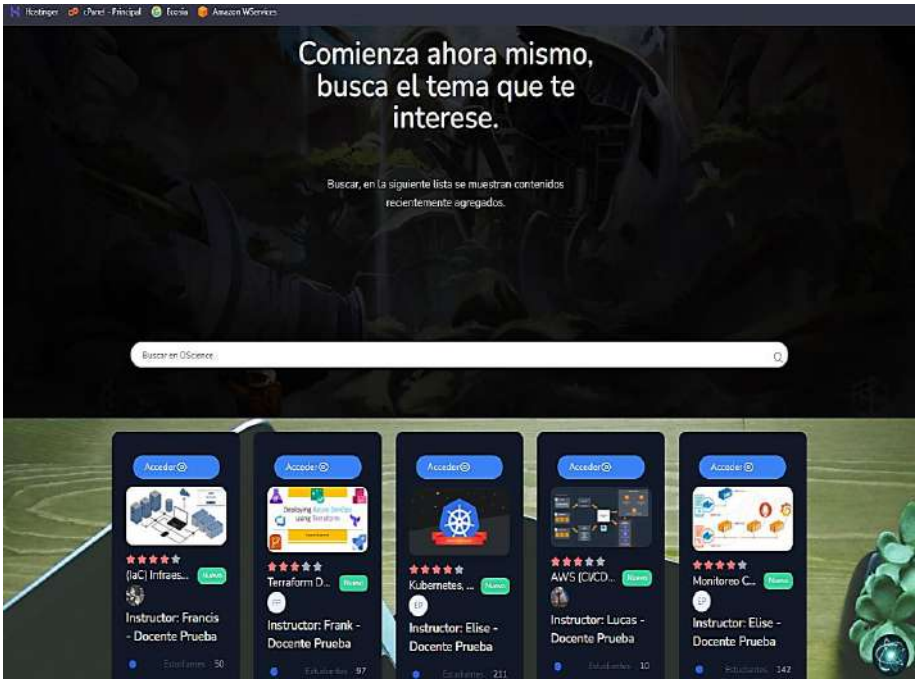


Fig. 46. Página Principal, Zona Contenidos de Interés – ejecución.

Fuente: Propia.

Menú Principal - Open Science Contenidos

El segundo elemento de navegación abarca toda la búsqueda global de los contenidos abiertos, tal como se visualiza en la Fig. 47.

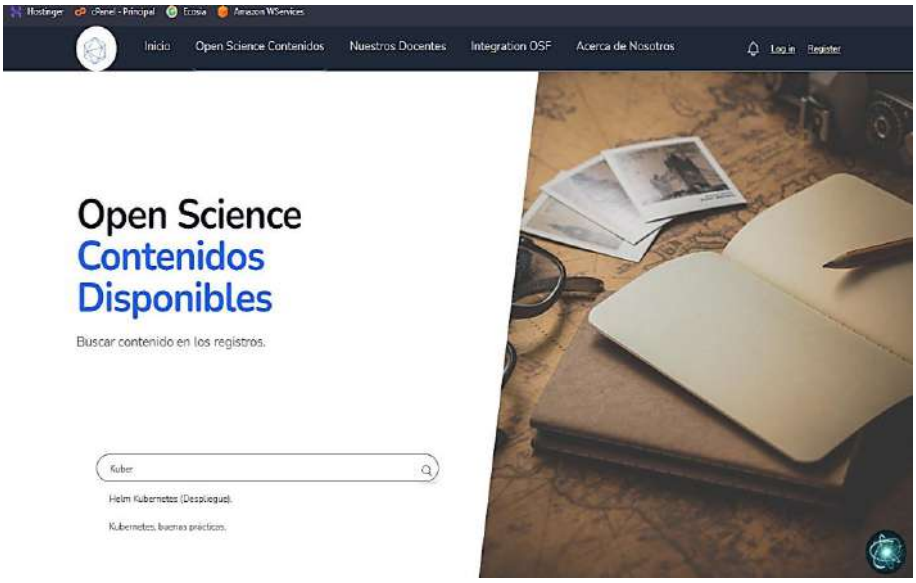


Fig. 47. Página Open Science Contenidos – ejecución.

Fuente: Propia.

Teniendo en cuenta la escalabilidad de la aplicación se crearon varios directorios con componentes y la flexibilidad que ofrecen para crear tareas asíncronas, ver Fig. 48.

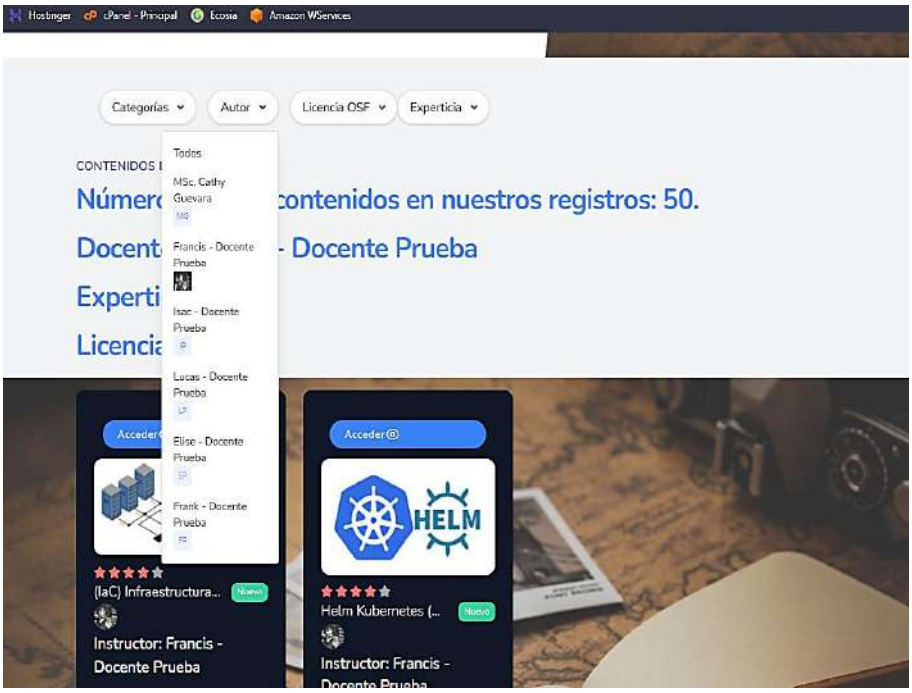


Fig. 48. Página Open Science Contenidos, filtros personalizados – ejecución.

Fuente: Propia.

Menú Principal - Nuestros Docentes

El tercer elemento de navegación abarca a todos los docentes que forman parte de la aplicación, se pueden visualizar de manera general con la información relevante, ver Fig. 49.

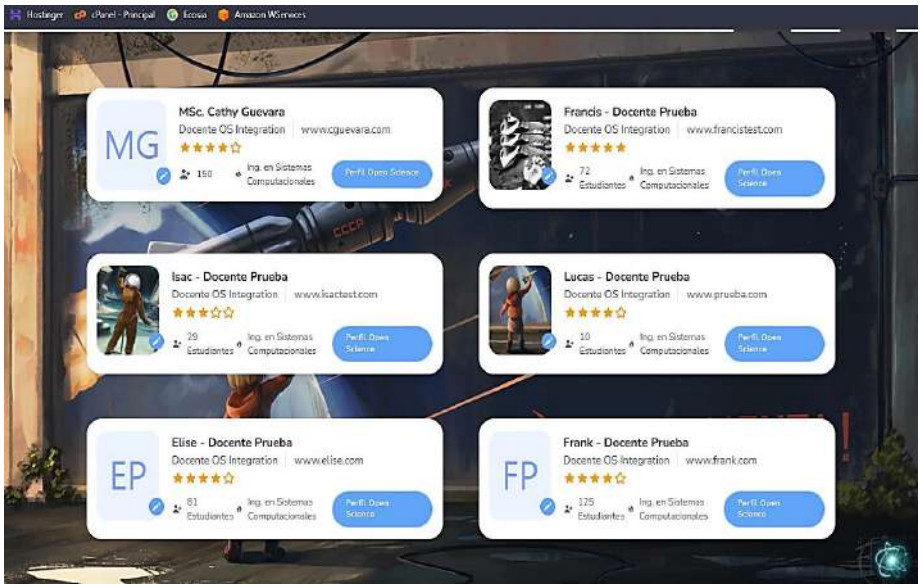


Fig. 49. Página Nuestros Docentes – ejecución.

Fuente: Propia.

Después de hacer clic en “Perfil Open Science” es posible acceder a la información detallada del docente, así como a las publicaciones que hayan pasado los filtros Open Science que se han establecido en la aplicación, ver Fig. 50.

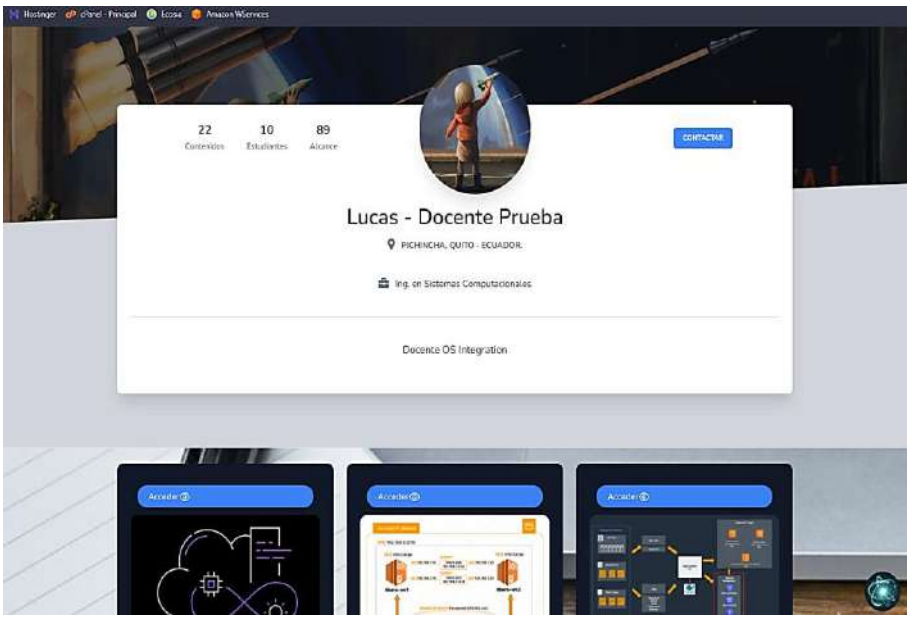


Fig. 50. Página Nuestros Docentes, Perfil – ejecución.

Fuente: Propia.

Menú Principal - Acerca de Nosotros

El quinto elemento de navegación contiene la información sobre aquello que pretende el movimiento Open Science y un enlace a la aplicación nativa alojada en Firebase, ver Fig. 51.

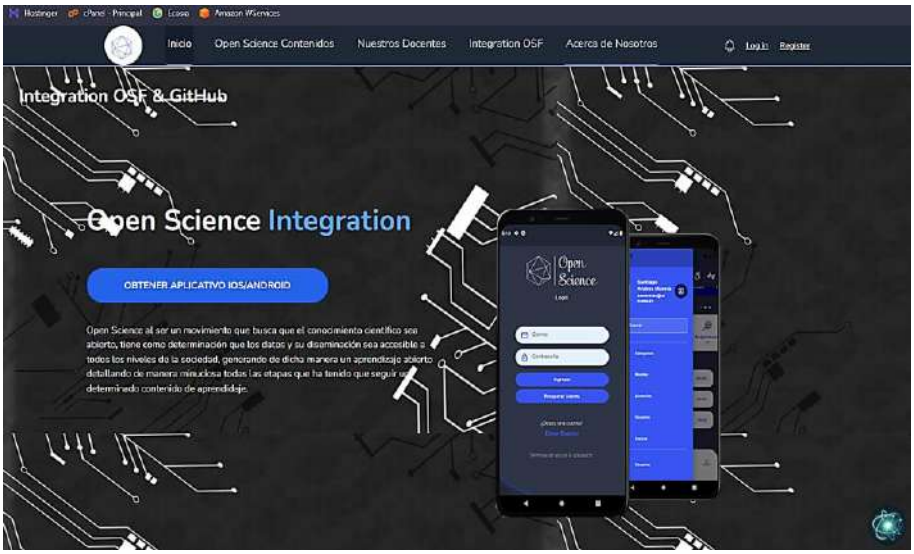


Fig. 51. Página Acerca de Nosotros – ejecución.

Fuente: Propia.

Menú Principal – Login - Acceso al Sistema

En la Fig. 52, se puede visualizar la página que permitirá iniciar sesión. Para el ingreso es necesario un usuario y contraseña correctamente validados o en su defecto el usuario podrá autenticarse con sus cuentas personales de GitHub – Google.

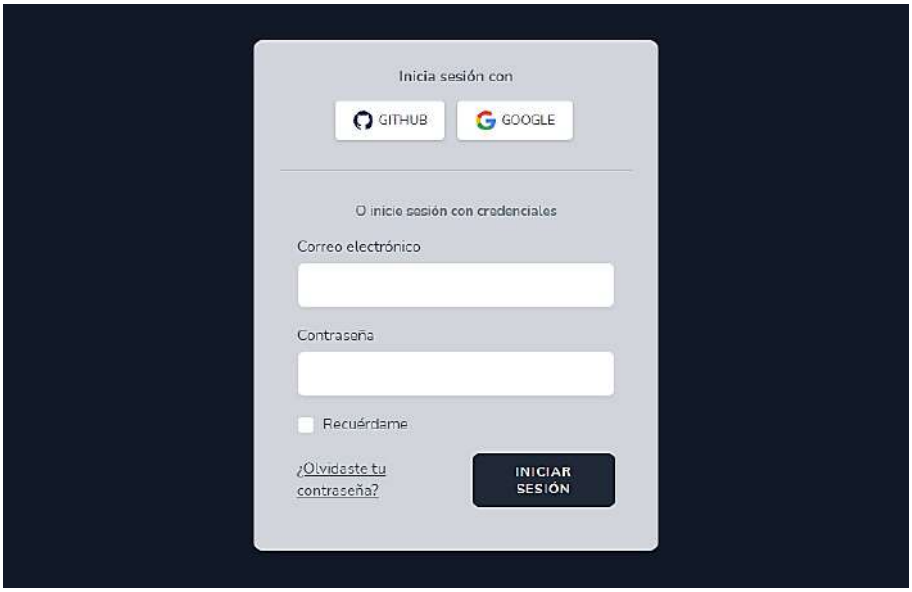


Fig. 52. Ingreso a la aplicación.

Fuente: Propia.

En caso de que el usuario opte por ingresar con su cuenta de GitHub o Google, la aplicación le redireccionará a la página Oficial de cada una en la cual deberá ingresar sus credenciales y permitir el acceso a la aplicación de desarrollador “Integration GitHub & OSF”, ver Fig. 53.

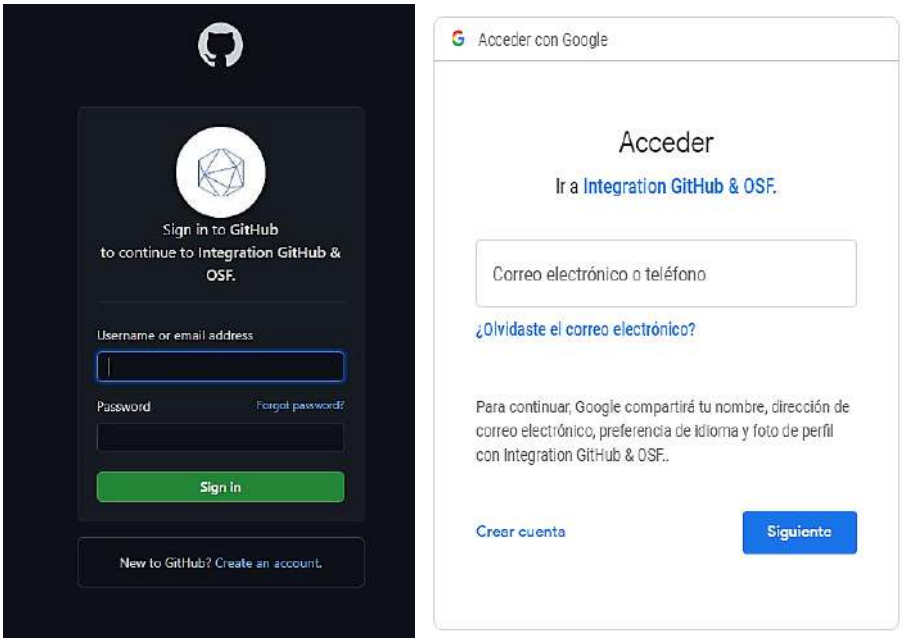


Fig. 53. Ingreso al sistema con GitHub y Google.

Fuente: Propia.

Para recuperar la cuenta el usuario debe dar clic en “¿Olvidaste tu contraseña?”, después el usuario deberá ingresar su correo electrónico y finalmente, el sistema enviará automáticamente un token de recuperación de sesión al correo del usuario, ver Fig. 54.

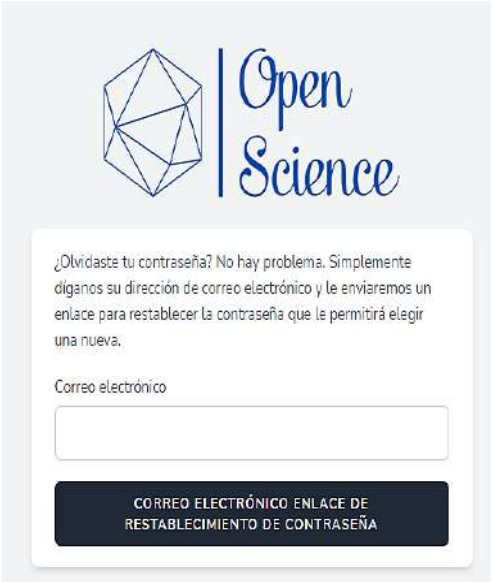


Fig. 54. Solicitud de recuperación de clave.

Fuente: Propia.

Después el usuario debe comprobar la bandeja de entrada de su correo electrónico y encontrará un mensaje con un enlace con un token que expirará en una hora, ver Fig. 55.

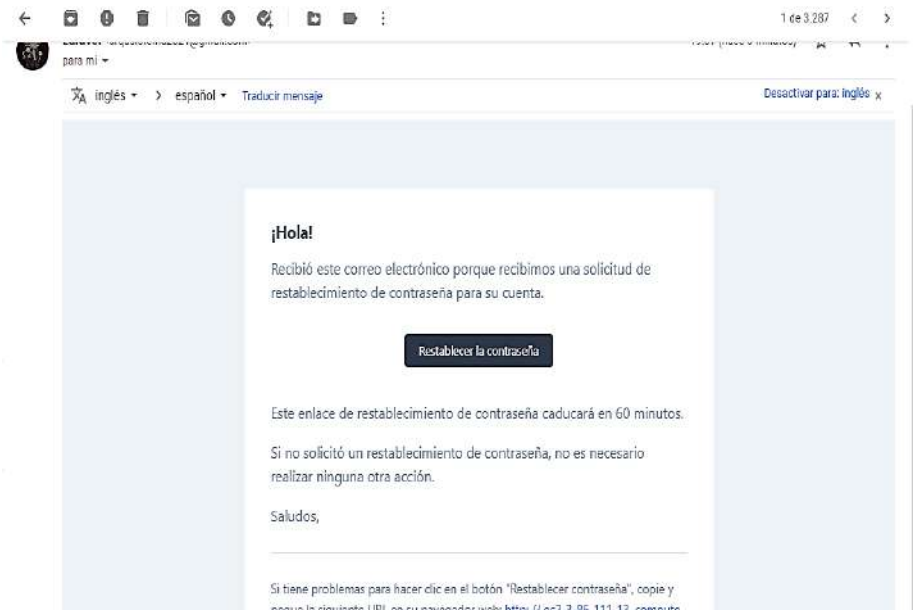


Fig. 55. Enlace de recuperación de clave.

Fuente: Propia.

Finalmente, el usuario deberá ingresar la nueva contraseña dos veces para que el sistema valide los datos, ver Fig. 56.

Fig. 56. Recuperar cuenta de usuario.

Fuente: Propia.

Menú Principal – Registro - Acceso al Sistema

Si el usuario opta por registrarse sin aplicar los métodos anteriormente mencionados, deberá ingresar los datos específicos en el siguiente formulario, ver Fig. 57.

Fig. 57. Registro al sistema.

Fuente: Propia.

Perfil de usuario

El usuario podrá personalizar 12 campos que van desde el nombre presentado en la aplicación hasta los tokens personales de GitHub – OSF para dar uso a las funcionalidades de integración; incluyendo la personalización de seguridad, ver Fig. 58.

Fig. 58. Perfil de usuario en la aplicación.

Fuente: Propia.

El sistema posee seguridades de alto nivel los cuales permiten administrar: autenticación de dos factores y gestión de sesiones abiertas en otros navegadores que el sistema registra con la dirección IP pública del usuario y la hora, ver Fig. 59.

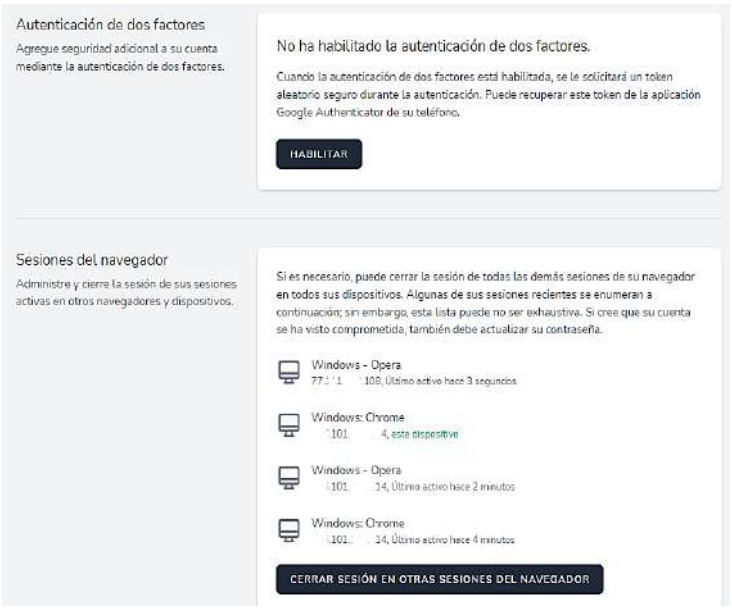


Fig. 59. Autenticación de dos factores y sesiones de navegador.

Fuente: Propia.

2.2.3.4. Sprint Review

Tabla 10. Sprint Review 2.

Código	Título	Cumple
OSI.08	Implementar las entidades por defecto en el proyecto. (migrations, users, etc).	Si
OSI.09	Implementar las entidades de Usuario y Login.	Si
OSI.10	Instalar las dependencias Laravel y Vue, iniciar las interfaces de usuario.	Si
OSI.11	Implementar las entidades de permisos y roles del sistema.	Si
OSI.12	Implementar las entidades de los contenidos abiertos.	Si
OSI.13	Implementar la entidad contenidos.	Si
OSI.14	Implementar la entidad objetivos-contenidos abiertos.	Si
OSI.15	Implementar la entidad información-contenidos abiertos.	Si
OSI.16	Implementar la entidad requerimientos-contenidos abiertos.	Si
OSI.17	Implementar la entidad de artículos.	Si
OSI.18	Implementar la entidad de progreso de contenidos abiertos.	Si
OSI.19	Implementar la entidad de docente.	Si
OSI.20	Implementar la entidad de estudiantes-contenidos abiertos.	Si
OSI.21	Implementar la entidad de almacenamiento de datos Open Science.	Si
OSI.22	Implementar las entidades de relación: contenido Open Science, archivos Open Science, Log del Contenido, Categorías de contenidos abiertos.	Si
OSI.23	Implementar la entidad creative common.	Si

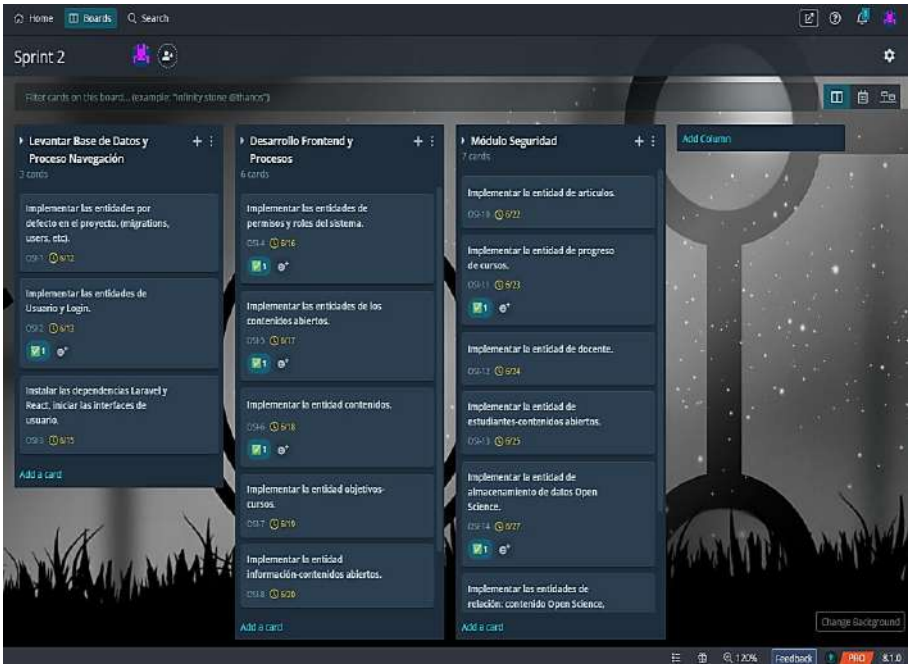


Fig. 60. GitKraken Board - Sprint 2.

Fuente: Propia.

2.2.3.5. Sprint Retrospective

¿Qué salió bien en el sprint?

Una vez que se realizaron las tareas descritas del presente Sprint se obtuvieron interfaces de usuario completamente amigables y responsivas, ya que al hacer uso del software Adobe Xd fue posible ejecutar pruebas UX necesarias que permiten identificar pequeños cambios que se desarrollaron de tal modo que no afecte la integridad de lo planificado.

Utilizar y aplicar las tecnologías Frontend permiten generar la estructura del aplicativo con las recomendaciones de buenas prácticas de cada una de las herramientas establecidas en las dependencias, también se instalaron los paquetes necesarios que agregan valor al producto ya que permiten su escalabilidad de manera correcta y eficaz.

Los módulos fueron creados de manera estructurada para lograr que el sistema sea sencillo de desarrollar a medida que incrementa su volumen.

¿Qué se aprendió en el sprint para mejorar la forma de llevar a cabo el proyecto?

Tal cual como lo recomiendan las buenas prácticas fue necesario generar las políticas de acceso, servicios de routing, y políticas por tarea asignada, con lo anteriormente mencionado fue posible asegurar las rutas de acceso al sistema para salvaguardar la integridad del sistema y la información que maneja.

2.2.4. Sprint 3

2.2.4.1. Sprint Planning

El propósito del Sprint número 3 fue continuar con el diseño de la arquitectura del sistema y de los datos que serán circunstanciales a la misma, para ello se establecieron los siguientes objetivos:

a) Implementar las funcionalidades a detalle en cada una de las historias de usuario.
b) Generar las rutas de accesos seguras para cada uno de los roles asignados en el sistema.
c) Almacenamiento de datos mediante el contenedor de AWS.
d) Implementar a detalle las funcionalidades para cada uno de los roles del sistema.
e) Ingresar datos de manera masiva para realizar las pruebas respectivas.
f) Generar las tablas polimórficas necesarias en la base de datos.
g) Migrar la base de datos a Hostinger.
h) Realizar la migración al repositorio AWS, mediante Docker y GitHub Actions.
i) Desarrollar las funcionalidades de la aplicación nativa e instalar las dependencias necesarias.
j) Realizar las configuraciones para desplegar el aplicativo nativo en Firebase.

2.2.4.2. Sprint Backlog

Para el Sprint número 3, se tomaron en cuenta las historias de usuario, mismas que están descritas en el Anexo Nro. 5. Para ello se definieron criterios de aceptación y tareas a realizar.

2.2.4.3. Ejecución del Sprint

En el presente Sprint se desarrollaron las funcionalidades de cada uno de los roles de la aplicación, teniendo en cuenta los actores principales: Administrador de la Aplicación, Colaborador, Docente, Estudiante y Root.

El resultado esperado del Sprint número 3 fue implementar las funcionalidades de los roles de la aplicación, al realizar esto se logró obtener flujos de trabajo en los cuales cada rol tiene ciertas funcionalidades y privilegios dentro de la aplicación. Las tareas que destacan fueron:

- Realizar las funcionalidades del estudiante en las áreas asignadas, con interfaces amigables y responsivas.
- Implementar servicio de correos para emitir detalles de alguna tarea establecida.
- Implementar flujos de trabajo para descargar contenido abierto y reporte.

- Realizar las funcionalidades del docente en las áreas asignadas, con interfaces de usuario y responsivas
- Implementar servicios AWS para almacenamiento de contenido.
- Realizar las funcionalidades del usuario administrador en las áreas asignadas, con interfaces amigables y responsivas.
- Implementar las políticas de acceso para el consumo de datos en GitHub, OSF, Google y otros servicios a usar.
- Implementar los elementos necesarios para navegar en el Backend.
- Implementar políticas de tareas asíncronas para emitir correos de revisión.
- Implementar una plantilla por defecto para el Backend de la aplicación, para ello se instalarán los componentes necesarios Bootstrap.
- Implementar funcionalidades para realizar reportes de cada una de las zonas de navegación del Backend.

Flujo de trabajo estudiante

El propósito del presente apartado fue crear el flujo de trabajo y desarrollar las funcionalidades necesarias para lograr tal fin. Teniendo en cuenta lo anteriormente mencionado se levantó el proceso del Estudiante en la aplicación, ver Fig. 61.

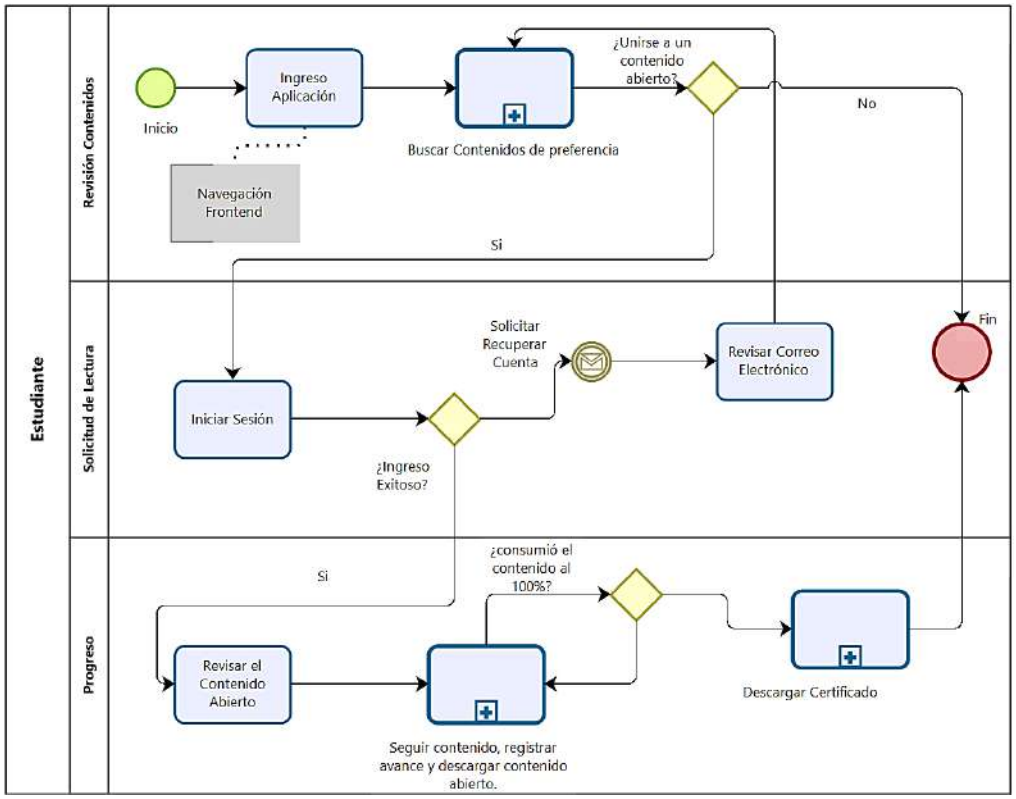


Fig. 61. Mapa de Procesos Global - Estudiante.

Fuente: Propia.

Una vez establecido el proceso se procedió a desarrollar las funcionalidades, es necesario mencionar que una vez un usuario cree una cuenta la aplicación le asignará una cuenta de Estudiante, dicho rol tiene acceso a los contenidos abiertos que se encuentran publicados o Golden Road (Open Access), ver Fig. 62.

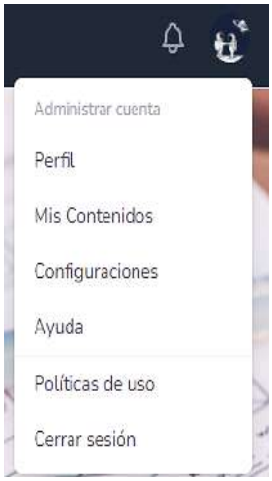


Fig. 62. Menú de Navegación, estudiante.

Fuente: Propia.

El estudiante debe acceder a la página Contenidos Open Science para escoger el contenido que sea de su interés con los filtros personalizados. En la vista general el estudiante podrá visualizar los detalles del contenido abierto, ver Fig. 63.

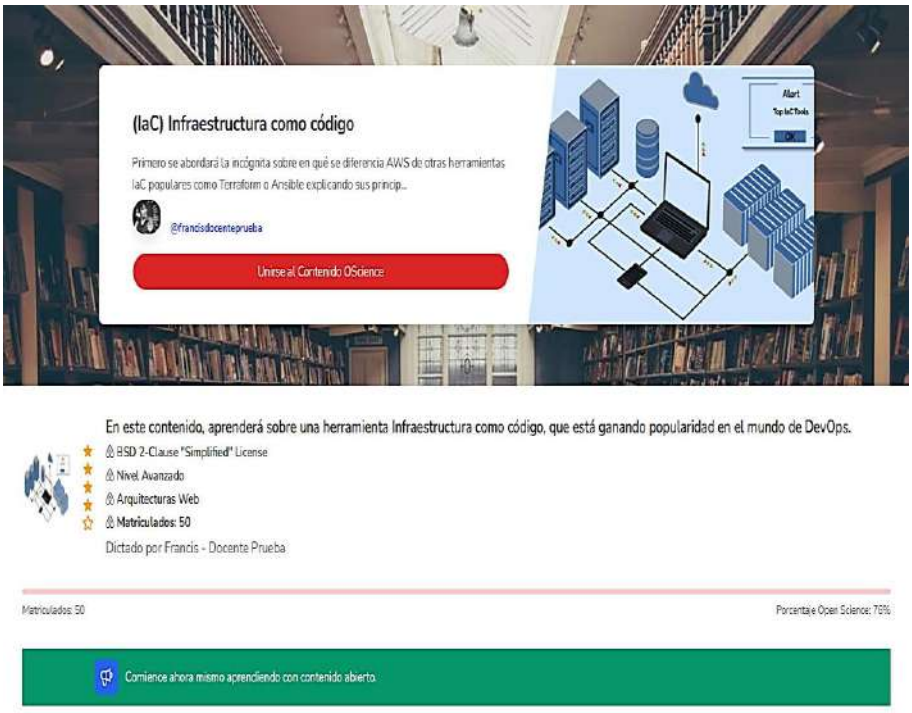


Fig. 63. Unión a contenido específico, estudiante.

Fuente: Propia.

En la Fig. 64, se visualiza los detalles del contenido, cabe recalcar que la entidad contenido abierto está relacionada a otras tablas que contienen los recursos abiertos.

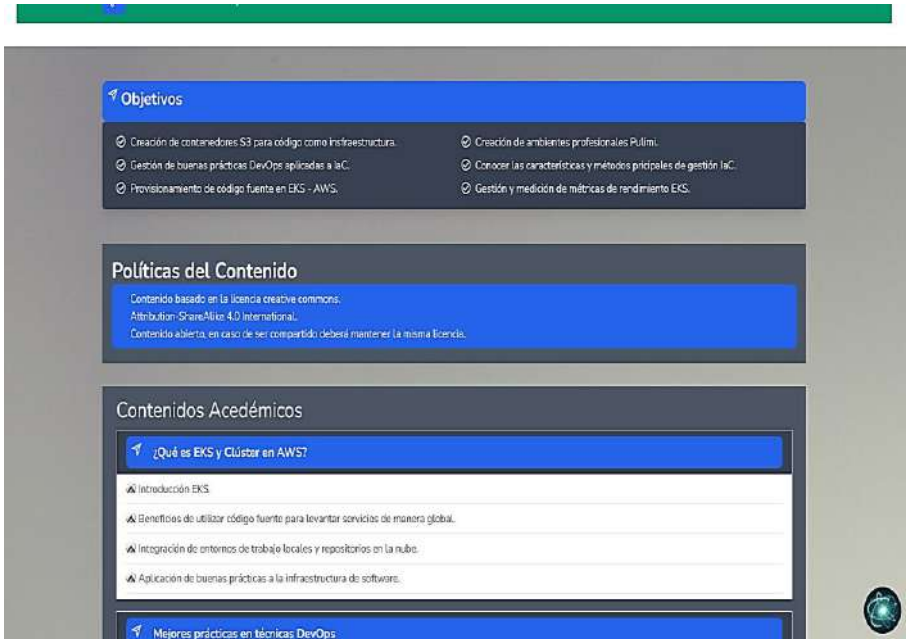


Fig. 64. Detalle en un contenido específico, estudiante.

Fuente: Propia.

Para obtener una retroalimentación general de aquellas opiniones que han dado los estudiantes del contenido, será necesario ir a la zona de comentarios. Por efecto de prueba en la Fig. 65, se visualiza como el estudiante deberá dejar un comentario en el contenido.

Fig. 65. Comentario establecido en un contenido específico, estudiante.

Fuente: Propia.

En la Fig. 66, se visualiza el semáforo de navegación:

- Color Rojo: Contenido que aún no es revisado por el usuario.
- Color Verde: Contenido que ya fue marcado como revisado por el usuario.
- Color Azul: Contenido que está siendo revisado por el usuario.

Fig. 66. Plantilla de un contenido abierto específico, estudiante.

Fuente: Propia.

En cada actividad el docente adjuntará contenido en distintos formatos multimedia, al dar clic en el elemento de un tema se desplegará el contenido en la parte superior y se habilitará un botón que abrirá un enlace al documento abierto, ver Fig. 67.

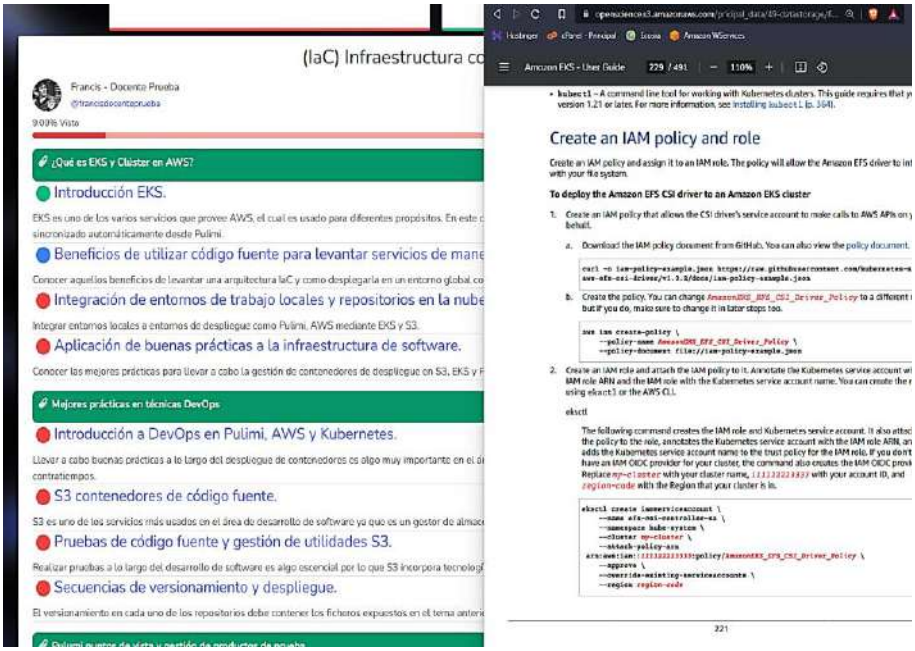


Fig. 67. Descargar ficheros de un contenido abierto específico, estudiante.

Fuente: Propia.

Después que el estudiante haya consumido todo el contenido con un determinado número de temas y el porcentaje de revisión del contenido haya llegado al 100% se habilitará un botón “Descargar Certificado” y se generará el certificado en formato PDF, ver Fig. 68.

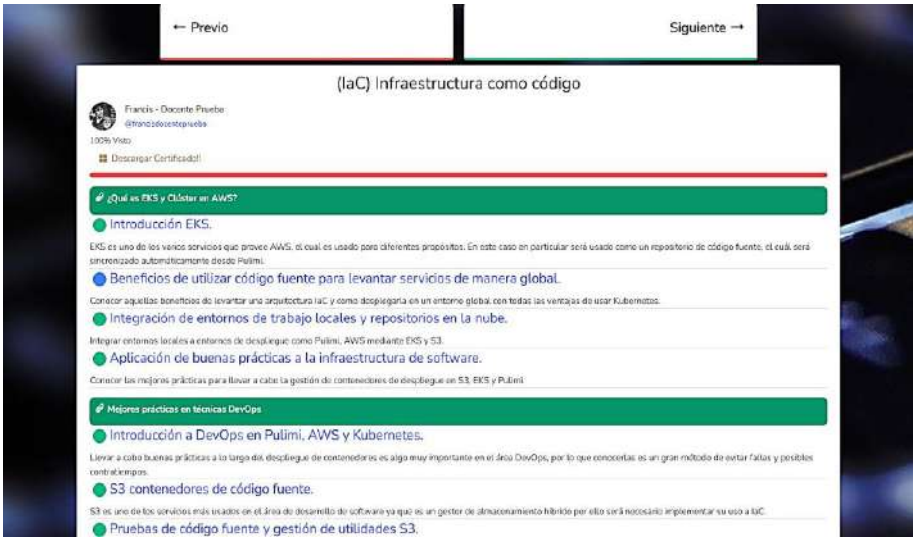


Fig. 68. Porcentaje de aprendizaje de un contenido específico, estudiante.

Fuente: Propia.

En la Fig. 69 se visualiza la simulación en el que un estudiante ha concluido el contenido como se evidencia todas las temáticas están en color verde lo que muestra que es posible descargar el certificado en formato PDF.

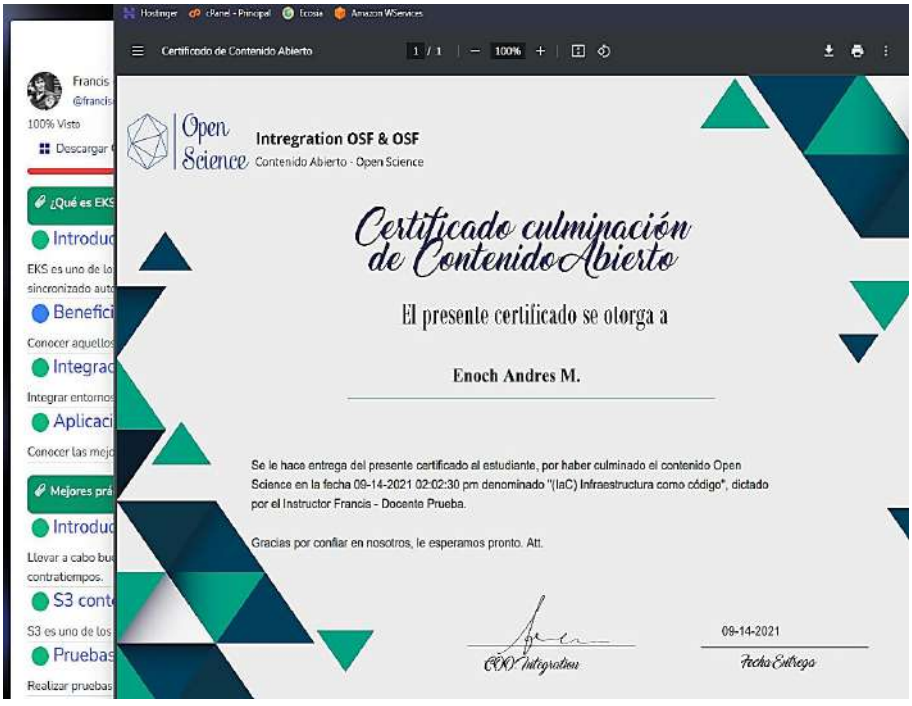


Fig. 69. Certificado PDF de un contenido específico, estudiante.

Fuente: Propia.

El flujo de trabajo del Estudiante contiene otras funcionalidades que son de índole de acceso, configuración de almacenamiento, y otros. Sin embargo, fueron omitidas para optimizar la presentación del resultado.

Flujo de trabajo docente

El propósito del presente apartado fue crear el flujo de trabajo y desarrollar las funcionalidades necesarias para lograr tal fin. Teniendo en cuenta lo anteriormente mencionado se levantó el proceso del Docente en la aplicación. Es necesario recalcar que se hará la documentación completa de todas las funcionalidades en los anexos de manual de usuario, por ello en este bloque se hará hincapié en aquellas tareas que son:

- Levantar los procesos del usuario docente.
- Crear las interfaces responsivas para la administración de contenidos por docente.
- Crear las políticas de acceso a los repositorios de almacenamiento abierto.
- Hacer cambios a la imagen docker con el propósito de generar nuevas políticas de almacenamiento en AWS.
- Generar los reportes necesarios de los datos que manipula el usuario.

Se levantó el siguiente flujo de trabajo del docente, es necesario recalcar que durante la publicación de contenido el proceso se vuelve iterativo, por ello se omitieron funcionalidades las cuales no están relacionadas al contenido que publica el docente, ver Fig. 70.

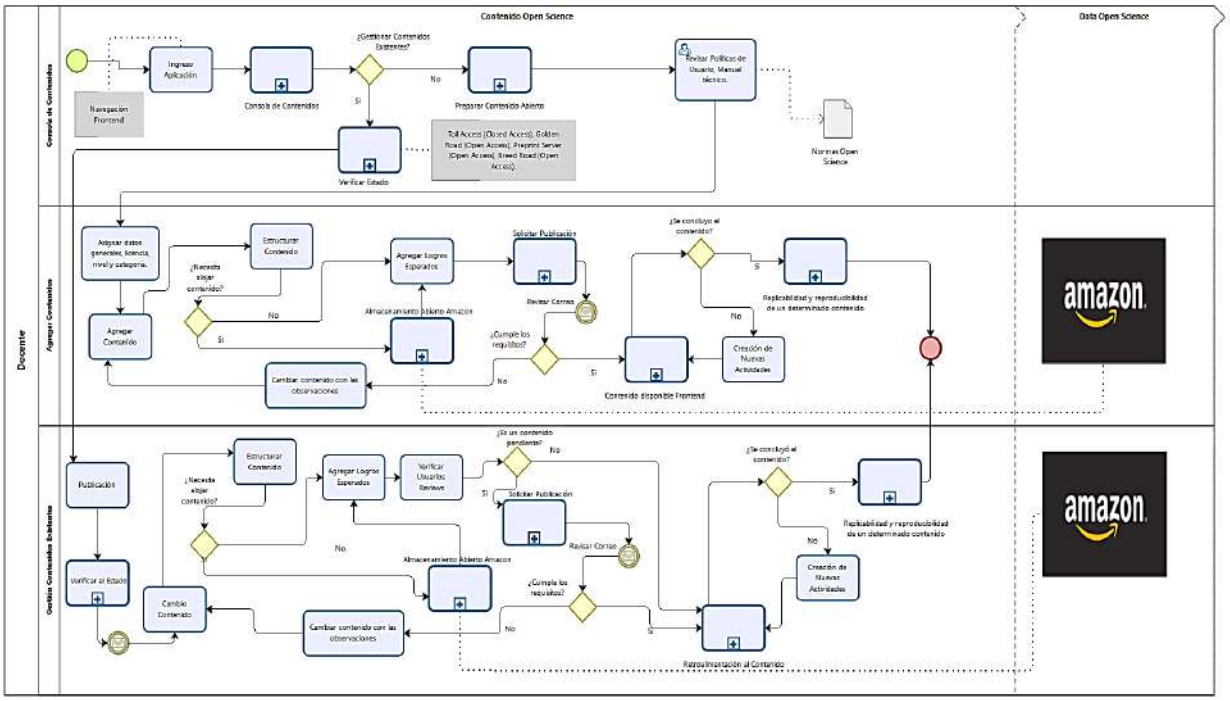


Fig. 70. Mapa de Procesos Global - Docente.

Fuente: Propia.

Fig. 71. Menú de navegación, docente.

Fuente: Propia.

El flujo de trabajo Open Science establece cuatro posibles estados para cada determinado contenido, por ello se lo ha representado de la siguiente manera:

- Color Rojo – Toll Access (Closed Access).
- Color Verde – Green Road (Open Access).
- Color Amarillo – Golden Road (Open Access).
- Color Gris Suave – Preprint Server (Open Access).

El usuario docente podrá visualizar todos los contenidos que ha creado una vez haya ingresado a la consola de contenidos, ver Fig. 72. Los contenidos poseen dos iconos en cada una de las imágenes de las fichas; el icono azul muestra que el contenido se encuentra en las tres primeras fases y el de color verde ha terminado el proceso Open Access por ello se encuentra totalmente abierto para la replicabilidad, reutilización y revisión.

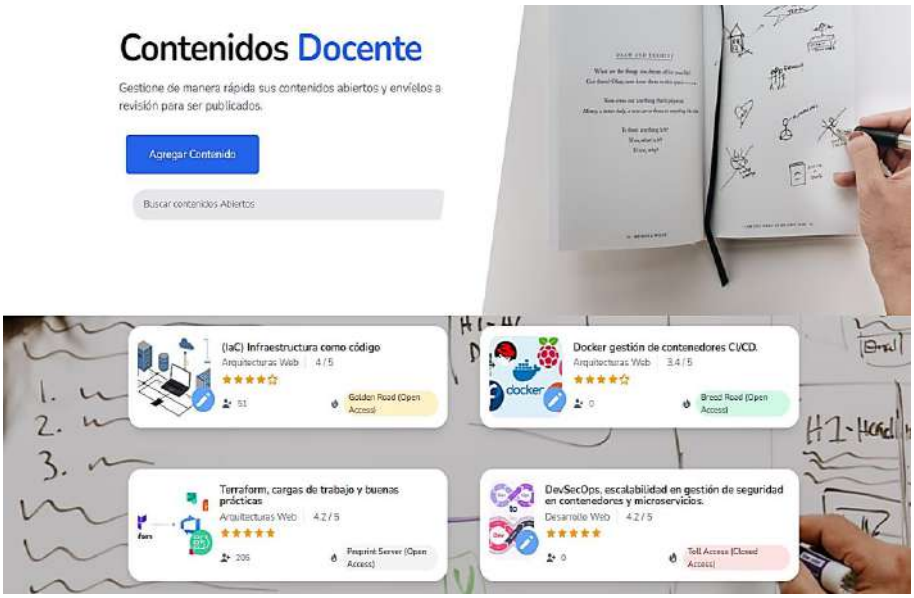


Fig. 72. Consola de contenidos, docente.

Fuente: Propia.

Si el usuario opta por el botón “Agregar Contenido Abierto”, de desplegará el siguiente formulario en el cual deberá agregar el contenido preliminar del contenido, ver Fig. 73.

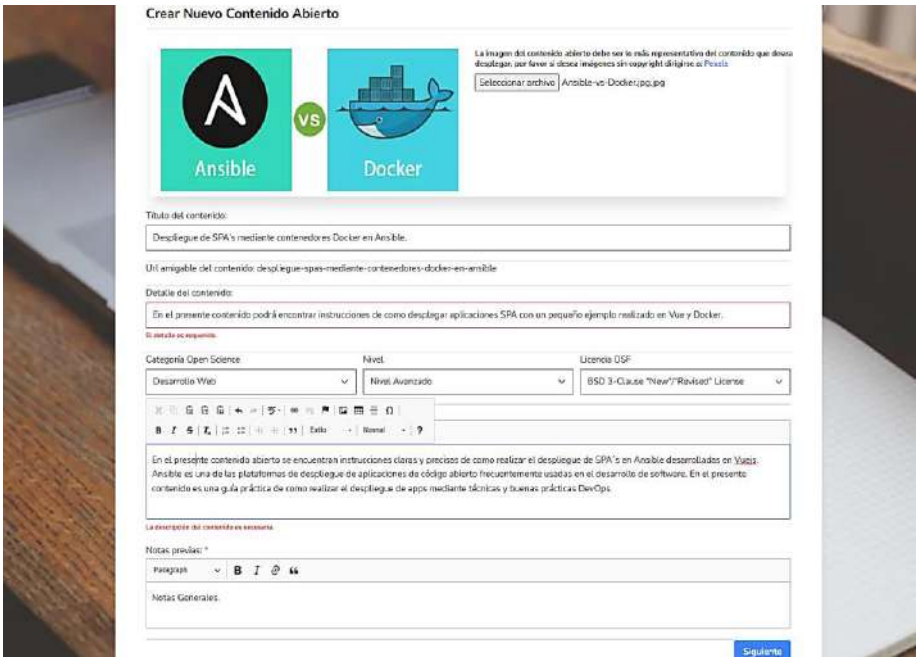


Fig. 73. Creación del contenido, docente.

Fuente: Propia.

Después de crear el contenido, se desplegará una hoja de ruta en la que el usuario podrá generar: información general, archivos, palabras clave, logros de aprendizaje, comentarios y usuarios anexos; ver Fig. 74.

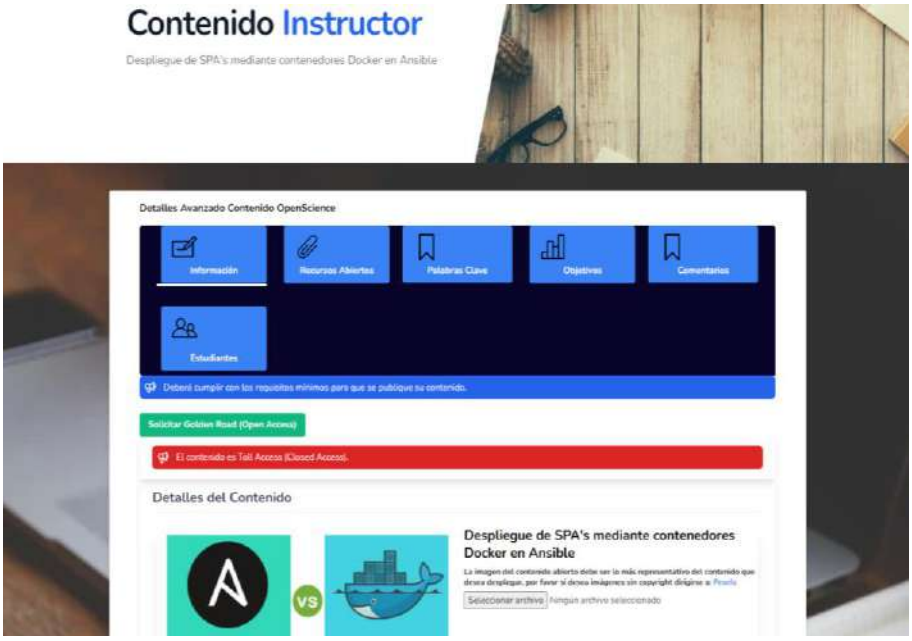


Fig. 74. Cambiar detalles del contenido, docente.

Fuente: Propia.

Para agregar contenido abierto será necesario dar clic en “Recursos abiertos” lo que hará posible subir archivos independientemente del formato de que se trate, ver Fig. 75.

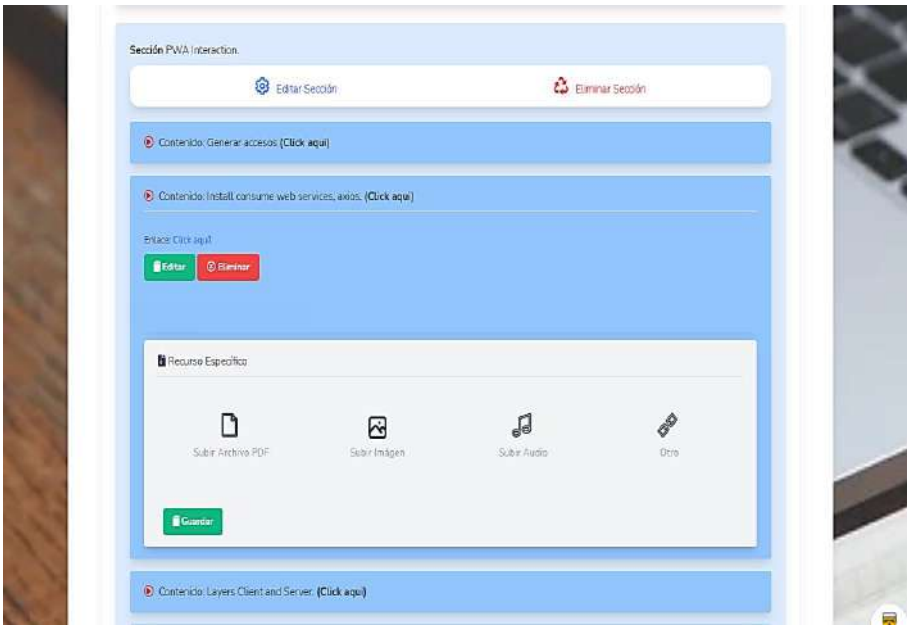


Fig. 75. Agregar ficheros al contenido abierto, docente.

Fuente: Propia.

Si el caso requiere que el usuario guarde un recurso PDF se abrirá la pantalla del explorador de archivos, ver Fig. 76.

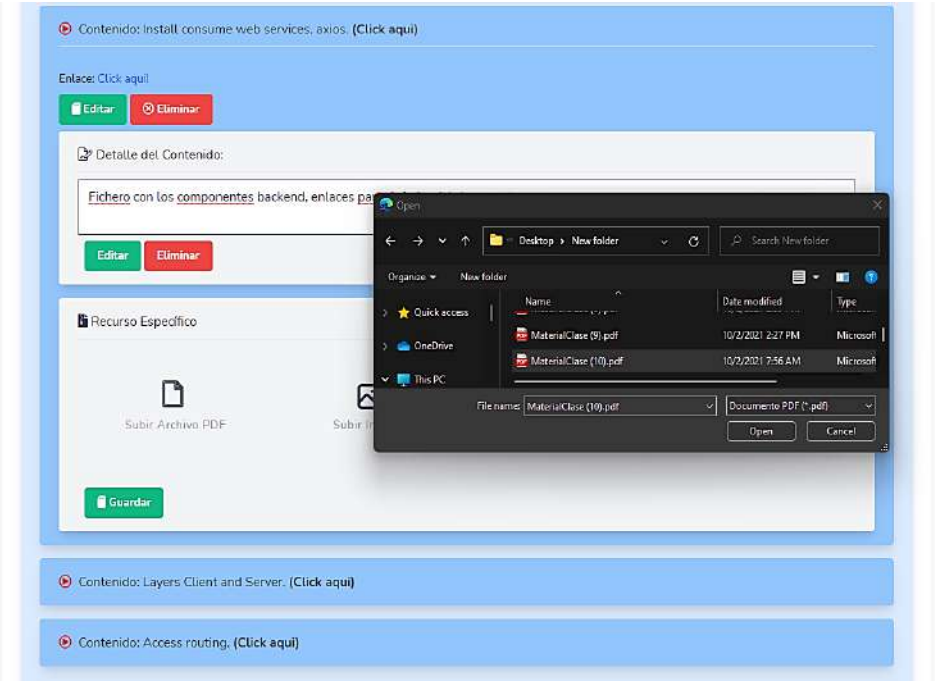


Fig. 76. Subir archivos formato PDF, docente.

Fuente: Propia.

Se ha creado un elemento llamado “Agregar Sección”, en el cual el docente deberá estructurar el contenido de manera ordenada y cronológica, ver Fig. 77.

En las historias de usuario y requisitos del presente documento se describen los campos específicos que deben ser completados en los formularios anidados.

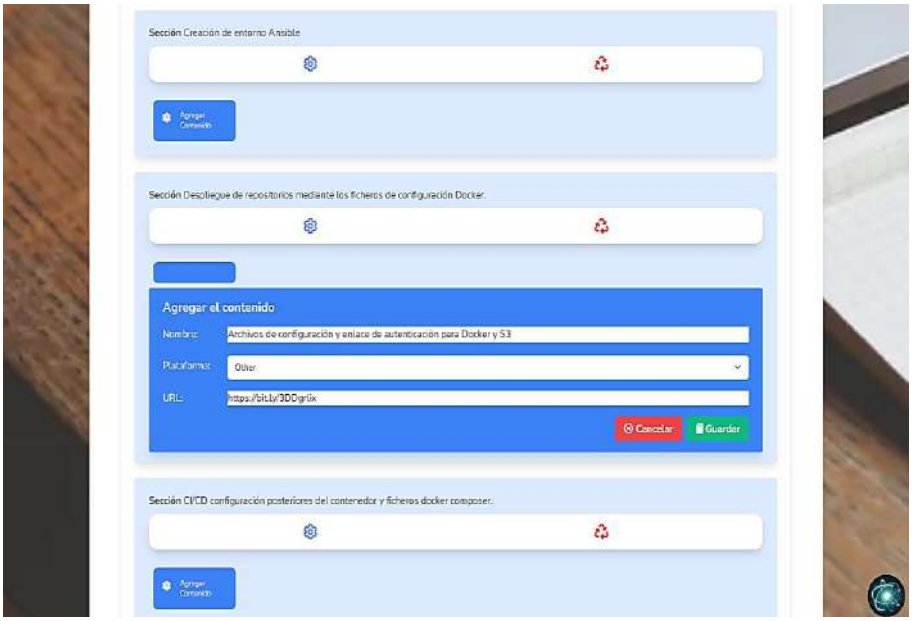


Fig. 77. Estructura del contenido abierto, docente.

Fuente: Propia.

Si el caso requiere subir ficheros de formatos diferentes a los tres elementos iniciales, el usuario puede subir contenido con el elemento “Otro”, ver Fig. 78.

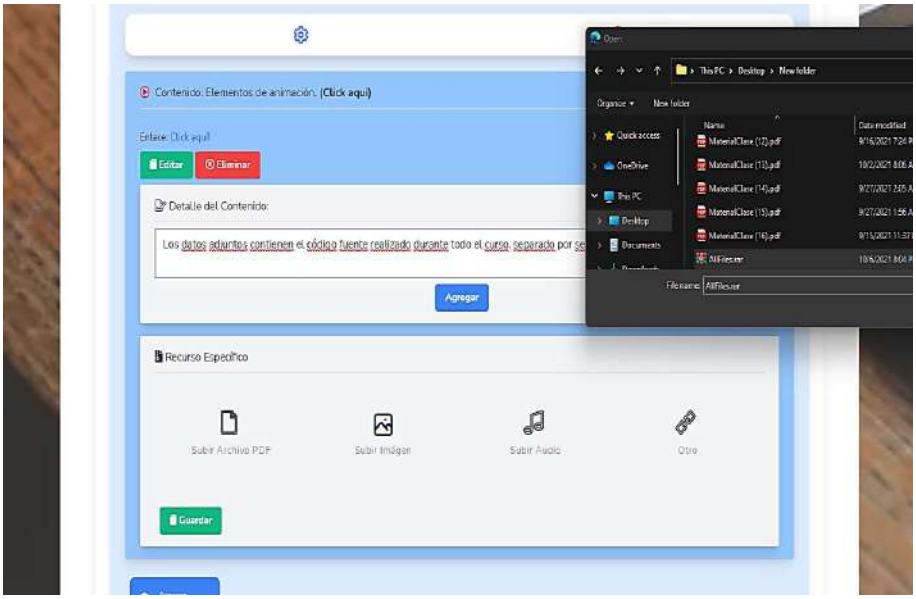


Fig. 78. Subir archivos formato no definido, docente.

Fuente: Propia.

Los elementos serán mostrados en una ventada del navegador, cabe recalcar que en el enlace del recurso se puede visualizar el contenedor de archivos abiertos en AWS y la capa de seguridad establecida en las políticas de acceso en S3, ver Fig. 79.

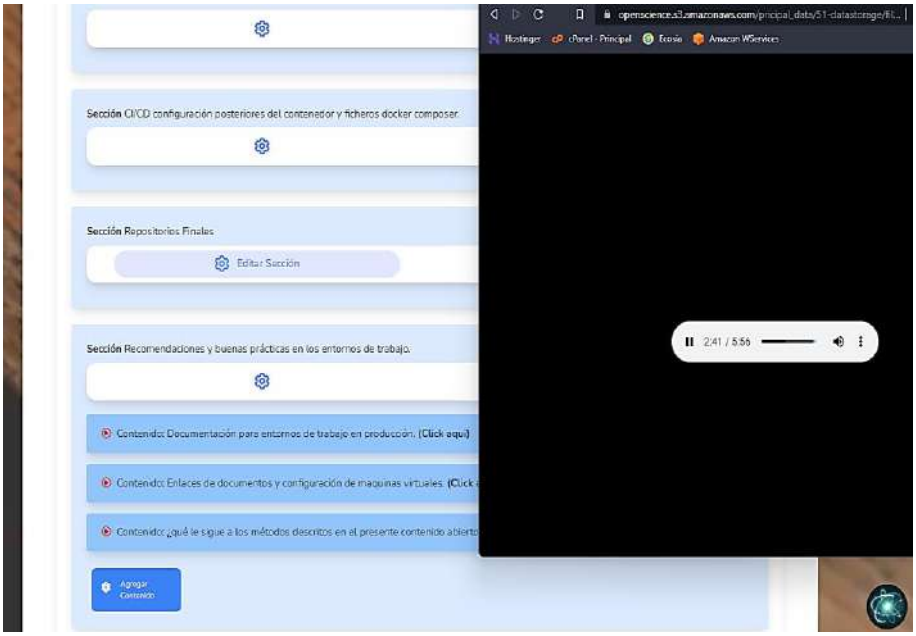


Fig. 79. Reproducción de contenido, docente.

Fuente: Propia.

El usuario deberá estructurar el contenido abierto en base a los principios Open Science, para efectos de optimización se han omitido varias funcionalidades que requieren completar varios pasos para dar por finalizado el proceso global del Docente.

La publicación del contenido abierto se realizará mediante el consumo de servicios de Amazon S3; para ello el usuario deberá estructurar el contenido abierto y adjuntar ficheros que se irán almacenando en un nodo de almacenamiento, ver Fig. 80.

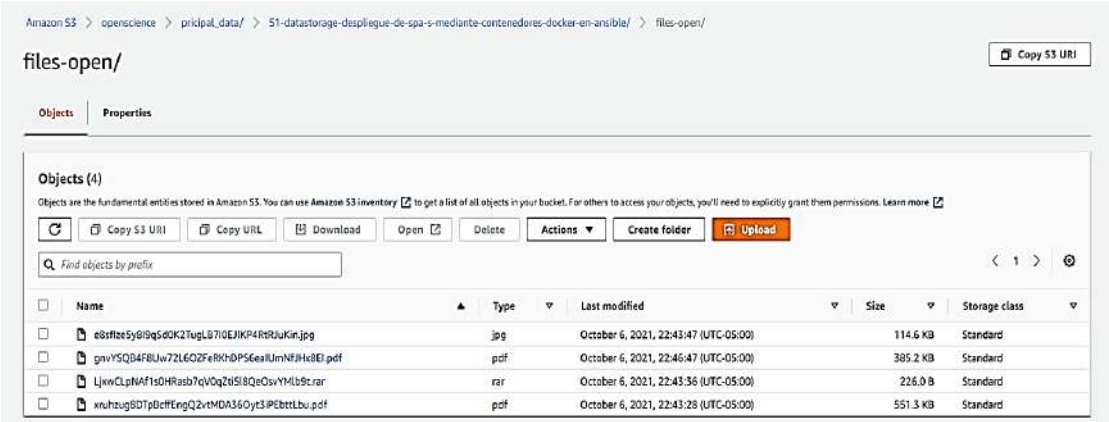


Fig. 80. Árbol de almacenamiento híbrido, AWS.

Fuente: Propia.

Después de realizar los cambios necesarios el usuario podrá visualizar el nuevo contenido agregado a la consola de contenidos abiertos y por ende después de que sea aprobado por el Administrador pasará a ser Golden Road, ver Fig. 81.

Fig. 81. Consola contenidos abiertos actualizada, docente.

Fuente: Propia.

El contenido abierto enviado a ser revisado por el administrador de la aplicación deberá cumplir con todos indicadores necesarios, de no ser así el ciclo seguiría en el mismo estado, por lo que el docente debe modificar el contenido y enviarlo nuevamente a estado Green Road.

En la Fig. 82, se puede visualizar la última integración realizada durante la etapa de desarrollo de los módulos relacionados a la gestión del contenido. Tal como se observa en la figura, el despliegue no ha mostrado ningún error y entre las varias tareas que ha ejecutado el contenedor se encuentran las respectivas pruebas de código, lo que quiere decir que no hay ningún problema en cuanto al aspecto de codificación, dependencia o paquete de software que pueda causar algún tipo de inconveniente.

Fig. 82. CI/CD última integración del Frontend, pruebas de calidad.

Fuente: Propia.

Flujo de trabajo Administrador

El propósito del presente apartado fue crear el flujo de trabajo del administrador y desarrollar las funcionalidades necesarias para lograr tal fin, ver Fig. 83. Una vez definidas las tareas del rol se crearon actividades a realizar para gestionar los servicios que proporciona la aplicación. La aplicación nativa se realizó de tal manera que el flujo de la información sea estable y seguro, por ello se instalaron paquetes de seguridad recomendados por Flutter y también se hizo uso de las seguridades proporcionadas por JWT; al realizar tal acción el acceso al sistema será seguro manteniendo así la integridad de la información.

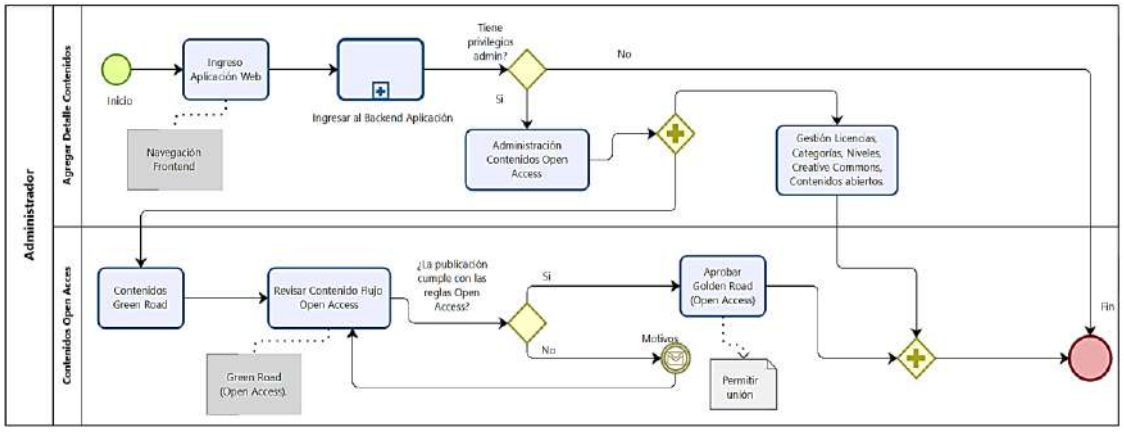


Fig. 83. Mapa de Procesos Global - Administrador.

Fuente: Propia.

Los accesos a la aplicación nativa se muestran en la Fig. 84, así mismo el usuario podrá crear una cuenta o ingresar con una existente.

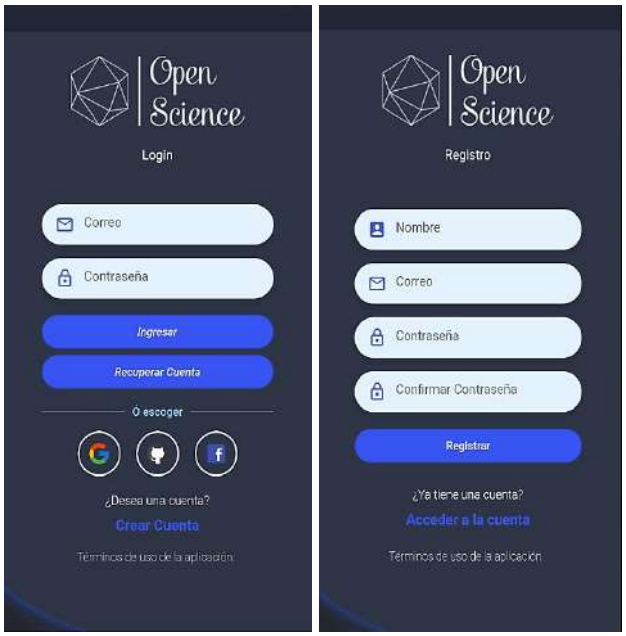


Fig. 84. Mockup ingreso aplicativo móvil, administrador.

Fuente: Propia.

Una vez abierta la aplicación nativa será posible desplegar el menú de la aplicación y la pantalla principal que contiene la información relevante del sistema, así mismo se puede visualizar las operaciones que pueden ser gestionadas con el rol establecido, ver Fig. 85.

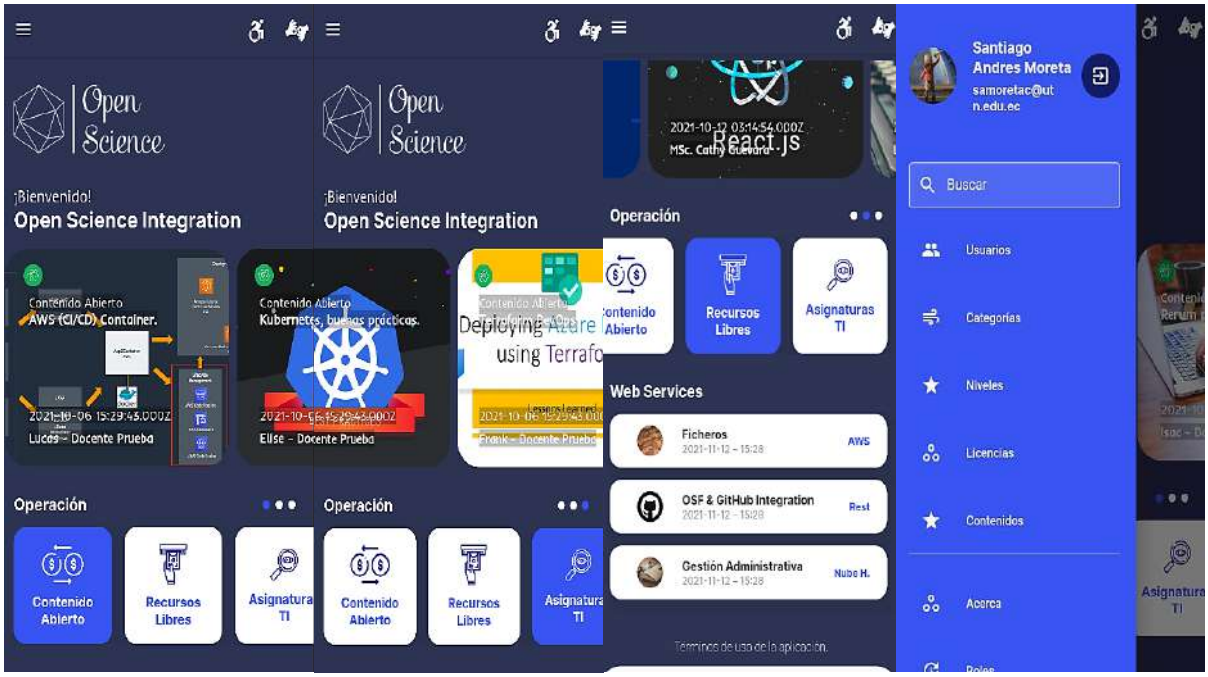


Fig. 85. Mockup inicio aplicativo móvil, administrador.

Fuente: Propia.

Para llevar a cabo la gestión de licencias se desplegará una pantalla con todas las licencias OSF disponibles, al igual que el respectivo detalle de cada una de ellas, ver Fig. 86.

Fig. 86. Mockup licencias aplicativo móvil, administrador.

Fuente: Propia.

El usuario podrá editar o agregar licencias con los campos establecidos en los criterios de aceptación de las historias de usuario del presente Sprint, ver Fig. 87.

Fig. 87. Mockup pantalla agregar - aplicativo móvil, administrador.

Fuente: Propia.

Las alertas serán mostradas con Widgets cupertino mismos que se adaptan a dispositivos Android y iOS según sea necesario, ver Fig. 88.

Fig. 88. Mockup alerta cupertino aplicativo móvil, administrador.

Fuente: Propia.

Los contenidos abiertos que se muestran en el Frontend deberán pasar varios filtros una vez que haya sido enviado y se encuentre en estado “Green Road”. En la presente simulación se procederá a revisar los datos que fueron creados en el flujo docente, ver Fig. 89.

Fig. 89.Revisión de contenidos Green Road, administrador.

Fuente: Propia.

En la Fig. 90, se visualiza el formato del mensaje que será enviado al correo del usuario Docente en caso de que el contenido abierto no haya aprobado los filtros necesarios.

Fig. 90. Términos de publicación y contenido no aprobado, administrador.

Fuente: Propia.

En la presentación de los resultados de la aplicación nativa se hizo una breve explicación de “licencias”, sin embardo la aplicación nativa tiene otras funcionalidades relacionadas a la administración de las principales entidades de cada contenido abierto y los métodos que permiten la gestión su publicación. En la aplicación nativa fueron instalados varios paquetes que permitieron consumir los servicios producto de la integración mediante JWT, ver Fig. 91.

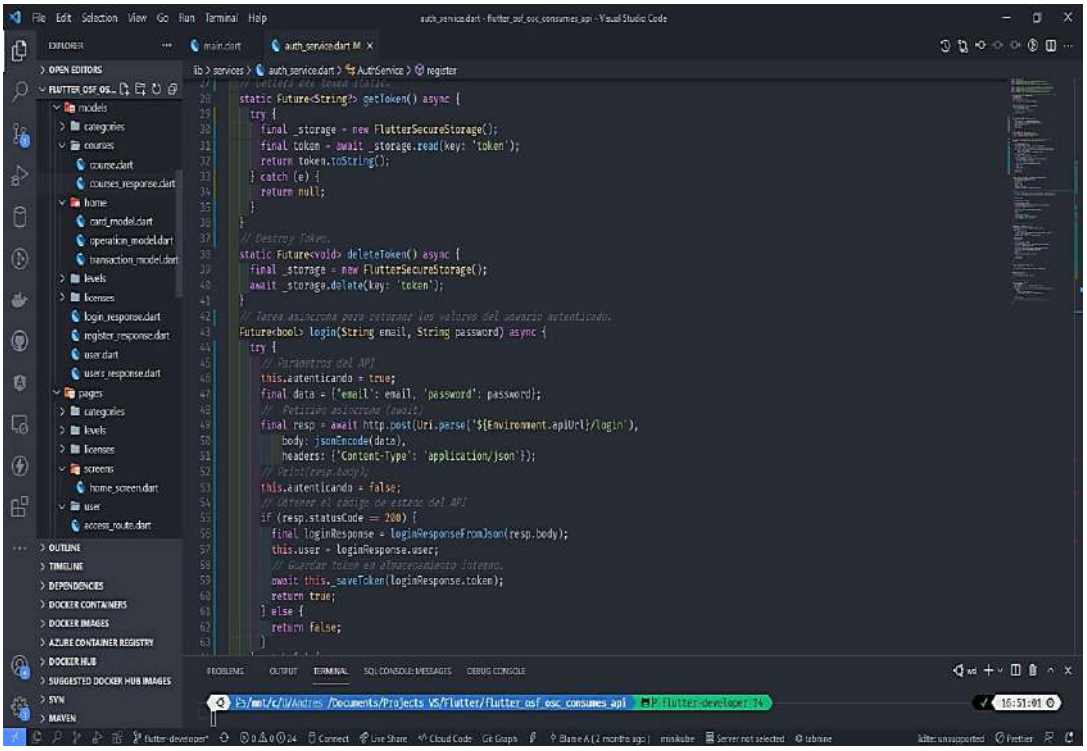


Fig. 91. Código fuente protección de acceso JWT, administrador.

Fuente: Propia.

Flujo de trabajo Root

El propósito del presente apartado fue crear el flujo de trabajo del usuario root y desarrollar las funcionalidades necesarias para lograr tal fin. Una vez definidas las tareas del rol que se expone se establecieron las actividades a ser automatizadas, ver Fig. 92.

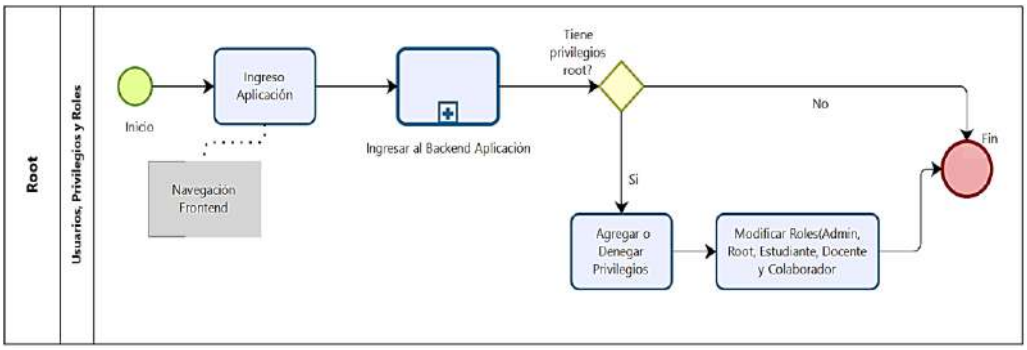


Fig. 92. Mapa de Procesos Global - Root.

Fuente: Propia

Para hacer uso de las funcionalidades de Root será necesario ingresar al Backend de la aplicación y con ello se desplegará el respectivo menú de navegación, ver Fig. 93.

Fig. 93. Menú de navegación backend, root.

Fuente: Propia.

En el submenú “Usuarios” se presenta una tabla paginada con todos aquellos usuarios registrados en la aplicación tal cual se visualiza en la Fig. 94. Usuarios con rol Administrador no podrán ser eliminados, pero si modificados. El usuario Root no podrá ser modificado ni eliminado al ser un usuario sumamente necesario.

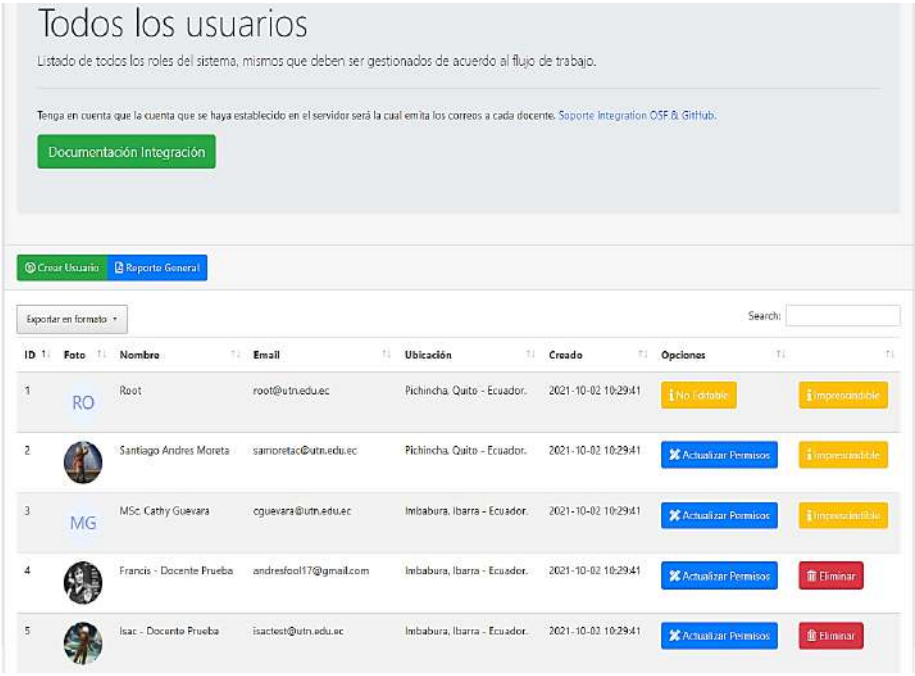


Fig. 94. Lista paginada todos los usuarios, root.

Fuente: Propia.

A medida que el volumen del proyecto aumentaba al implementar las diversas funcionalidades de cada rol del sistema, fue necesario establecer las rutas de acceso de cada módulo para un determinado usuario. Se aplicó una estructura basada en privilegios dentro del sistema; al realizar tal acción será posible mantener la integridad de la información y asegurar las rutas de acceso a cada funcionalidad, ver Fig. 95.

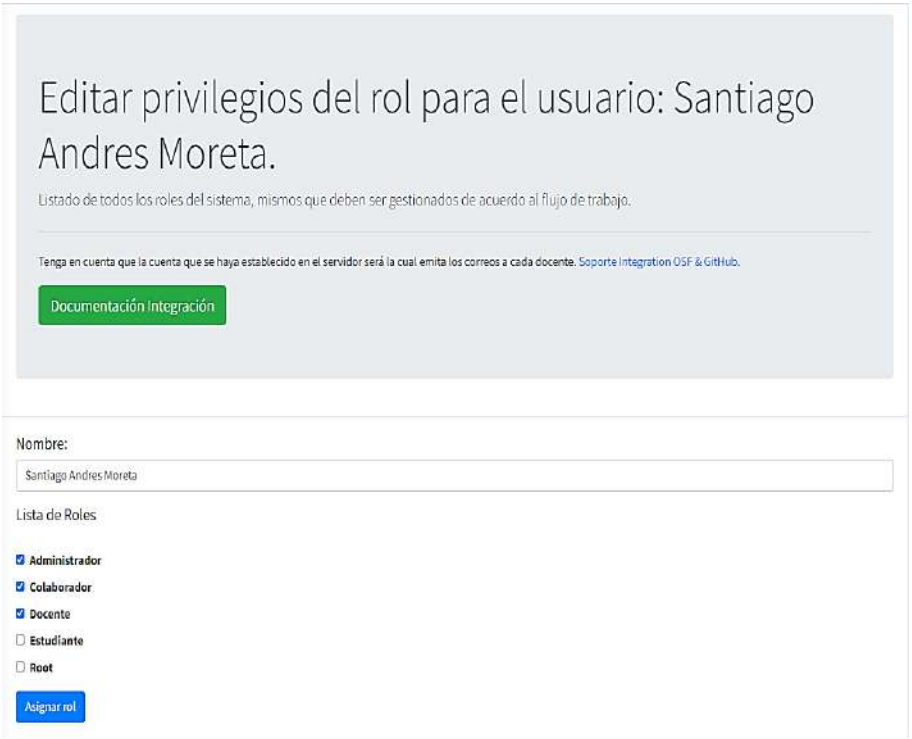


Fig. 95. Asignación de roles a usuarios, root.

Fuente: Propia.

En la Fig. 96, se puede visualizar la pantalla que está validada con modales que muestran alertas de acuerdo con cada acción que se encuentre realizando el usuario autenticado, dichas alertas son mostradas en casos tales como: agregar datos, actualizar datos, eliminar datos, y dependencia de datos que por defecto no sea posible eliminar.

Fig. 96. Modales por tareas, root.

Fuente: Propia.

En la Fig. 97, se muestran los cinco roles establecidos en la aplicación con sus respectivos privilegios, sin embargo, es posible aumentar otros roles para versiones posteriores.

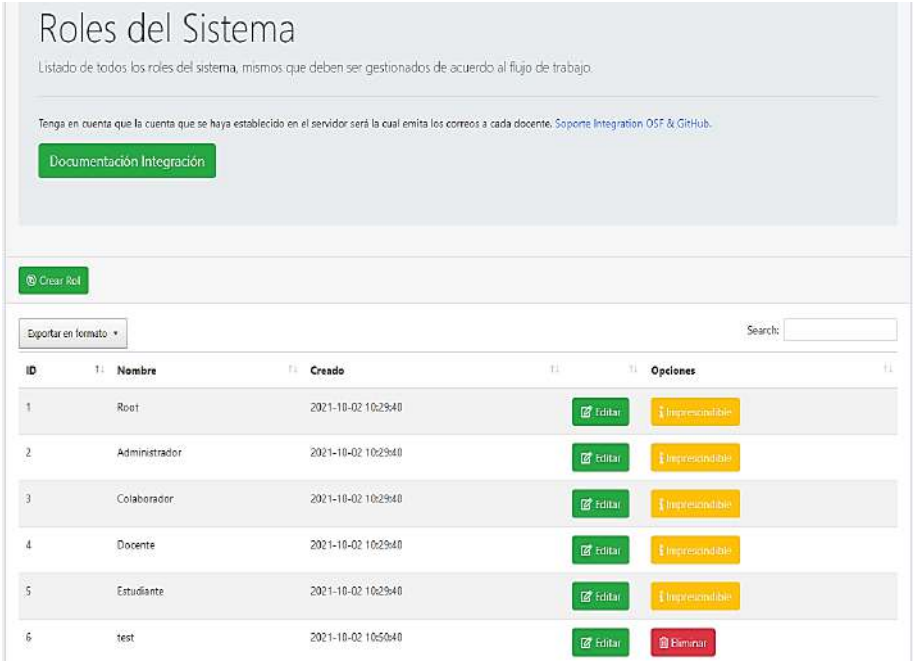


Fig. 97. Vista de roles, root.

Fuente: Propia.

En la Fig. 98, se muestra la lista de privilegios establecidos por cada rol del sistema.

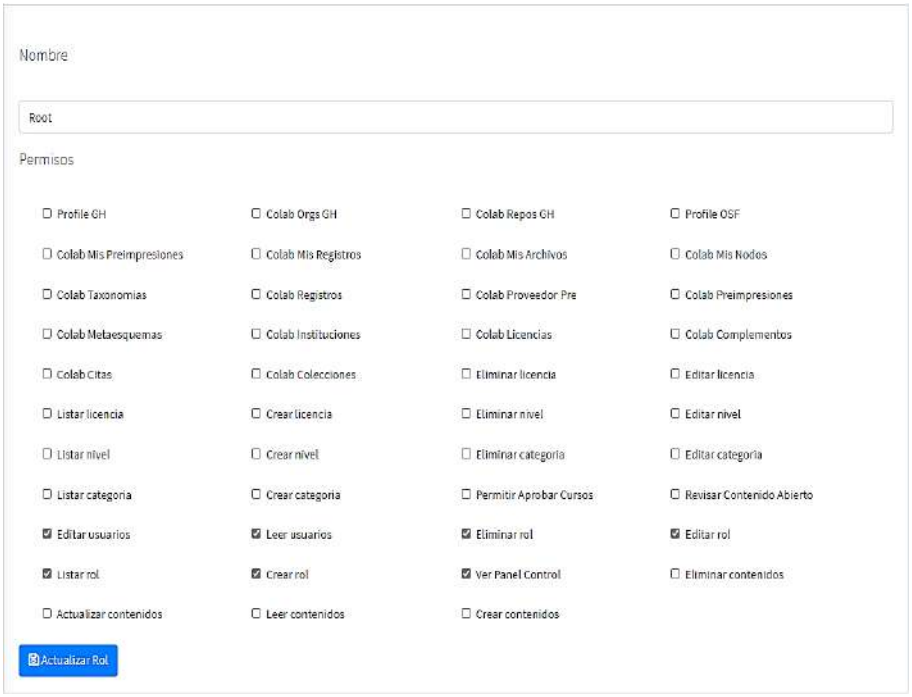


Fig. 98. Asignación de privilegios por rol, root.

Fuente: Propia.

Independientemente del rol que posea el usuario, si este intenta ingresar a rutas de acceso a funcionalidades a las que no se le ha dado privilegios se mostrará una pantalla mostrando el error de acceso denegado, ver Fig. 99.

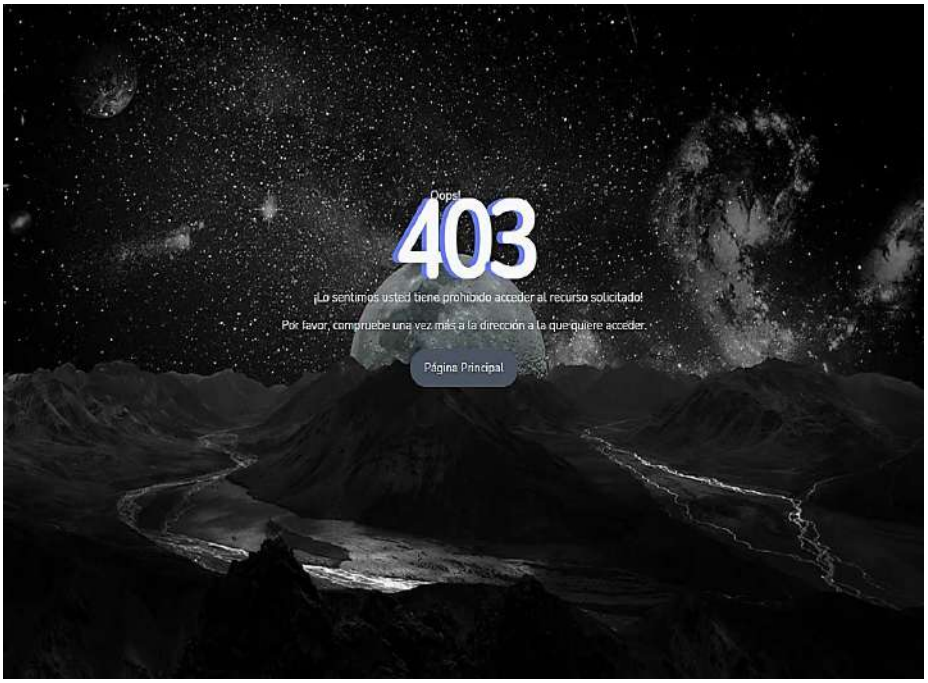


Fig. 99. Página Acceso no Autorizado - roles.

Fuente: Propia.

En la Fig. 100, se visualiza el reporte general de usuarios de la aplicación que contiene los datos principales y los roles que desempeña.

ID	Nombre	Email	Título	Roles
1	Root	root@utn.edu.ec	Súper Administrador	Root -
2	Santiago Andres Moreta	samoretac@utn.edu.ec	Estudiante UTN	Administrador - Colaborador - Docente -
3	MSc. Cathy Guevara	cguevara@utn.edu.ec	Ing. en Sistemas Computacionales	Docente -
4	Francis - Docente Prueba	francistest@utn.edu.ec	Ing. en Sistemas Computacionales	Docente -
5	Isac - Docente Prueba	isactest@utn.edu.ec	Ing. en Sistemas Computacionales	Docente -

Fig. 100. Reporte usuarios PDF, root.

Fuente: Propia.

Flujo de trabajo Colaborador

Por último, se creó las funcionalidades necesarias para la revisión de los componentes que conforman la arquitectura SOA: código fuente, mockups web y aplicación nativa, citas bibliográficas, bizagi process, contenido Open Science, métricas ISO/IEC 25022, etc.

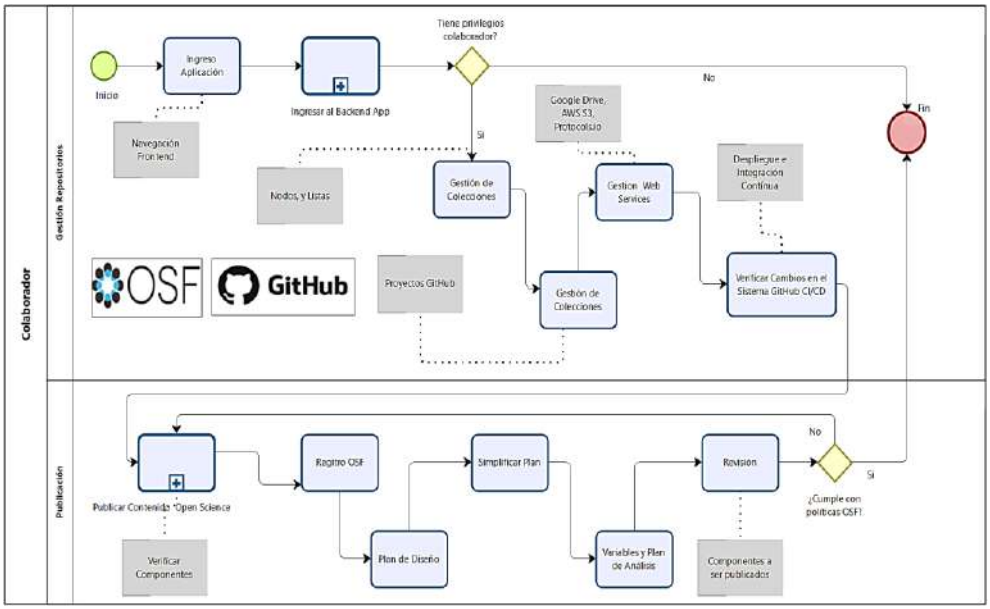


Fig. 101. Mapa de Procesos Global - Colaborador.

Fuente: Propia.

Workflow Final AWS – CI/CD

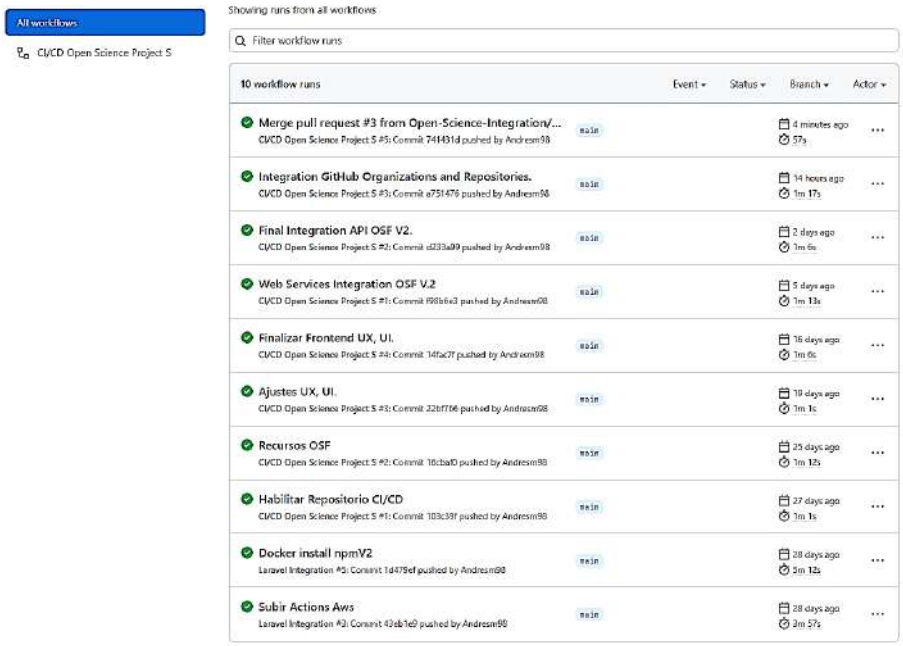


Fig. 102. CI/CD última integración de la aplicación.

Fuente: Propia.

2.2.4.4. Sprint Review

Tabla 11. Sprint Review 3.

Código	Título	Cumple
OSI.24	Al ser usuario administrador, es posible acceder a todos los componentes, repositorios, documentos de investigación.	Si
OSI.25	Al ser usuario administrador, se puede gestionar los usuarios, licencias, categorías, niveles, creative commons, publicaciones, etc.	Si
OSI.26	Al ser usuario administrador es posible aprobar contenido Open Science en la página principal.	Si
OSI.27	Al ser usuario docente es posible editar el perfil que mostrará a los estudiantes de sus contenidos abiertos.	Si
OSI.28	Al ser usuario docente es posible crear contenidos abiertos dada la categoría, nivel, licencias, etc.	Si
OSI.29	Al ser usuario docente es posible agregar un conjunto de contenido a partir de la temática.	Si
OSI.30	Al ser usuario docente es posible publicar el contenido a los alumnos.	Si
OSI.31	Al ser usuario estudiante es posible presentar matrícula a un determinado y hacer uso de las funcionalidades.	Si
OSI.32	Los datos que se consumen a través del API Backend, OSF y GitHub se reflejan automáticamente en el contenedor AWS	Si
OSI.33	La aplicación orientada a servicios es subida al clúster de aplicaciones AWS con el script Docker y GitHub Actions.	Si
OSI.34	Se instalan los paquetes necesarios para el aplicativo mediante el fichero yml.	Si
OSI.35	Integración de las entidades colecciones de OSF.	Si
OSI.36	Integración de las entidades nodos de OSF.	Si
OSI.37	Integración de las entidades repositorios de OSF.	Si
OSI.38	Integración de las entidades citaciones, preimpresiones, registros OSF.	Si
OSI.39	Integración de elementos usuario OSF.	Si
OSI.40	Gestionar componentes OSF y GitHub.	Si

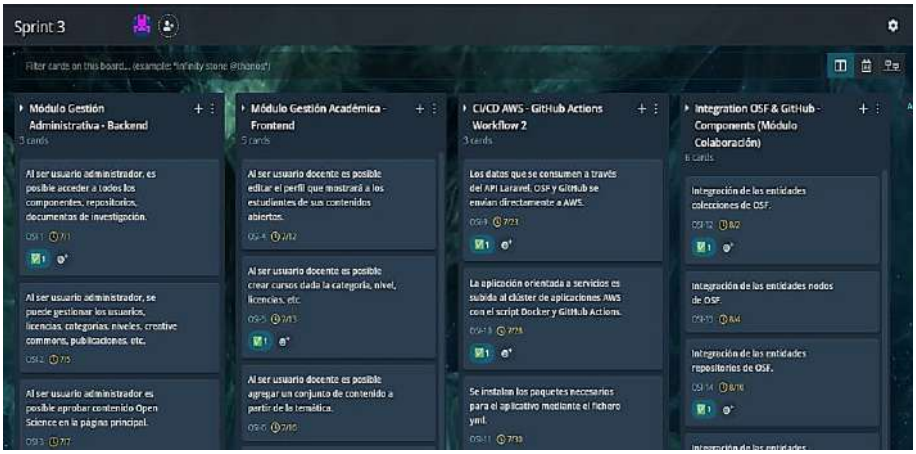


Fig. 103. GitKraken Board - Sprint 3.

Fuente: Propia.

2.2.4.5. Sprint Retrospective

¿Qué salió bien en el sprint?

En todas las etapas del desarrollo del presente sprint fue posible implementar de manera eficaz todos aquellos componentes necesarios, del mismo modo se hizo uso de las herramientas de cada una de las nubes privadas AWS, GitHub, OSF, Google y otras. La

arquitectura fue desplegada bajo CI/CD en la última versión, en esta ocasión en el repositorio Docker se instalaron componentes que permiten validar el código fuente.

El uso de Docker generó varias ventajas entre las que destacan el tener un entorno de desarrollo controlado ya que al tener las imágenes de versionamiento resulta eficaz el implementar nuevas versiones en el clúster y repositorio AWS. Sin embargo, para mantener la compatibilidad en todas las dependencias es necesario revisar la documentación de cada herramienta; el modelo de la base de datos final se adjuntó en el Anexo. 7.

¿Qué se aprendió en el sprint para mejorar la forma de llevar a cabo el proyecto?

Durante el desarrollo fue necesario implementar componentes para lograr que cada uno de los roles sean intuitivos, se hizo uso de estándares de buenas prácticas UI que recomienda la documentación de cada una de las herramientas. A medida que la aplicación y su arquitectura se vuelve robusta se hace evidente que usar tecnologías, frameworks, web services y otras herramientas de vanguardia permiten dar un mayor valor al producto.

2.3. Consumo de las APIs Rest de OSF y GitHub

Después de dar por concluida la integración de los servicios de las dos plataformas expuestas se realizó el consumo de los datos, componentes, código fuente y demás artefactos producto del presente proyecto, para ello se hará uso de los tokens personales de cada usuario.

Flujo de trabajo Colaborador - Ejecución

Para gestionar los componentes de OSF & GitHub, fueron desarrollados dos objetos macro que permitirán la gestión de cada plataforma por separado, ver Fig. 104.

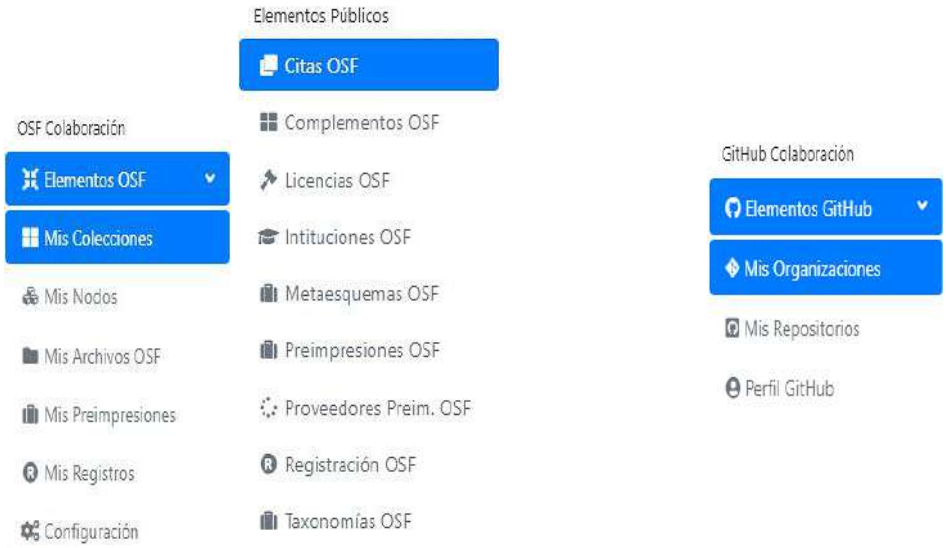


Fig. 104. Menú de navegación Backend, colaborador.

Fuente: Propia.

El usuario deberá solicitar los accesos pertinentes de cada una de las plataformas y tener los privilegios en los repositorios de la presente investigación. Al realizar tal acción se le permitirá el acceso a todos los componentes que conforman cada uno de los repositorios con la licencia establecida en OSF mediante el uso del token.

OSF Colaboración

Para que el usuario pueda gestionar los componentes del repositorio OSF es necesario que se le asigne los privilegios respectivos, ver Fig. 105.

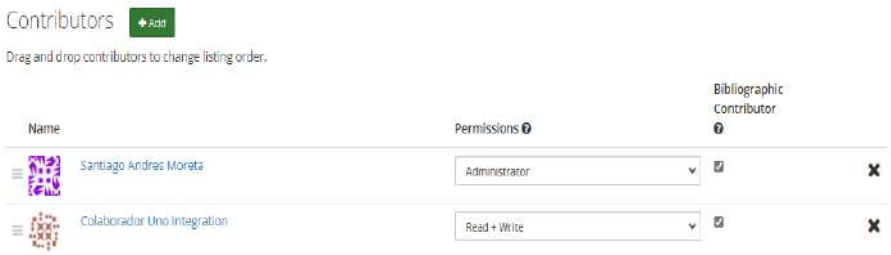


Fig. 105. Accesos al repositorio OSF, colaborador.

Fuente: Propia.

Elementos Públicos

Los complementos públicos de OSF son accesos a varios servicios entre los que destacan Dataverse, Dropbox, Figshare, GitLab, Mendeley, OneDrive, BitBucket, Box y otros. Dichos contenedores deben ser habilitados con un token de acceso en OSF, en varios casos es necesaria una configuración avanzada en el servicio de origen, ver Fig. 106.

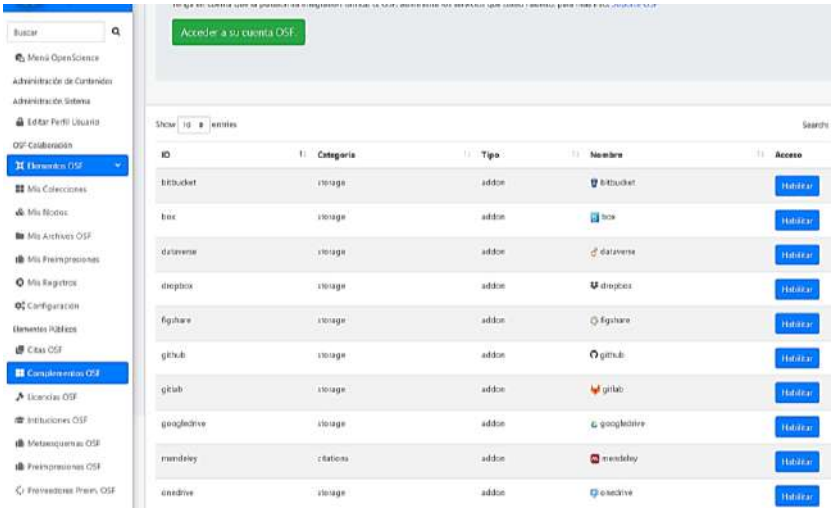


Fig. 106. Complementos públicos OSF, colaborador.

Fuente: Propia.

Si es necesario acceder a grandes volúmenes de datos el usuario debe asignar el repositorio de origen, en este caso se encuentran dos repositorios globales **harvard.edu** y **lib.virgina.edu**. En el caso de integrar un espacio de datos que sea escalable y seguro en Microsoft Dataverse se deberán habilitar los accesos necesarios de lectura y escritura a las estructuras en árbol que sea requiera mostrar.

El complemento de origen Dataverse no será instalado mediante la integración, sin embargo, Google Drive, OneDrive, Mendeley y otros servicios si fueron configurados, ver Fig.107.

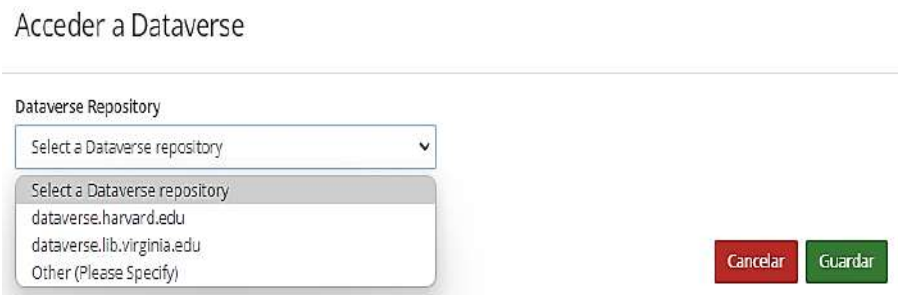


Fig. 107. Integración de Complementos OSF, colaborador.

Fuente: Propia.

Licencias OSF, es necesario recalcar que las licencias fueron usadas también para los contenidos abiertos que se muestran en el Frontend. Al ser un proceso experimental funciona de tal manera que a un determinado contenido se le asigna una licencia al detalle, mostrando cada una de las cláusulas que debe tener un elemento que sea necesario publicar o enviar a proceso Golden Road, ver Fig. 108.

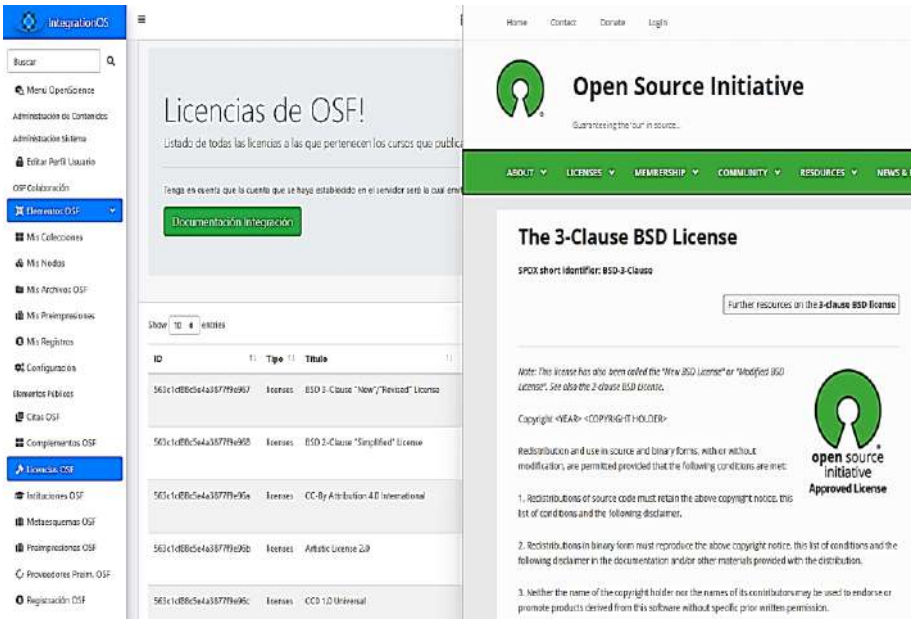


Fig. 108. Licencias Públicas OSF, colaborador.

Fuente: Propia.

Las instituciones públicas o privadas permiten acceder a sus investigaciones científicas relacionadas a varias áreas del conocimiento entre las que se encuentran ciencias de la computación. Pueden ser visualizadas directamente; en el presente caso se ha hecho los filtros para encontrar todas investigaciones científicas realizadas en el MIT y que han sido enviadas a proceso de registro, ver Fig. 109.

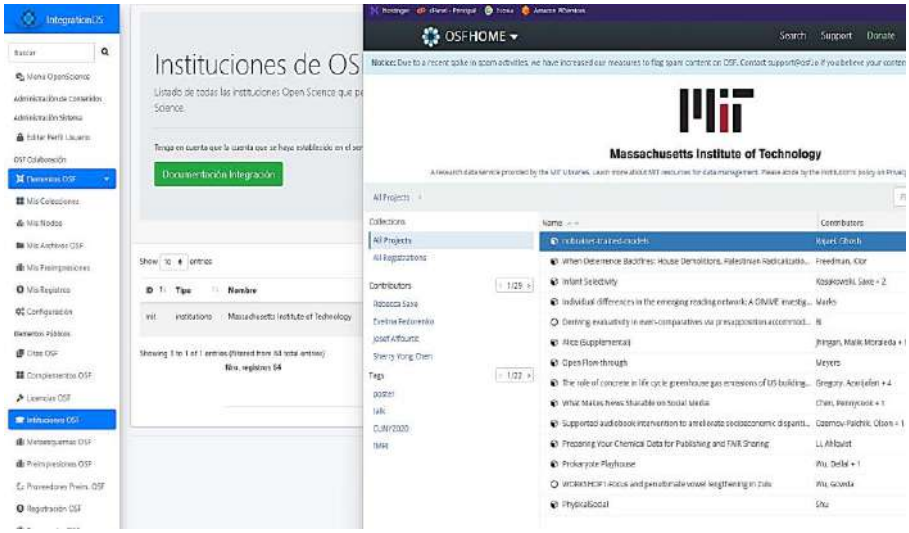


Fig. 109. MIT Repositorios Públicos, colaborador.

Fuente: Propia.

El proceso de preimpresión requiere que el usuario complete varios formularios en los cuales deberá ingresar todos aquellos detalles de la investigación, por ejemplo: muestras de datos, resultados estadísticos, artículos científicos y otros elementos a los que se les asignará un DOI (Digital Object Identifier), en un metadato no modificable de OSF, ver Fig. 110.

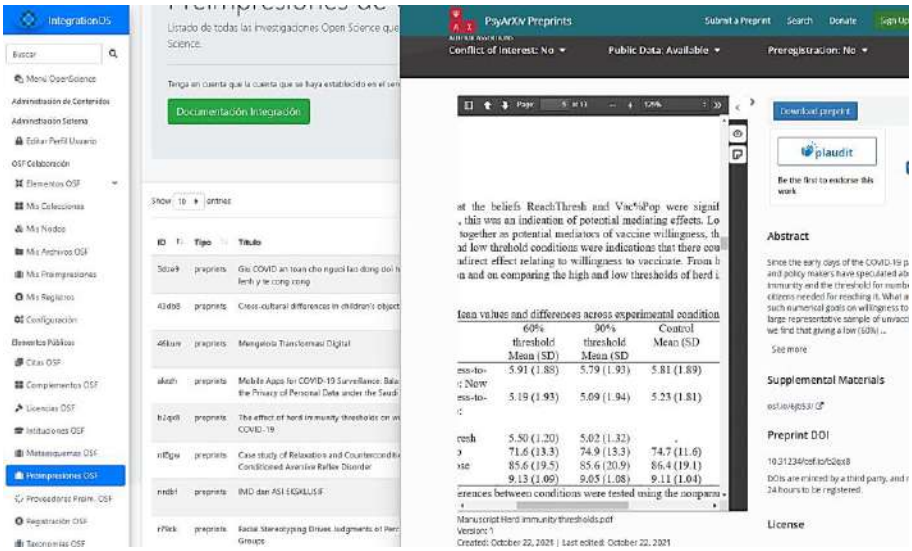


Fig. 110. Preimpresiones en OSF, colaborador.

Fuente: Propia.

Los proveedores de preimpresiones como se ha mencionado con anterioridad requieren un espacio de almacenamiento para los metadatos y elementos que se pretendan hacer públicos, en su mayoría requieren una suscripción. Los principales proveedores son: BioHackrXiv, MindRxiv, Thesis Commons y otros, ver Fig. 111.

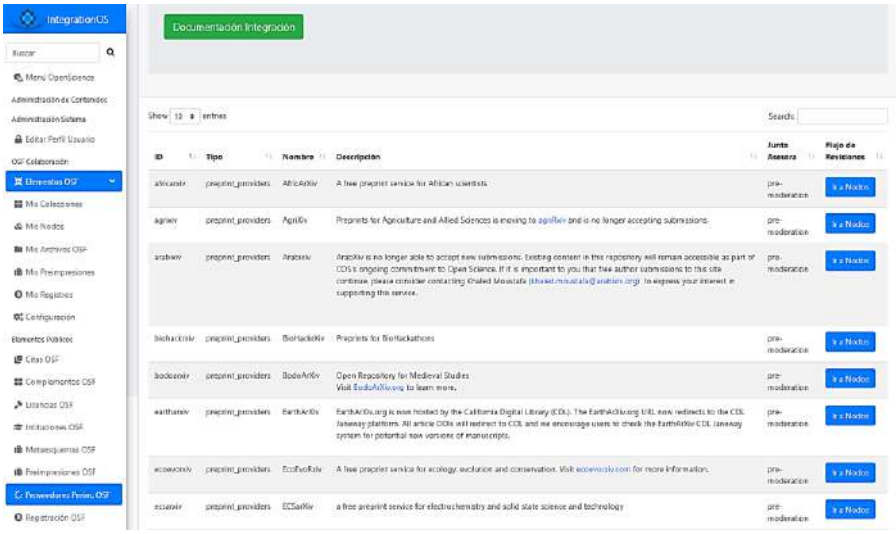


Fig. 111. Proveedores de Preimpresiones OSF, colaborador.

Fuente: Propia.

Los registros públicos muestran la información detallada de cada una de las investigaciones y las versiones de todos los documentos que contienen, un registro debe contener la siguiente información: Overview (Vista General), Files (Ficheros), Wiki (Pequeño trabajo Informático), Components (Componentes), Links (Enlaces), Analytics (Analítica) y Comments (Comentarios); ver Fig. 112.

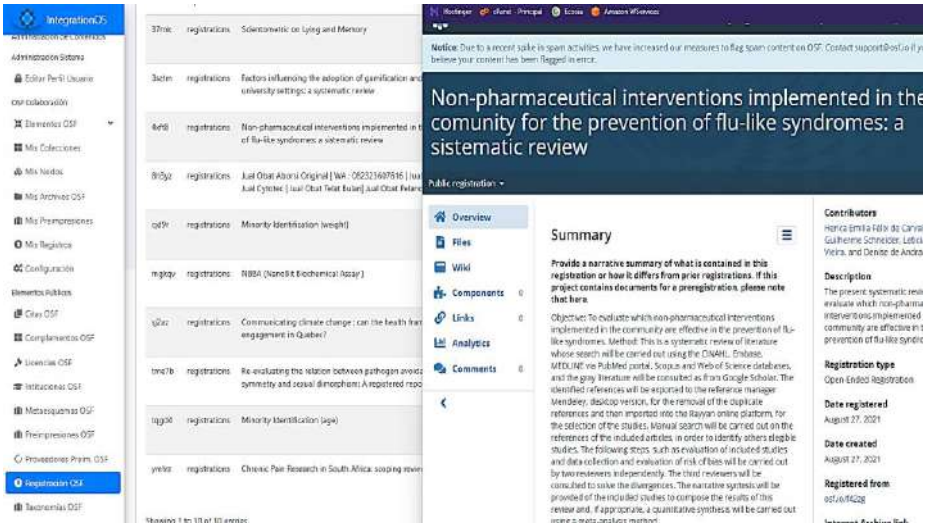


Fig. 112. Registros Publicados en OSF, colaborador.

Fuente: Propia.

Las taxonomías contienen los lineamientos de aquellas temáticas generales en las que se encuentra alojado un determinado contenido abierto. En la Fig. 113, se puede visualizar un total de 11033 registros, las taxonomías van desde Astronomía, Ciencias de la Computación, Física, Ingeniería Aeroespacial, Ingeniería Mecánica, Jurisprudencia, Psicología Social y de la Personalidad y otras ramas del conocimiento.

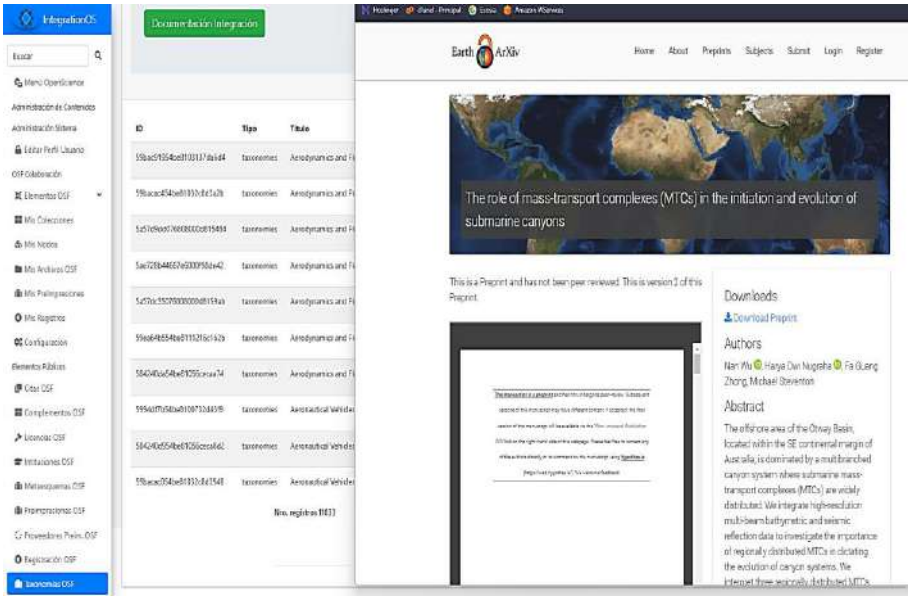


Fig. 113. Taxonomías OSF, colaborador.

Fuente: Propia.

Componentes OSF

Colección es un objeto que puede tener uno o más proyectos. El usuario colaborador ha creado una colección llamada “Integration Colaboration” en la que se aloja el repositorio principal y todos sus componentes, ver Fig. 114.

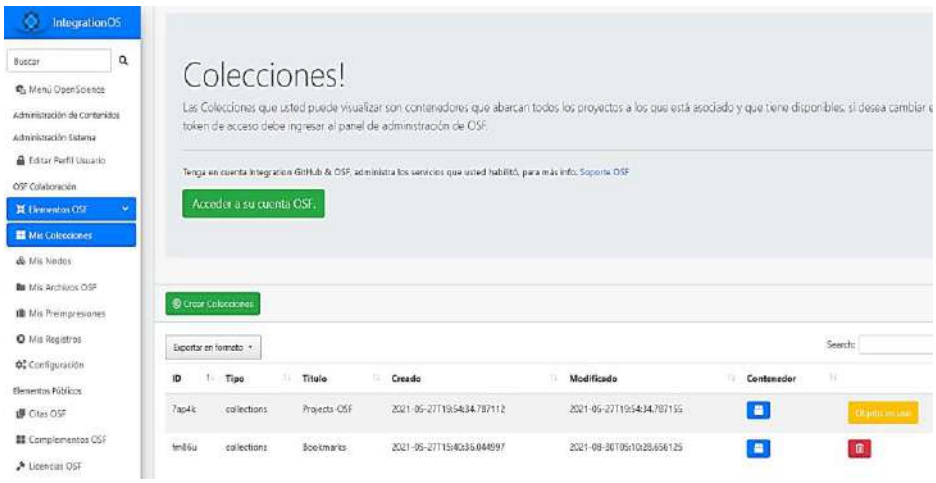


Fig. 114. Colecciones OSF, colaborador.

Fuente: Propia.

En la colección “Integration Colaboration” el usuario apartó el nodo “Integration GitHub & OSF” el cual contiene todos los elementos del presente proyecto, ver Fig. 115.

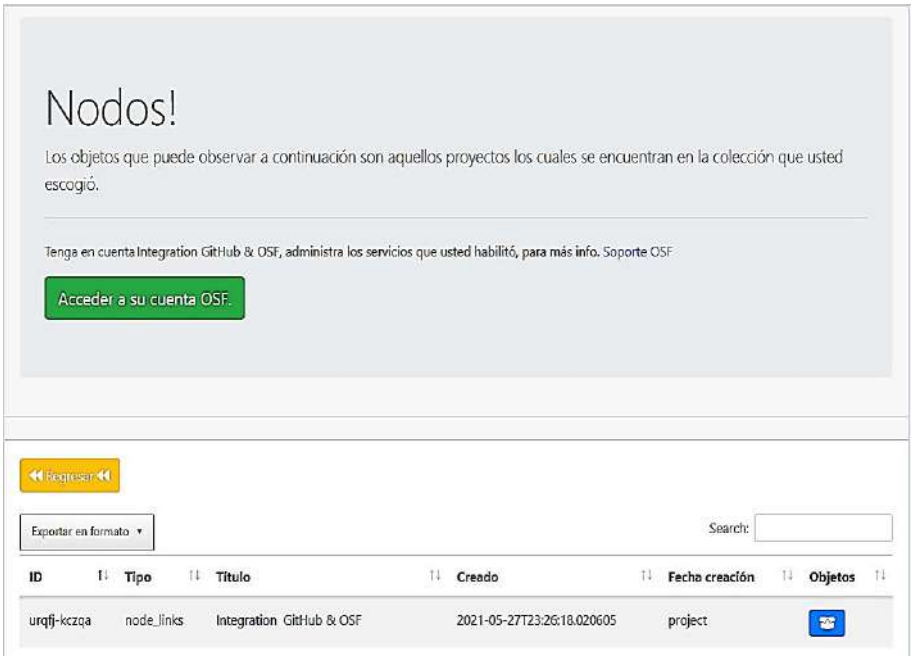


Fig. 115. Proyectos OSF, colaborador.

Fuente: Propia.

Una vez el usuario ingrese al proyecto del cual es un miembro colaborador podrá acceder a una consola en la cual se puede visualizar los detalles y datos importantes acumulados durante toda la ruta del proyecto, ver Fig. 116.

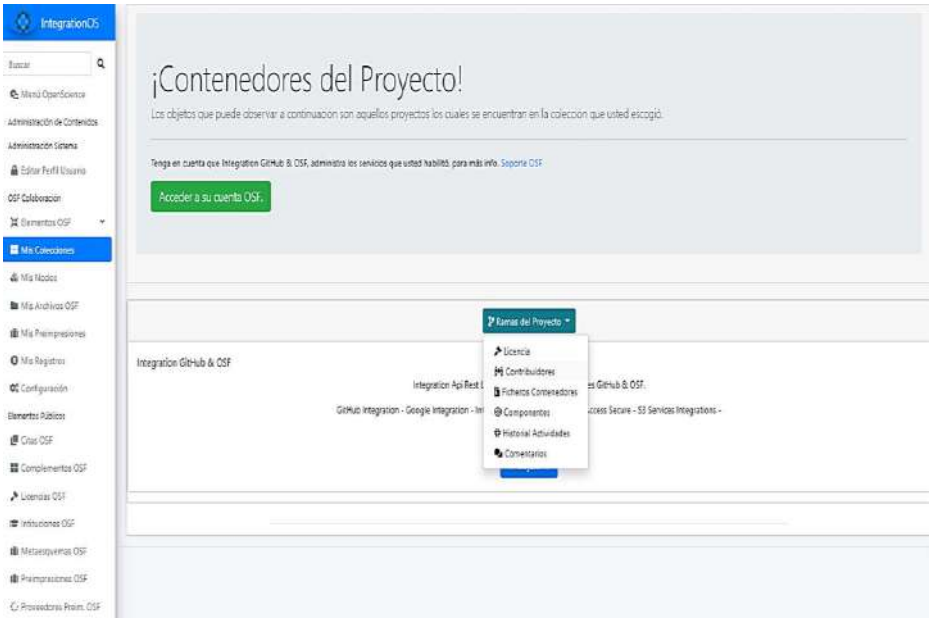


Fig. 116. Detalle del proyecto OSF, colaborador.

Fuente: Propia.

Los componentes que se encuentran en el nodo del proyecto llamado “Integration GitHub & OSF” está sujeto a la licencie The 3 - Clause BSD License, cabe recalcar que dicha licencia fue establecida para los contenidos que se encuentran en proceso de desarrollo, ver Fig. 117.

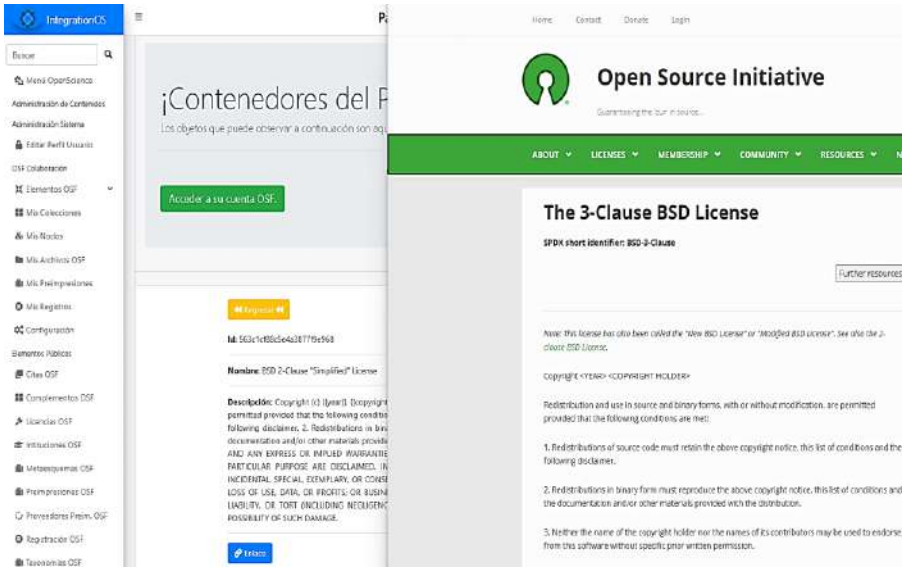


Fig. 117. Licencias Iniciales del proyecto, colaborador.

Fuente: Propia.

En la Fig. 118, se visualizan los ficheros contenedores abarcan los microservicios habilitados en cada componente. En el componente principal se establecieron cuatro (GitHub, AWS, Google Drive y OSFStorage).

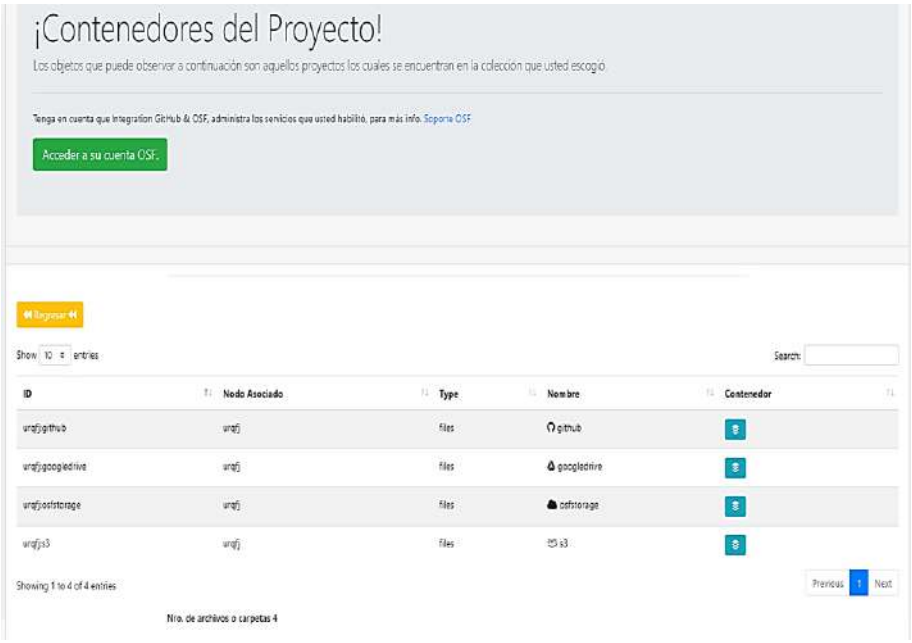


Fig. 118. Microservicios del Repositorio, colaborador.

Fuente: Propia.

El usuario podrá gestionar los repositorios de código fuente alojados en GitHub, de tal manera que pueda tener una copia local y accesos directos a cada uno de los cambios que se hayan agregado durante la etapa de desarrollo, ver Fig. 119.

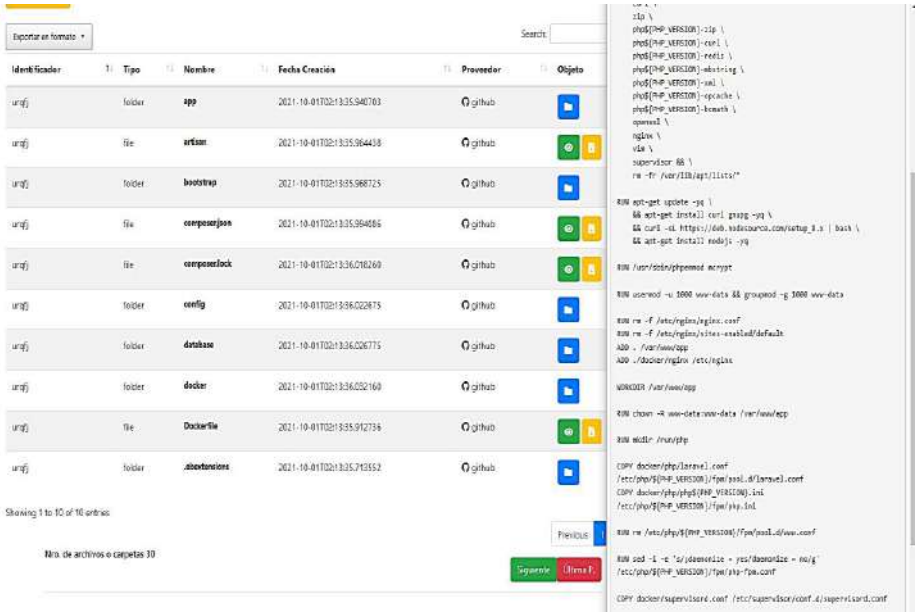


Fig. 119. Contenedor GitHub en OSF, colaborador.

Fuente: Propia.

En la Fig. 120, se visualizan los contenidos abiertos que serán agregados por los docentes y también podrán ser monitoreados por el colaborador, de tal manera que sea posible realizar minería de datos para próximas versiones en el contenedor de AWS.

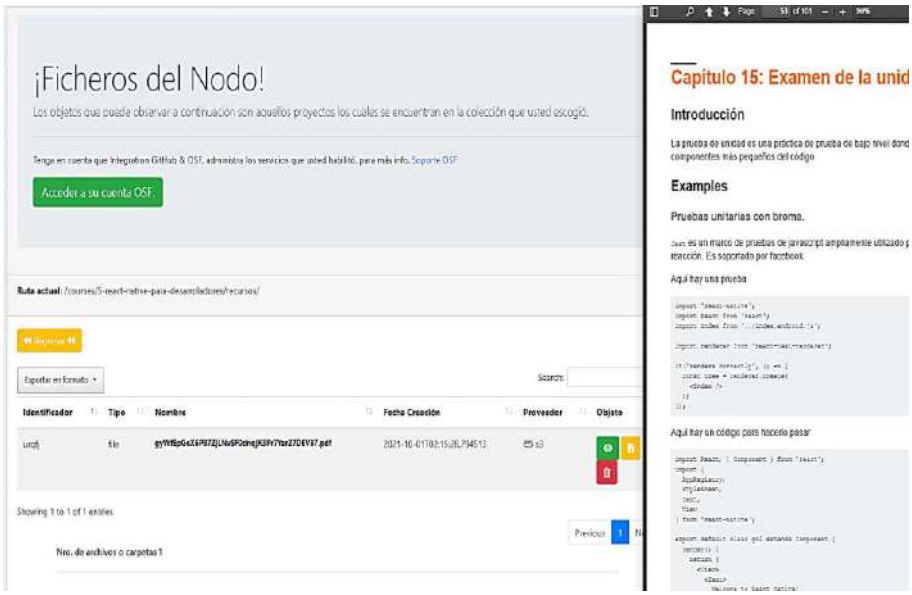


Fig. 120. Contenedor AWS en OSF, colaborador.

Fuente: Propia.

La eliminación en cualquier fichero OSF dejará un metadato, el cual mantiene información de la fecha, hora y demás datos. Cabe recalcar que una vez los elementos sean enviados a preimpresión, el nodo almacenará un log con metadatos, ver Fig. 121.

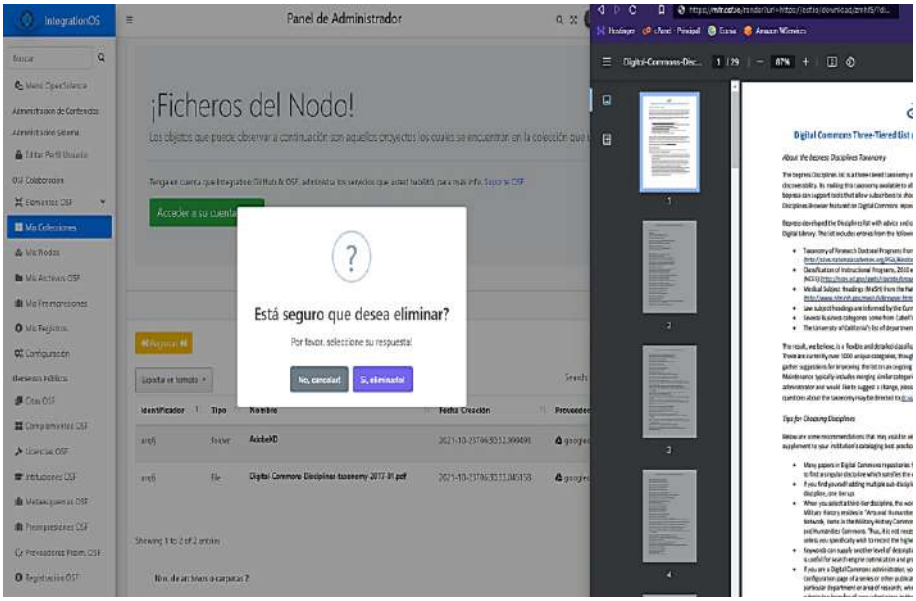


Fig. 121. Registro Metadata, colaborador.

Fuente: Propia.

A medida que el proyecto fue incrementando su contenido se hizo evidente la necesidad de crear nuevos nodos hijos “componentes hijos”, dando como resultado los siguientes nodos relacionados al nodo principal, ver Fig. 122.

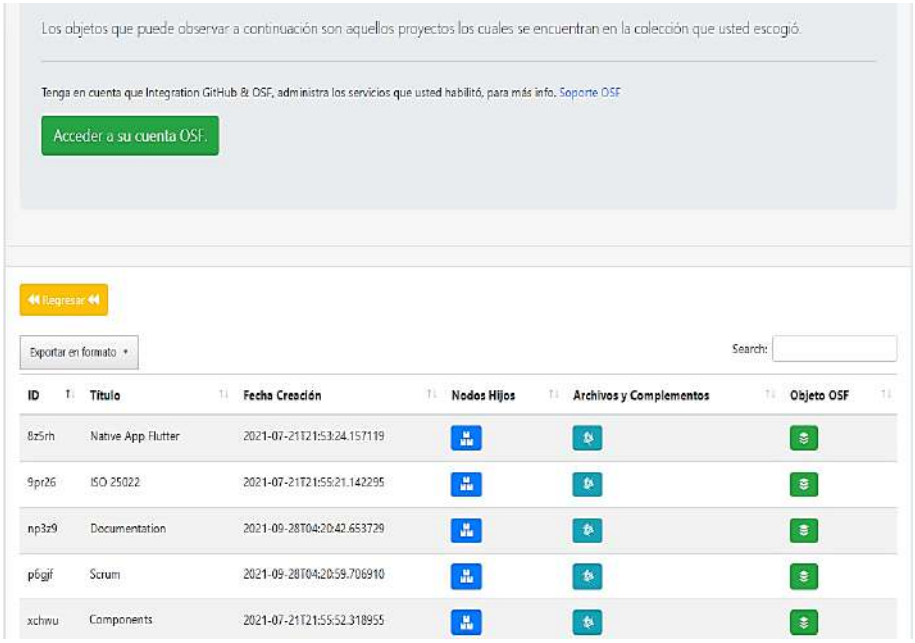


Fig. 122. Nodos Hijos del repositorio en OSF, colaborador.

Fuente: Propia.

Para optimizar la visualización de los flujos de trabajo se alojaron las 40 historias de usuario de cada uno de los Sprint en el nodo “Scrum”, ver Fig. 123.

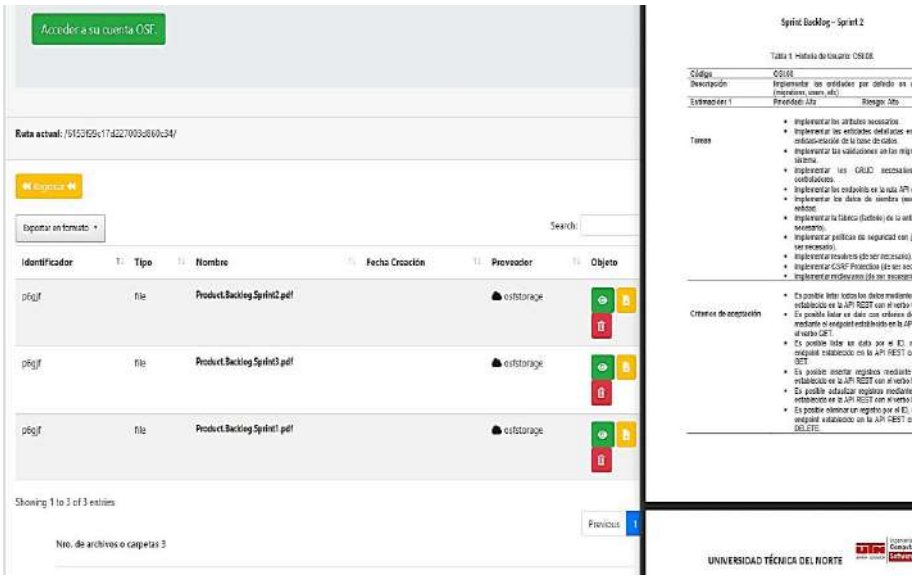


Fig. 123. Nodos Hijos Scrum en OSF, colaborador.

Fuente: Propia.

En el apartado Logs será posible visualizar todas las actividades que se han llevado a cabo en el repositorio, por ejemplo: nuevos archivos, código fuente modificado, nuevos componentes, nuevos nodos, etc. Hasta el momento el proyecto lleva 840 logs, ver Fig. 124.

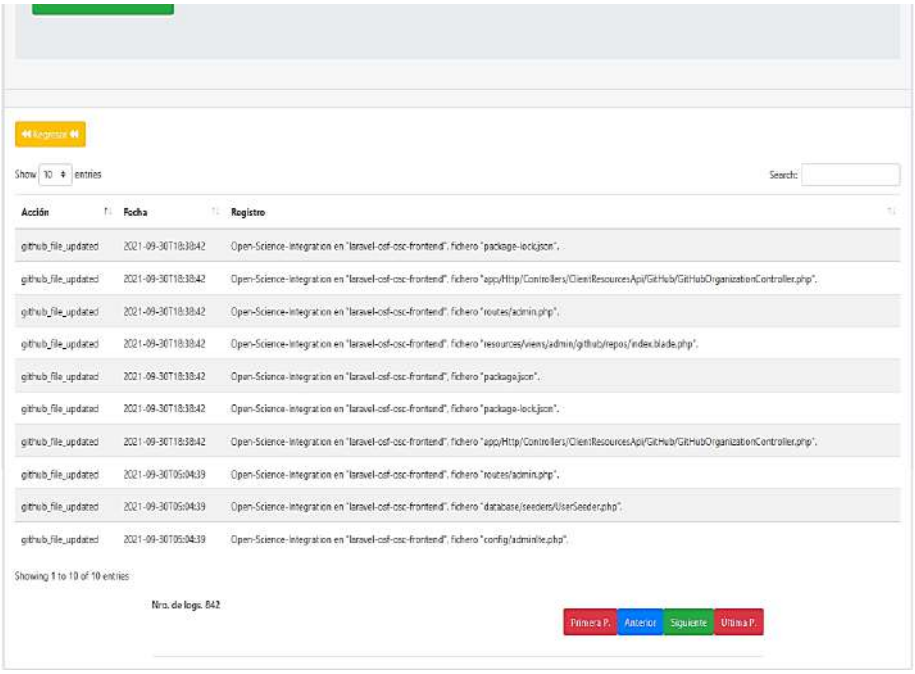


Fig. 124. Logs del Repositorio en OSF, colaborador.

Fuente: Propia.

En la Fig. 125, se visualizan los comentarios realizados en cada uno de los nodos del proyecto y las acciones que se pueden realizar en cada uno de ellos.

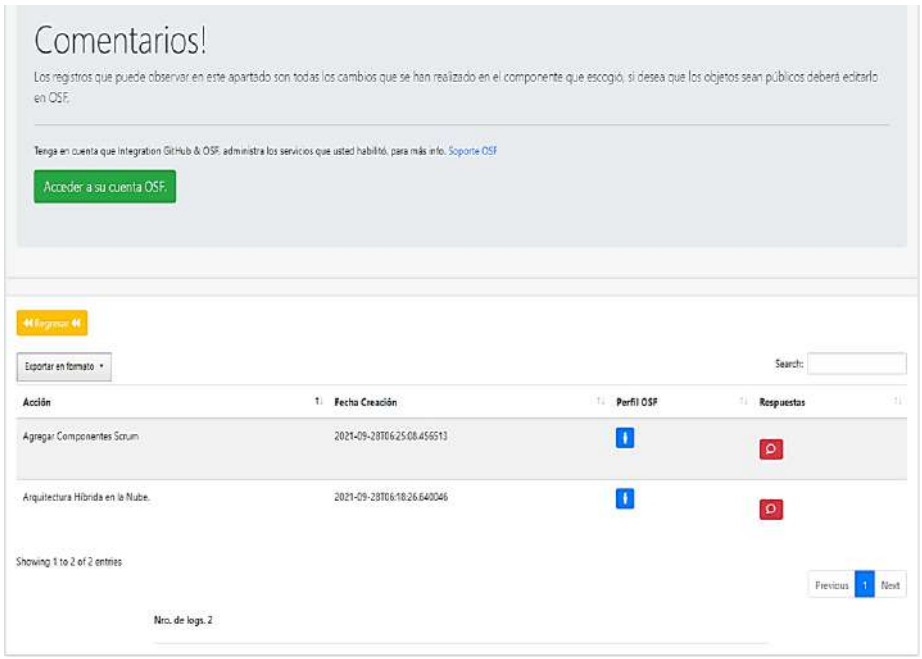


Fig. 125. Comentarios del Repositorio en OSF, colaborador.

Fuente: Propia.

El usuario en OSF posee un contenedor privado de archivos los cuales generalmente son productos de otras investigaciones o proyectos, por lo que es necesario que dichos datos sean gestionados de manera local, ver Fig. 126.

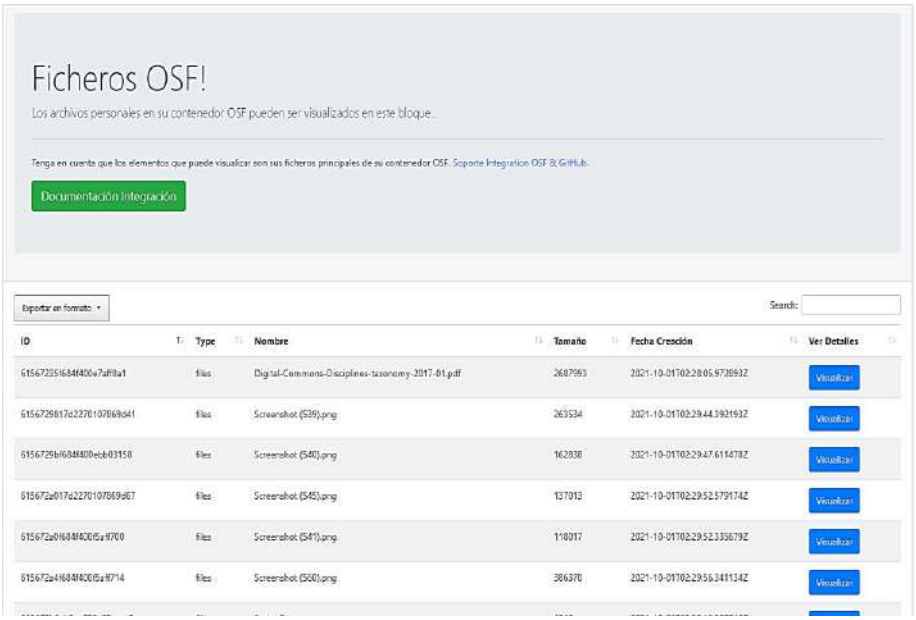


Fig. 126. Listado de Ficheros Privados en OSF, colaborador.

Fuente: Propia.

Los ficheros CSV que contienen las tarjetas generadas en GitKraken fueron almacenados de manera local para tener disponible un respaldo de la información recopilada a lo largo del proyecto, ver Fig. 127.

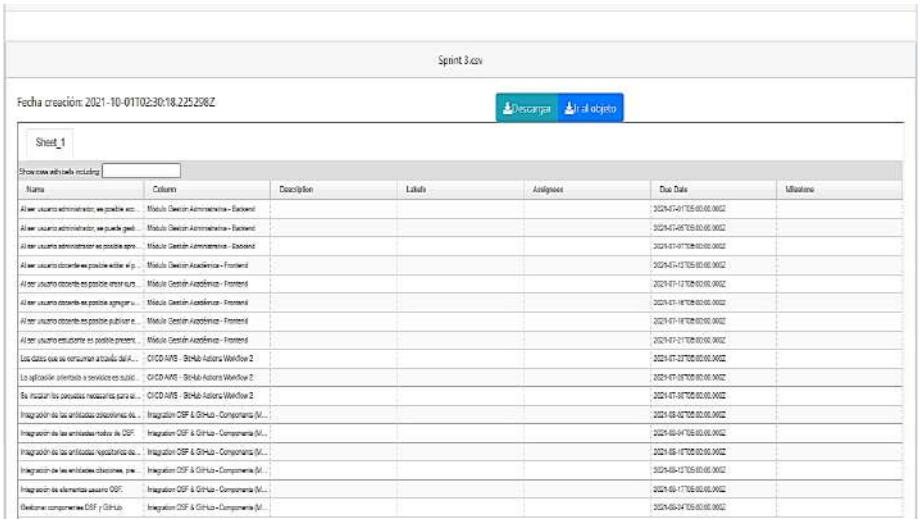


Fig. 127. Tarjetas GitKraken CSV en OSF, colaborador.

Fuente: Propia.

GitHub Colaboración

La colaboración en los repositorios contiene los flujos necesarios para manipular el código fuente; por motivos de optimización se omitieron varias funcionalidades y serán mostradas de manera general. En la Fig. 128, se visualizan todas las organizaciones públicas o privadas a las que se encuentra asociado el usuario, en el presente caso el usuario se encuentra asociado a varias organizaciones, entre las que se encuentra “Open-Science-Integration”.

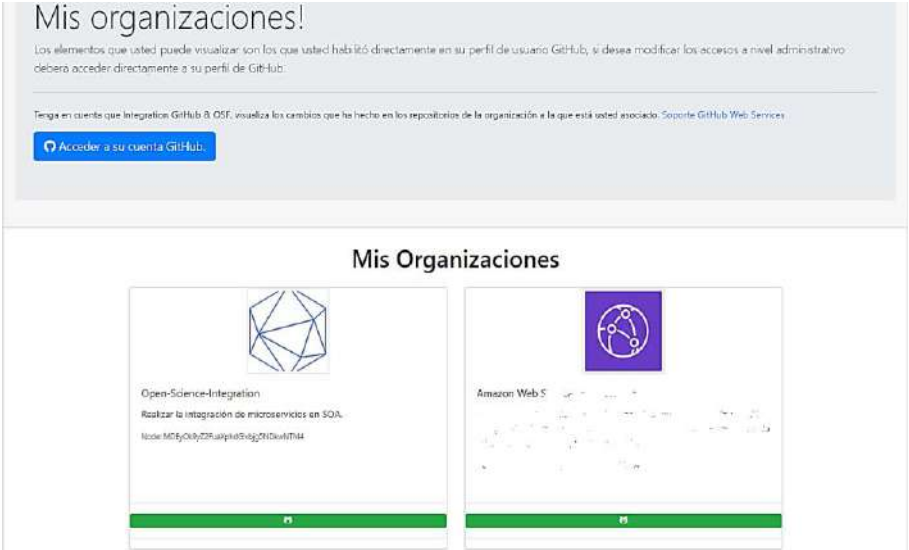


Fig. 128. Organizaciones en GitHub, colaborador.

Fuente: Propia.

En la Fig. 129, se visualizan los repositorios alojados en la organización escogida.

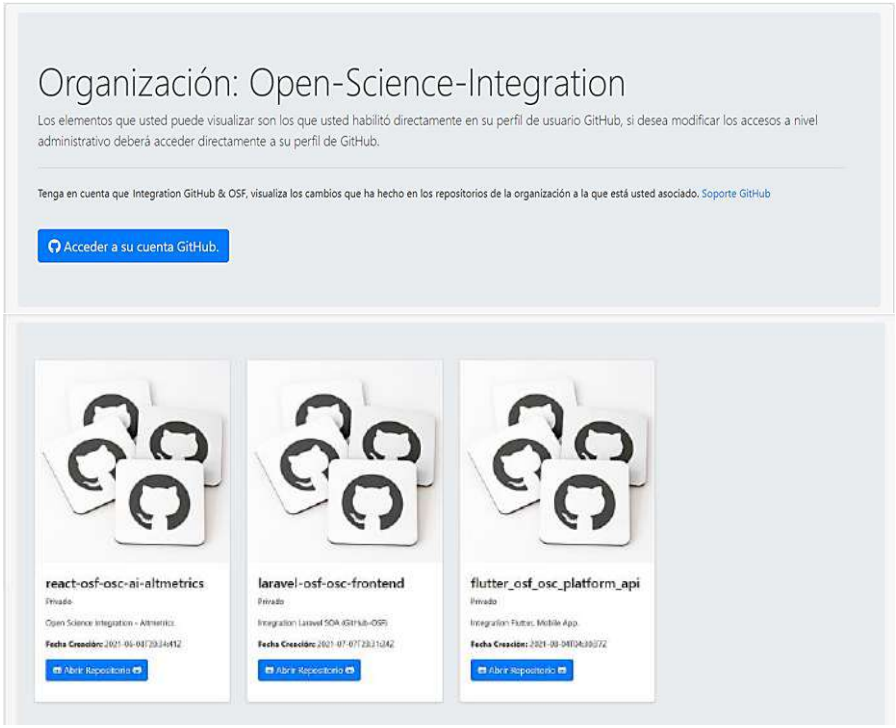


Fig. 129. Repositorios por organización en GitHub, colaborador.

Fuente: Propia.

El usuario podrá visualizar los componentes principales del repositorio, las ramas y también clonar el código fuente a un entorno de trabajo local mediante SSH y HTTPS, ver Fig. 130.

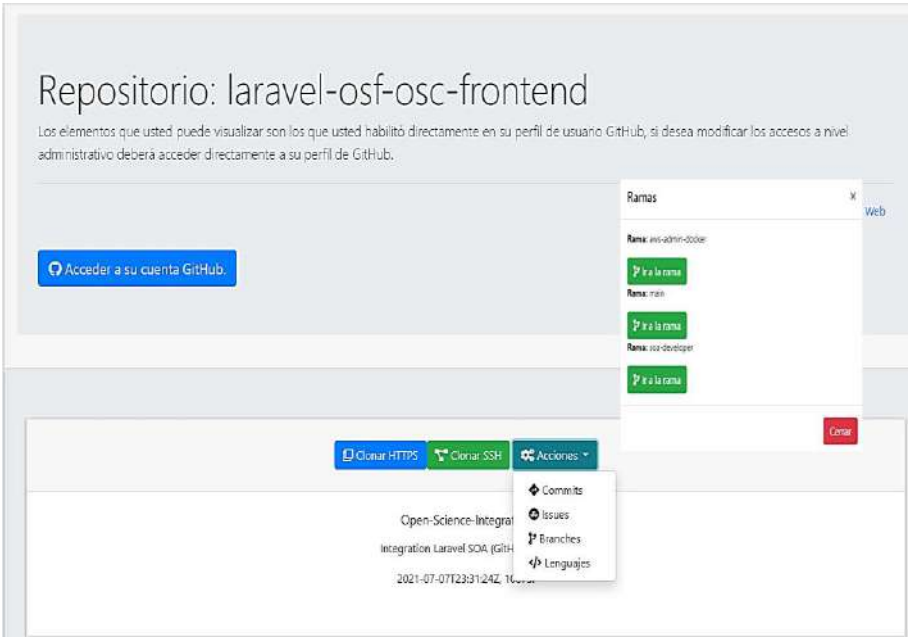


Fig. 130. Detalle del repositorio GitHub, colaborador.

Fuente: Propia.

Los elementos Commits de cada uno de los repositorios podrán ser visualizados en la tabla paginada, ver Fig. 131.

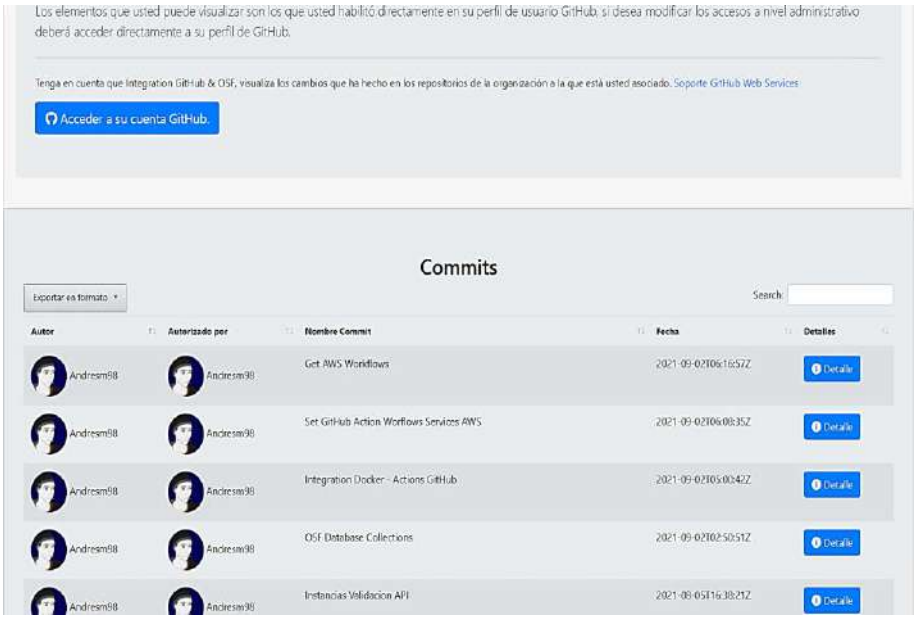


Fig. 131. Commits del repositorio GitHub, colaborador.

Fuente: Propia.

En cada rama se mostrará el último commit a detalle con todos los cambios que se hayan realizado en los ficheros del repositorio y también la unión por rama, ver Fig. 132.

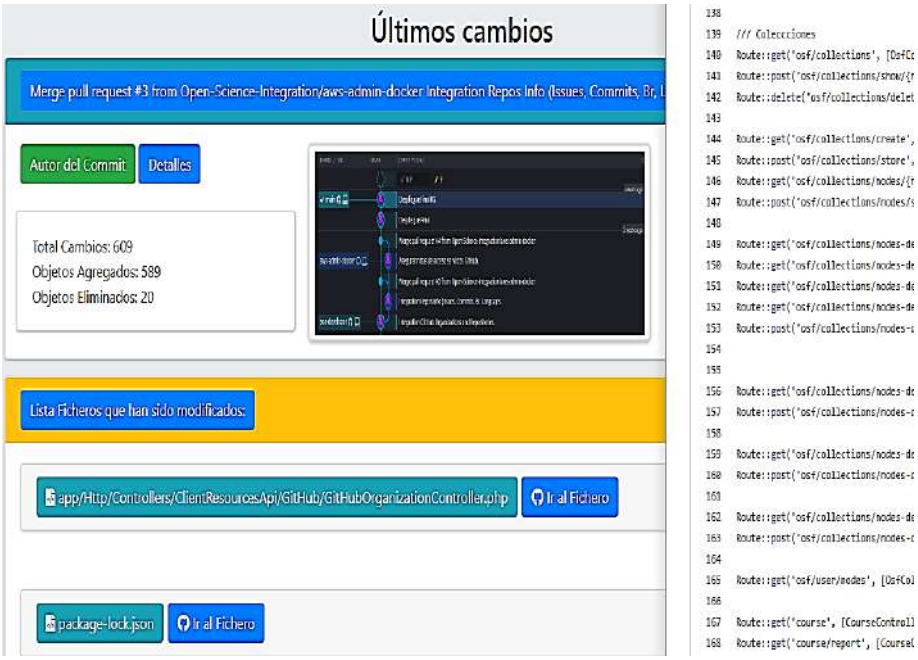


Fig. 132. Detalle Commit repositorio GitHub, colaborador.

Fuente: Propia.

En la Fig. 133, se visualiza la pantalla en la que el usuario podrá acceder a los repositorios que tenga en su cuenta GitHub, por lo que será sencillo reutilizar contenido Open Source alojados en la plataforma.

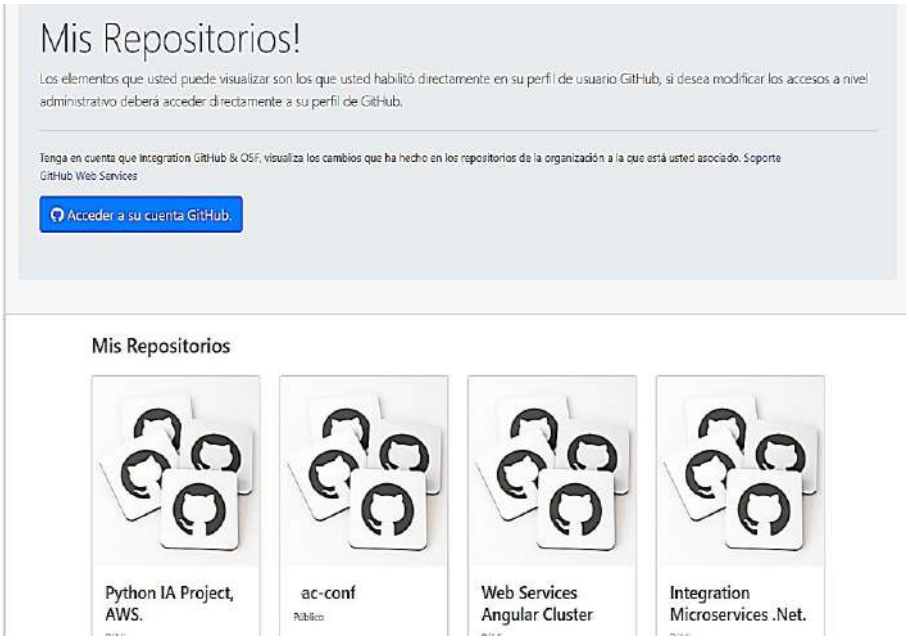


Fig. 133. Lista repositorios GitHub, colaborador.

Fuente: Propia.

2.4. Métricas de rendimiento DevOps en la Arquitectura

Después de concluir con los flujos de trabajo de cada uno de los roles de la aplicación, se procedió a evaluar por última vez el rendimiento de la arquitectura, la razón por la que se realizaron pruebas de rendimiento durante todo el proceso fue que las herramientas Prometheus y Grafana se ajustan a la supervisión de arquitecturas empresariales orientadas a servicios altamente escalables. Por lo general, es necesario implementar un entorno controlado de pruebas que garantice que una vez sea desplegado un servicio, este pueda ser monitoreado para tener claras las posibles fallas o puntos de estrés que deben mejorarse.

En el presente proyecto por motivos de optimización no serán presentados varios pasos a seguir para la instalación y explicación de las herramientas, pues son un tema relativamente extenso. Sin embargo, se mostrarán las pruebas de estrés realizadas al clúster Docker, base de datos, balanceo de carga y otros elementos. Ver Anexo 2 para detalles de la instalación.

De manera general, los instrumentos necesarios para obtener métricas de rendimiento lo más cercanas a la realidad en una arquitectura TI son:

- Máquina Virtual Linux ejecutando Grafana Versión 8.2.1.
- Instalación de Prometheus Versión 2.30.3 mediante la imagen docker de AWS.

- Herramientas de estrés e instalación del fichero de evaluación mediante SSH.
- Balanceo de carga con puertos abiertos a tráfico de entrada y salida para verificar el comportamiento en todos los posibles escenarios.
- Creación del DataSource que tendrá acceso a las métricas de rendimiento.

En la Fig. 134, se puede visualizar la consola de administración de Prometheus ejecutándose en el puerto 9090, una vez levantado el servicio en la instancia AWS inmediatamente comenzarán a guardarse las métricas de rendimiento en la base de datos respectiva.

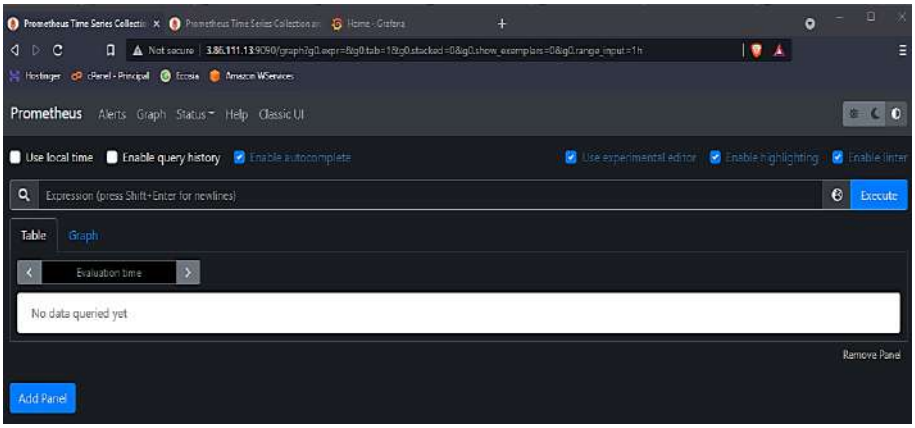


Fig. 134. Instancia Prometheus en el contenedor Docker.

Fuente: Propia.

En la Fig. 135, se puede visualizar las métricas que presenta node exporter, las cuales describen los datos a nivel de máquina y kernel. Los datos que permitirá visualizar son aquellos como uso de memoria, uso del disco, tiempos de lectura y escritura, tráfico de red, temperatura del servidor, datos de entrada y salida en servidores de correo o proxy, y otros.

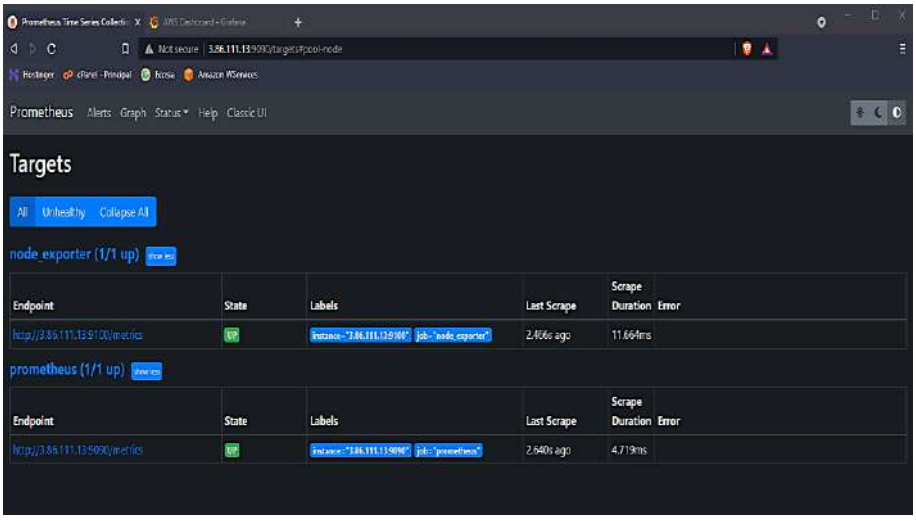


Fig. 135. Configuración node exporter en el contenedor Docker.

Fuente: Propia.

En la Fig. 136, se puede visualizar las alertas que se agregaron para verificar los puntos de latencia, si alguna de ellas llega a fallar en los tiempos establecidos la alerta inmediatamente entrará en proceso de error. En la sección pueden estar *n* alertas, de acuerdo con la necesidad de la organización o empresa; para efectos de prueba se deshabilitó el primer servicio para comprobar la funcionalidad.

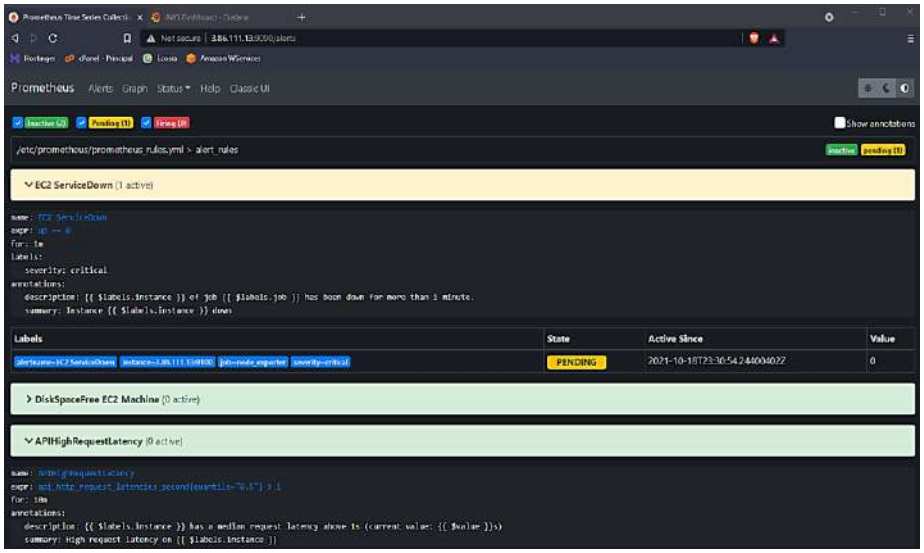


Fig. 136. Alerting Rules en el contenedor Docker.

Fuente: Propia.

Prometheus almacena los datos de toda la información relacionada al contenedor y la máquina física, dichos datos recolectados se muestran en tabulaciones. Los registros previos a las pruebas de estrés indican que el servidor está en óptimas condiciones, ver Fig. 137.

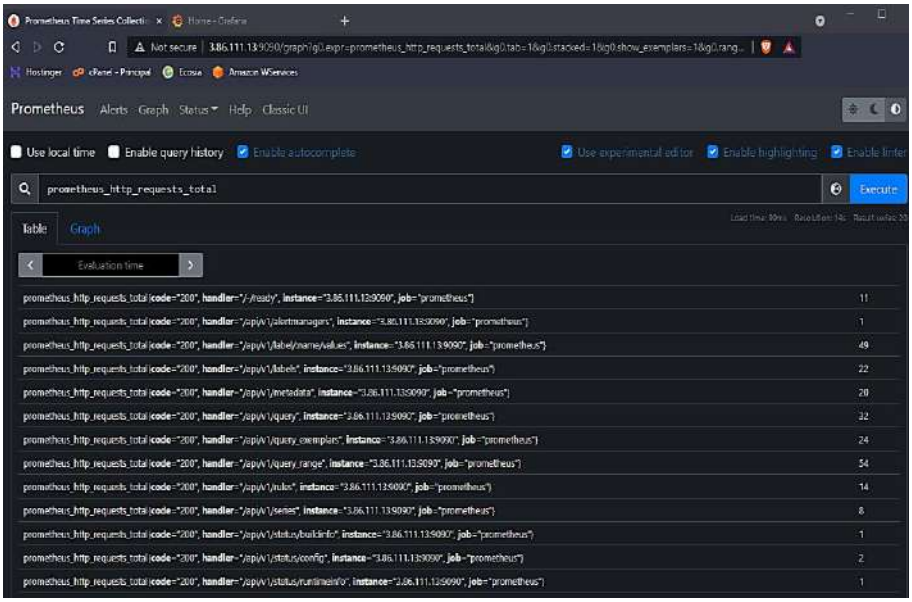


Fig. 137. Tráfico de Red previo a las pruebas de rendimiento.

Fuente: Propia.

Grafana se instaló en un contendor privado y se generaron las configuraciones necesarias para acceder a Prometheus. En la Fig. 138, se visualiza la configuración del DataSource AWS mediante la dirección IP del servidor, el acceso a IAM Role de AWS y permisos pertinentes.

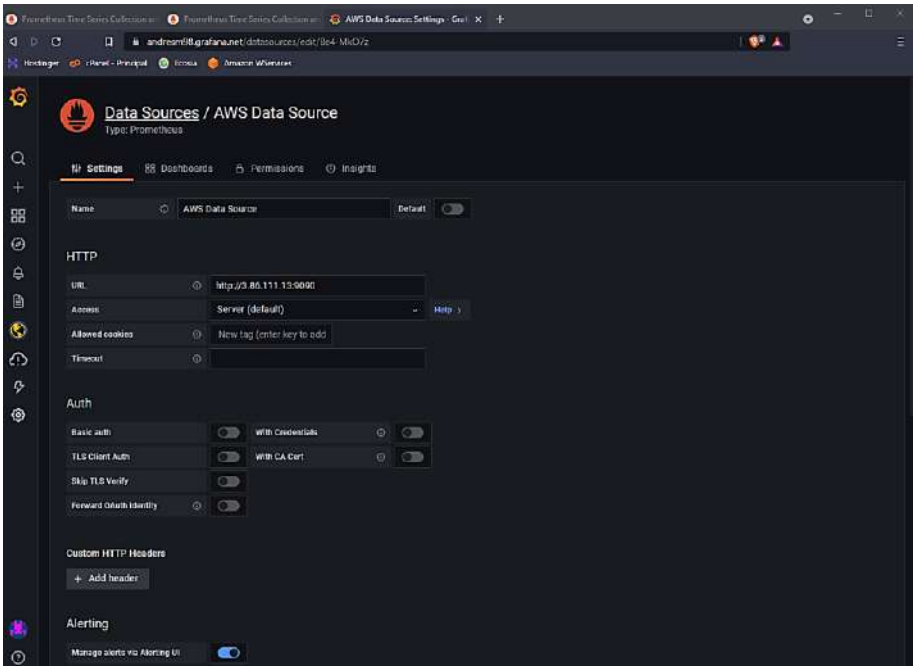


Fig. 138. DataSource para el acceso a Prometheus Docker.

Fuente: Propia.

En la Fig. 139, se visualiza el acoplamiento del entorno de pruebas, cabe recalcar que para acceder al sitio se almacenó un fichero de configuración en la raíz del servidor vía SSH.

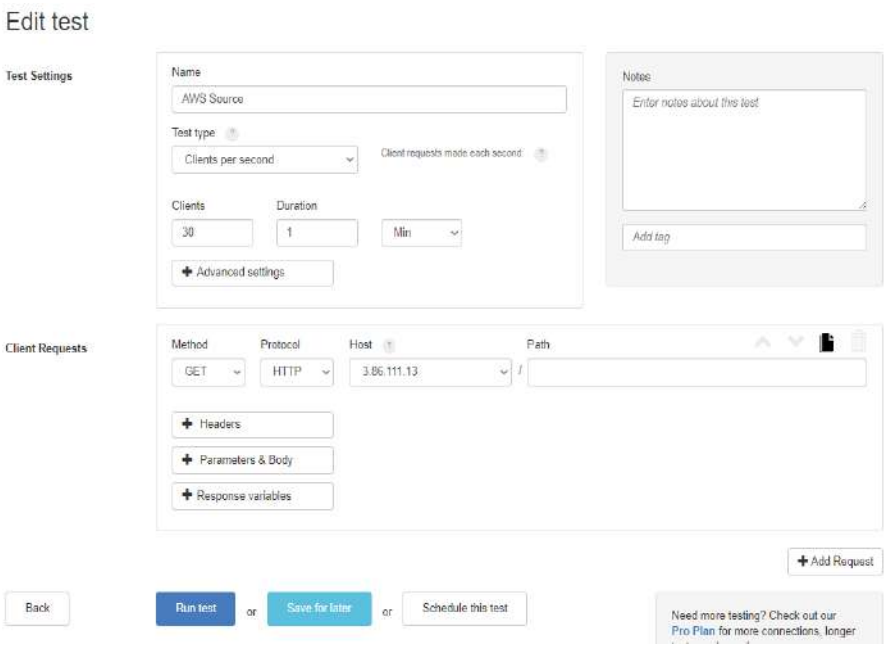


Fig. 139. Preparación de entorno para pruebas de Rendimiento.

Fuente: Propia.

En la Fig. 140, se puede observar las métricas al simular 30 usuarios haciendo solicitudes por segundo durante un minuto, los resultados son eficientes teniendo en cuenta las características del clúster AWS, el uso del CPU es solamente de 2.31%, en la carga al sistema el porcentaje es de 12%, el uso de la memoria RAM de 1,5 GB muestra un 52% de uso.

Las solicitudes que se envían al contenedor son respondidas de inmediato, por lo que la tasa de errores se mantiene en números decimales cercanos a cero. El panel de métricas posee más de 50 paneles de servicios, hilos y procesos que se están monitoreando en tiempo real mediante la API Prometheus y consultas de datos PromQL.

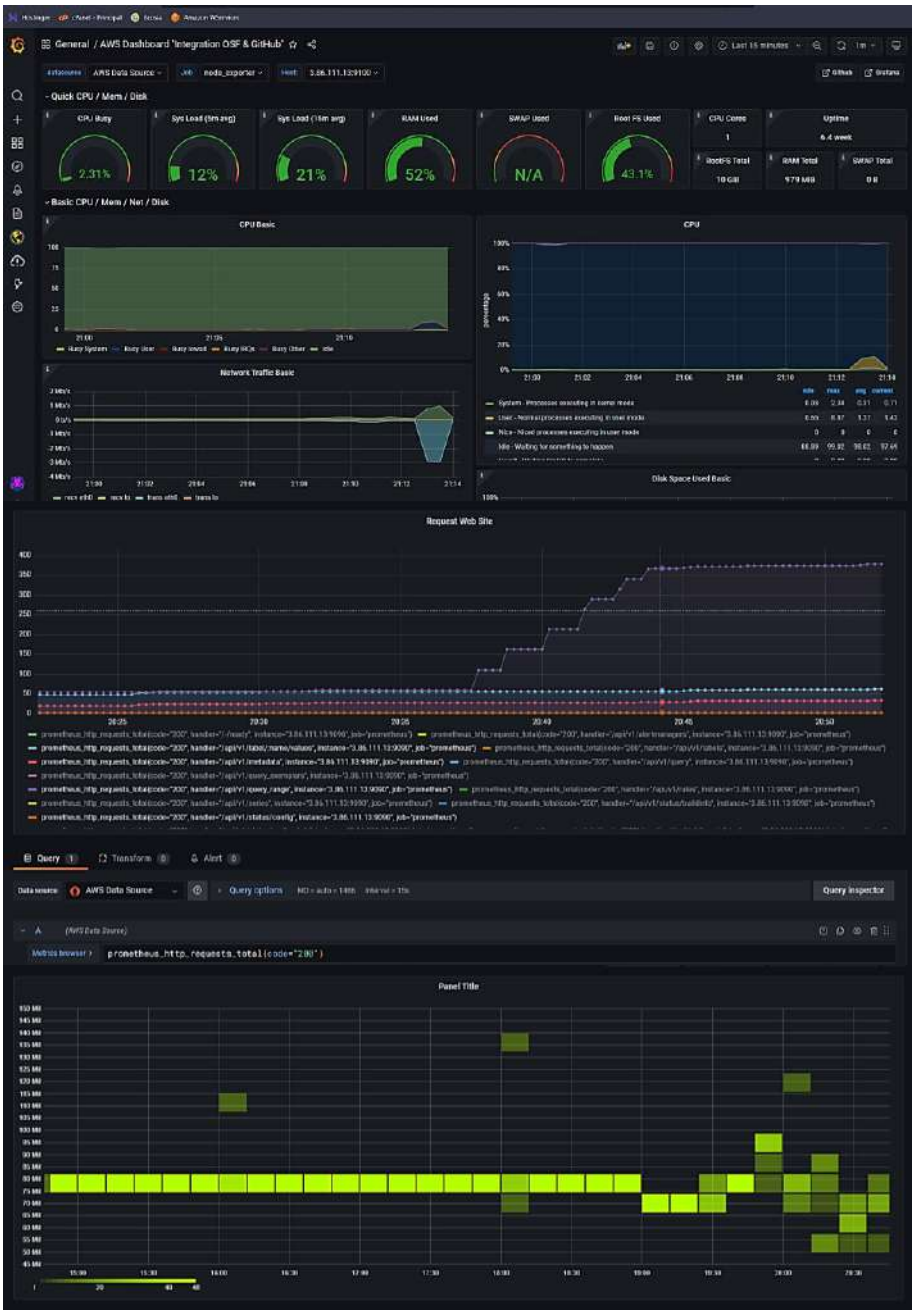


Fig. 140. Dashboard de métricas de rendimiento, vista general.

Fuente: Propia.

La segunda ronda de evaluación de rendimiento fue realizada para la simulación de 50 usuarios accediendo cada segundo durante un minuto al “endpoint” de licencias OSF, fue necesario especificar el token de acceso a la API y también el verbo, ver Fig. 141.

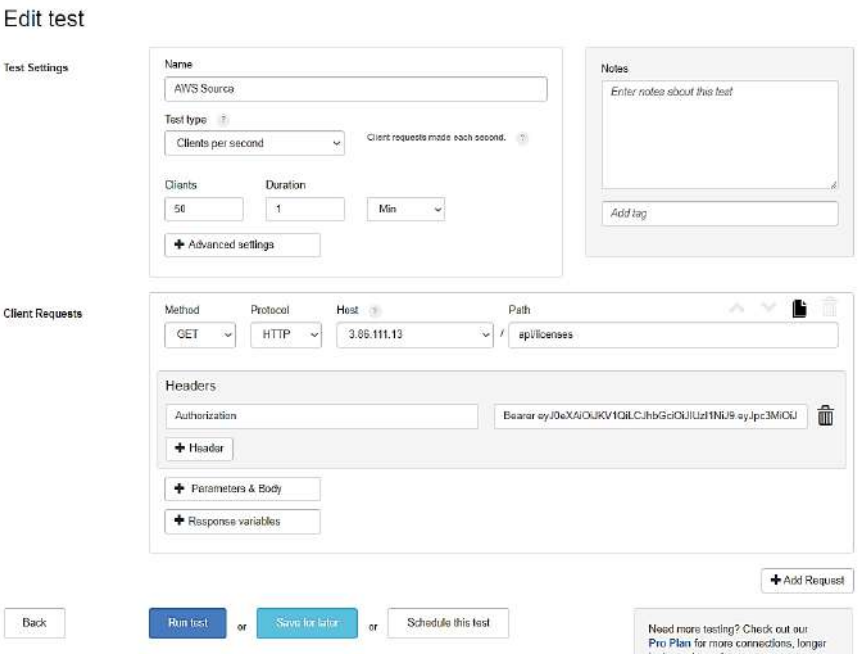


Fig. 141. Segundo Entorno para pruebas de rendimiento.

Fuente: Propia.

Los resultados muestran que el servidor se encuentra en óptimas condiciones en cuanto al rendimiento, el CPU se encuentra con un porcentaje del 59.7%, la memoria RAM se encuentra al 43% de su uso, la carga en el sistema es del 29.0%, ver Fig. 142.

Fig. 142. Dashboard de métricas de rendimiento, segundo escenario.

Fuente: Propia.

En la Fig. 143, se puede visualizar el rendimiento de cada uno de los sockets del sistema, el socket TCP muestra que el tráfico de red es normal y por ende el servidor no ha sufrido ningún problema relacionado a cuellos de botella. Por otro lado, el socket de monitoreo de memoria muestra que está a punto de llegar a su límite, es algo de esperarse al hacer 50 solicitudes.

Fig. 143. Dashboard de métricas de rendimiento, Sockets Network.

Fuente: Propia.

En la Fig. 144, se puede visualizar el tráfico de red por paquetes que no ha mostrado ningún problema, en cuando a la caída del tráfico de red no se ha mostrado ninguna alerta.

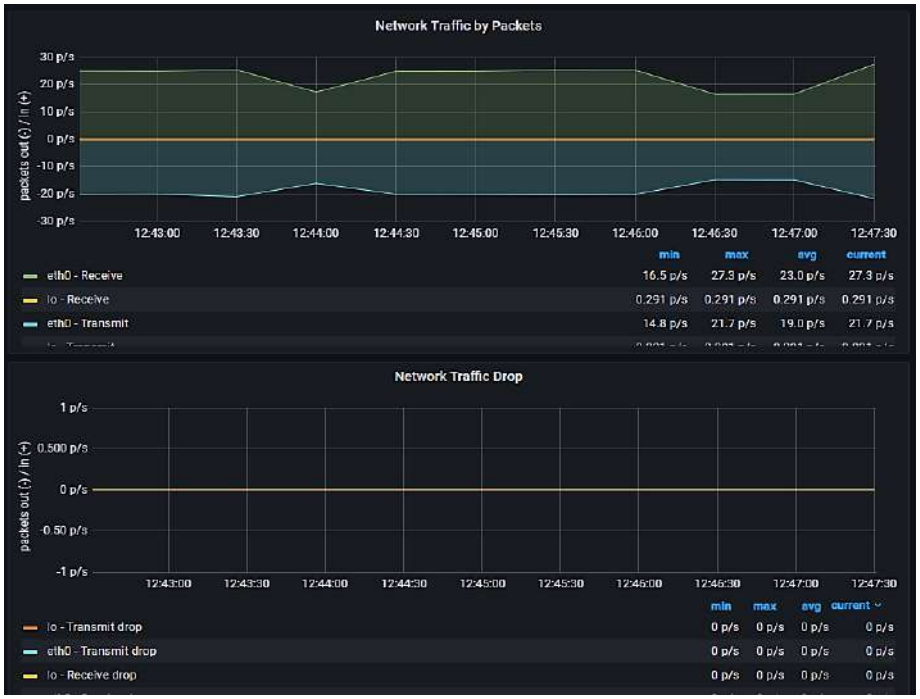


Fig. 144. Dashboard de métricas de rendimiento, Tráfico de Red.

Fuente: Propia.

En la Fig. 145, se puede visualizar las métricas del disco duro, el servidor al ser un entorno de pruebas muestra que está a punto de llegar al límite en cuanto al almacenamiento, sin embargo, no causará problemas ya que los ficheros se almacenan en un contenedor S3.

Fig. 145. Dashboard de métricas de rendimiento, Disco Duro.

Fuente: Propia.

Los resultados que arrojan las métricas de rendimiento referentes a los componentes de la arquitectura muestran ser satisfactorios, teniendo en cuenta las capacidades de cada uno de los servicios integrados y el flujo de información a los que se ven expuestos. Para ello se hizo uso de las herramientas anteriormente presentadas; es necesario mencionar que el clúster, servidor, balanceo de carga, y otros servicios que provee AWS fueron también puestos a pruebas de rendimiento para obtener los posibles puntos de error y darles solución.

El servidor Grafana el cual fue instalado mediante la imagen Docker que presta a servicio el proveedor fue instalado de tal manera que los datos de monitoreo comenzaron a recolectarse después de crear el respectivo Data Source que contiene la dirección IP/dominio, puertos abiertos, enlaces de entrada, credenciales IAM Role y otros datos privados de AWS.

Alerting Rules que proporciona Prometheus a través de la captura de datos con Node Exporter no generó ninguna alerta de rendimiento al realizar las pruebas, para el presente proyecto de investigación fueron mostrados los datos de las métricas de rendimiento más relevantes, sin embargo, el panel Grafana tiene varios paneles los cuales pueden ser manipulados a través de Queries de PromQL o PostgreSQL.

2.5. Publicación de contenido Open Science

Preregistración

En cuanto al preregistro de la etapa inicial del proyecto fue necesario revisar todos aquellos contenidos que se fueron desarrollando, haciendo hincapié en que aquellos contenidos producto de la investigación sean registrados con un metadato único. El Framework OSF provee varios tipos de registros y el investigador deberá elegir el que se relacione al flujo de trabajo:

1. OSF Preregistration: Colección de datos previa a la investigación, en la cual el usuario debe detallar el proceso de diseño, planificación y recolección de datos.
2. Open-Ended Registration: Es el resumen de aquello que contiene el repositorio de investigación.
3. Qualitative Preregistration: Una plantilla creada para que investigadores desde una comunidad cualitativa puedan realizar la preinscripción de la investigación cualitativa.
4. Secondary Data Preregistration: Segunda colección de datos, en dicho registro debe ser detallado el plan global de investigación, plan de análisis y diseño.
5. Registered Report Protocol Preregistration: Plantilla para completar una serie de preguntas y de ser necesario adjuntar un informe.
6. OSF-Standard Pre-Data Collection Registration: Plantilla que contiene preguntas sobre la recolección de datos de la investigación.

Para realizar el proceso de preregistración fue necesario llenar varios formularios en la aplicación, se dividen en las siguientes etapas:

1. Metadata: Datos que se aplican solamente al registro, pero no al proyecto.
2. Stydy Information: Lista de hipótesis específicas, concisas y comprobables. Si algún tipo de interacción ha sido establecida al contenido deberá ser dallada.
3. Design Plan: El plan de diseño se refiere a todos los datos acumulados que pueden ser la recopilación de diversos experimentos o rondas de recolección de datos.
4. Sampling Plan: Es el plan de recolección de muestras, el número de muestras y la razón fundamental de tal decisión. De ser necesario se pueden definir conjuntos de datos “dataset” (moderados o extensos).
5. Variables: Se refiere a todas las variables que serán usadas para la confirmación del plan de análisis.
6. Analysis Plan: Describe uno o más análisis confirmatorios que deben especificarse en el artículo final. El análisis debe anotarse como exploratorio o generador de

hipótesis, cabe recalcar que el análisis confirmatorio debe establecer variables predictoras “independientes” y resultados “dependientes”.

7. Other: Información adicionar que sea necesario incluir en la preregistración.

Tabla 12. OSF Preregistration Project.

Registration Metadata, OSF		
Código	**Campos**	**Descripción**
MT.01	Title	Integration of OSF and GitHub REST APIs through a service oriented application for publishing open science content.
MT.02	Description	Integration REST API.
MT.03	Contributors	Santiago Andres Moreta – Admin
MT.04	Category	Software
MT.05	License	BSD 2-Clause “Simplified” License. (2021, SOAINTG)
MT.06	Subjects	Education, Educational Methods. Engineering, Computer Engineering, Computer and Systems Architecture.
MT.07	Tags	AWS Integration, GitHub Integration, Cloud Services Integration, OSF Integration, SOA.
		Registration Stydy Information
SI.01	Hypotheses	What is the goal of publishing Open Science content and the methods applied to do so? What can Open Science contribute to Education? How to implement multiple Open Science platform resources using REST API technologies? If the services of an open content platform can manage several public and private clouds through an SOA architecture, it will be possible to build an incremental hybrid cloud.
		Registration Design Plan
DP.01	Study type	Integration Open Science Services, SOA.
DP.02	Blinding	No blinding is involved in this study.
DP.03	Study design	This is a randomized block design between subjects, with students randomized within a given educational unit. FILES (Process, Architecture).
DP.04	Randomization	It is a study based on a questionnaire that measures the metrics established in ISO/IEC 25022:2016.
		Registration Sampling Plan
SP.01	Existing Data	Registration prior to creation of data.
SP.02	Data. C. Procedures	Quality metrics in use, ISO/IEC 25022:2016.
SP.03	Sample size	The target sample size is 30 users.
SP.04	Stopping rule	Data collection will not be interrupted due to restrictions. All users are established regardless of the role of the application.
		Registration Variables
V.01	Manipulated variables	The only variable that is manipulated is the assignment of acceptance levels for open content. Students are randomly assigned to complete each course with all milestones that comprise it, once they have finished, the percentage of consumption of the content in an S3 container is established.
V.02	Measured variables	The values obtained are generated by the metrics established by ISO 25022 of the characteristics: Effectiveness, Efficiency and Satisfaction.
V.03	Indices	Quality metrics in use, ISO 25022.
		Analysis Plan
AP.01	Statistical models	The data of each one of the characteristics established by the ISO is obtained by means of the metrics, formula, desired value, weighting, partial value, level of importance, percentage of importance and final value.
AP.02	Transformations	No additional transformations, beyond those indicated in the attached "OSF student measures" are planned.
AP.03	Data exclusion	Risk freedom and context coverage, ISO/IEC 25022:2016.
		Other
OT.01	Other	The quality in use is closely related to the behavior of the system while it executes a certain process, therefore, the value obtained from an acceptance test will seek to quantify usability through the sub-characteristics when a user has completed a certain objective. (Estdale & Georgiadou, 2018) The components that were generated to develop the application based on SOA architecture, were deployed to a hybrid cloud. This project aims to evaluate the three main characteristics of ISO 25022: 2016, they are the following: Effectiveness, Efficiency and Satisfaction. Estdale, J., & Georgiadou, E. (2018). Applying the ISO/IEC 25010 Quality Models to Software Product: 25th European Conference, EuroSPI 2018, Bilbao, Spain, September 5-7, 2018, Proceedings (pp. 492–503). https://doi.org/10.1007/978-3-319-97925-0_42

La etapa de desarrollo del presente proyecto de investigación se culminó sin ningún tipo de inconveniente en el intervalo de tiempo establecido en la metodología Scrum; por lo que después de dar por concluida la arquitectura SOA y todos los componentes que la conforman, se procedió a realizar la registración del contenido en OSF con el formato de la Tabla 12.

En la Fig. 146, se pueden visualizar todos los componentes finales del proyecto; al igual que se realizó en el preregistro se llenaron los datos con el respectivo formulario, es importante mencionar que dicho proceso toma varios días después de dar por concluido el proyecto y depende de los lineamientos en materia de creative commons del investigador, colaboradores y el flujo de trabajo de la plataforma COS.

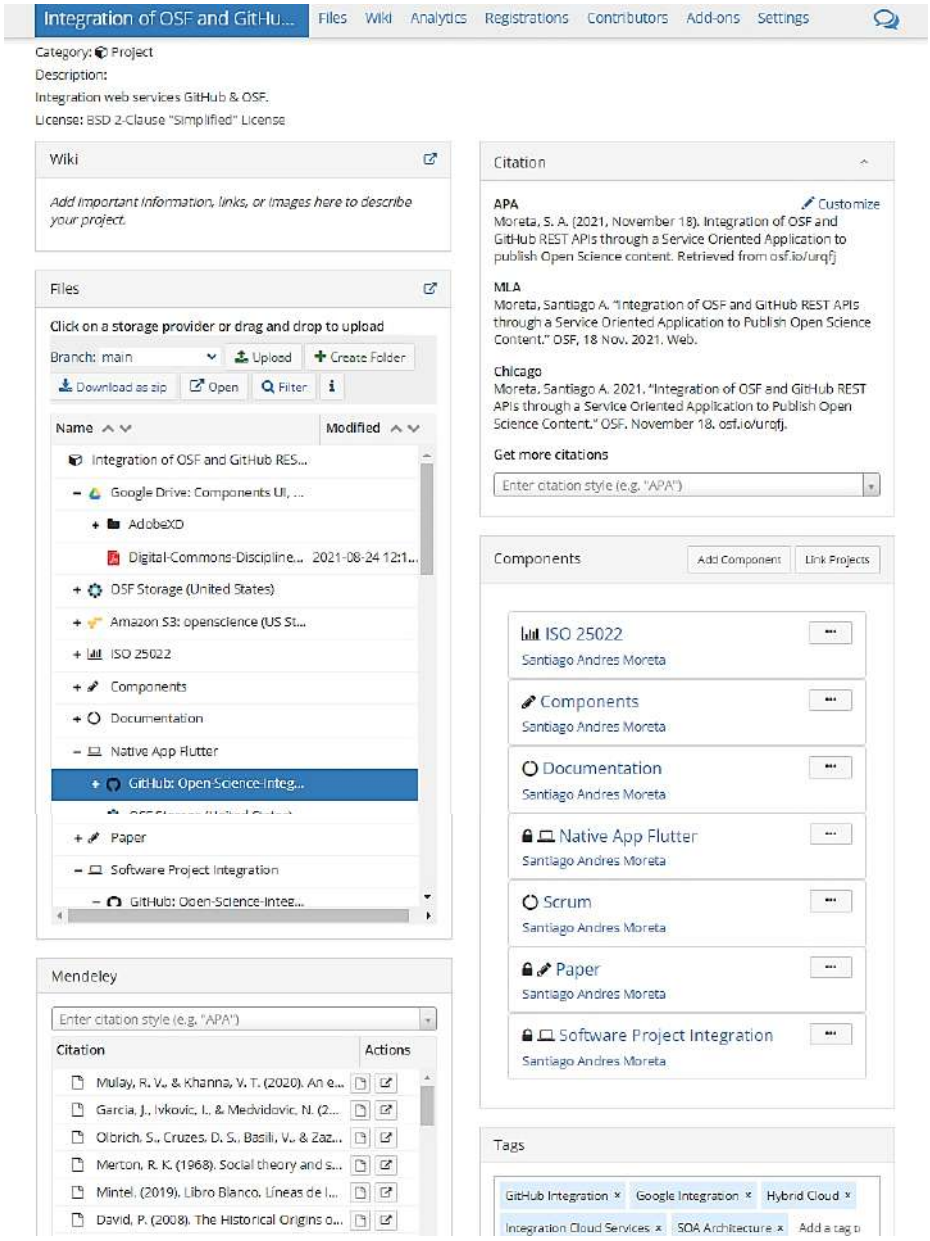


Fig. 146. Componentes producto de la investigación, OSF.

Fuente: Propia.

CAPÍTULO 3

Resultados y Discusión

3.1. Evaluación de productos de Software ISO/IEC 25040

Para garantizar la calidad de software, se establecen métricas que permiten obtener un buen producto, en varias ocasiones incluso es necesario implementar otras normas para realizar una correcta evaluación; con lo anteriormente mencionado se hace evidente la necesidad de un proceso secuencial y ordenado que permita llevar a cabo la evaluación del software.

La norma ISO/IEC 25040 define el proceso que se debe seguir para llevar a cabo la evaluación de un determinado producto de software, dicho proceso tiene cinco actividades principales, ver Fig. 147.

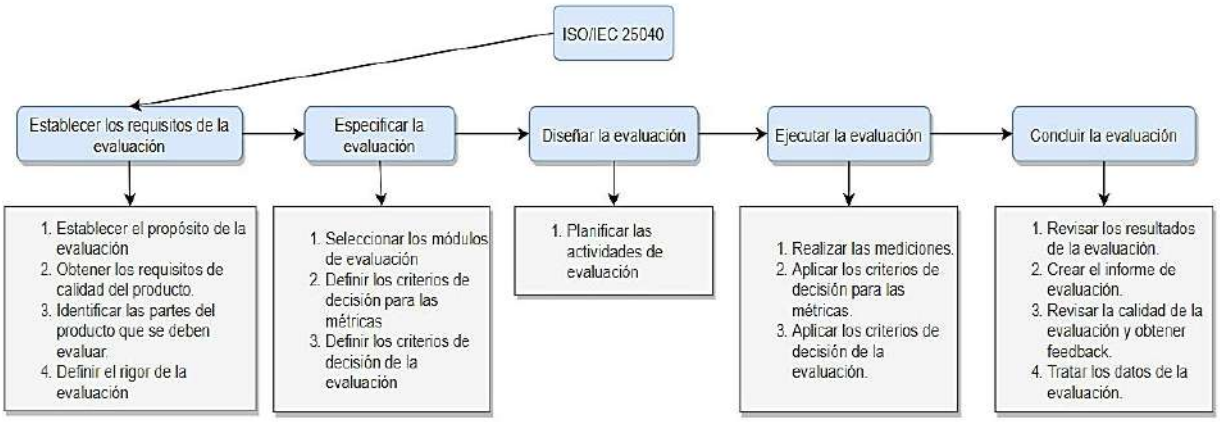


Fig. 147. ISO/IEC 25040:2011.

Fuente: https://www.iso.org/standard/35765.html.

3.1.1. Establecer los requisitos de la evaluación

La evaluación del producto de software será ejecutada en la arquitectura híbrida en la nube, para lo que fue necesario establecer los requisitos y las funcionalidades de la aplicación con el estándar IEEE 830. Después se identificaron los requerimientos de evaluación, ejecución de la evaluación y análisis de los datos obtenidos durante la evaluación.

El presente proyecto se sustenta en los principios Open Science en sintonía con la educación y la investigación, por lo tanto, durante las etapas que han llevado a levantar la aplicación se ha hecho referencia a un elemento macro “Integration OSF & GitHub” basado en la arquitectura orientada a servicios. A consecuencia, fueron desarrolladas las siguientes características generales:

1) Aplicación con contenidos abiertos, ficheros disponibles referentes a diversas áreas del conocimiento con OSF y GitHub.
2) Colabore con otros investigadores mediante la Organización alojada en GitHub.
3) Gestión de componentes alojados en repositorios abiertos en OSF.
4) Aplicación desarrollada con fines académicos, recursos libres y acceda a ellos 24/7.
5) Acceda desde su dispositivo celular con nuestro sistema y aplicación móvil iOS/Android, los contenidos serán siempre accesibles.
6) Verifique su progreso en cada uno de los contenidos que decida tomar, para ello tendrá a disposición un indicador con el porcentaje que ha ido generando.

Requisitos de Software IEEE 830

Los requisitos de la presente aplicación fueron descritos mediante la norma IEEE 830, misma que describe los requerimientos de software y por ende las funciones que posee la aplicación. Aquellos componentes de la arquitectura fueron descritos para obtener la idea global de como interactúan entre sí. La documentación completa se encuentra adjunta en el Anexo 6.

Elementos del producto que serán evaluados

Los elementos que serán objeto de evaluación son:

- Procesos automatizados en la aplicación.
- Instrumentos de desarrollo.
- Aplicativos desarrollados durante la etapa de investigación.
- Requisitos funcionales y no funcionales.
- Evaluar la arquitectura SOA en base a las características de la ISO/IEC 25022.
- Establecer de manera general el rendimiento, seguridad y usabilidad de la aplicación mediante un instrumento de evaluación.

3.1.2. Especificar la evaluación

La aplicación ha sido desarrollada de manera que provee servicios a cualquier sistema externo, por lo que es capaz de interactuar y consumir datos de plataformas mediante la integración de servicios, por tal motivo los datos son manejados de manera centralizada y segura. Lo anteriormente mencionado ha sido logrado mediante la creación de varios módulos los cuales se encargan de gestionar todos los procesos que fueron automatizados a lo largo del desarrollo del proyecto.

Módulos Experimentales

1. **Gestión de proyectos de investigación OSF:** Se encarga de manipular los datos de los proyectos que se encuentran en proceso de investigación, para ello utiliza nodos anexos como nodos, colaboradores, licencia, ficheros contenedores, componentes, historial de actividades y comentarios.
2. **Gestión de versionamiento en repositorios GitHub:** Se encarga de manipular los datos de versionamiento de los repositorios que han sido desarrollados durante el proceso de investigación.
3. **Integración con servicios:** Se encarga de gestionar y consumir los contenidos que han sido producidos durante el proceso de investigación y que están alojados en nubes públicas y privadas: Google, OneDrive, Dropbox, S3 y otros contenedores.
4. **Gestión de seguridad, roles y accesos:** Se encarga de manipular los elementos de acceso, usuario y roles de la aplicación.
5. **Gestión de publicación de contenido abierto:** Se encarga de los procesos que manejan el flujo de publicación del contenido en la aplicación y de acceder a los datos de contenidos que se encuentran en proceso de publicación.
6. **Gestión seguimiento contenidos abiertos:** Se encarga de gestionar los procesos en los que cada usuario deberá hacer seguimiento de los contenidos que publique, administre o consuma.

Criterios de decisión

Para establecer la metodología es necesario especificar cual es el propósito por el cual se pretende realizar la evaluación. Según (Balseca, 2014), las métricas de calidad en uso miden si un determinado sistema computacional satisface las necesidades específicas de los usuarios, por lo que para llevar a cabo la medición es necesario aplicar la evaluación en un ambiente en producción.

Los campos importantes de una determinada métrica independientemente del contexto al que se refiera, deben abarcar los siguientes campos, ver Tabla 13.

Tabla 13. Descripción de campos por métrica.

Fuente: Recuperado de (Balseca, 2014).

Característica	Motivo de elección
Subcaracterística	Subcaracterística de calidad.
Nombre de la métrica	Nombre asignado a la métrica de la calidad.
Fase del ciclo de vida de calidad del producto	Fase del ciclo: calidad en uso.

Propósito de la métrica de calidad	Motivo por el cual se selecciona la métrica.
Método de aplicación	Como se aplicará la métrica.
Fórmula y cálculo de datos	Establecer la fórmula de medición y especifica los significados de los datos que se van a utilizar.
Valor deseado	Proporciona el rango, valores preferibles y recomendados.
Tipo de medida	Especifica el tipo de medida que será seleccionado: tamaño (tamaño de la función, tamaño de la fuente), tiempo (lapso, tiempo de usuario), contador (número de cambios, número de fallas).
Recursos utilizados	Especifica los recursos que serán utilizados para medir cada métrica, entre los recursos pueden estar: entrevistas a usuarios, calidad código fuente, documentación, entre otras.

El nivel de importancia fue establecido con los porcentajes necesarios para cada una de las características, teniendo en cuenta cada uno de los componentes que conforman la aplicación, ver Tabla 14.

Tabla 14. Evaluación de calidad de productos de software aplicando la norma ISO/IEC 25022.

CARACTERÍSTICAS DE CALIDAD EN USO		
Característica	**Nivel y Porcentaje de Importancia**	**Motivo de elección**
Efectividad	H (Alto), 40%	Para el valor de importancia se le asigna el valor ***H*** porque es necesario verificar si la aplicación producto de la integración permite alcanzar los objetivos o necesidades del usuario.
Eficiencia	M (Medio), 30%	Para el valor de importancia se le asigna el valor ***M*** porque es necesario verificar si la aplicación producto de la integración permite alcanzar los objetivos o necesidades del usuario utilizando recursos mínimos.
Satisfacción	M (Medio), 30%	Para el valor de importancia se le asigna el valor ***M*** porque es necesario verificar si la aplicación producto de la integración satisface las necesidades del usuario al utilizarlo.
Libertad de Riesgo	L (Bajo), 0%	Para el valor de importancia se le asigna el valor ***L*** porque no es necesario verificar si la aplicación producto de la integración produce alguna consecuencia en relación con la salud.
Cobertura de Contexto	L (Bajo), 0%	Para el valor de importancia se le asigna el valor ***L*** porque no es necesario verificar dicha característica.

3.1.3. Diseño de la evaluación

Usabilidad y Calidad en uso

Generalmente la usabilidad y calidad en uso son conceptos que han estado durante varios años estrechamente relacionados en áreas de ciencias computacionales. Teniendo en cuenta lo anteriormente mencionado, se hace evidente que para obtener resultados que permitan obtener datos más cercanos a la realidad es necesario utilizar algún instrumento de usabilidad que permita encuestar a los usuarios que hagan uso de la aplicación.

PSSUQ The Post-Study System Usability Questionnaire

Existen varios instrumentos para evaluar productos de software, entre los que destaca PSSUQ The Post-Study System Usability Questionnaire, en español Cuestionario de Usabilidad del Sistema Posterior al Estudio; dicha herramienta consta de un cuestionario de 16 preguntas y todas se miden en una escala de Likert que van desde totalmente de acuerdo hasta totalmente en desacuerdo.

Según (Lewis, 1992), creador del cuestionario lo define como un instrumento para medir la usabilidad percibida de un sistema. Las preguntas cubren la usabilidad, al basar el cuestionario en la definición de usabilidad de ISO 9241-11. Varios trabajos de investigación confirman la solidez del puntaje y los resultados que arroja el cuestionario PSSUQ. La escala puede ser aplicada para detectar posibles problemas de usabilidad, como es de esperarse si el resultado arroja valores bajos indica que los usuarios tuvieron problemas con el sistema y es necesario descubrir cuales son para realizar versiones de mejora.

Tabla 15. Estimaciones fiabilidad, versiones PSSUQ adaptado de (Sauro & Lewis, 2012).

			Subdimensiones			
	Número de ítems	Alfa Cronbach global	Utilidad del sistema (Sysuse)	Calidad de la información (InfoQual)	Calidad de la interfaz (IntQual)	Estudios realizados
PSSUQ v1	18	0,97	0,96	0,91	0,91	Lewis 1993
PSSUQ v2	19	0,96	0,96	0,92	0,83	Lewis 1996
PSSUQ v3	16	0,94	0,9	0,91	0,83	Lewis 2002

Se hará uso de la versión 3 del cuestionario, que tiene 4 subescalas:

- General: los puntajes promedio de las preguntas 1 a 16.
- Utilidad del sistema (SYSUSE): los puntajes promedio de las preguntas 1 a 6.
- Calidad de la información (INFOQUAL): puntajes promedio de las preguntas 7 a 12.
- Calidad de la interfaz (INTERQUAL): las puntuaciones medias de las preguntas 13 a 15.

El cuestionario consta de las siguientes preguntas:

1. En general, estoy satisfecho con lo fácil que es utilizar este sistema.
2. Fue sencillo utilizar este sistema.
3. Pude completar las tareas y escenarios rápidamente usando este sistema.
4. Me sentí cómodo usando este sistema.
5. Fue fácil aprender a usar este sistema.
6. Creo que podría volverme productivo rápidamente usando este sistema.
7. El sistema dio mensajes de error que me indicaban claramente cómo solucionar problemas.
8. Siempre que cometía un error al utilizar el sistema, podía recuperarme fácil y rápidamente.

9. La información (como ayuda en línea, mensajes en pantalla y otra documentación) proporcionada con este sistema era clara.
10. Fue fácil encontrar la información que necesitaba.
11. La información fue eficaz para ayudarme a completar las tareas y los escenarios.
12. La organización de la información en las pantallas del sistema fue clara.
13. La interfaz de este sistema fue agradable.
14. Me gustó usar la interfaz de este sistema.
15. Este sistema tiene todas las funciones y capacidades que espero que tenga.
16. En general, estoy satisfecho con este sistema.

Evaluación en 7 niveles de escala Likert:

1. Totalmente de acuerdo.
2. Bastante de acuerdo.
3. De acuerdo.
4. Neutral.
5. En desacuerdo.
6. Bastante en desacuerdo.
7. Totalmente en desacuerdo.

El cuestionario PSSUQ, está estrechamente relacionado a las características de calidad en uso descritas en la Tabla 15. A continuación, se presenta la relación de las preguntas con cada una de las subcaracterísticas que serán evaluadas.

Tabla 16. Relación de PSSUQ e ISO/IEC 25022.

PSSUQ e ISO/IEC 25022		
Subcaracterística	**Métrica**	**Pregunta (P)**
Efectividad	Completitud de la tarea	P11
	Efectividad de la tarea	P10, P12
	Frecuencia de error	P7, P8, P9
Eficiencia	Tiempo de la tarea	P2, P4
	Tiempo relativo de la tarea	P1, P5, P6
	Frecuencia relativa de la tarea	P3
Utilidad	Nivel de satisfacción	P13, P14
	Uso discrecional de las tareas	P15
	Porcentaje de quejas de los clientes	P16

3.1.4. Ejecución de la evaluación

Tabla 17. Métricas de Calidad en Uso - Matriz de evaluación para productos de Software.

MATRIZ DE EVALUACIÓN CALIDAD EN USO (ISO/IEC 25022)											
Sub característica	**Métrica**	**Propósito**	**Fórmula**	**Valor deseado (Umbral)**	**Aplica**	**Valor Obtenido**	**Ponderación**	**Valor Parcial Total**	**Nivel y Porcentaje de Importancia**	**Valor Final**	**Calidad de la aplicación**
Efectividad	Completitud de la tarea	¿Qué cantidad de tareas son completadas correctamente?	X = A/B A = Número de tareas completadas. B = Número total de tareas intentadas. Donde: B > 0	1	Si	A = 15 B = 15 X = 1	10,00	10,00	H – 40%	4,00	
	Efectividad de la tarea	¿Qué cantidad de los objetivos de la tarea se realiza Completamente?	X = A/B A = Cantidad de objetivos completados por la tarea. B = Cantidad de objetivos planteados por la tarea.	1	Si	A = 3 B = 3 X = 1	10,00				
	Frecuencia de error	¿Cuál es la frecuencia de los errores cometidos por el usuario en comparación con lo planeado?	X = A/B A = Número de errores cometidos por los usuarios. B = Número de tareas. B > 0	0	SI	A = 0 B = 13 X =	10,00				
Eficiencia	Tiempo de la tarea	¿Cuánto tiempo se tarda en completar una tarea en comparación de lo planeado?	X = A/B A = Tiempo planeado. (min) B = Tiempo actual. (min) B > 0	1	Si	A = 25 B = 33 X = 0,8	7,14				
	Tiempo relativo de la tarea	¿Cuánto tiempo necesita un usuario normal en completar una tarea en comparación de un experto?	X = A/B A = Tiempo que completa una tarea un experto. (min) B = Tiempo que completa una tarea un usuario normal (min). B > 0	1	Si	A = 30 B = 40 X = 0,75	7,50				

	Eficiencia relativa de la tarea	¿Qué tan eficiente es un usuario comparado con lo planeado?	X = A/B A = Número de tareas eficientes realizadas por un usuario ordinario. B = Número de tareas eficientes planeadas. B > 0	1	Si	A = 6 B = 6 X = 1	10,00			
	Productividad Económica	¿Qué tan redituables resultan ejecutar las tareas por los usuarios?	X = A/B A = Número de tareas efectivas. B = Número total de las tareas. B > 0	1	No	A = B = X = N/A	N/A			
	Porcentaje Productivo	¿Qué tan productiva resulta una tarea ejecutada?	X = A/B A = Tiempo de la tarea. B = Tiempo de productividad, B > 0	0	No	A = B = X = N/A	N/A	8,21	M - 30%	2.46
	Número relativo de las acciones del usuario	¿Cuál es el número relativo de acciones al ejecutar una tarea?	X = A/B A = Número de acciones realizadas por los usuarios. B = Número de acciones necesarias actualmente. B > 0	1	No	A = B = X = N/A	N/A			
Utilidad	Nivel de satisfacción	¿Qué tan satisfecho está el usuario?	X = A/B A = Número de preguntas con respuestas satisfactorias. B = Número total de preguntas realizadas en el cuestionario. B > 0	1	Si	A = 15 B = 18 X = 0.83	7.33			

9,19

	Uso discrecional de las funciones	¿Qué porcentaje de los usuarios optan por utilizar las funciones del sistema?	X = A/B A = Número de veces que se utilizan las funciones del software. B = Número de veces que están destinados a ser usados. B > 0	1	Si	A = 6 B = 6 X = 1	10,00	9,11	M - 30%	2.73
	Porcentaje de quejas de los clientes	¿Cuál es el porcentaje de quejas realizadas por los clientes?	X = A/B A = Número de clientes que se quejan. B = Número total de clientes. B > 0	0	Si	A = 0 B = 30 X = 0	10,00			

La socialización de la aplicación se realizó de manera remota mediante la herramienta Zoom, al cual asistieron 30 personas, miembros de una unidad educativa; del mismo modo una vez finalizado el proceso se compartió la encuesta realizada en Microsoft Forms, ver Anexo 9.

Tabla 18. Resultados Parciales de la Encuesta.

Pregunta	Totalmente de acuerdo (1)	Bastante de acuerdo (2)	De acuerdo (3)	Neutral (4)	En desacuerdo (5)	Bastante en desacuerdo (6)	Totalmente en desacuerdo (7)
P – 1	25	5	0	0	0	0	0
P – 2	17	13	0	0	0	0	0
P – 3	17	8	5	0	0	0	0
P – 4	14	15	0	1	0	0	0
P – 5	19	8	1	1	1	0	0
P – 6	14	8	7	0	1	0	0
P – 7	20	9	1	0	0	0	0
P – 8	23	6	1	0	0	0	0
P – 9	19	9	1	1	0	0	0
P – 10	19	10	1	0	0	0	0
P – 11	23	7	0	0	0	0	0
P – 12	21	9	0	0	0	0	0
P – 13	24	6	0	0	0	0	0
P – 14	23	7	0	0	0	0	0
P – 15	23	7	0	0	0	0	0
P - 16	24	6	0	0	0	0	0

La aplicación orientada a servicios fue monitoreada con el panel de métricas de Grafana durante todo el proceso con el fin de prevenir cualquier inconveniente en cuanto a los servicios implementados. Ninguna alerta establecida en Alerting Rules mostró cambio alguno, lo que indica que las simulaciones realizadas con anterioridad mostraban el margen de rendimiento correcto que tendrían los componentes al ser usados por varios usuarios.

El criterio de puntaje por usuario según PSSUQ, se establece con los siguientes cálculos:

$$Utilidad\ del\ Sistema\ = \frac{\left[\frac{\sum Puntaje\ preguntas\ (1-6)}{\text{Nro. Encuestados (30)}}\right]}{6} = 1.54\ (Muy\ Bueno)$$

$$Calidad\ de\ la\ Información = \frac{\left[\frac{\sum Puntaje\ preguntas\ (7-12)}{\text{Nro. Encuestados (30)}}\right]}{6} = 1.34\ (Muy\ Bueno)$$

$$Calidad\ de\ la\ Interfaz = \frac{\left[\frac{\sum Puntaje\ preguntas\ (13-15)}{\text{Nro. Encuestados (30)}}\right]}{3} = 1.22\ (Excelente)$$

$$Calidad\ General\ Sistema = \frac{\left[\frac{\sum Puntaje\ de\ todas\ las\ preguntas\ (1-16)}{\text{Nro. Encuestados (30)}}\right]}{16} = 1.39\ (Muy\ Bueno)$$

3.1.5. Conclusión de la evaluación

Después de ejecutar la evaluación de calidad en uso de la aplicación “Integration OSF & GitHub”, los resultados muestran de manera general que en la escala establecida se obtuvo un valor *9,19/10,00.* El puntaje indica que el porcentaje en calidad total se obtiene un porcentaje del 92%, ver Tabla 17.

Tabla 19. Matriz Global Resultados ISO/IEC 25000.

Fuente: Recuperado de *SQuaRE* (System and Software Quality Requirements and Evaluation).

Calidad	Calidad de la Aplicación	Nivel y Porcentaje de la Calidad en Uso	Nivel
USO	9,19	H – 92%	SATISFACTORIA
INTERNA	N/A	N/A	NO EVALUADA
EXTERNA	N/A	N/A	NO EVALUADA
TOTAL	9,19	H – 92%	SATISFACTORIA

Resultados obtenidos de la evaluación calidad en uso

Los datos obtenidos en cada una de las métricas que formaron parte de la evaluación de la calidad en uso arrojaron datos satisfactorios en cuanto a usabilidad.

Tabla 20. Resultados ISO/IEC 25022.

	Característica	Valor Parcial Total	Nivel y Porcentaje de Importancia	Valor Final	Calidad de la Aplicación
CALIDAD EN USO	Efectividad	10,00	H – 40%	4,00	9,19
	Eficiencia	8,21	M – 30%	2,46	
	Satisfacción	9,11	M – 30%	2,73	
	Libertad de Riesgo	N/A	L – N/A	N/A	
	Cobertura de Contexto	N/A	L – N/A	N/A	

Revisar los Resultados de la evaluación de Usabilidad

En la Fig. 148, se puede visualizar las preguntas relacionadas a la calidad de la información (INFOQUAL) y eficiencia, en cuanto a las preguntas 2 y 4 que están relacionadas a la percepción se obtuvo valores referentes al tiempo de las tarea de (%56,67 – 43,33%) y (46,67% – 50% – 3,33%) respectivamente, lo que demuestra que fue necesaria cierta retroalimentación para lograr terminar con las tareas. Por otro lado, las preguntas 1, 5 y 6 muestran que a los usuarios novatos les tomó más tiempo de lo planeado completar la tarea, el tiempo de completar una tarea superó lo planificado y a pesar de que los usuarios novatos terminaron las tareas tardaron más tiempo a comparación de un experto. Finalmente, en cuanto a la pregunta 3 se muestran resultados de (56,67% – 26,67% – 16,67%) respectivamente, lo que indica que todos los usuarios novatos completaron las tareas planificadas, pero con cierta dificultad y es algo previsible.

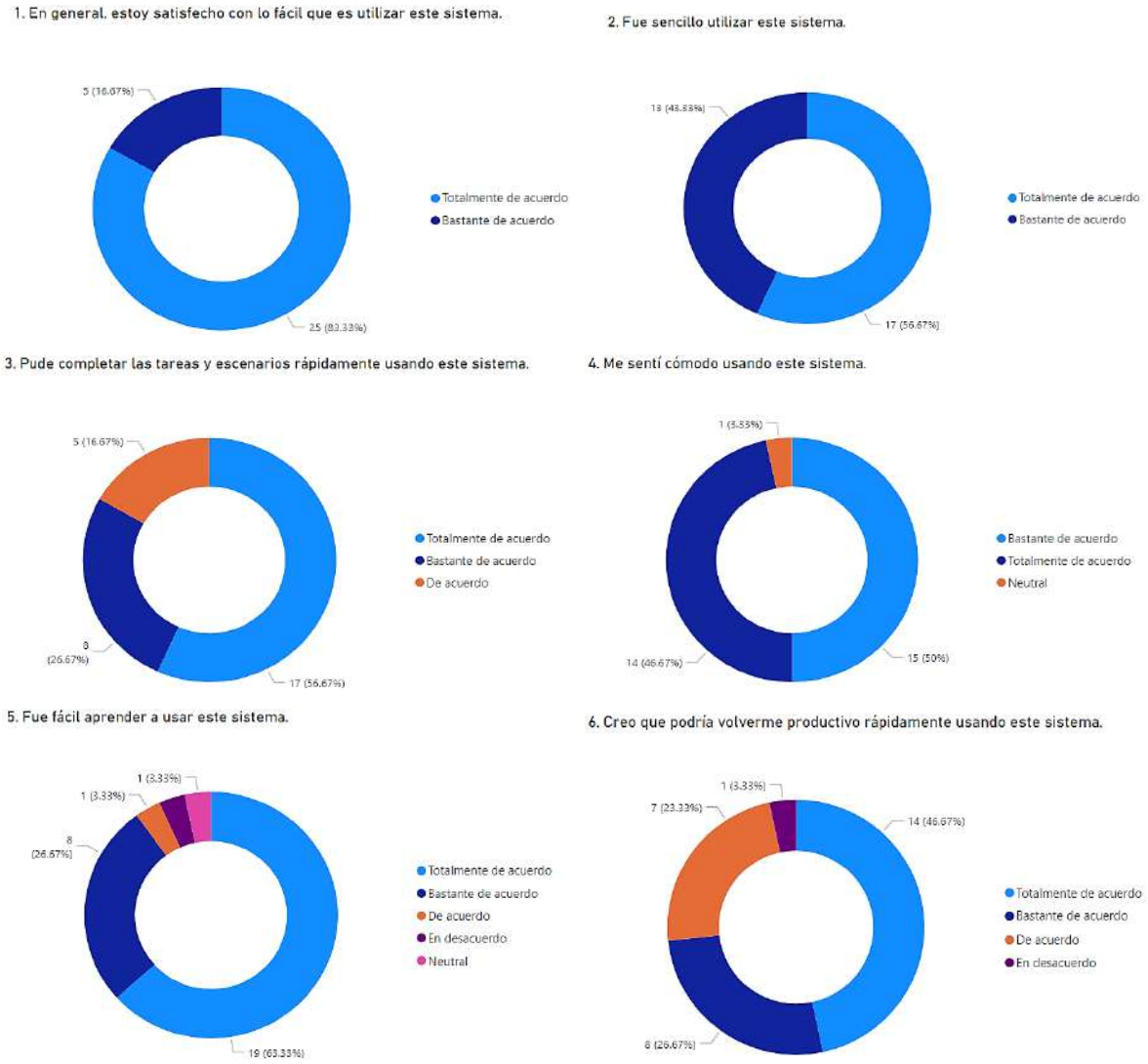


Fig. 148. Utilidad del sistema (SYSUSE).

Fuente: Propia.

En la Fig. 149, se puede visualizar las preguntas relacionadas a la utilidad del sistema (SYSUSE) y Efectividad, en cuanto a las preguntas 7, 8 y 9 que están relacionadas a como el sistema ayudó a minimizar el número de errores cometidos por los usuarios, se obtuvieron valores óptimos (66,67% – 30%, 76,67 – 20% y 63,33 – 30%) respectivamente en las dos primeras escalas. Por otro lado, en la efectividad de la tarea y las preguntas 10 y 12 se obtuvo porcentajes de (63,33% – 33,33% y 70% – 30%) respectivamente, lo que indica que el modo en el que se mostraban las interfaces ayudó a culminar con los objetivos planteados. Finalmente, en cuanto a la pregunta 11 y la completitud de la tarea se muestra que se obtuvieron los porcentajes (76,67% – 23,33%) para las dos escalas altas, lo que indica que para terminar las tareas fueron cruciales las interfaces dinámicas y minimalistas.

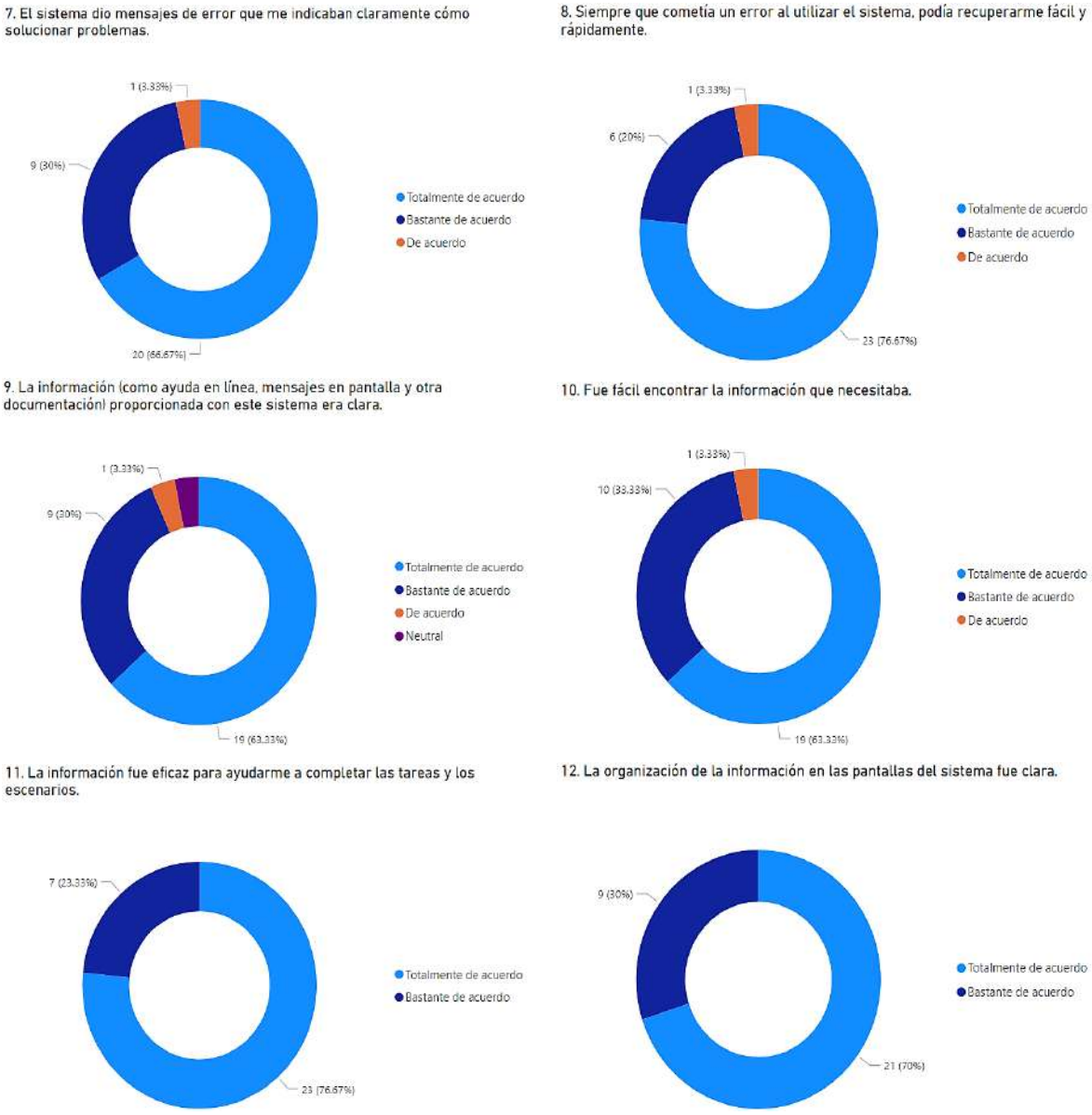


Fig. 149. Calidad de la información (INFOQUAL).

Fuente: Propia.

En la Fig. 150, se puede visualizar las preguntas relacionadas calidad de la interfaz (INTERQUAL) y Utilidad, en cuanto a las preguntas 13 y 14 los porcentajes fueron óptimos y es importante mencionar que la métrica especifica es nivel de satisfacción, la cual fue de 18/20 ya que algunos usuarios tuvieron problemas en completar los escenarios, sin embargo, después de dar las indicaciones necesarias culminaron las tareas y objetivos, ver Tabla 16.

En cuanto a la pregunta 15, los usuarios hicieron uso de las funciones establecidas, por lo que se muestra que el porcentaje fue de (76,67% – 23,33%) para las dos escalas altas.

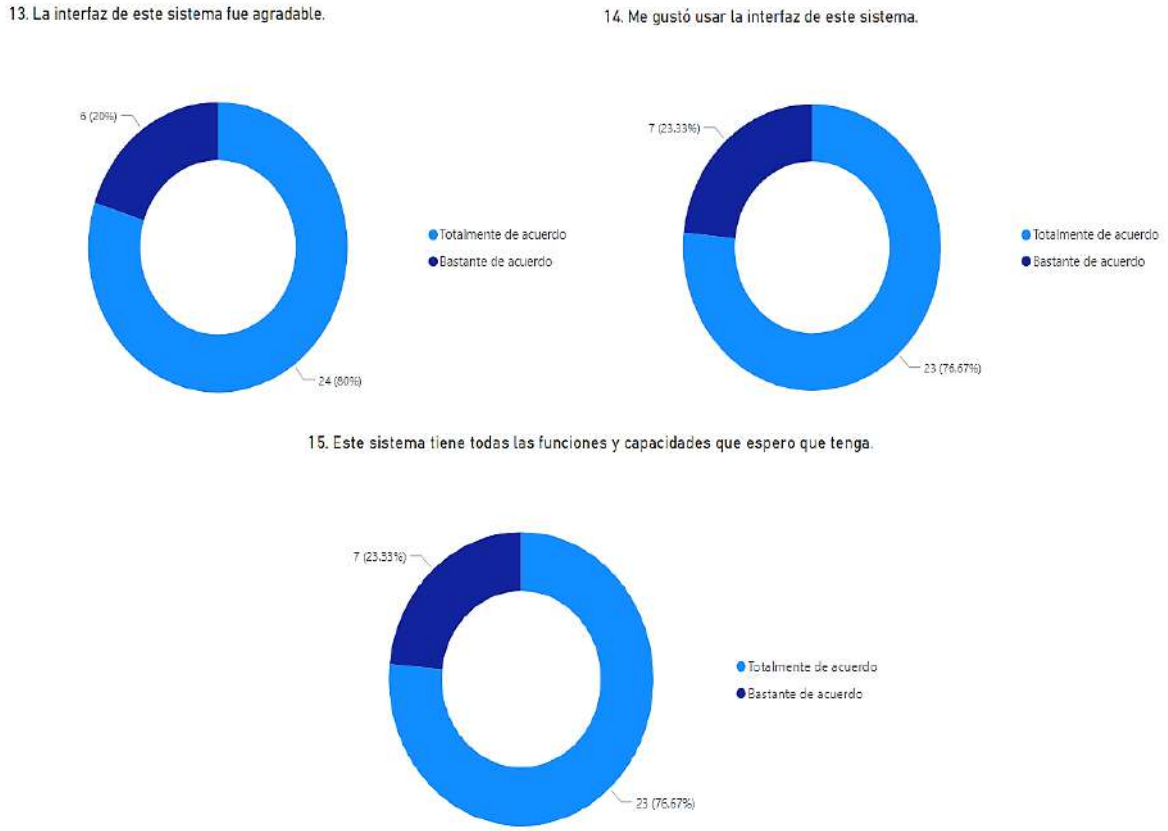


Fig. 150. Calidad de la interfaz (INTERQUAL).

Fuente: Propia.

Finalmente, en la pregunta 16 se muestran los porcentajes de (80,00% – 20,00%) que muestra que los usuarios no presentaron quejas en cuanto a la aplicación, ver Fig. 151.

Es necesario recalcar que la métrica (porcentaje de quejas de los clientes) fue evaluada en base a problemas que pudiesen experimentar los usuarios, al implementar Prometheus y Grafana se evitaron posibles contratiempos en producción, ya que al tener un entorno de pruebas se estableció un límite de usuarios, además las reglas establecidas en node exporter no mostraron alertas en la consola de Prometheus.

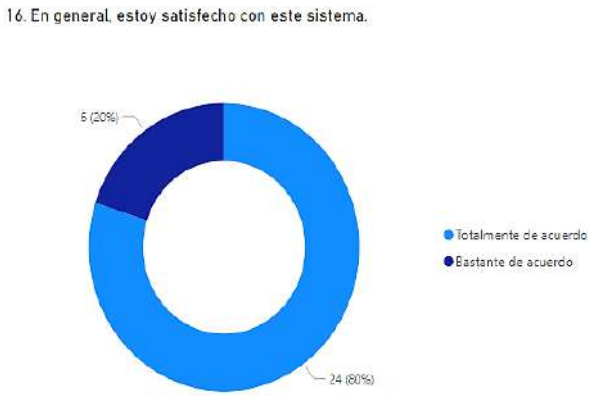


Fig. 151. Resultado Pregunta 16 encuesta PSSUQ

Fuente: Propia.

Interpretación de los Resultados

PSSUQ

Para determinar el porcentaje en cada una de las escalas PSSUQ es necesario realizar una transformación final de los datos. En el primer paso es necesario transformar los puntajes de cada escala 1 para “Totalmente en desacuerdo” y 7 para “Totalmente de acuerdo”

En el segundo paso el puntaje se le restará al total de la escala y se le suma 1; por ejemplo, el promedio en SYSUSE es: ((7 – 1,54) + 1) → 6,46. Finalmente, para obtener el porcentaje de percepción del usuario, el promedio será multiplicado por 100 y el resultado será dividido para 7; por ejemplo, el porcentaje en SYSUSE es: ((6,46 * 100) / 7) → 92,29 %.

Tabla 21. Resultados encuesta PSSUQ.

Resultados PSSUQ			
Escala	**Puntaje**	**Promedio**	**Porcentaje de percepción de Usabilidad**
Utilidad del sistema (SYSUSE)	1,54	6,46	92,29 % (Muy Buena)
Calidad de la Información (INFOQUAL)	1,34	6,66	95,14 % (Muy Buena)
Calidad de la Interfaz (INTERQUAL)	1,22	6,78	96,86 % (Excelente)
Calidad General del Sistema	1,39	6,61	94,42 % (Muy Buena)

En cuando a la valoración de la usabilidad percibida por los usuarios en la aplicación según el cuestionario PSSUQ. En la utilidad del sistema el resultado fue de 92,29 % Muy Buena, en la calidad de la información el resultado fue de 95,14 % Muy Buena, en la calidad de la interfaz

el resultado fue de 96,86 % Excelente y en la calidad general del Sistema el resultado fue de **94,42 % Muy Buena**. Los resultados anteriormente mencionados demuestran que la aplicación proporciona los servicios, funcionalidades, y otros elementos que generan que el usuario perciba un mayor nivel de aceptación en cuanto a usabilidad de los componentes producto de la arquitectura orientada a servicios “Integration OSF & GitHub”.

Calidad en Uso ISO/IEC 25022

En la Fig. 152, se puede visualizar los resultados de cada una de las características de calidad en uso que fueron seleccionadas para la evaluación. Tal como se muestra las características que obtuvieron mayores puntajes fueron Efectividad (40%/40%) y Satisfacción (27,30%/30%) respectivamente. Por otro lado, la característica Eficiencia obtuvo un valor de (24,60%/30%), refleja que al usuario tardó en completar alguna tarea de acuerdo con el tiempo planeado y por ende los mensajes de alerta e instrucciones recomendadas por las heurísticas de Nielsen resultaron cruciales para lograr que el usuario pueda recuperar el ritmo de la tarea.

Finalmente, el criterio de evaluación de la calidad en uso es Satisfactorio y demuestra que la arquitectura es óptima, sin embargo, para futuras versiones será necesario implementar otros elementos para que los niveles de Eficiencia y Satisfacción sean cercanos a 100%.

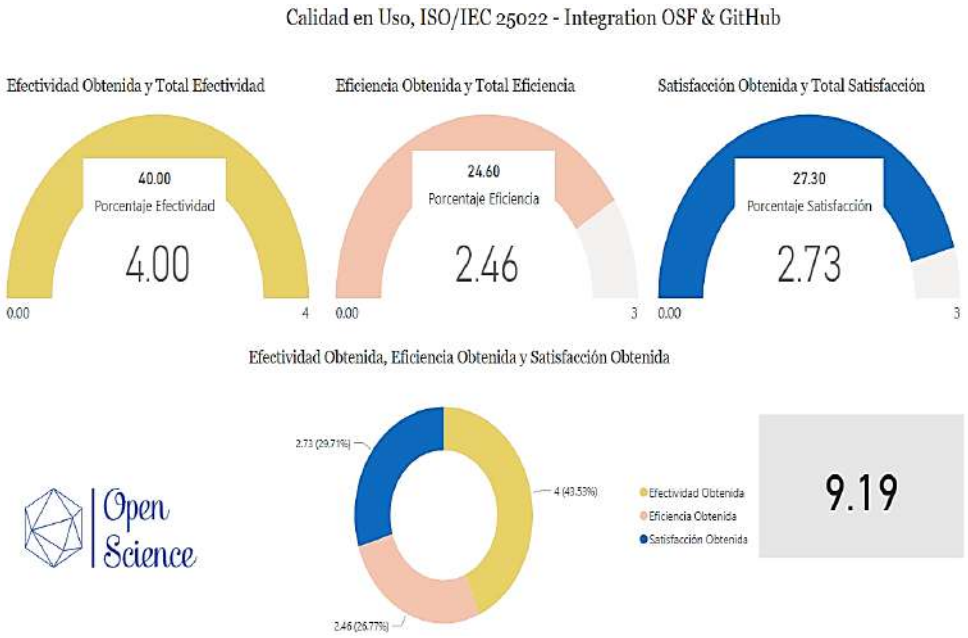


Fig. 152. Resultados Calidad en Uso de la evaluación de la Aplicación "Integration OSF & GitHub".

Fuente: Propia.

CONCLUSIONES

La integración de servicios relacionados a ciencia abierta permite gestionar contenido de diversas fuentes de una manera eficiente, sin embargo, es necesario tener claras las pautas y alcance del software a desarrollar. Del mismo modo se deben involucrar alternativas tecnológicas, tales como una arquitectura híbrida en la nube, ya que a medida que se consumen datos de diversas fuentes será posible dar seguimiento de manera oportuna a todos los componentes que sean implementados.

La publicación de contenido abierto será dentro de unos años una de las principales pautas para generar conocimiento científico, ya que permite la replicabilidad, reutilización y reinvención de componentes que son producto de una investigación. OSF resulta ser un excelente marco de trabajo de investigación aplicado en prestigiosas universidades, por ejemplo, el MIT *Massachusetts Institute of Technology* institución a la que están asociadas las licencias de gran parte de las tecnologías de software utilizadas en el proyecto.

La escalabilidad de una arquitectura orientada a servicios desde un punto de vista TI requiere herramientas que permitan gestionar todos los componentes desplegados en capas, tal como se aplicó a la investigación. La reutilización, modularidad, bajo acoplamiento y otras características deben ser tomadas en cuenta desde el inicio del desarrollo, ya que a medida que un proyecto incrementa su volumen necesita mecanismos para gestionar de manera eficiente los procesos y la información que operan. La plataforma GitHub con los servicios que posee permitió hacer el seguimiento de la escalabilidad, despliegue a producción y versionamiento de manera sencilla; no hubo contratiempos al desplegar los componentes de la arquitectura y el presente proyecto de investigación es un claro ejemplo de ello.

En materia de software, realizar pruebas de rendimiento durante todas las etapas de desarrollo es una de las varias normas indispensables a seguir, ya que así es posible garantizar productos de calidad. También es necesario realizar pruebas de usabilidad a pequeña escala que permitan mitigar los posibles contratiempos que pueden experimentar los usuarios al hacer uno de un determinado servicio, con ello la usabilidad está garantizada.

La norma ISO/IEC 25022:2016 proporciona un modelo para medir las métricas de la calidad en uso de un determinado producto de software, por lo que el proceso de evaluación permite involucrar a todos los elementos relacionados a las etapas del ciclo de vida del software. Los resultados arrojados por el análisis de la calidad en uso se encuentran por encima del 80% ya que fueron aplicadas heurísticas y buenas prácticas durante las etapas de desarrollo en conjunto con herramientas de AWS; que dicho sea de paso demuestra ser una de los mejores y más sofisticados proveedores de servicios en la nube.

RECOMENDACIONES

Indagar en metodologías emergentes en cuanto a publicación de contenido abierto, la dimensión del tema en efecto resulta ser extensa por tal razón en la presente investigación se hizo hincapié en ciencias de la computación, desarrollo de software y métodos afines, sin embargo, la taxonomía cambia de acuerdo con la rama científica.

GitHub y OSF, poseen varios servicios además de proveer herramientas para desarrollador modernas; con lo anteriormente mencionado se hace evidente que las posibilidades son muy extensas en cuando a contenido abierto se trata. Una investigación puede escalar hacia diversas ramas y generar nuevas técnicas en base a la colaboración, por ello se recomienda aplicar su uso a enfoques relacionados a Ingeniería de Software.

Aplicar los procesos y técnicas DevOps de la presente investigación en proyectos afines a ingeniería de software y arquitecturas TI; ya que las tecnologías, servicios en la nube, contenedores, lenguajes de programación, plataformas y otros elementos que fueron implementados son de gran demanda en la industria por su gran versatilidad y alta tasa de adopción en organizaciones prestigiosas.

El desarrollo del proyecto mantuvo directrices en base a estándares, normas, heurísticas, recomendaciones de expertos y otros elementos que permitieron generar productos escalables y eficientes. Por tal motivo, se recomienda seguir una pauta de buenas prácticas en todas las etapas de desarrollo y con más razón cuando se trata de un proyecto de investigación, ya que se pretende obtener los mejores resultados posibles.

Para garantizar un eficiente funcionamiento de una arquitectura híbrida como la que se construyó es necesario usar herramientas que permitan monitorear el rendimiento de todos los componentes desarrollados, independientemente de los servicios que sean consumidos. AWS posee servicios para migrar un ambiente nativo a una nube híbrida en su totalidad, por lo que si se pretende escalar servicios en la nube deben ser desplegados a VMware on AWS.

Las herramientas de monitoreo de eventos permiten manejar los errores de manera eficiente, por lo que es recomendable instalarlas desde el inicio del proyecto, ya que resulta óptimo conocer a detalle todos aquellos problemas que puedan surgir en un determinado sistema y solucionarlos para mitigar el índice de impacto.

Si se pretende llevar una pauta ordenada para la evaluación de software, se recomienda aplicar los procesos que establece la norma ISO/IEC 25040, ya que presenta varias actividades que deben ser seguidas secuencialmente. También es recomendable añadir instrumentos como evaluaciones e inclusive otras normas.

TRABAJO FUTURO

Durante el proceso de desarrollo del presente proyecto se evidenciaron otros trabajos a futuro que por el alcance, tiempos y recursos planificados no fue posible implementarlos, inclusive se obtuvieron sugerencias de implementación de otras funcionalidades macro que pueden ser desarrolladas en trabajos relacionados a la misma línea de investigación:

- Integración de los flujos de trabajo de GitLab para gestión de repositorios de índole empresarial, ya que resultaría oportuno generar nuevos proyectos en base a la colaboración de desarrolladores o estudiantes de carreras afines.
- Los módulos académicos realizados para el presente proyecto fueron experimentales, es decir fueron hechos para medir el impacto de generar contenido abierto mediante una de las ramas Open Science, por lo que resultaría fructífero explorar otra rama, por ejemplo, la rama Infrastructure School que basa la búsqueda de nuevos métodos científicos en base a la colaboración en la red para cubrir las necesidades de organizaciones públicas y privadas.
- Integración de servicios de inteligencia artificial, varias de las herramientas propuestas son utilizadas en áreas, tales como ciencia de datos, minería de datos y otras. En la arquitectura se estableció una capa de IA, misma que podría proveer servicios para análisis avanzado de texto, revisión de código automático, webscraping, chatbots, reducción en tiempo real y otras herramientas sofisticadas que posee AWS.
- Generar rutinas de monitoreo profundo TI aplicadas a flujos de CI/CD, ya que manejar entornos controlados mediante la orquestación de contenedores Docker demuestra tener buenos resultados en cuanto a escalabilidad y rendimiento de un determinado proyecto relacionado al despliegue de arquitecturas de software.
- Escalar el aplicativo de arquitectura nativa, que permita realizar transacciones mediante la reutilización de servicios que ofrecen organizaciones públicas, ya que con los accesos pertinentes es posible incrementar la arquitectura según sea necesario.
- Migrar los componentes de una arquitectura desplegada en un ambiente nativo a un ambiente híbrido demuestra tener enormes ventajas en cuanto a rendimiento, por ello aplicarlas en arquitecturas orientadas a servicios o microservicios resulta redituable. Por tal motivo es importante realizar trabajos que permitan explotar las características de servicios en la nube altamente adoptados a nivel global.

REFERENCIAS

Abadal, E., & Lluís, A. (2020). *CIENCIA ABIERTA: CÓMO HAN EVOLUCIONADO LA DENOMINACIÓN Y EL CONCEPTO*. http://dx.doi.org/10.6018/analesdoc.378171

Adobe. (2021). *AdobeXD Design like you always imagined.* https://www.adobe.com/products/xd.html

Altunay, M., Avery, P., Blackburn, K., Bockelman, B., Ernst, M., Fraser, D., Quick, R., Gardner, R., Goasguen, S., Levshina, T., Livny, M., Mcgee, J., Olson, D., Pordes, R., Potekhin, M., Rana, A., Roy, A., Sehgal, C., Sfiligoi, I., & Board, T. (2010). A Science Driven Production Cyberinfrastructure—the Open Science Grid. *Journal of Grid Computing*, *9*, 201–218. https://doi.org/10.1007/s10723-010-9176-6

Amazon Web Services - Documentation. (2021). https://docs.aws.amazon.com/index.html?nc2=h_mo

Anastasov, M. (2021). *CI/CD: Continuous Integration & Delivery Explained*. https://semaphoreci.com/cicd

Balseca, E. A. (2014). *Evaluación de calidad de productos de software en empresas de desarrollo de software aplicando la norma ISO/IEC 25000*. https://bibdigital.epn.edu.ec/handle/15000/9113

Bartling, S. (2014a). *How This Book was Created Using Collaborative Authoring and Cloud Tools* (pp. 313–315). https://doi.org/10.1007/978-3-319-00026-8_24

Bartling, S. (2014b). *Organizing Collaboration on Scientific Publications: From Email Lists to Cloud Services* (pp. 289–291). https://doi.org/10.1007/978-3-319-00026-8_20

Bartling, S., & Friesike, S. (2014). *Opening Science - The Evolving Guide on How the Web is Changing Research, Collaboration and Scholarly Publishing*.

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

Bokhari, S., Azam, F., & Habiba, U.-E. (2015). Limitations of Service Oriented Architecture and its Combination with Cloud Computing. *BUJICT*, *8*.

Boulton, G., Rawlins, M., Vallance, P., & Walport, M. (2011). Science as a public enterprise: The case for open data. *Lancet*, *377*, 1633–1635. https://doi.org/10.1016/S0140-6736(11)60647-8

CH Yeong; Abdullah BJJ. (2012). Altmetrics: The right step forward. *Biomedical Imaging and*

Intervention Journal, *8*(2), 1–2. https://doi.org/10.2349/biij.8.3.e15

Chesbrough, H. (2006). *Open Innovation: A New Paradigm for Understanding Industrial Innovation*. http://openinnovation.net/category/open-innovation/

Commision, E. (2020). *Open Science Monitor*. https://ec.europa.eu/info/research-and-innovation/strategy/strategy-2020-2024/our-digital-future/open-science/open-science-monitor_en

Commission, E. (2017). *HORIZON 2020 - Work Programme 2016 - 2017 Science with and for Society*. https://ec.europa.eu/research/participants/data/ref/h2020/wp/2016_2017/main/h2020-wp1617-swfs_en.pdf

Dart Overview. (2021). https://dart.dev/overview

David, P. (2008). The Historical Origins of `Open Science': An Essay on Patronage, Reputation and Common Agency Contracting in the Scientific Revolution. *Capitalism and Society*, *3*, 5. https://doi.org/10.2202/1932-0213.1040

De Roure, D., Goble, C., Aleksejevs, S., Bechhofer, S., Bhagat, J., Cruickshank, D., Fisher, P., Hull, D., Michaelides, D., Newman, D., Procter, R., Lin, Y., & Poschen, M. (2010). Towards Open Science: The myExperiment approach. *Concurrency and Computation: Practice and Experience*, *22*. https://doi.org/10.1002/cpe.1601

DigiKam. (2021). *Professional Photo Management with the Power of Open Source*. https://www.digikam.org/about/

Dixit, V. ., & chhabra, shruti. (2015). Cloud Computing: State of the Art and Security Issues. *ACM SIGSOFT Software Engineering Notes Volume 40 Issue 2, March 2015*, *40*. https://doi.org/10.1145/2735399.2735405

Docker Documentation. (2021). https://docs.docker.com

Estdale, J., & Georgiadou, E. (2018). *Applying the ISO/IEC 25010 Quality Models to Software Product: 25th European Conference, EuroSPI 2018, Bilbao, Spain, September 5-7, 2018, Proceedings* (pp. 492–503). https://doi.org/10.1007/978-3-319-97925-0_42

Faniran, V. T., Badru, A., & Ajayi, N. (2017). Adopting Scrum as an Agile approach in distributed software development: A review of literature. *2017 1st International Conference on Next Generation Computing Applications (NextComp)*, 36–40. https://doi.org/10.1109/NEXTCOMP.2017.8016173

Fecher, B., & Friesike, S. (2014). *Open Science: One Term, Five Schools of Thought BT - Opening Science: The Evolving Guide on How the Internet is Changing Research, Collaboration and Scholarly Publishing* (S. Bartling & S. Friesike (Eds.); pp. 17–47). Springer International Publishing. https://doi.org/10.1007/978-3-319-00026-8_2

Flutter Docs. (2021). https://flutter.dev/docs/get-started/learn-more

Foster, I. (2003). The Grid: A New Infrastructure for 21st Century Science. In *Phys. Today* (Vol. 55, pp. 51–63). https://doi.org/10.1002/0470867167.ch2

Friesike, S. (2014). *Creative Commons Licences* (pp. 287–288). https://doi.org/10.1007/978-3-319-00026-8_19

Fry, J., Schroeder, R., & Besten, M. (2009). Open science in e-science: Contingency or policy? *Journal of Documentation*, *65*, 6–32. https://doi.org/10.1108/00220410910926103

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. (2021). *GitHub Docs API REST*. https://docs.github.com/en/github

GitKraken Documentation. (2021). https://support.gitkraken.com

Golder, S., & Macy, M. (2012). *Social Science with Social Media.* https://www.asanet.org/sites/default/files/savvy/footnotes/jan12/socialmedia_0112.html

Grafana, L. (2021). *Get started or start exploring Grafana*. https://grafana.com/docs/

Harnad, S. (2007). *The Green Road to Open Access: A Leveraged Transition*.

IBM. (2021). *SOA vs. Microservices: What's the Difference?* https://www.ibm.com/cloud/blog/soa-vs-microservices

Jailia, M., Kumar, A., Agarwal, M., & Sinha, I. (2016). Behavior of MVC (Model View Controller) based Web Application developed in PHP and .NET framework. *2016 International Conference on ICT in Business Industry & Government (ICTBIG)*, 1–5. https://doi.org/10.1109/ICTBIG.2016.7892651

Laravel, T. (2021). *Laravel Documentation*. https://laravel.com/docs/8.x

LERU. (2018). *Open Science and its role in universities: A roadmap for cultural change LEaGUE OF EUROPEan RESEaRCH UnIVERSITIES*. *may*.

https://ec.europa.eu/research/openscience/index.cfm;

Lewis, J. (1992). Psychometric evaluation of the post-study system usability questionnaire: The PSSUQ. In *Proceedings of the Human Factors Society* (Vol. 2).

MacCallum, C. (2018). *Research Communication: Open Science & the perverse evaluation cycle*. https://doi.org/10.14324/111.1234

Mackier, D. (2019). *Flutter Architecture-ScopedModel, A complete guide to real world architecture*. https://medium.com/flutter-community/flutter-architecture-scopedmodel-a-complete-guide-to-real-world-architecture-205a24674964

Merton, R. K. (1968). Social theory and social structure. *New York: The Free Press*. https://psycnet.apa.org/record/1959-00989-000

Mintel. (2019). Libro Blanco, Líneas de Investigación, Desarrollo e Innovación y Transferencia del Conocimiento en TIC. *Telecomunicaciones.Gob.Ec*. https://www.telecomunicaciones.gob.ec/wp-content/uploads/2019/01/libro-blanco-lineas-de-investigacion.pdf

Moedas, C. (2020). *Commissioner for Research, Science and Innovation.* https://www.leru.org/files/LERU-AP24-Open-Science-full-paper.pdf

Nakai, H., Tsuda, N., Honda, K., Washizaki, H., & Fukazawa, Y. (2016). Initial Framework for Software Quality Evaluation Based on ISO/IEC 25022 and ISO/IEC 25023. *2016 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, 410–411. https://doi.org/10.1109/QRS-C.2016.66

Nentwich, M. (2003). Cyberscience: Research in the Age of the Internet. In *Information Research - IR* (Vol. 9).

Nielsen, J. (1994). *Enhancing the explanatory power of Usability Heuristics*. 152–158. https://doi.org/10.1145/191666.191729

Nielsen, M. (2013). *Reinventing Discovery: The New Era of Networked Science*. https://press.princeton.edu/books/paperback/9780691160191/reinventing-discovery

NodeJS Documentation. (2021). https://nodejs.org/es/docs/

Olbrich, S., Cruzes, D. S., Basili, V., & Zazworka, N. (2009). The evolution and impact of code smells: A case study of two open source systems. *2009 3rd International Symposium on Empirical Software Engineering and Measurement, ESEM 2009*, 390–400.

https://doi.org/10.1109/ESEM.2009.5314231

OSF.IO. (2021). *OSF API Developer*. https://developer.osf.io/

Pacheco, R., Nascimento, E., & Weber, R. (2018). *Digital Science: Cyberinfrastructure, e-Science and Citizen Science* (pp. 377–388). https://doi.org/10.1007/978-3-319-73546-7_24

Peters, Isabella. Frodeman, Robert. Wilsdon, James. Bar-Ilan, Judit. Lex, Elisabeth. Wouters, P. (2017). *Next-generation metrics Responsible metrics and evaluation for open science*. https://op.europa.eu/en/publication-detail/-/publication/b858d952-0a19-11e7-8a35-01aa75ed71a1/language-en

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

Priem, J., & Hemminger, B. (2012). Decoupling the Scholarly Journal. *Frontiers in Computational Neuroscience*, *6*, 19. https://doi.org/10.3389/fncom.2012.00019

Sauro, J., & Lewis, J. (2012). Quantifying the User Experience. In *Quantifying the User Experience*. https://doi.org/10.1016/C2010-0-65192-3

Schöpfel, J., Prost, H., & Rebouillat, V. (2017). Research Data in Current Research Information Systems. *Procedia Computer Science*, *106*. https://doi.org/10.1016/j.procs.2017.03.030

Sidler, M. (2014). *Open Science and the Three Cultures: Expanding Open Science to all Domains of Knowledge Creation* (pp. 81–85). https://doi.org/10.1007/978-3-319-00026-8_5

TypeScript Documentation. (2021). https://www.typescriptlang.org/docs/

Urbina, V., Vera-Rivera, F., & Pérez, B. (2016). *Modelo de Nube Híbrida (Hybrid Cloud) de Infraestructura como Servicio para Mejorar el Rendimiento de la Plataforma Sandbox-UFPS*.

VueJs. (2021). *Vue JS, Introduction*. https://vuejs.org/v2/guide/

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

ANEXOS

ANEXO 1: Product Backlog Sprint 1

https://bit.ly/3D4a1mv

ANEXO 2: Guía instalación Prometheus y Grafana AWS

(Prometheus) https://go.aws/3pqygIq - https://bit.ly/3kPWo46

(Grafana) https://go.aws/2Xko2xx - https://bit.ly/3ARfAmT

ANEXO 3: Product Backlog Sprint 2

https://bit.ly/2ZzpEof

ANEXO 4: Product Backlog Sprint 3

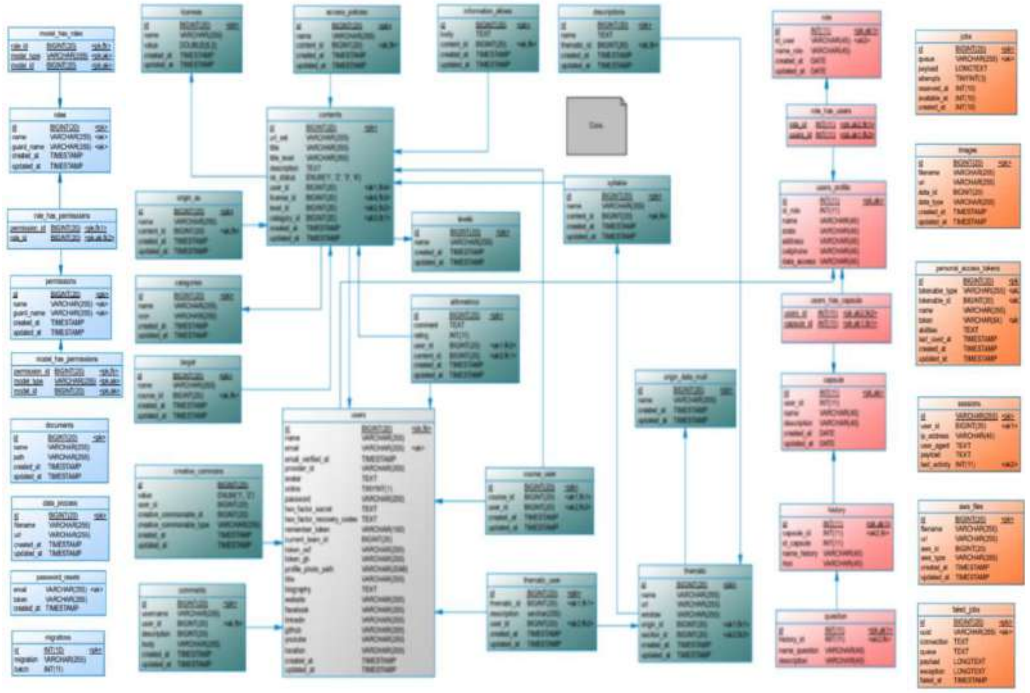
https://bit.ly/31cjQl2

ANEXO 5: Especificación de Requisitos de Software IEEE 830

https://bit.ly/2ZxORiK

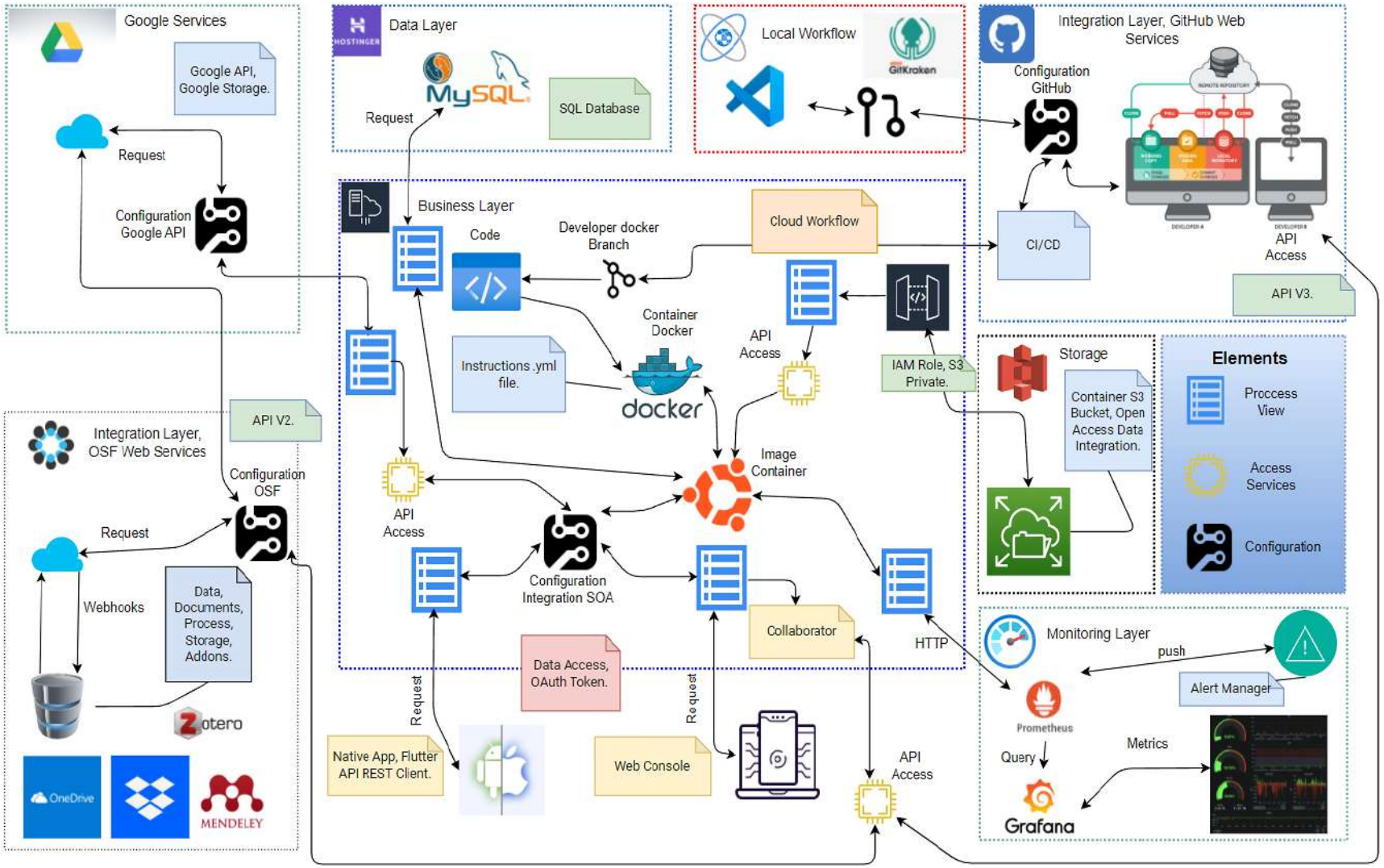
ANEXO 6: Modelo Físico, Base de Datos Final

A continuación, se muestra el bosquejo final del modelo físico de la base de datos producto de la integración de las APIs REST de OSF y GitHub.



ANEXO 7: Arquitectura Final

A continuación, se muestra la arquitectura orientada a servicios con todos los componentes, capas, conexiones, y otros elementos.



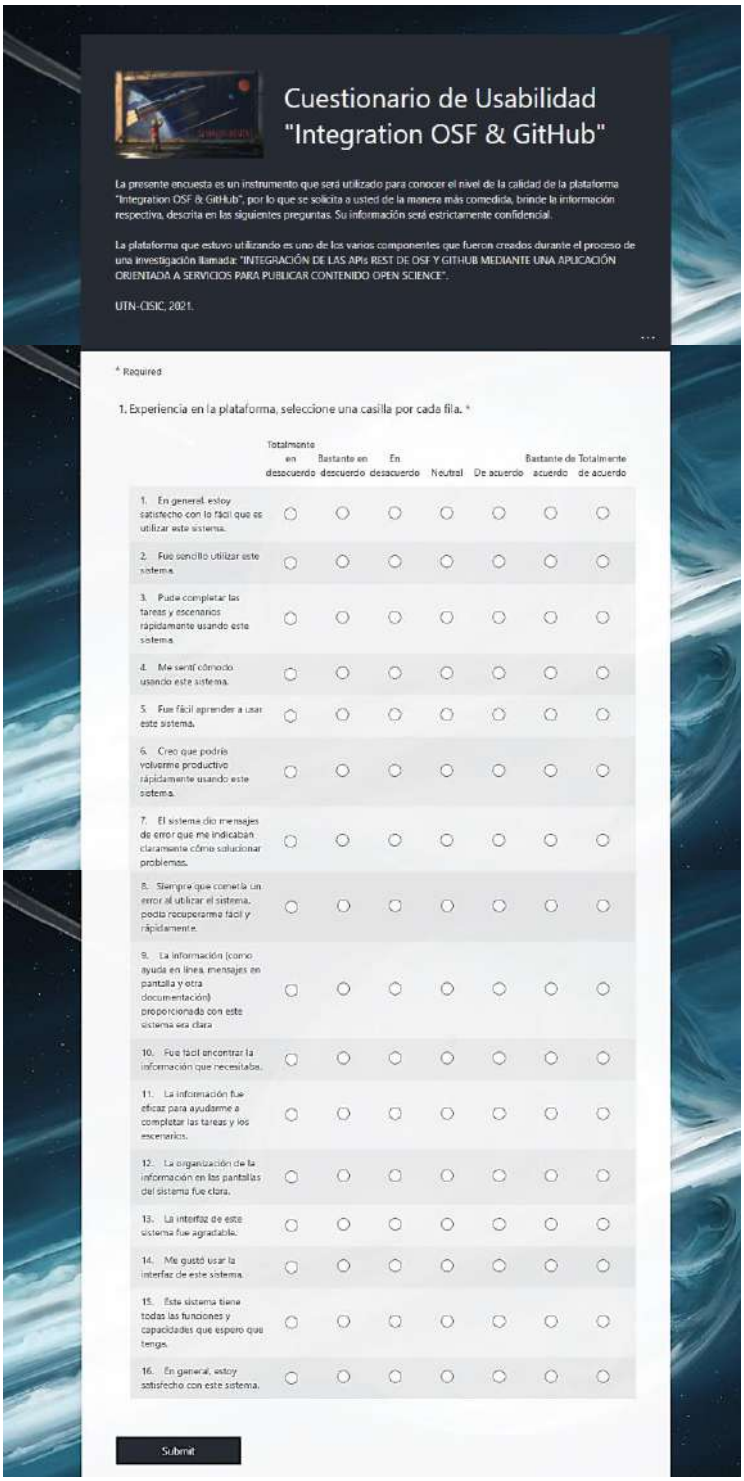
ANEXO 8: Firebase App Distribution

(Android) https://appdistribution.firebase.dev/i/9f69333b83b9bc05

(iOS) https://appdistribution.firebase.dev/i/8ab5f5b19f802569

ANEXO 9: Formato Encuesta PSSUQ

Instrumento para validación de la calidad en uso de la norma ISO/IEC 25022.



Cuestionario de Usabilidad
"Integration OSF & GitHub"

La presente encuesta es un instrumento que será utilizado para conocer el nivel de la calidad de la plataforma "Integration OSF & GitHub", por lo que se solicita a usted de la manera más comedida, brinde la información respectiva, descrita en las siguientes preguntas. Su información será estrictamente confidencial.

La plataforma que estuvo utilizando es uno de los varios componentes que fueron creados durante el proceso de una investigación llamada: "INTEGRACIÓN DE LAS APIs REST DE OSF Y GITHUB MEDIANTE UNA APLICACIÓN ORIENTADA A SERVICIOS PARA PUBLICAR CONTENIDO OPEN SCIENCE".

UTN-CISIC, 2021.

* Required

1. Experiencia en la plataforma, seleccione una casilla por cada fila. *

	Totalmente en desacuerdo	Bastante en desacuerdo	En desacuerdo	Neutral	De acuerdo	Bastante de acuerdo	Totalmente de acuerdo
1. En general, estoy satisfecho con lo fácil que es utilizar este sistema.							
2. Fue sencillo utilizar este sistema.							
3. Pude completar las tareas y escenarios rápidamente usando este sistema.							
4. Me sentí cómodo usando este sistema.							
5. Fue fácil aprender a usar este sistema.							
6. Creo que podría volverme productivo rápidamente usando este sistema.							
7. El sistema dio mensajes de error que me indicaban claramente cómo solucionar problemas.							
8. Siempre que cometía un error al utilizar el sistema, podía recuperarme fácil y rápidamente.							
9. La información (como ayuda en línea, mensajes en pantalla y otra documentación) proporcionada con este sistema era clara							
10. Fue fácil encontrar la información que necesitaba.							
11. La información fue eficaz para ayudarme a completar las tareas y los escenarios.							
12. La organización de la información en las pantallas del sistema fue clara.							
13. La interfaz de este sistema fue agradable.							
14. Me gustó usar la interfaz de este sistema.							
15. Este sistema tiene todas las funciones y capacidades que espero que tenga.							
16. En general, estoy satisfecho con este sistema.							

Submit