UNIVERSIDAD TÉCNICA DEL NORTE

Facultad de Ingeniería en Ciencias Aplicadas

Carrera de Ingeniería en Sistemas Computacionales

DESARROLLO DE UN ENVOLTORIO DEL API-REST DE MENDELEY CON GRAPHQL

Trabajo de grado previo a la obtención del título de Ingeniero en Sistemas Computacionales

Autor:

Kevin David Rodríguez Barahona

Director:

MSc. José Antonio Quiña Mera

Ibarra – Ecuador

2020

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 DEL CONTACTO			
CÉDULA DE IDENTIDAD:	1004012627		
APELLIDOS Y NOMBRES:	RODRÍGUEZ BARAHONA KEVIN DAVID		
DIRECCIÓN:	URCUQUÍ – BARRIO EL ROSARIO, CALLE ALBERTO AMADOR Y HERIBERTO VALLEJO		
EMAIL:	kdrodriguezb@utn.edu.ec		
TELÉFONO FIJO:	062-939-804	**TELÉFONO MÓVIL:**	0991932724

DATOS DE LA OBRA	
TÍTULO:	DESARROLLO DE UN ENVOLTORIO DEL API-REST DE MENDELEY CON GRAPHQL
AUTOR (ES):	RODRÍGUEZ BARAHONA KEVIN DAVID
FECHA: DD/MM/AAAA	05/03/2020
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA:	[X] PREGRADO [ ] POSTGRADO
TÍTULO POR EL QUE OPTA:	INGENIERO EN SISTEMAS COMPUTACIONALES
DIRECTOR:	MSc. ANTONIO QUIÑA

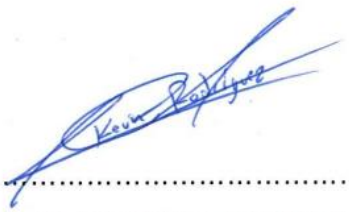
2. CONSTANCIAS

2. CONSTANCIAS

El autor(es) manifiesta(n) que la obra objeto de la presente autorización es original y se 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 tanto, que asume(n) la responsabilidad sobre el contenido de la misma y saldrá(n) en defensa de la Universidad en caso de reclamación por pate de terceros.

Ibarra, a los cinco días del mes de marzo de 2020

EL AUTOR:



.....

KEVIN DAVID RODRÍGUEZ BARAHONA

1004012728

CERTIFICACIÓN DIRECTOR

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN DIRECTOR

Por medio de la presente, yo MSc. Antonio Quiña, certifico que el Sr. **KEVIN DAVID RODRÍGUEZ BARAHONA**, portador de la cédula de identidad Nro. 1004012728, ha trabajado en el desarrollo del proyecto de grado **"DESARROLLO DE UN ENVOLTORIO DEL API-REST DE MENDELEY CON GRAPHQL."**, previo a la obtención del título de Ingeniero en Sistemas Computacionales, realizando con interés profesional y responsabilidad, que verifico en honor a la verdad.

Atentamente,

MSc. Antonio Quiña
DOCENTE FICA UTN

.....

MSc Antonio Quiña

DIRECTOR DEL TRABAJO DE GRADO

AUTORÍA

AUTORÍA

Yo, KEVIN DAVID RODRÍGUEZ BARAHONA, portador de la cédula de ciudadanía número 1004012728, declaro bajo juramento que el trabajo aquí descrito es de mi autoría, **DESARROLLO DE UN ENVOLTORIO DEL API-REST DE MENDELEY CON GRPHQL**, que no ha sido previamente presentada para ningún grado, ni calificación profesional, y que se ha respetado las diferentes fuentes y referencias.

KEVIN DAVID RODRÍGUEZ BARAHONA

C.I. 1004012728

DEDICATORIA

Dedico el presente proyecto de titulación a mi madre Gladys Barahona, a mi padre Luis Rodríguez, a mis hermanas Katy, Wendy y Micaela, a mis sobrinos Briana, Danae, Dayelin y Zaid y a toda mi familia quienes con su cariño y apoyo fueron mi fuente de inspiración para que culminara con éxito este trabajo de grado.

“Si algún día sientes que ya no puedes más, tomate el tiempo de pensar en tu fuente de inspiración que motivo a iniciar el camino hacia una meta, ello hará que vuelvas a sentir las ganas que sentiste al iniciarlo todo”.

Kevin Rodríguez

AGRADECIMIENTO

Agradezco primeramente a Dios, por haberme permitido culminar esta etapa de mi vida con éxito y por haberme hecho fuerte a pesar de haber tenido momentos muy difíciles.

A mis queridos padres Gladys y Luis que me educaron y me enseñaron a vivir la vida siempre con humildad y respeto y quienes, con su apoyo incondicional y motivación inspiraron la culminación de este proyecto. También agradezco a mis hermanas y sobrinos a los cuales los quiero mucho.

A mis apreciados tíos Marcelo, Rosa, Fanny y Sonia que siempre estuvieron pendientes de mi bienestar y quienes me dieron su total apoyo y cariño en todo el proceso de titulación, a ellos un inmenso Gracias.

A mi grupo de amigos Adriana, Bryan, Cristofer, Jorge, Josue, Luis Carlos, Maritza, Nelson, Richard con quienes he compartido, momentos duros, momentos difíciles, pero también momentos felices, ellos saben el difícil camino que hemos atravesado, pero con dedicación y esfuerzo cualquier objetivo se lo puede cumplir.

A mi director de tesis y amigo MSc. Antonio Quiña que con su guía y conocimiento hizo posible la realización y culminación de este proyecto.

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

La creación de sistemas ha experimentado cambios en su proceso de construcción, la separación del servidor y cliente ha impulsado la innovación tecnológica por medio de la creación de tecnologías más útiles y eficientes. La integración entre el cliente y servidor normalmente es realizada por una API, comúnmente usando tecnología REST la cual es muy aceptada en la comunidad de desarrollo de software, aunque es muy aceptada también presenta inconvenientes como la complejidad que presenta su consumo lo que hace necesario la creación y utilización de alternativas, una de estas alternativas es GraphQL que es un lenguaje de consultas para las APIs y que pretende ser más eficiente y mejorar circunstancialmente la experiencia de desarrollo del lado del cliente.

El presente trabajo describe la creación de una API que envuelve la tecnología API-REST y la convierte en tecnología GraphQL (envoltorio), se tomó a la API-REST de Mendeley como caso de desarrollo. Para la creación del envoltorio se definió una base teórica que permitió conocer conceptos que involucra crear el envoltorio. El desarrollo del producto se lo hizo siguiendo la metodología ágil Scrum que comprende ciclos interactivos llamados Sprints. Para validar la usabilidad del software, se usó un taller práctico y una encuesta que fueron validadas estadísticamente y que permitieron recolectar datos los cuales fueron usados para evaluar la calidad en uso con un marco de trabajo basado en el estándar ISO/IEC 25000, por lo cual se obtuvo resultados satisfactorios.

***Palabras clave*:** arquitectura de software, GraphQL, REST, envoltorio, APIs, gestor de referencias bibliográficas, Mendeley.

ABSTRACT

The creation of systems has undergone changes in its construction process, the separation of the server and client has driven technological innovation through the creation of more useful and efficient technologies. The integration between the client and server is usually done by an API, commonly using REST technology which is widely accepted in the software development community, although it is very accepted it also presents problems such as the complexity of its consumption, which makes it necessary to creation and use of alternatives, one of these alternatives is GraphQL which is a query language for APIs and that aims to be more efficient and circumstantially improve the client-side development experience.

This work describes the creation of an API that involves API-REST technology and converts it into GraphQL technology (wrapper), Mendeley API-REST was taken as a development case. For the creation of the wrapper, a theoretical basis was defined that allowed to know concepts that involve creating the wrapper. Product development was done following the agile Scrum methodology that includes interactive cycles called Sprints. To validate the usability of the software, a practical workshop and a survey were used that were statistically validated and that allowed data to be collected which were used to evaluate the quality in use with a framework based on the ISO/IEC 25000 standard, so which obtained satisfactory results.

***Keywords*:** software architecture, GraphQL, REST, wrapper, APIs, bibliography reference manager, Mendeley.

INTRODUCCIÓN

Antecedentes

En el año 2012, Facebook implementó GraphQL como alternativa al bajo rendimiento en la presentación de las vistas de la aplicación móvil de Facebook. Las excesivas conexiones que hacía este consumo con el servidor para cada vista de las aplicaciones móviles se traducía a un ineficiente rendimiento tanto a nivel de servidor como para el lado del cliente ya que Facebook manejaba grandes cantidades de datos. Ante este problema, se reestructuró este proceso mediante la obtención de la información que se necesitaba por medio de una única petición; así apareció GraphQL (Lee, Code, 2015). GraphQL permitió que la obtención de datos de las aplicaciones móviles de Facebook se enfocará en las aplicaciones cliente y con un formato sencillo y óptimo para los diseñadores y desarrolladores (Lee, 2016). En el año 2015 Facebook lanza la versión open source de GraphQL como alternativa a la optimización de consulta de datos, pretendiendo adaptarse a las necesidades actuales (Wittern, Cha, & Laredo, 2018).

El gestor bibliográfico Mendeley fue creado con el propósito de administrar las búsquedas bibliográficas de la comunidad de investigadores. Mediante un servicio REST, este gestor bibliográfico permite usar sus funciones en aplicaciones de terceros; para consumir estas funciones se debía tener conocimientos de la arquitectura REST, y, además el API presentaba el inconveniente de no tener flexibilidad para los desarrolladores ya que, al tener muchos endpoints, se dificultaba el consumo de dichas funciones.

Situación Actual

En la actualidad, las API-REST han mostrado sus limitaciones debido a su modelo de funcionamiento, arquitectura y al tamaño de aplicaciones en el lado del cliente (Ghebremicael, 2017), lo cual genera cierta incertidumbre a la hora de elegir un método de consumo de información.

El gestor de referencias bibliográficas Mendeley, posee un API con la arquitectura REST el cual puede ser consumido por la comunidad de desarrolladores interesados en este servicio. sin embargo, este gestor no cuenta con un API GraphQL que gestione de una manera más eficiente y personalizada el servicio que ofrece a la comunidad.

La comunidad de desarrolladores ha visto como alternativa para solucionar algunos problemas de las RESTful APIs a las APIs GraphQL por los tantos beneficios que presenta el modelo de funcionamiento de esta tecnología (Ghebremicael, 2017). En el caso del gestor de referencias Mendeley, al tener la arquitectura REST; presenta los inconvenientes que tiene este

tipo de arquitectura en términos de optimización y flexibilidad; además el consumo se vuelve algo tedioso del lado del cliente ya que debe realizar varias peticiones si desea obtener información personalizada (información de varios endpoints).

Prospectiva

La optimización del funcionamiento del software siempre debe ser un parámetro importante a la hora de crear productos. Mediante el análisis de algoritmos se puede cuantificar medidas físicas, uso de memoria y tiempo de ejecución y ver que para la resolución de un problema existen varias soluciones, unas más óptimas que otras (Salcedo Parra & Herrera, 2013).

Con el presente trabajo, se pretende desarrollar un envoltorio del API-REST de Mendeley con GraphQL, lo cual implica la creación de un API GraphQL (backend) y de un cliente (frontend) por medio de un marco de trabajo de calidad en uso basado en las ISO 25000, que permita demostrar la eficiencia de esta nueva tecnología y la flexibilidad de consumo por parte de los clientes al implementar este tipo de tecnología.

Planteamiento del problema

Las limitaciones en la eficiencia y la dificultad de consumo del API-REST de Mendeley, hacen que usar este servicio sea un poco molesto del lado del cliente por la complejidad de adaptación de dicho servicio con el producto del cliente (aplicaciones web, aplicaciones de escritorio, aplicaciones móviles etc.). La falta de una herramienta (Api GraphQL) más eficiente y más intuitiva para el cliente hacen que el proceso de consumo no sea muy óptimo ni fácil de implementar.

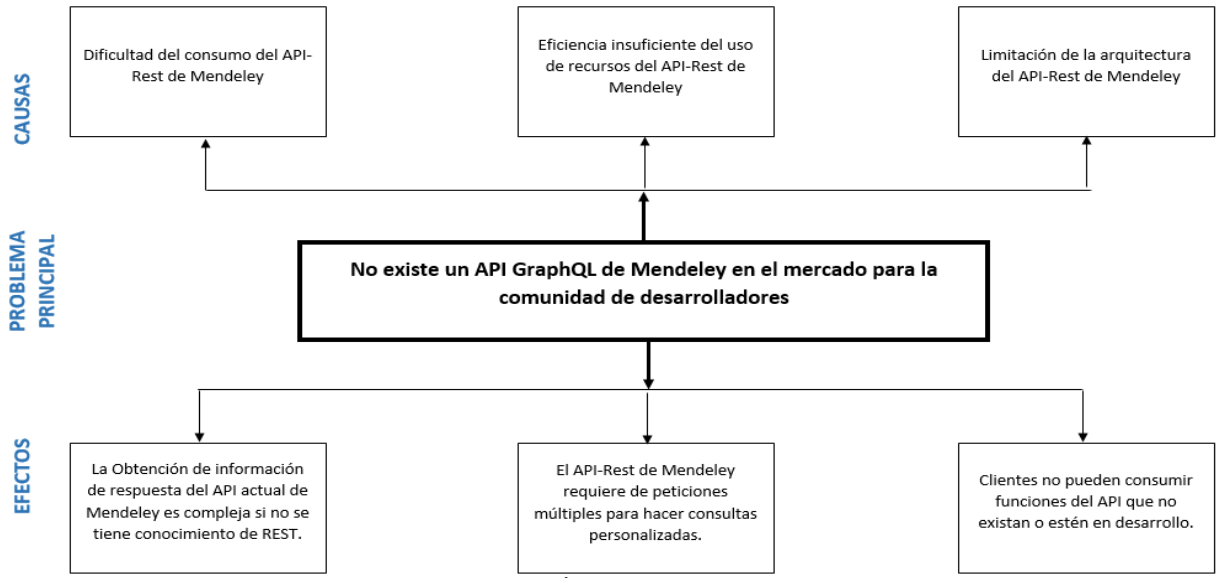


Fig. 1. Árbol de problemas

Objetivos

Objetivo General

Desarrollar un envoltorio del API-REST del gestor bibliográfico Mendeley utilizando el lenguaje de consultas GraphQL y validado con un marco de trabajo de calidad en uso basado en el estándar ISO/IEC 25000

Objetivos Específicos

- Elaborar un marco teórico que involucre una investigación acerca de la creación de envoltorios GraphQL.
- Implementar un envoltorio GraphQL de Mendeley que permita el consumo de las funcionalidades del gestor bibliográfico Mendeley, mediante la aplicación de una prueba de concepto realizada en REACT con Apollo Client.
- Evaluar el API-GraphQL de Mendeley mediante el marco de trabajo de calidad en uso basado en el estándar ISO/IEC 25000.
- Validar los resultados del trabajo realizado

Alcance

El presente trabajo pretende desarrollar un envoltorio del API-REST de Mendeley con la herramienta GraphQL por medio de un marco de trabajo de calidad de uso basado en las ISO/IEC 25000, lo cual permitirá a desarrolladores específicos usar los servicios de Mendeley, pero por medio del API resultante del envolvimiento mencionado, esto permitirá que el consumo de los servicios de Mendeley sea más flexible.

El envoltorio va enfocado a diferentes comunidades de desarrollo como por ejemplo las comunidades de APIs como APIs Gurus, APIs addicts, entre otras; a la comunidad académica y de investigación como por ejemplo el grupo ISA - Ing. De software de la Universidad de Sevilla, la Universidad Técnica del Norte (personal de investigación); a los desarrolladores de soluciones enfocados a la gestión bibliográfica.

Servicios de Mendeley que se implementarán al proyecto.

- Conexión con cuenta Mendeley
- Listado de carpetas
- Listado de documentos y sus detalles
- Archivos adjuntos
- Metadatos de los registros
- Consulta de grupos de colaboración

- Opción de búsqueda o autocompletado (opcional).

El proceso de desarrollo de este trabajo estará dividido en etapas complementarias, las cuales permitirán el cumplimiento de los objetivos establecidos. Una de estas etapas es la construcción de un marco teórico, por medio de una investigación la cual permita conocer conceptos fundamentales acerca de la creación de envoltorios a partir de orígenes de datos múltiples, específicamente de origen API-REST; además conocer los beneficios del lenguaje de consultas GraphQL y como se ha tomado a esta herramienta como una alternativa a las APIs de origen REST. En esta investigación también se mencionará los servicios que brinda Mendeley para los desarrolladores y como estos pueden ser consumidos por clientes para el desarrollo de aplicaciones multiplataforma.

En la etapa de creación del producto, se generará un prototipo GraphQL con las funcionalidades más importantes del servicio del API-REST de Mendeley. Para la comprobación se hará una prueba de concepto para verificar que el producto es apto para ser utilizado como herramienta tecnológica operativa; esta prueba se la realizará con React y Apollo Client del envoltorio.

Para la validación de resultados, se determinará un modelo o método estadístico que se ajuste a las características del trabajo realizado.

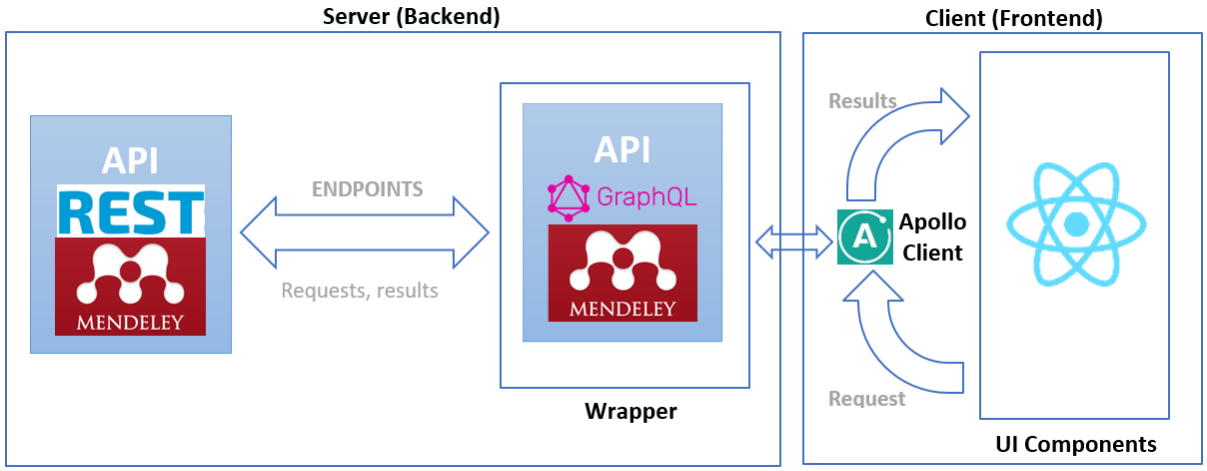


Fig. 2. Arquitectura de implementación
Fuente: Elaboración propia

Justificación

La contribución de herramientas tecnológicas accesibles a la sociedad se ha vuelto hoy en día una práctica muy habitual, una de las ideas que más se acopla a esta definición es las herramientas de software libre, las cuales permiten a cierta comunidad con conocimientos

específicos, explotar una gran cantidad y variedad de productos o servicios informáticos. El gestor bibliográfico Mendeley ofrece a la comunidad de desarrolladores un servicio tecnológico por medio del cual se puede acceder a las funcionalidades de gestión de bibliografía por medio de un API-REST.

Con la implementación de una manera más flexible (API GraphQL) para obtener los servicios de Mendeley, el presente trabajo contribuye a la generación de oportunidades de trabajo ya que profesionales pueden usar la herramienta desarrollada y crear sus propias soluciones sin necesidad de tener experticia en áreas específicas.

El enfoque del presente trabajo hacia los objetivos de desarrollo sostenible (ODS) se contempla en tres de ellos.

Educación de calidad (Objetivo 4): Mediante la investigación realizada en este trabajo, se pretende contribuir al acceso a formación técnica y a obtener conocimiento acerca del contenido de este trabajo (Programa de las Naciones Unidas para el Desarrollo, 2018a).

Trabajo decente y crecimiento económico (Objetivo 8): Estimular el crecimiento económico mediante la generación de trabajo por medio de la innovación tecnológica (Ryder, 2015).

Industria, innovación e infraestructura (Objetivo 9): Mediante soluciones tecnológicas que ayuden a generar un crecimiento económico y laboral (Programa de las Naciones Unidas para el Desarrollo, 2018b).

Contexto

Los trabajos relaciones con el desarrollo o estudio de APIs GraphQL hoy en día son un poco escasos por el motivo de que esta tecnología es nueva, pero al mismo tiempo está teniendo una gran aceptación por la comunidad de desarrolladores los cuales han visto de esta tecnología las ventajas que ofrece en comparación con otras tecnologías tales como REST (Vázquez-Ingelmo, Cruz-Benito, & García-Peñalvo, 2017).

En la Universidad Técnica del Norte, no existen trabajos relacionados respecto al desarrollo de tecnologías con GraphQL debido a la reciente liberación de esta tecnología como herramienta de desarrollo para aplicaciones informáticas académicas o de investigación.

El proyecto de investigación titulado *Experiences on Migrating RESTful Web Services to GraphQL,* hace referencia a un caso de estudio de la migración de un API-REST a GraphQL, en el cual se realizó una comparativa de rendimiento entre la API-REST y la GraphQL; los resultados demostraron el tiempo de respuesta de cada una de estas APIs y se evidenció los beneficios del

enfoque de obtención de datos con GraphQL (Vogel, Weber, & Zirpins, 2018). Lo resultados del proyecto antes mencionado muestran la ventaja de migrar de APIs REST a GraphQL por lo que la envoltura GraphQL que se desarrollará a partir del API REST de Mendeley obtendrá algunas ventajas de las que se menciona en el proyecto.

Un trabajo relacionado a la generación de envoltorios GraphQL de un API-REST es el artículo titulado *Generating GraphQL-wrappers for REST(-like) APIs*; en esta investigación se realizó el estudio de que tan factible es envolver las API-REST, para lo cual se tomó como prueba a 959 OAS (OpenApi Specification, describe a una API-REST) y por medio de una herramienta de generación de envoltorios se obtuvo los resultados que dicen que el 89,5% de los casos fue exitosa la creación del envoltorio, sin embargo hubo ciertas limitaciones las cuales se mencionan en dicho trabajo (Wittern et al., 2018). El proyecto de generación de envoltorios permitirá tener información acerca de las limitaciones que podría llegar a tener el desarrollo del envoltorio GraphQL de Mendeley, sin embargo, se evidencia un porcentaje alto de casos exitosos en la creación de dichos envoltorios.

CAPÍTULO 1

Marco Teórico

A continuación, se muestra de una manera estructurada el contenido del presente capítulo el cual conceptualiza los puntos esenciales que involucra la creación de envoltorios GraphQL, ver TABLA 1**.**

TABLA 1 Esquematización marco teórico

Tema principal	Tema secundario	Sumario
Arquitecturas de software	Arquitectura monolítica	
	Arquitecturas Orientadas a Servicios	
	Arquitectura microservicios	
	Estilo arquitectónico REST	Recurso Representación Estado URI Operaciones
	Interfaz de programación de aplicaciones	API SOAP API REST API GRAPHQL
Lenguaje de consultas GraphQL	Consultas y mutaciones	Operaciones Campos Argumentos Variables
	Esquema y sistema de tipos	Esquema Campos y tipos de objetos Argumentos Tipos de consulta y mutación Tipos de entrada
	Validación	Fragmento repetitivo Campos inexistentes Sin campos para devolver Campo escalar
	Ejecución	Resolutores Respuesta
Envoltorios (Wrappers)	Envoltorios de base de datos	
	Envoltorios a nivel de lenguaje de programación	
	Envoltorios basados en API REST	
Gestores bibliográficos	Gestor bibliográfico Mendeley	Servicio para desarrolladores (API REST Mendeley)
Herramientas tecnológicas	Backend	Node.js Npm (Gestor de librerías) Express

	Frontend	React.js Apollo client
Metodologías de desarrollo de software	Marco de trabajo Scrum	Roles Artefactos Eventos
ISO/IEC 25000	ISO/IEC 25010	Modelo de calidad en uso
	ISO/IEC 25022	Medición de calidad en uso
	ISO/IEC 25040	Evaluación de calidad en uso

1.1. Arquitecturas de Software

La arquitectura de software es considerada como un diseño de alto nivel, que muestra la estructura de un sistema mediante la organización y relación de los componentes que lo conforman (Endrei et al., 2004), sin embargo no existe una definición concreta y aceptada, es por esto que algunos autores han propuesto algunas definiciones, por ejemplo GARLAN & SHAW (1993) los cuales mencionan que la arquitectura de software es “a collection of computational components—or simply components—together with a description of the interactions between these components—the connectors”. [una colección de componentes computacionales, o simplemente componentes, junto con una descripción de las interacciones entre estos componentes]. (p.4).

Para tener una visión general del diseño de software, es necesario construir los sistemas basados en alguna arquitectura, esto dependiendo de las necesidades que abarque la creación de los mismos. Existen algunos tipos de arquitecturas, entre las que están vigentes se tiene a la arquitectura monolítica, la arquitectura orientada a servicios, la arquitectura microservicios.

1.1.1. Arquitectura Monolítica

Los componentes de un sistema monolítico están empaquetados en una sola base contenedora: la interfaz de usuario, la lógica de negocio y la capa de acceso a datos, ver Fig. 3, es decir los componentes no son independientes, estos funcionan complementariamente entre ellos. Un error que presenta un servicio se traduce en una caída total de la aplicación; normalmente estas aplicaciones se empaquetan en archivos jar o war (Villamizar et al., 2015). A continuación se muestra un ejemplo de una arquitectura monolítica de un sistema académico, ver Fig. 3.

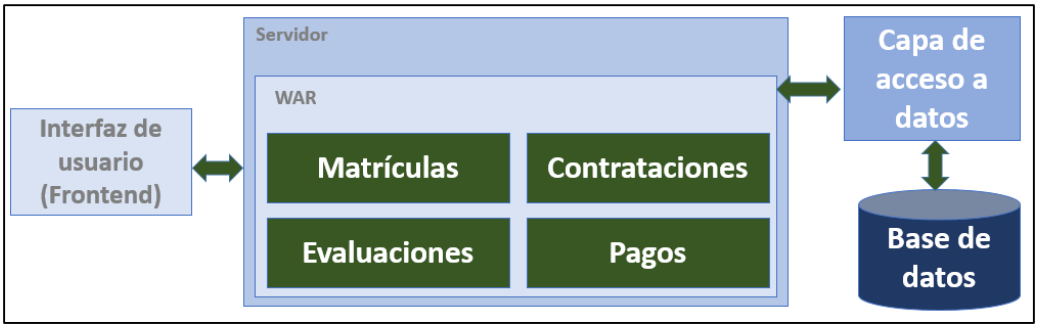


Fig. 3. Estructura de la arquitectura monolítica
Fuente: Elaboración propia

1.1.2. Arquitectura Orientada a Servicios (SOA)

La arquitectura orientada a servicios se define como un modelo de organización y uso de recursos distribuidos los cuales son independientes y pueden estar controlados por diferentes dominios. El termino servicio hace referencia a un recurso; a nivel organizacional es aquel que posee un nombre y una serie de tareas (MacLennan & Van Belle, 2014).

La separación de la lógica de integración del negocio de la implementación de servicios es uno de los objetivos de SOA, la estandarización de dicha implementación hace que los componentes de una aplicación puedan ser usados por cualquier usuario que tenga los privilegios necesarios. Estas estandarizaciones pueden ser el bus de servicios Business Process Execution Language (BPEL) o Web Services Descriptive Languaje (WSDL) para describir servicios web (IBM Knowledge Center, 2017).

1.1.3. Arquitectura Microservicios

Las aplicaciones con este tipo de arquitectura se caracterizan por estar compuestas por un conjunto de servicios pequeños e independientes, el conjunto de servicios que componen una aplicación se comunican por medio de mecanismos ligeros como la tecnología REST. Los despliegues de estos servicios se los hace de manera independiente, sin afectar a los demás servicios y sin limitar la disponibilidad de la aplicación en general. La Fig. 4 muestra un ejemplo de un sistema acadèmico en donde cada servicio al ser independiente, tiene su propia base de datos y su propia base de codificación por lo que cada uno de estos servicios puede estar desarrollado en un lenguaje de programación diferente (Dmitry Namiot, 2014).

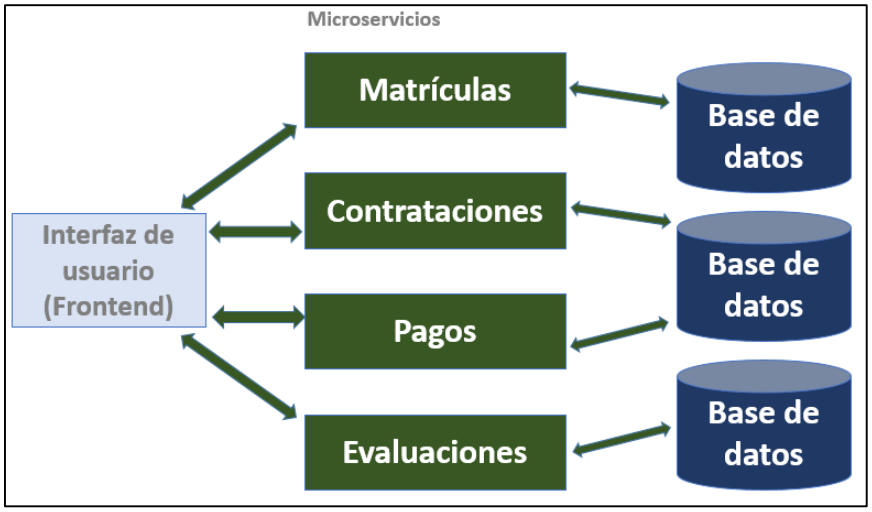


Fig. 4. Estructura de la arquitectura microservicios
Fuente: Elaboración propia

Según (Newman, 2017), los microservicios presentan una serie de beneficios entre los cuales se pueden mencionar:

a) Tecnología heterogénea ya que permite incluir diferentes tipos de tecnologías.
b) Cada servicio es independiente, por lo que el fallo de algún servicio no afecta al rendimiento de la aplicación en general, esto garantiza la disponibilidad de la aplicación.
c) La escalabilidad hace que los microservicios crezcan independientes entre sí.
d) El despliegue ante cambios se hace en cada servicio, evitando así la interrupción del funcionamiento de la aplicación.

1.1.4. Estilo arquitectónico REST

Representational State Transfer (REST) es una serie de restricciones que, al ser plasmadas al diseño de un software, crea un estilo arquitectónico. REST utiliza la arquitectura cliente-servidor[1] que se caracteriza por él envió de peticiones hacia el servidor por parte del cliente y respuestas por parte del servidor. Además cada solicitud es independiente ya que tiene la información necesaria para ser realizada, así el servicio no guarda un estado; los recursos son obtenidos por medio de una dirección única y válida (Sandoval, 2009).

Recurso, representación y estado son conceptos importantes que definen el estilo arquitectónico REST.

[1] **Arquitectura cliente-servidor**: Arquitectura en donde interviene dos partes, el servidor que provee recursos y el cliente el cual los solicita. Los clientes hacen peticiones a un programa (servidor) y este brinda una respuesta en relación a dicha petición.

- **Recurso**

  Un recurso es algo que puede ser almacenado y que hace referencia a un objeto físico o abstracto (Xinyang, Jianjing, & Ying, 2009). Los recursos en REST son identificados mediante un Uniform Resource Identifier (URI) también conocidos como endpoints, la manipulación de estos se la realiza a través de métodos HTTP, por lo que todo sistema que soporte este protocolo puede usar REST (Kobusińska & Hsu, 2018).

- **Representación**

  Una representación en REST es la información o contenido de un recurso en forma de un conjunto de bytes y metadatos que los describen; algunos mensajes de respuesta contienen los metadatos del recurso y metadatos de la representación (Fielding & Taylor, 2002). Un recurso puede tomar diferentes representaciones, esto depende del cliente y las características del recurso que dicho cliente solicita, es por esto que una representación puede adoptar diferentes formas; la más común es una secuencia Extensible Markup Language (XML), aunque también puede ser texto plan, una imagen, un archivo JavaScript Object Notation (JSON), entre otros (Sandoval, 2009).

- **Estado**

  El estado en REST es la información acerca de un recurso o de la aplicación; el estado de un recurso se mantiene en el lado del servidor, mientras que el estado de la aplicación se encuentra en el lado del cliente (Xinyang et al., 2009). El estado de un recurso es persistente ya que se mantiene en el servidor, este puede ser accedido por cualquier cliente con su respectivo permiso ya que se encuentra en el servidor, en cambio el estado de la aplicación solo le pertenece al cliente activo por ejemplo en una sesión.

  Un recurso en REST es identificado por una URI y es manipulado por operaciones HTTP; la identificación del recurso permite por medio de una petición hacia el servidor realizar una operación hacia el recurso.

- **Uniform Resource Identifier (URI)**

  En REST, una URI es un identificador de un recurso por medio del cual el cliente puede solicitar al servidor una representación, es decir la URI direcciona hacia el recurso que está en el servidor. Una URI no es exactamente un hipervínculo, pero al usar la Web como tecnología de soporte se lo asume de esta manera. Normalmente en un servidor web no REST, se tiene un recurso identificado por algún tipo de URI, si se cambia el

nombre o dirección de este, su URI también cambia; en REST la URI de un recurso no está destinado a cambiarse ya que en una arquitectura que usa los principios de REST se dificultaría la localización de los recursos (Sandoval, 2009).

- **Operaciones**

  El intercambio de representaciones entre el cliente y el servidor, se hace a través de las operaciones definidas en el protocolo HTTP y es la manera en que el cliente puede manipular los recursos del servidor de datos. En la TABLA 2 se observa la equivalencia entre las operaciones HTTP y las acciones Create, Read, Update and Delete (CRUD).

TABLA 2 Equivalencia de acciones de datos y operaciones HTTP

Data action	HTTP protocol equivalent
CREATE	POST
RETRIEVE/READ	GET
UPDATE	PUT
DELETE	DELETE

Fuente: (Sandoval, 2009)

Las operaciones que utiliza REST para el intercambio de representaciones entre el cliente y el servidor se las realiza a través de peticiones HTTP que según (Sandoval, 2009) se definen:

a) GET: Esta operación es equivalente a Read o Retrieve que permite obtener la información del estado actual de un recurso realizando una petición HTTP del tipo GET y obteniendo una representación.

b) POST: El método POST permite la creación de nuevos recursos por medio de una petición HTTP, esto adjuntando la información del recurso a ser creado en dicha petición.

c) PUT: La actualización de un recurso se la realiza con esta operación, pero para realizar esta operación primero se obtiene la representación del recurso en el cliente, al obtener esta información se cambia por nuevos valores y se realiza la petición de tipo PUT adjuntando la representación con los nuevos valores.

d) DELETE: Para eliminar algún recurso se usa una petición HTTP de tipo DELETE.

1.1.5. Interfaz de Programación de Aplicaciones (API)

Una API es definida como la manera de comunicación de dos aplicaciones por medio de un lenguaje que las dos partes entienden; en dicha comunicación, una aplicación accede a datos o servicios disponibles por medio de un contrato en donde se define la estructura de la API, privilegios y otras consideraciones (Jacobson, Brail, & Woods, 2011).

- **API SOAP**

  Simple Object Access Protocol (SOAP) es un protocolo de mensajería para la comunicación de aplicaciones que se especifica sobre XML por lo cual este protocolo se basa en estándares de esta especificación tales como XML Schema, XML Namespaces. Los mensajes SOAP son Documentos XML que tienen como elemento raíz un sobre (<envelope>) del cual derivan dos partes que son el encabezado en donde se especifica la ruta de envió del mensaje y el cuerpo que contiene el mensaje en sí. SOAP incluye un documento WSDL con información de las prestaciones del servicio y ubicación en caso de web services mediante un contrato (Snell, Tidwell, & Kulchenko, 2002).

- **API REST**

  Según (Ribas, 2018) detalla una API con el estilo arquitectónico REST tomando en cuenta una serie de características clave que lo definen:

  a) Protocolo cliente servidor sin estado.
  b) Un recurso es un objeto representacional como puede ser un producto, un estudiante, etc.
  c) Para la identificación de recursos se usa una URI.
  d) Usa cuatro operaciones principales para manejar los recursos; GET para obtener un recurso, POST para crear un recurso, PUT para editar un recurso y DELETE para borrar un recurso.
  e) Una API REST es adaptable a cualquier lenguaje de programación ya que usa una manera universal de comunicación como respuesta de las solicitudes, normalmente en formato JSON.

- **API GRAPHQL**

  La implementación de un API con GraphQL se la hace por medio de un lenguaje de consultas creado por Facebook ante la necesidad de optimizar el uso de recursos y flexibilizar el consumo del lado cliente. En la siguiente sección se detalla sobre esta especificación.

1.2. Lenguaje de consultas GraphQL

1.2.1. Introducción

GraphQL es un lenguaje de consultas desarrollado para las APIs que ejecuta consultas del lado del servidor, basándose en la definición de un sistema de tipos para los datos predefinido por el desarrollador. GraphQL hace flexible el consumo de la API lo cual da el control de la

obtención de datos al cliente; por medio de consultas personalizadas, el cliente obtiene solo lo que necesita y por medio de una única petición. GraphQL elimina el versionamiento utilizado por REST, ya que solo se necesita cambiar la estructura de la solicitud para obtener una nueva estructura de datos (Linux-Fundation, 2019).

Esta tecnología creada por Facebook, nació en el año 2012 ante la necesidad de optimizar las aplicaciones móviles de dicha organización ya que, al ser cada vez más complejas, su rendimiento se volvía limitado e ineficiente; es por eso que se inició el proyecto GraphQL lo cual enfatizó la obtención de datos desde el punto de vista del cliente (Lee, 2015). En 2015 el proyecto fue lanzado públicamente por medio de un borrador de especificación del lenguaje (Bryant, 2017).

1.2.2. Estructura de consultas y mutaciones

- **Operaciones**

  GraphQL tiene la capacidad de realizar las cuatro funciones fundamentales para gestionar información almacenada conocido como CRUD (crear, consultar, actualizar y eliminar). En relación con REST; este posee estas operaciones en forma de verbos (GET, POST, PUT, DELETE) las cuales son especificadas en la consulta HTTP, en cambio GraphQL especifica tres tipos de operaciones las cuales son independientes de la solicitud HTTP; estas operaciones son: consultas, mutaciones y suscripciones (Vázquez-Ingelmo et al., 2017).

  a) Consultas: Obtención de datos (solo lectura).
  b) Mutaciones: Manipulación de datos; realiza una modificación de los datos y obtiene como respuesta una consulta. Creación, actualización, eliminación.
  c) Suscripciones: Solicitudes en respuesta a eventos fuente (tiempo real).

- **Campos**

  Los campos en GraphQL son representaciones de varios tipos de datos los cuales pueden ser escalares (Int, Float, String, Boolean e ID) u objetos (recuperación de objetos de una relación), estos pueden ser obtenidos mediante una función de resolución (Ghebremicael, 2017). Estos campos son solicitados mediante una estructura de petición en la cual se especifican campos contenidos dentro de un objeto, por ejemplo la Fig. 5 muestra el objeto *groups* y sus campos escalares *name* y *description*; la respuesta será un JSON que tendrá la misma estructura de la solicitud. (Linux-Fundation, 2019).

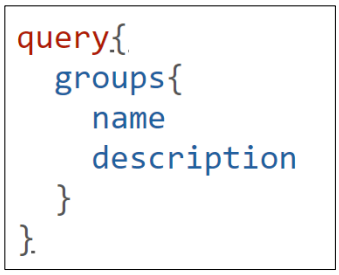


Fig. 5. Campos de una consulta GraphQL
Fuente: Elaboración propia

- **Argumentos**

  En una consulta GraphQL, los argumentos son información que se relaciona con un campo de consulta para realizar una relación tipo clave-valor. También, los argumentos permiten filtrar ciertos datos de una consulta (Porcello & Banks, 2018).

  La ventaja de usar argumentos es que facilita el filtro de datos de un objeto, y ya que en la estructura de una operación posee campos objetos (objeto anidado) dentro de una solicitud, se puede filtrar información de estos campos, por ejemplo se muestra el argumento *id* que filtra a un grupo específico, ver Fig. 6. Los tipos de datos escalares también pueden contener argumentos lo que lo convierte en una transformación de datos en el servidor (Linux-Fundation, 2019).

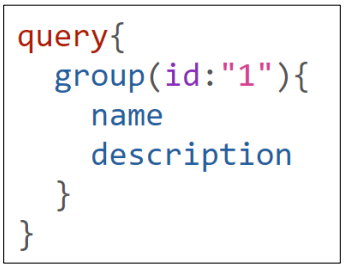


Fig. 6. Argumentos en una consulta GraphQL
Fuente: Elaboración propia

- **Variables**

  Son un dato de entrada de una operación de consulta o mutación los cuales reemplazan valores estáticos que se asignan a las operaciones; estos valores se hacen dinámicos. Las variables son valores dinámicos que pueden ser asignados a los argumentos; la estructura de una variable dentro de una consulta es similar a $var, el símbolo $ indica que var es una variable, y var es el nombre de la variable, por ejemplo en la Fig. 7 se observa la variable *topic* que toma como valor *React* y es usada como argumento de la consulta; y para enviar el valor de la variables se usa el formato JSON en donde los nombres de las variables son una clave de JSON (Porcello & Banks, 2018).

```
query TopicCourses($topic: String) {
  courses(topic: $topic) {
    title
    topic
  }
}
```

QUERY VARIABLES

```
{
  "topic": "React"
}
```

Fig. 7. Variables en una consulta GraphQL
Fuente: Elaboración propia

1.2.3. Esquema y sistema de tipos

- **Esquema**

Así como existen las definiciones de esquemas de base de datos, GraphQL estructura los datos por medio del esquema GraphQL (Kimokoti, 2018). El esquema del servidor GraphQL define en su estructura los tipos de datos y las relaciones que existen en estos, en este esquema también se definen las operaciones de consulta y mutaciones de los datos (Wittern et al., 2018).

En resumen, un esquema es considerado una colección de las definiciones de tipos establecidos para la aplicación (Porcello & Banks, 2018). A continuación se muestra los tipos *query* y *mutation* con sus respectivas operaciones, y el tipo *Course* correspondientes al esquema GraphQL, ver Fig. 8.

```
type Query{
  message: String
  course(id: Int!): Course
  courses(topic: String): [Course]
}

type Mutation{
  updateCourseTopic(id: Int!, topic: String!): Course
}

type Course{
  id:Int
  title: String
  author: String
  topic: String
  url: String
}
```

Fig. 8. Ejemplo de un esquema GraphQL
Fuente: Elaboración propia

- **Campos y tipos de objetos**

  Un esquema GraphQL está compuesto por tipos, los cuales son la unidad central del esquema. Un tipo representa a un objeto personalizado y cada uno de estos objetos representa la funcionalidad principal de la aplicación. Un tipo está compuesto por campos que no son más que las características o propiedades del objeto (Porcello & Banks, 2018).

  Los campos de cada tipo de objeto son definidos por un tipo especial de dato denominado tipo escalar que viene incorporado en GraphQL tales como ID, Int, String, Float, Boolean (Linux-Fundation, 2019). En la Fig. 9 se muestra algunos ejemplos de campos como el *id* de tipo Int, *title* de tipo String, *author* de tipo vector Author, todos estos pertenecientes al tipo *Course*.

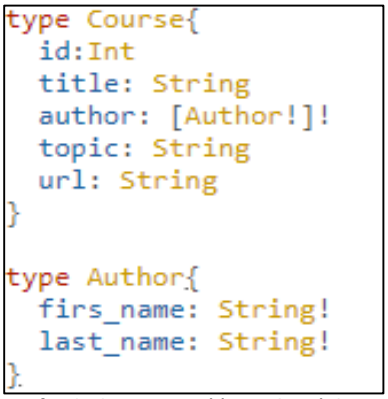


Fig. 9. Campos y objetos GraphQL
Fuente: Elaboración propia

- **Argumentos**

  La definición de argumentos dentro del esquema permite que la información que se obtendrá del origen de datos sea filtrada por medio del valor de dichos argumentos; los argumentos tienen un tipo ya sean tipo escalar o tipo objeto que esté definido en el esquema (Porcello & Banks, 2018). Los tipos de datos escalares que posee GraphQL de manera predeterminada pueden ser: flotante, entero, cadena, booleano, ID, estos campos tienen una limitación del valor que pueden tomar. Cuando un argumento es obligatorio se específica con el símbolo "!", esto quiere decir que se necesita estrictamente el valor para el argumento (Kimokoti, 2018). En la Fig. 10 se observa el argumento first_*name* de tipo String y obligatorio por contar con el símbolo *!* en su estructura.

```
type Course{
   id:Int
   title: String
   author(first_name: String!): [Author!]!
   topic: String
   url: String
 }
```

Fig. 10. Argumentos en campos de Tipos
Fuente: Elaboración propia

- **Tipos consulta y mutación (type query, type mutation)**

  En un esquema GraphQL se puede ver diferentes definiciones de tipos, ya sean tipos de objetos, escalares, entre otros; pero hay dos tipos con una funcionalidad específica e importante; los tipos consulta y mutación que se pueden observar en la Fig. 11, que son un punto de entrada para las operaciones de la API (Linux-Fundation, 2019). En los tipos de consulta y mutación del esquema GraphQL, los puntos de entrada están definidos por los campos, argumentos y tipos de resultados, estos tipos se conectan con los denominados resolutores (Kimokoti, 2018).

```
type Query{
  message: String
  course(id: Int!): Course
  courses(topic: String): [Course]
}

type Mutation{
  updateCourseTopic(id: Int!, topic: String!): Course
}
```

Fig. 11. Tipo Query y Tipo Mutation
Fuente: Elaboración propia

- **Tipos de entrada (Input type)**

  En el esquema GraphQL, normalmente se define a un objeto con la palabra reservada type, estos tipos se usan para obtener información de salida, sin embargo, existe un tipo que permite el ingreso de información lo cual es usado por las operaciones de mutación; por medio de la palabra reservada input, este tipo se convierte en un objeto de entrada de datos como se puede ver en la Fig. 12. Un tipo de objeto y un tipo de entrada se ven igual en su estructura, con la variación de la palabra type e input, también los tipos de entrada carecen de argumentos en sus campos (Linux-Fundation, 2019).

  La ventaja de los tipos de entrada es la organización que se obtiene al escribir un esquema GraphQL; estos tipos son útiles en el mejoramiento del filtrado de datos o la

paginación de los resultados; además mejoran en gran medida la documentación y la accesibilidad de la API haciéndola más fácil de entender e implementarla para el cliente (Porcello & Banks, 2018).

```
input Author{
  first_name: String!
  last_name: String!
}
```

Fig. 12. Tipo Input (tipo de entrada)
Fuente: Elaboración propia

1.2.4. Validación

Para saber si una consulta es válida, existen un conjunto de reglas de validación que garantiza que el contenido de una consulta sea semánticamente correcto. Por medio del sistema de tipos se puede realizar la validación de la consulta, evitando acceder a comprobaciones en tiempo real. Algunos tipos de errores semánticos que se especifican en (Linux-Fundation, 2019).

- **Fragmento repetitivo**

  Si en el contenido de una consulta por medio de un fragmento se llama al mismo fragmento, se estaría creando un ciclo repetitivo infinito dando como resultado una sobrecarga en procesamiento e ineficiencia en general

- **Campos inexistentes**

  Los campos que se establecen en una consultan, deben existir en el tipo de la definición del esquema de la aplicación, si no existen el resultado mostrará un error de campo inválido.

- **Sin campos para devolver**

  Si la estructura de la consulta no tiene como último nivel un tipo escalar, el resultado mostrará un error; este error se produce cuando de un tipo objeto no se selecciona ningún campo de nivel inferior; es decir, no existe un anidamiento completo en la consulta.

- **Campo escalar**

  Los campos escalares son el último nivel en la estructura de un tipo; es decir no existen niveles más debajo de estos, por esta razón la respuesta ante una petición de campos tipos escalares generará un error.

Existen otras especificaciones de validación como la inserción obligatoria de campos requeridos, que obliga a establecer en la consulta todos los campos que se especificaron con el símbolo “!”

1.2.5. Ejecución

En la ejecución de operaciones GraphQL, la validación de campos es fundamental para consultas o mutaciones, sin embargo, para mostrar resultados, un servidor debe ejecutar estas operaciones por medio de resolutores. Cada campo establecido es denominado como una función que devuelve un valor para el siguiente tipo, si el valor producido por estos campos es escalar, se finaliza la ejecución de dicho campo, caso contrario, continua al siguiente nivel (Linux-Fundation, 2019).

- **Resolutores**

  Los resolutores en GraphQL son funciones las cuales son usadas para obtener los datos de los campos definidos en el esquema. Cada campo del esquema definido tiene una función de correspondencia (Ghebremicael, 2017)**.** Las instrucciones de los resolutores, permiten la conversión de las operaciones en datos; cada campo tiene una función de resolución y este conjunto de funciones son llamadas mapa de resolución. Los resolutores tienen por defecto cuatro argumentos, los cuales son: parent, args, context, info (Apollo-GraphQL, 2019). En la Fig. 13 se muestra el resolutor del tipo *course* con los argumentos mencionados como parámetros, en la imagen se usa el argumento *args* para obtener el *id* enviado desde la estructura de la consulta en el esquema.

  a) Parent (obj): Es el objeto anterior; este parámetro es inservible para el campo raíz; así se evidencia el anidamiento de las consultas en GraphQL.
  b) Args:Contiene los parámetros del campo que fueron pasados en la consulta.
  c) Context: Este argumento contiene el estado de cada solicitud, donde se incluye la autenticación, y toda la información de resolución de la consulta; este es un objeto compartido para los resolutores.
  d) Info: La información correspondiente al estado de la consulta es contenida en este argumento, también contiene información del nombre y la ruta jerárquica del campo.

Existen resolutores asíncronos, estos resolutores se usan cuando existe una carga a una fuente de datos ya sea una base de datos, una API, etc. Estas operaciones al ser asíncronas necesitan las Promises, Futures, Tasks o Deferred dependiendo del lenguaje (Linux-Fundation, 2019).

```
Query: {
  course(obj, args, context, info) {
    return getCourse(args.id);
  }
}
```

Fig. 13. Ejemplo de resolutor
Fuente: Elaboración propia

- **Respuesta**

  Al finalizar el proceso de ejecución de una consulta GraphQL, la respuesta es mostrada en un mapa clave-valor en formato JSON; este mapa contiene el alias de cada campo definido en la consulta, y el valor del mismo, todo esto denotando la anidación de las consultas lo cual muestra la estructura definida en la consulta (Linux-Fundation, 2019).

  Específicamente la respuesta puede tomar diferentes estructuras ya que en ocasiones la consulta podría tener errores en su contenido, produciendo un error (Apollo-GraphQL, 2019); estos son las posibles respuestas que produce una consulta GraphQL.

  a) Valor escalar: Respuesta ocasional que en algunos casos no tiene un significado especial
  b) Matriz: Respuesta en forma de lista; solo si está definido en el esquema.
  c) Promesa: Cuando las operaciones son asíncronas se produce una respuesta tipo promesa.
  d) Indefinido o nulo: Esta respuesta se da cuando no se encontró el objeto.

1.3. Envoltorios (Wrappers)

Los envoltorios de orígenes de datos también denominados wrappers, son herramientas que reconocen y obtienen objetos útiles para su uso; estos objetos son datos extraídos de diferentes fuentes, ya sea páginas web, archivos de texto, base de datos, etc. La función principal de un wrapper es la devolución de la información extraída de forma estructurada, ya sea en formato XML u otro tipo de estructuración de datos. La comprobación de la calidad de un wrapper es fundamental ya que en ocasiones se puede tener datos erróneos o algún otro tipo de error en la generación del envoltorio que comprometa el objetivo principal por la que se obtuvo dichos datos (Vargas et al., 2013).

1.3.1. Envoltorios de Base de Datos

Los envoltorios de base de datos son herramientas que transforman los datos de una base de datos a un tipo adecuado de estructura de datos, esta transformación implica la conversión de las consultas de la base de datos a un nuevo formato de consultas que se implementa en el envoltorio y viceversa. Las traducciones a nivel del envoltorio son generalmente expresadas a cierto tipo de modelo como por ejemplo a un lenguaje de marcado como XML, o a un paradigma de programación como la programación orientada a objetos.

Los wrappers de base de datos permiten obtener nuevas o mejorar ciertas capacidades funcionales de dicha base de datos, aumentar la utilidad de los componentes de sistemas de datos, haciéndolos fácil de integrar en sistemas distribuidos modernos (Thiran, Hainaut, & Houben, 2005).

1.3.2. Envoltorios a nivel de lenguaje de programación

Los wrappers envuelven cierta funcionalidad de un recurso por medio de otra interfaz para que sea más fácil usar; a nivel de lenguajes de programación, existen envolturas de tipos primitivos tales como: int, char, String, etc (Oracle, 2019). Aunque también existe envolturas de tipos de datos más complejos como Objetos y otros, se dará énfasis a la envoltura de los tipos escalares mencionados.

Las envolturas de tipos primitivos nacen por la necesidad de usar estos tipos en forma de objetos, un caso de ejemplo es Java el cual usa las clases de envoltura para tipos numéricos; estas clases tales como Integer, Double, Float, Byte y otros son los envoltorios finales, en la Fig. 14 se muestra la jerarquía de este envoltorio.

Existen otros envoltorios tales como Boolean y Character de los tipos primitivos boolean y char respectivamente, el tipo String no es considerado tipo primitivo sino un Objeto el cual tiene establecidos métodos para su uso (Oracle, 2019).

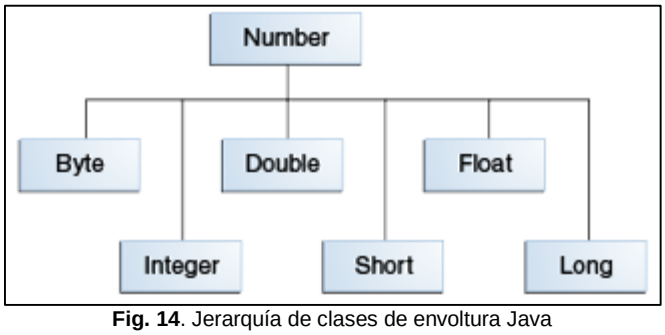


Fig. 14. Jerarquía de clases de envoltura Java
Fuente: (Oracle, 2019)

1.3.3. Envoltorios basados en API-Rest

Las API REST en la actualidad son el enfoque de desarrollo y prestación de servicios web más usado en el campo de desarrollo; sin embargo, para el lado del cliente, se dificulta el entendimiento y uso de estas APIs ya que el nivel de abstracción de una API REST es muy bajo considerando que el desarrollo cliente no está familiarizado con el protocolo HTTP. En el artículo de (Cho & Ryu, 2014) se menciona a las API Javascript como un recurso que facilita al desarrollador cliente el consumo de API REST por medio de comunicaciones bidireccionales mantenimiento dicha comunicación, siendo la API Javascript el envoltorio.

Existen varias maneras de envolver este tipo de APIs, una de estas es la generación de un API GraphQL a partir de un API REST, que es la idea principal de este trabajo. Un envoltorio GraphQL flexibiliza el desarrollo del cliente y además optimiza recursos. Estas envolturas reciben consultas estructuradas de GraphQL, las cuales son transformadas en peticiones a la API envuelta; en la investigación de (Wittern et al., 2018) se concluye que este tipo de envolturas presentan cierto margen de error, aunque es mínimo, es un margen que dificulta un ajuste exacto y completo.

1.4. Gestores bibliográficos

Un gestor bibliográfico es una herramienta que contribuye a la investigación mediante la recolección de referencias bibliográficas que se encuentran alojadas en bases de datos; entre las referencias se encuentran revistas y artículos científicos, libros, tesis, etc. Mediante estos gestores se puede realizar citas de las referencias, generar la bibliografía de la investigación.

Los gestores bibliográficos permiten realizar una revisión bibliográfica lo cual ayuda a ver el panorama de investigación mediante una visión general de fuentes exploradas, lo cual afianza el trabajo científico o académico e indica a los lectores como el proyecto de investigación se ajusta a un campo general de estudio más amplio (Ramdhani, Abdullah, Ramdhani, & Amin, 2014).

Existen varios tipos de estos gestores que cumplen funciones similares entre estos, algunos de estos gestores son: EndNote, RefWorks, Zotero, Mendeley.

1.4.1. Gestor bibliográfico Mendeley

El gestor bibliográfico Mendeley es una herramienta informática que administra referencias bibliográficas la cual ayuda a constituir una estructura de investigación. Mendeley también posee una red social académica que ayuda a que los investigadores trabajen conjuntamente, además de la colaboración entre usuarios y el acceso a un extenso catálogo de referencias de este gestor.

La organización Elseiver es la propietaria de Mendeley que posee recursos informáticos que se detallan en la Fig. 15. Entre las principales características de este gestor, se detallan (Elsevier, 2019).

a) Generación de citas bibliográficas por medio de plug-ins.

b) Creación de cuentas de usuario privadas y sincronización de dichas cuentas en las diferentes plataformas.

c) Colaboración en grupos de investigación.

d) Creación de bibliografía

e) Extracción de los metadatos de archivos PDF.

f) Inserción manual y digital de documentos de investigación en el catálogo privado.

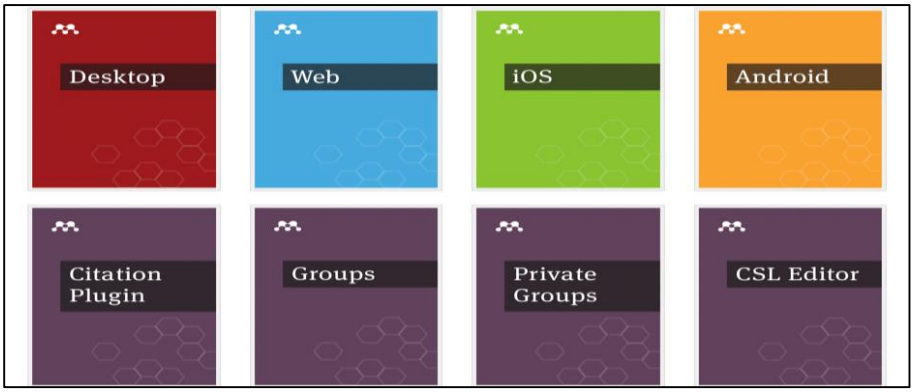


Fig. 15. Recursos informáticos de Mendeley
Fuente: (Mendeley, 2019b)

- **Servicio para desarrollores (API-REST Mendeley)**

  Mendeley cuenta con un servicio para la comunidad de desarrolladores e investigadores por medio de una API que se basa en los principios de REST, por medio de la cual se puede acceder a los recursos que ofrece este gestor, estos recursos son usados por investigadores, profesionales de la investigación, bibliotecarios e instituciones.

  En forma general, la API Mendeley usa los principios de REST, el formato de texto de respuesta es JSON, para la seguridad usa HTTPS para proteger las conexiones, y finalmente para la autorización usa el protocolo OAuth 2.0

- **Recursos básicos**

Una característica principal de REST es que identifica los recursos disponibles por medio de una url, así que la API REST de Mendeley ofrece los recursos por medio de la url base[2]. Los principales recursos de Mendeley se detallan en la TABLA 3.

TABLA 3 Principales recursos del API Mendeley

Recurso	Descripción	Uri recurso
Documentos	Recursos que representan artículos, revistas, páginas web, conferencias y todo tipo de material de investigación. Cada documento tiene metadatos de dicho documento. Existen dos tipos de documentos, los de catálogo que están en el catálogo colaborativo del gestor y los documentos de usuario que se encuentran en la biblioteca o grupos de cada usuario.	https://api.mendeley.com/documents
Archivos	Es un recurso multimedia el cual esta enlazado a un documento, estos archivos contienen metadatos del documento; cada documento puede tener uno o más archivos asociados, generalmente los archivos son PDF.	https://api.mendeley.com/files
Grupos	Son asociaciones de usuarios que trabajan colaborativamente y comparten recursos. Existen grupos privados, abiertos y de solo invitación. Cada grupo tiene usuarios con roles como: propietario, administrador, miembro, seguidor.	https://api.mendeley.com/groups https://api.mendeley.com/gropus/v2
Carpetas	Son estructuras jerárquicas que contienen colecciones de documentos o a su vez, mas carpetas. Una carpeta puede estar asociada a un usuario o a un grupo colaborativo.	https://api.mendeley.com/folders

Fuente: (Mendeley, 2019a)

1.5. Herramientas Tecnológicas

El presente trabajo busca la creación de una interfaz de programación de aplicaciones, la cual estará compuesta por un servidor backend. Para verificar la funcionalidad del servidor se realizará un cliente frontend, a continuación, se detalla las herramientas tecnológicas a utilizarse.

1.5.1. Backend

- **Node.js**

  "Concebido como un entorno de ejecución de JavaScript orientado a eventos asíncronos, Node está diseñado para construir aplicaciones en red escalables" (Node, 2019). Node.js en su filosofía usa conceptos como la estructura modular de sus proyectos, esto ayuda a que los proyectos sean más manejables, entendibles y reutilizables. Además, permite usar funcionalidades limitadas de proyectos ya que se puede exportar operaciones

[2] **url base**: https://www.api.mendeley.com

especificas del código, haciéndolo así menos propenso a errores de uso y más fácil de entender. La eficiencia de Node.js se debe a que usa el motor v8 creado por Google para el lenguaje JavaScript; lo cual hace óptima a la administración de recursos y la rapidez del entorno de ejecución (Casciaro, 2014).

- **Npm (Gestor de librerías)**

  Para el desarrollo en Node.js se usa Node Package Manager (NPM), que es el gestor de librerías oficial de este entorno, contiene aproximadamente 250 000 librerías (Abdalkareem, Nourry, Wehaibi, Mujahid, & Shihab, 2017). Las aplicaciones basadas en JavaScript ayudan de gran manera al desarrollo web, existen aplicaciones JavaScript que funcionan como librerías para aplicaciones independientes.

- **Express**

  Según (Express.js, 2017), "Express es un marco de aplicación web Node.js mínimo y flexible que proporciona un conjunto robusto de características para aplicaciones web y móviles". Express es muy usado por los desarrolladores ya que cuenta con una amplia red de apoyo a su código fuente y el soporte que se recibe es alto. En la documentación de Mozilla (Mozilla developers, 2019) se detallan algunas funciones de express:

  - Generación de respuestas por medio de ingreso de información en plantillas; esto se lo hace con la incorporación de mecanismos de renderización de vistas.
  - La ubicación de las plantillas de renderización de las respuestas, el uso del puerto para la conexión.
  - La ejecución de solicitudes middleware para el procesamiento de peticiones HTTP y las respuestas correspondientes.

1.5.2. Frontend

- **React.js**

  Basada en JavaScript, react.js es una librería de componentes para la creación de interfaces de usuario reutilizables; en el patrón de diseño MVC, se encarga de la vista. Esta librería trabaja en base a componentes, los cuales son piezas de Interfaz de usuario reutilizables que son escritos por un lenguaje de marcado llamado JSX que se encarga de transformar los componentes con sintaxis de HTML a JavaScript (Aggarwal, 2018).

  En el lado del cliente, el DOM (Document Objet Model) de las aplicaciones web es la estructura interna de un documento HTML, ante cambios en este documento las aplicaciones web actualizan este documento y así se recarga completamente la página

lo que es muy costoso en términos de recursos; ante este inconveniente, React usa el concepto de Virtual DOM que es un DOM guardado en memoria, este objeto se encarga de identificar los cambios en el DOM, pero con la particularidad de que identificado el cambio, solo se actualizará dicho cambio y no todo el DOM (Aggarwal, 2018).

- **Apollo Client**

  Apollo es una plataforma completa para la gestión de datos, existe Apollo para el lado del servidor y Apollo para el lado del cliente. Apollo Client es una implementación del lado del cliente para la obtención de datos de GraphQL y el despliegue de estos en interfaces de usuario, Apollo Client se usa generalmente con React.js, pero también se lo puede usar con otras plataformas; el desarrollo cliente es flexible, adaptable, sencillo, las aplicaciones que usan Apollo Client son adaptables de forma incremental, entendibles, interactivas, con compatibilidad universal ya que funcionan con cualquier servidor GraphQL (Apollo-GraphQL, 2019).

1.6. Metodologías de desarrollo de software

1.6.1. Introducción

Las metodologías de desarrollo de software nacen ante la necesidad de crear productos de calidad ya que estos carecían o tenían un limitado índice de calidad, por esta razón nació la idea de que el desarrollo de software se ajuste a un conjunto de metodologías para así garantizar la calidad en sus productos. Las principales metodologías establecidas para el desarrollo de software son dos: metodologías de desarrollo de software tradicional y ágil (Molina, Vite, & Dávila, 2018). En la TABLA 4 se detallan las diferencias entre las dos metodologías.

TABLA 4 Diferencia entre Metodología tradicional y ágil

Metodologías tradicionales	Metodologías ágiles
Predictivos	Adaptativos
Orientado a procesos	Orientado a personas
Proceso rígido	Proceso flexible
Se concibe como un proyecto	Un proyecto es subdividido en varios proyectos más pequeños.
Poca comunicación con el cliente.	Comunicación constante con el cliente.
Entrega de software al finalizar el desarrollo	Entregas constantes de software
Documentación extensa	Poca documentación

Fuente: Adaptado de (Molina et al., 2018)

1.6.2. Marco de trabajo Scrum

Según (Scrum, 2019), Scrum se define como un marco simple que ayuda a que proyectos grandes puedan ser abordados de manera eficiente; caracterizado por el desarrollo incremental, es por este motivo que Scrum genera entregables cada cierto tiempo. El marco técnico de Scrum se forma de: roles, artefactos y eventos los cuales se resumen en la Fig. 16

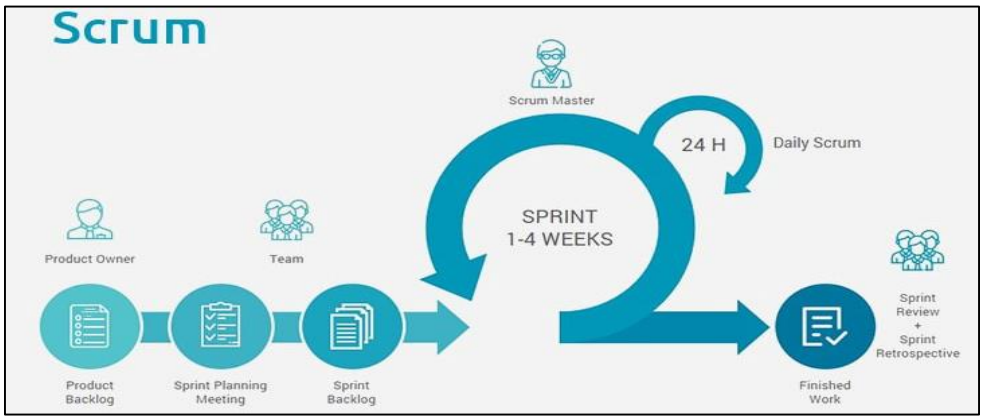


Fig. 16. Desarrollo ágil de software (Scrum)
Fuente: (Porras, 2019)

- **Roles de Scrum**

  Un proyecto Scrum está compuesto por un equipo de trabajo en el cual, cada uno de los miembros tiene un rol específico. Los roles que componen Scrum son: Propietario del Producto (Product Owner), el Líder o Administrador del proyecto (Scrum Master) y Miembro del proyecto que la reunión de estos se denomina (Team) Grupo de desarrollo (Greene & Stellman, 2014).

  La definición de los roles de Scrum según (Palacio, Menzinsky, & López, 2016) son:

  a) El Product Owner debe ser una única persona, experta en el proceso a automatizarse, quien realiza la toma de decisiones de los clientes. El Propietario del Producto decide cómo se verá finalmente el producto y también define partes del producto para que se conviertan en entregable, debe tener una comunicación continua con el Scrum Master.

  b) El Scrum Master es la persona que lidera el desarrollo del producto, este miembro conoce el plan de desarrollo del producto y además es el encargado de guiar al grupo de desarrollo en el cumplimiento de los entregables por medio de decisiones técnicas, tiene constante comunicación con el Propietario del Producto y el equipo de desarrollo.

  c) El equipo de desarrollo es un equipo compuesto por personas cualificadas para desarrollar el producto, este grupo debe trabajar de forma simultanea cumpliendo tareas expuestas en la planificación de ciertas actividades preestablecidas. Trabajan en colaboración con el Scrum Master.

- **Artefactos de Scrum**

  Los requisitos en un proyecto scrum pueden ser exhibidos por medio de dos artefactos denominados Pila del producto (Product Backlog) y Pila del sprint (Sprint Backlog), también existe el denominado incremento que es el resultado de cada sprint. En la guía de (Palacio et al., 2016) detalla a los artefactos de la siguiente manera:

  a) La pila del producto o Product Backlog especifica los requerimientos del producto a desarrollarse mediante un tipo de requisito denominado historia de usuario, el Propietario del Producto es el encargado de obtener estos recursos en los cuales se especifica de manera clara la información detallada y categorizada del software a desarrollarse.

  b) La pila del sprint o Sprint Backlog contiene cierto contenido de la pila del producto que se desarrollara en un periodo de tiempo, pero con la particularidad de que las historias de usuario están desglosadas en tareas, las cuales son distribuidas en un plan para desarrollarlas. El pla de la pila del sprint es dinamico ya que las actividades cambian de estado; hay tres posibles estados: Por hacer, En proceso y Hecho.

  c) El incremento es un entregable del software, es una parte que fue desarrollada durante un sprint y que se convierte en una versión utilizable y demostrable. Este incremento se presenta en la revisión del sprint.

- **Eventos de Scrum**

  En Scrum existe el termino sprint que se define como un período de tiempo en el cual por medio del desarrollo de software se genera un incremento de dicho software, este incremento es usable y funcional, el sprint contiene eventos que se desarrollan de manera específica.

  Los eventos de Scrum según (Greene & Stellman, 2014) se detallan a continuación.

  a) La planificación del sprint o Sprint Planning es el inicio del sprint, esta se la realiza por todos los que integran el equipo Scrum; en esta reunión se definen prioridades para el desarrollo del producto y se establecen las tareas a realizarse para que se conviertan en el siguiente entregable del producto.

  b) La reunión diaria o Daily Scrum es realizada por el equipo Scrum, tiene una duración de 15 minutos y su finalidad es dar a conocer información e inquietudes en el proceso de desarrollo de software. Existen tres preguntas base de las reuniones diarias que son: “¿Qué hizo ayer?”, “¿Qué hará hoy?” y “¿Qué limitaciones ha tenido durante el desarrollo?”. El Scrum Master es el encargado de resolver toda clase de inconvenientes.

c) La revisión del sprint o Sprint Review es una reunión en la cual se entrega una versión del proyecto por medio de una demostración la cual deberá tener las especificaciones hechas en la planificación del sprint, si no se cumplieron dichas especificaciones, se deberá entregar una versión funcional aprobada por el Propietario del Producto.

d) La retrospectiva es una especie de reunión en donde el equipo de desarrollo junto al Scrum Master hacen un análisis del sprint del cual se pueden realizar mejoras del proceso de desarrollo o de alguna otra actividad específica.

1.7. ISO/IEC 25000

La norma ISO/IEC 25000 constituye una familia de estándares que evalúa la calidad del producto de software. Esta familia también es conocida como System and Software Quality Requirements and Evaluation (SQuaRE) y fue creada en base a otras normas tales como las normas ISO/IEC 9126 e ISO/IEC 14598 (ISO 25000, 2019). La división de esta norma se la detalla en la Fig. 17.

Fig. 17. División de la norma ISO/IEC 25000
Fuente: (ISO 25000, 2019)

1.7.1. ISO/IEC 25010

El sistema para evaluar la calidad de un producto, deriva de un modelo de calidad en uso en donde se determina las características de calidad que debe tener dicho producto a la hora de evaluarlo. Para verificar la calidad del producto de software, es necesario que plasmar ciertas condiciones que derivan de un modelo de calidad del producto compuesto por una serie de características y sub-características (adecuación funcional, eficiencia de desempeño, compatibilidad, usabilidad, fiabilidad, seguridad, mantenibilidad y portabilidad) que representan una categorización del modelo de calidad en uso (ISO 25000, 2019).

- **Modelo de calidad en uso**

  Las características que abarcan la calidad en uso son: Eficiencia, eficacia, satisfacción, libertad de riesgo y cobertura de contexto que se muestran en la Fig. 18, las cuales evalúan el grado en que un producto puede ser usado por un público de usuarios que satisfacen necesidades y logran objetivos (ISO/IEC 25010, 2011).

Eficacia
Eficiencia
Satisfacción
Utilidad
Confianza
Placer
Comodidad
Libertad de riesgo
Mitigación de riesgos económicos.
Mitigación de riesgos de salud y seguridad.
Mitigación de riesgos ambientales.
Cobertura de contexto
Completitud del contexto
Flexibilidad

Fig. 18. Características y subcaracterísticas de calidad de uso
Fuente: (ISO/IEC 25010, 2011)

(ISO/IEC 25010, 2011) menciona que la usabilidad se define como un subconjunto de calidad en uso que consiste en efectividad, eficiencia y satisfacción, para mantener la coherencia con su significado establecido. La calidad de uso tiene un subconjunto denominado usabilidad por lo cual será evaluado el envoltorio.

1.7.2. ISO/IEC 25022

- **Medición de calidad en uso**

  Para la evaluación de calidad en uso de productos, es necesaria una normativa de medición que se asocie al modelo de calidad en uso. La ISO/IEC 25022 proporciona una serie de medidas para evaluar el modelo de calidad de un sistema informático, siendo estas medidas flexibles a modificación de acuerdo al criterio del usuario. Las medidas tienen como objetivo asegurar la calidad del producto en función de su utilización, los resultados de estas medidas son los desarrolladores, evaluadores o cualquier persona autorizada para la valorización de la medición de la evaluación (ISO/IEC 25022, 2016).

1.7.3. ISO/IEC 25040

- **Evaluación de calidad en uso**

  El proceso de evaluación de calidad en uso que detalla el estándar ISO/IEC 25040, contempla conceptos generales y requisitos de evaluación, además de una descripción general de este proceso, que junto con la norma ISO/IEC 25010 que define el modelo de calidad en uso y la norma ISO/IEC 25022 que define las métricas de medición, completan la evaluación del producto de software desde la perspectiva de calidad de uso, calidad interna y calidad externa. Entre los responsables de usar esta normativa están: los desarrolladores, persona adquiriente del producto, evaluadores certificados, una organización o cualquier persona autorizada para ejecutar este proceso de evaluación (ISO/IEC 25040, 2011).

CAPÍTULO 2

Desarrollo

La fase de desarrollo del presente proyecto consta de dos software desarrollados bajo el marco de trabajo ágil SCRUM. El primer software es un servidor que envuelve los servicios REST del API Mendeley, el cual tiene la función de brindar recursos y servicios por medio de una API GraphQL. El software cliente es una prueba de concepto que consume los servicios y recursos de la API GraphQL de Mendeley y tiene como función probar la utilización de la API desarrollada. En la Fig. 19 Se observa el proceso de desarrollo del proyecto.

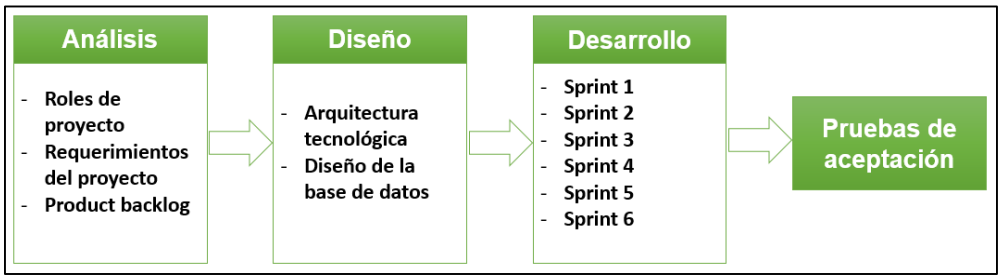


Fig. 19 Estructura de la fase de desarrollo del proyecto

2.1. Análisis

2.1.1. Roles de Proyecto

Para el desarrollo del API GraphQL de Mendeley (envoltorio del API-REST de Mendeley) y la prueba de concepto, se ha compuesto un grupo de trabajo colaborativo según el Marco de Trabajo Scrum; en la TABLA 5 se muestra los roles de cada miembro del equipo.

TABLA 5 Roles del Proyecto

Miembro	Descripción	Rol
Ing. Antonio Quiña	Director del presente Trabajo de Grado y Docente de la Carrera de Ingeniería en Sistemas Computacionales de la Universidad Técnica del Norte.	Propietario del Producto (Product Owner)
Sr. Kevin Rodríguez	Estudiante de la Carrera de Ingeniería en Sistemas Computacionales de la Universidad Técnica del Norte.	Jefe del Proyecto (Scrum Master).
Sr. Kevin Rodríguez	Estudiante de la Carrera de Ingeniería en Sistemas Computacionales de la Universidad Técnica del Norte..	Equipo de Desarrollo (Development Team)

2.1.2. Requisitos del Proyecto

- **Historias de Usuario**

  Los requisitos para el envoltorio y para la prueba de concepto fueron levantados por el propietario del producto por medio de Historias de usuario, las cuales son reconocidas como un instrumento en la metodología Scrum.

TABLA 6 Historia de Usuario 1

Historia de Usuario	**Numero:** HU-01
Nombre: Definición de estructura del software	**Usuario:**
Dependencias: Ninguna	**Estimación:** 32 horas
Descripción: Establecer las principales funcionalidades del envoltorio GraphQL, mediante pruebas de consumo al API de Mendeley para verificar los servicios que ofrece el gestor y así estructurar la base técnica del software a desarrollarse detallando la información necesaria para poder usar dichos servicios.	
Pruebas de aceptación • La autenticación de la API REST de Mendeley debe ser establecida en un formato entendible de documentación para su posterior desarrollo. • La información obtenida del consumo de los servicios del API debe ser clara y fácil de comprender para su uso en el desarrollo. • Conocer los servicios adicionales del API que no son considerados como principales pero que son necesarios para la funcionalidad del gestor.	

TABLA 7 Historia de Usuario 2

Historia de Usuario	**Numero:** HU-02
Nombre: Gestión del recurso Documentos	**Usuario:** Desarrollador frontend
Dependencias: HU-01	**Estimación:** 14 horas
Descripción: Como desarrollador frontend, requiero tener un servicio API de administración de documentos de Mendeley, Este servicio debe poder obtener un documento o un listado de documentos del usuario que inició sesión, también debe poder crear, actualizar y eliminar documentos.	
Observación: Los campos necesarios para la gestión de documentos se detallan en la documentación del API-REST https://api.mendeley.com/apidocs/docs#!/documents/getDocuments.	
Pruebas de aceptación • La obtención exitosa de un listado de documentos con sus parámetros (opcional), generará una respuesta HTTP OK y un archivo json con el contenido de la consulta.	

- El ingreso erróneo de los parámetros de listado de documentos generará un mensaje informando el error.
- La creación, la actualización o eliminación exitosa de un documento, generará una respuesta HTTP OK.
- El ingreso de un ID de documento erróneo mostrará un mensaje informando el error.
- El ingreso de parámetros válidos en la creación, actualización o eliminación de un documento generará una respuesta satisfactoria y un archivo json informando el estado de la operación, caso contrario, generará una respuesta informando el error.

TABLA 8 Historia de Usuario 3

Historia de Usuario	**Numero:** HU-03
Nombre: Gestión del recurso Carpetas	**Usuario:** Desarrollador frontend
Dependencias: HU-01	**Estimación:** 12 horas

Descripción:

Como desarrollador frontend, requiero tener un servicio API de administración de carpetas de Mendeley, Este servicio debe poder obtener una carpeta o un listado de carpetas del usuario que inició sesión, también debe poder crear, actualizar y eliminar carpetas.

Observación:

Los campos necesarios para la gestión de carpetas son: id, name, parent_id, group_id, created, modified; todos de tipo String.

Pruebas de aceptación

- La obtención exitosa de un listado del recurso carpetas con sus parámetros los cuales son opcionales, generará una respuesta HTTP OK satisfactoria y un archivo json con el contenido de la consulta
- El ingreso erróneo de los parámetros de listado de carpetas generará un mensaje informando el error.
- La creación, la actualización o eliminación exitosa de una carpeta, generará una respuesta HTTP OK.
- El ingreso de un ID de carpeta erróneo mostrará un mensaje informando el error.
- El ingreso de parámetros válidos para la creación, actualización o eliminación de una recurso carpeta generará una respuesta satisfactoria y un archivo json informando el estado de la operación, caso contrario, generará una respuesta informando el error.

TABLA 9 Historia de Usuario 4

Historia de Usuario	**Numero:** HU-04
Nombre: Gestión del recurso Grupos	**Usuario:** Desarrollador frontend
Dependencias: HU-01	**Estimación:** 14 horas

Descripción:

Como desarrollador frontend, requiero tener un servicio API de administración de grupos de Mendeley, Este servicio debe poder obtener un grupo o un listado de grupos del usuario que proporciono el token de acceso; también debe poder crear, actualizar y eliminar grupos.

Observación:

Los campos necesarios para la gestión de grupos de Mendeley se detallan en: https://dev.mendeley.com/methods/?shell#groups-v2.

Pruebas de aceptación

- La obtención exitosa de un listado de grupos con sus parámetros (opcional), generará una respuesta HTTP OK y un archivo json con el contenido de la consulta
- El ingreso erróneo de los parámetros de listado de grupos generará un mensaje informando el error.
- La creación, la actualización o eliminación exitosa de un grupo, generará una respuesta HTTP OK.
- El ingreso de un ID de grupo erróneo mostrará un mensaje informando el error.
- El ingreso de parámetros válidos en la creación, actualización o eliminación de un grupo generará una respuesta satisfactoria y un archivo json informando el estado de la operación, caso contrario, generará una respuesta informando el error.

TABLA 10 Historia de Usuario 5

Historia de Usuario	**Numero:** HU-05
Nombre: Gestión del recurso Archivos	**Usuario:** Desarrollador frontend
Dependencias: HU-01	**Estimación:** 12 horas

Descripción:

Como desarrollador frontend, requiero tener un servicio API de administración de archivos de Mendeley, Este servicio debe poder descargar un archivo u obtener un listado de archivos del usuario que proporciono el token de acceso; también debe poder eliminar archivos.

Observación:

Los campos necesarios para la gestión de archivos se detallan en: https://dev.mendeley.com/methods/?shell#files

Pruebas de aceptación

- La obtención exitosa de un listado de archivos con su parámetro límite (opcional), generará una respuesta HTTP OK y un archivo json con el contenido de la consulta.

- El ingreso erróneo del parámetro de listado de archivos generará un mensaje informando el error.
- La descarga o eliminación exitosa de un grupo, generará una respuesta HTTP OK.
- El ingreso de un ID de archivo erróneo mostrará un mensaje informando el error.

TABLA 11 Historia de Usuario 6

Historia de Usuario	**Numero:** HU-06
Nombre: Autenticación de API	**Usuario:** Desarrollador frontend
Dependencias: HU-01	**Estimación:** 16 horas

Descripción:

Como desarrollador frontend requiero un servicio API de autenticación que permita iniciar sesión a usuarios de Mendeley previamente registrados. El servicio debe pedir el correo y la contraseña de usuario. Se espera que la autenticación con credenciales correctas permita el uso de los servicios de Mendeley; si las credenciales son incorrectas, se espera un mensaje informando el error.

Pruebas de aceptación

- El ingreso de un correo valido y una contraseña válida, permite a los usuarios Mendeley el acceso a las funcionalidades del gestor.
- El ingreso de credenciales inválidas genera un mensaje de error informando de dicho error y permitiendo al usuario volver a ingresar las credenciales.

TABLA 12 Historia de Usuario 7

Historia de Usuario	**Numero:** HU-07
Nombre: Modelo de gestión de la aplicación web.	**Usuario:** Investigador
Dependencias: HU-02, HU-03, HU-04, HU-05	**Estimación:** 11 horas

Descripción:

Como investigador, requiero un portal de gestión de recursos de Mendeley como una prueba de concepto de la API, el portal deberá contener una barra de navegación con el logo de Mendeley y opciones de usuario. La gestión de los diferentes recursos debe ser realizada de modo que simule las funciones de la app oficial de Mendeley (Sin prioridad funcionalidades del lado del cliente). Los datos del usuario (nombres, correo electrónico, nivel académico) deberán ser mostrados en una ventana.

Pruebas de aceptación

- Los datos del usuario serán desplegados en una ventana emergente.
- Los paneles de los recursos de grupos, carpetas, documentos y detalle de documento mostrarán una interfaz que indique la relación de estos componentes.
- El panel de carpetas y grupos debe contener a los recursos mencionados en el mismo componente, pero diferenciando cada uno de estos.

TABLA 13 Historia de Usuario 8

Historia de Usuario	**Numero:** HU-08
Nombre: Gestión de carpetas y grupos	**Usuario:** Investigador
Dependencias: HU-07, HU-03, HU-04	**Estimación:** 16 horas

Descripción:

Como investigador, requiero el manejo de carpetas y grupos; las carpetas y grupos deberán listarse en sus respectivos paneles. Para ingresar una nueva carpeta, deberá pedir como datos de entrada el nombre de la misma, en el caso de un nuevo grupo, deberá pedir como datos de entrada el nombre del grupo y el tipo de acceso (público, privado, solo invitación). Para eliminar cualquiera de los recursos mencionados, se deberá mostrar la respectiva confirmación.

Pruebas de aceptación

- El listado de carpetas y grupos deberá mostrar opciones de edición y eliminación de cada recurso.
- La información ingresada en el registro o edición de carpetas o grupos será validada según el tipo de dato, mostrará un mensaje de operación satisfactoria si se realizó la operación exitosamente, caso contrario mostrará un mensaje de error.
- La eliminación de un grupo o carpeta mostrará un mensaje de confirmación para realizar la acción.

TABLA 14 Historia de Usuario 9

Historia de Usuario	**Numero:** HU-09
Nombre: Gestión de documentos y archivos	**Usuario:** Investigador
Dependencias: HU-07, HU-02, HU-05	**Estimación:** 18 horas

Descripción:

Como investigador, requiero el manejo de documentos; estos documentos deberán listarse en un panel; en la lista se debe identificar si un documento tiene un archivo adjunto. Para el ingreso de un nuevo documento, los datos de entrada serán: título, tipo, autores, resumen, año, páginas, DOI, lista de URLs, etiquetas; estos documentos tienen más propiedades que no se abarcan en la aplicación de prueba, pero se los puede manipular en la aplicación oficial. Los detalles de un documento deberán mostrarse en un panel que contendrá toda la información del documento (incluyendo el archivo y su respectiva descarga y visualización si así lo tuviere), esta información será desplegada al seleccionar un documento. Para eliminar un documento, se deberá mostrar la respectiva confirmación.

Pruebas de aceptación

- Los documentos se mostrarán mediante un listado con un filtro de carpetas o grupos.
- La información de un documento será desplegada por medio de la selección del mismo en un panel que muestre dicha información.
- Si un documento tiene un archivo adjunto, descargarlo desde los detalles del documento.

- La información ingresada en los formularios de registro o edición de un documento será validada según el tipo de dato, mostrará un mensaje satisfactorio si se realizó la operación exitosamente, caso contrario mostrará un mensaje de error.
- La eliminación de un documento mostrará un mensaje de confirmación para realizar la acción.
- Un archivo podrá mostrarse (solo PDF) de forma online y descargarse, así como eliminarse.

TABLA 15 Historia de Usuario 10

Historia de Usuario	**Numero:** HU-10
Nombre: Portal de desarrollo del envoltorio	**Usuario:** Desarrollador
Dependencias: HU-01	**Estimación:** 23 horas

Descripción:

Como desarrollador requiero un portal de desarrollo del envoltorio en el cual, se presente información acerca del mismo, una explicación de cómo usarlo; presentar las soluciones de como probar la API, enlaces a los repositorios GitHub de los proyectos y la documentación respectiva. También, requiero la implementación de una página web que presente información de uso de su identidad para realizar peticiones. Requiero que los proyectos(envoltorio y prueba de concepto) sean públicos y sean desplegados en un PaaS..

Pruebas de aceptación

- El portal de desarrollo mostrará información intuitiva sobre el envoltorio.
- La documentación del envoltorio será mostrada en un documento PDF descargable.
- Los repositorios de los proyectos deben estar en GitHub.
- El portal de desarrollo debe tener links a los repositorios GitHub.
- La página de información de uso de identidad debe mostrar información de permiso de uso de credenciales para peticiones a la API.
- La funcionalidad del envoltorio en ambiente de producción debe ser igual al ambiente local.
- La funcionalidad de la prueba de concepto en ambiente de producción debe ser igual al ambiente local.

TABLA 16 Historia de Usuario 11

Historia de Usuario	**Numero:** HU-11
Nombre: Actividad del envoltorio	**Usuario:** Desarrollador
Dependencias: HU-10	**Estimación: 17** horas

Descripción:

Como desarrollador, requiero una base de datos basado en el modelo realizado previamente para brindar reportes de actividad del envoltorio; realizar una conexión hacia esta por medio de un administrador de base de datos. Requiero una página de reportes en base a la base de datos de

actividad del envoltorio en donde se liste el uso de cada petición realizada, los usuarios que lo han usado, detalles de operaciones REST y GraphQL y una cronología de uso.

Pruebas de aceptación

- La base de datos será alojada en la PaaS del envoltorio.
- Las peticiones realizadas se listarán en una tabla.
- Los usuarios que han usado la API se mostrarán en una tabla independiente.
- Los recursos y operaciones REST se mostrarán en tarjetas indicando el número de veces que se accedió a dichos recursos y operaciones.
- Las operaciones GraphQL se mostrarán en tarjetas y un gráfico de porcentajes, indicando el número de veces que se han hecho peticiones de cada tipo.
- Las consultas de la actividad del API desde el administrador de base de datos deben coincidir con los datos arrojados desde la presentación de actividad de la aplicación.

2.1.3. Product Backlog

El Product Backlog detallado en la TABLA 17 contiene las historias de usuario definidas en orden según la prioridad definida por le Product Owner.

TABLA 17 Product Backlog

PRODUCT BACKLOG

Orden	ID HU	DESCRIPCIÓN	ROL	ESTIMACIÓN
1	HU-01	Definición de estructura del software	--	32
2	HU-05	Gestión del recurso Archivos	Desarrollador fronted	12
3	HU-04	Gestión del recurso Grupos	Desarrollador fronted	14
4	HU-03	Gestión del recurso Carpetas	Desarrollador fronted	12
5	HU-02	Gestión del recurso Documentos	Desarrollador fronted	14
6	HU-06	Autenticación de API	Desarrollador fronted	16
7	HU-07	Modelo de gestión de la aplicación web.	Investigador	11
8	HU-10	Portal de desarrollo del envoltorio	Desarrollador	23
9	HU-09	Gestión de documentos y archivos	Investigador	18
10	HU-08	Gestión de carpetas y grupos	Investigador	16
11	HU-11	Actividad del envoltorio	Desarrollador	17

2.2. Diseño

El desarrollo del proyecto está distribuido en Sprints, sin embargo, para definir el diseño del proyecto se realizó un intervalo de desarrollo denominado Sprint 0 en donde se definió la arquitectura tecnológica y el diseño de la base de datos inicial que guarda información de la actividad del envoltorio.

TABLA 18 Detalle del Sprint 0

Sprint	Fecha inicio	Fecha fin	Duración
Sprint 0	10-jun 2019	24-jun 2019	20

Planificación del Sprint 0

Reunión de planificación del proyecto y definición de base tecnológica del proyecto.

Fecha: 10/06/2019

Objetivo: Definir la arquitectura tecnológica y la base de datos inicial del proyecto

Miembros: Product Owner , Scrum Master y Equipo de desarrollo.

TABLA 19 Planificación Sprint 0

Fase de desarrollo	Tarea	Estimación de tiempo
Diseño	Definir la arquitectura tecnológica del proyecto	5
Diseño	Diseñar el modelo de la base de datos de registro de actividad del envoltorio.	4
Diseño	Crear la base de datos en algún gestor de base de datos.	3
Planificación	Detallar las tareas a realizar en el Sprint actual	4
Revisión	Revisar los resultados del desarrollo del Sprint	3
Retrospectiva	Analizar los resultados del Sprint	1
	TOTAL HORAS	**20**

Revisión del Sprint 0

2.2.1. Arquitectura tecnológica

La arquitectura tecnológica consta de dos proyectos, un servidor (envoltorio) y un cliente (prueba de concepto). El servidor está desarrollado en Node.js con la implementación de tecnologías como, graphql-yoga, express, el API Rest de Mendeley; la base de datos se encuentra en

PostgreSQL. El cliente está desarrollado en React.js con el uso de tecnologías tales como Apollo client, bootstrap, reactstrap. Ver Fig. 20.

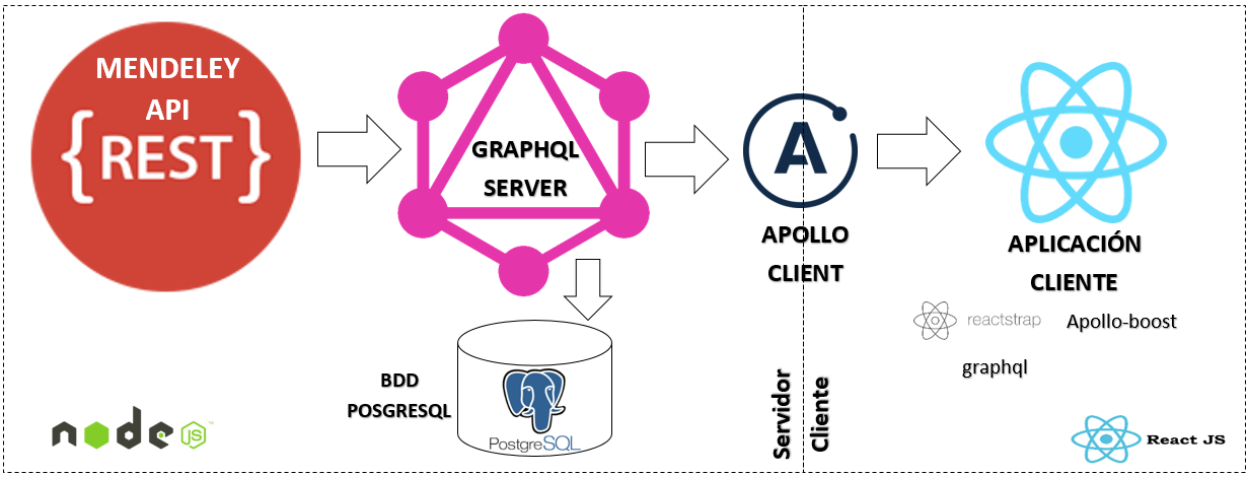


Fig. 20 Arquitectura tecnológica

2.2.2. Diseño inicial de la base de datos

La base de datos tiene la función de registrar las operaciones que realiza el envoltorio, el modelo de esta se la puede observar en la Fig. 21.

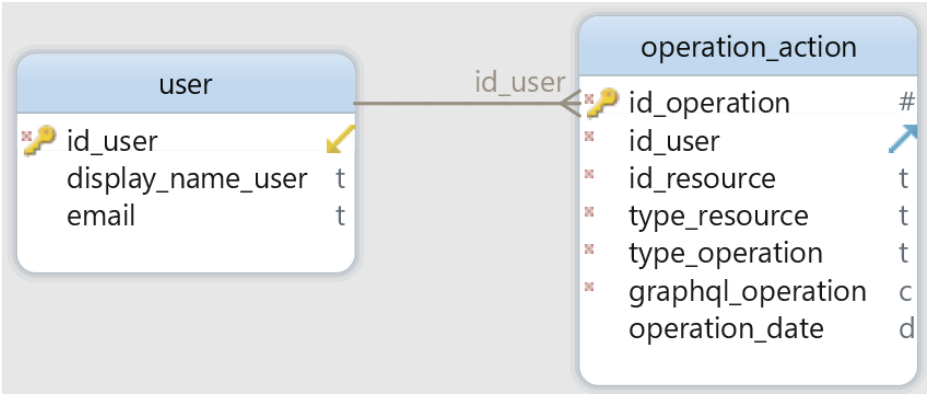


Fig. 21 Modelo de la Base de Datos

2.3. Desarrollo

La ejecución del proyecto se lo realizó en Sprints, cada sprint tiene una duración de 40 horas distribuidas en tres semanas a excepción del Sprint 6 que tiene una duración de 50 horas, Ver TABLA 20.

TABLA 20 Detalle general de los Sprints

Sprint	Fecha inicio	Fecha fin	Duración
Sprint 1	01-jul 2019	22-jul 2019	40
Sprint 2	22-jul 2019	12-ago 2019	40
Sprint 3	12-ago 2019	02-sep 2019	40

Sprint 4	02-sep 2019	23-sep 2019	40
Sprint 5	23-sep 2019	14-oct 2019	40
Sprint 6	14-oct 2019	04-nov 2019	50

2.3.1. SPRINT 1

- **Planificación del Sprint 1**

Establecimiento de tareas de exploración de la API actual de Mendeley para la generación de un documento el cual contenga información de uso de los servicios de dicha API que sirva como insumo para el desarrollo del producto.

a) Reunión de planificación

Fecha: 01 de julio de 2019

Asistentes: Scrum Master, Product Owner, Team Development

Objetivo: Planificación del Sprint 1 (Creación del Sprint backlog)

b) Sprint Backlog

TABLA 21 Sprint 1 Backlog

PLANIFICACIÓN

Historia de usuario	Fase de desarrollo	Tarea	Estimación de tiempo
HU-01	Análisis	Leer y analizar la documentación de la API de Mendeley para conocer los requisitos de uso de la API y definir los recursos a utilizar.	8
	Análisis	Realizar el proceso de autenticación de API para tener acceso a los recursos de Mendeley.	3
	Pruebas	Pruebas de concepto de las tecnologías (Sug. Postman).	3
	Documentación	Diseñar un documento con la información necesaria de la autenticación de los recursos del API	4
	Pruebas	Consumir los recursos principales de Mendeley con alguna herramienta de peticiones de API.	6
	Pruebas	Pruebas de concepto de las tecnologías (Sug. Postman).	3
	Documentación	Diseñar un documento con la información de consumo de los recursos definidos.	5
Eventos	Planificación	Detallar las tareas a realizar en el Sprint actual	4
	Revisión	Revisar los resultados del desarrollo del Sprint	3
	Retrospectiva	Analizar los resultados del Sprint	1
		TOTAL HORAS	**40**

- **Revisión del Sprint 1**

La culminación del primer Sprint determinó la estructura para el desarrollo de la API. La siguiente información es el resultado de la revisión y análisis de la documentación del portal de desarrolladores de Mendeley, esta información sirve para estructurar el funcionamiento de la API GraphQL de Mendeley y facilitar el desarrollo del envoltorio.

a) Reunión de revisión

Fecha: 22 de Julio de 2019

Asistentes: Scrum Master, Product Owner, Team Development

Resultado: Revisión del desarrollo del incremento del producto.

b) Incremento del producto potencialmente entregable

URL base del API REST Mendeley: https://api.mendeley.com/

AUTENTICACIÓN

La autenticación del envoltorio de Mendeley se lo hace usando el protocolo para autorizaciones OAuth2 que permite la autorización de solicitudes a la API por medio de tokens de acceso. Este protocolo permite tres tipos de concesiones para el uso de los servicios de Mendeley, estos se detallan en la TABLA 22. Para el desarrollo del envoltorio se usará el tipo de concesión *código de autorización*.

TABLA 22 Tipos de conseciones Autorización Mendeley

Tipo de subvención:	Código de Autorización	Implícito	Credenciales del cliente
Delegado para:	Usuario	Usuario	Solicitud
Requiere confidencialidad del cliente:	Y		Y
Inicio sesión en la interfaz de usuario:	Y	Y	
Fichas de acceso de larga duración:	Y		

Fuente: Adaptada de (Mendeley, 2019c)

Para obtener el token de acceso que permita a los usuarios hacer peticiones al envoltorio GraphQL de Mendeley, se realizó una revisión del proceso de Autorización mediante el flujo *código de autorización;* este proceso se detalla en el diagrama de flujo de la Fig. 22

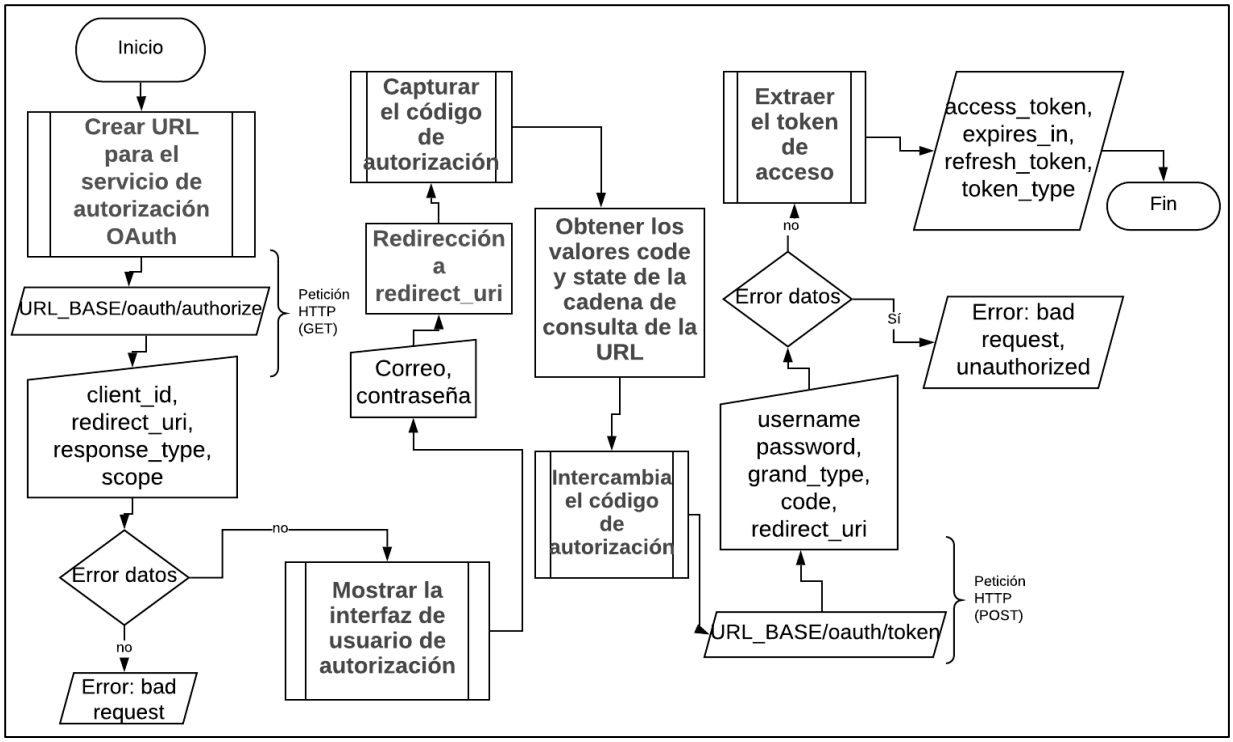


Fig. 22 Flujograma del proceso de autenticación API Mendeley (Código de Autorización)
Fuente: Adaptada de (Mendeley, 2019c)

RECURSO DOCUMENTOS

- **Documentación del recurso:** https://dev.mendeley.com/methods/#documents
- **URI:** /documents

- **Atributos principales**

id	string		Identificador UUID del document, establecido por el servidor al crear el recurso, no modificable
title	string	255	Título del document, es requerido
type	string		Tipo de documento: journal, book, book_section, generic, conference_proceedings, working_paper, report, thesis web_page, magazine_article, statute, patent, hearing, case newspaper_article, computer_program, film, bill television_broadcast, encyclopedia_article.
profile_id	string		Id de Perfil del usuario Mendeley que agrego el documento
group_id	string		Id de grupo al cual el document pertenece.
created	string		Fecha creación del documento, establecido por el servidor.
last_modified	string		Fecha última edición del recurso, establecido por el servidor.
abstract	string	10000	Resúmen del documento
source	string	255	Punto de publicación del documento
year	integer		Año de publicación o emición del document.
authors	array		Lista de autores del document.
identifiers	array	500 each	Identificadores del documento tales como: arxiv, doi, isbn, issn, pmid (PubMed), scopus and ssrn.

keywords	array	50 each	Palabras clave del documento establecidas por el autor del mismo.

El recurso tiene atributos secundarios que se los puede obtener con el parámetro de cuerpo view, este parámetro se lo detalla en la operación GET.

- **Operaciones**

GET	POST	PATCH	DELETE
Response Content Type: application/vnd.mendeley-document.1+json **Parámetros:** ***limit*** (string) - de 1 a 500 (Default 20) ***sort*** (string) – title, last_modified, created (Default created) . ***order*** (string) – asc, desc (Defacult asc) ***view*** (string) - bib, all, client, tags, patent () ***id:*** ID para obtener un documento en específico	**Response Content Type:** application/vnd.mendeley-document.1+json **Parameter content type:** application/vnd.mendeley-document.1+json **Parámetros:** ***body***: Cuerpo solicitud	**Response Content Type:** application/vnd.mendeley-document.1+json **Parameter content type:** application/vnd.mendeley-document.1+json **Parámetros:** ***body***: Cuerpo solicitud ***id:*** ID del documento a modificar	**Parámetros:** ***id****:* ID del documento a eliminar

A continuación, se presenta un ejemplo del consumo del recurso Documento, ver Fig. 23

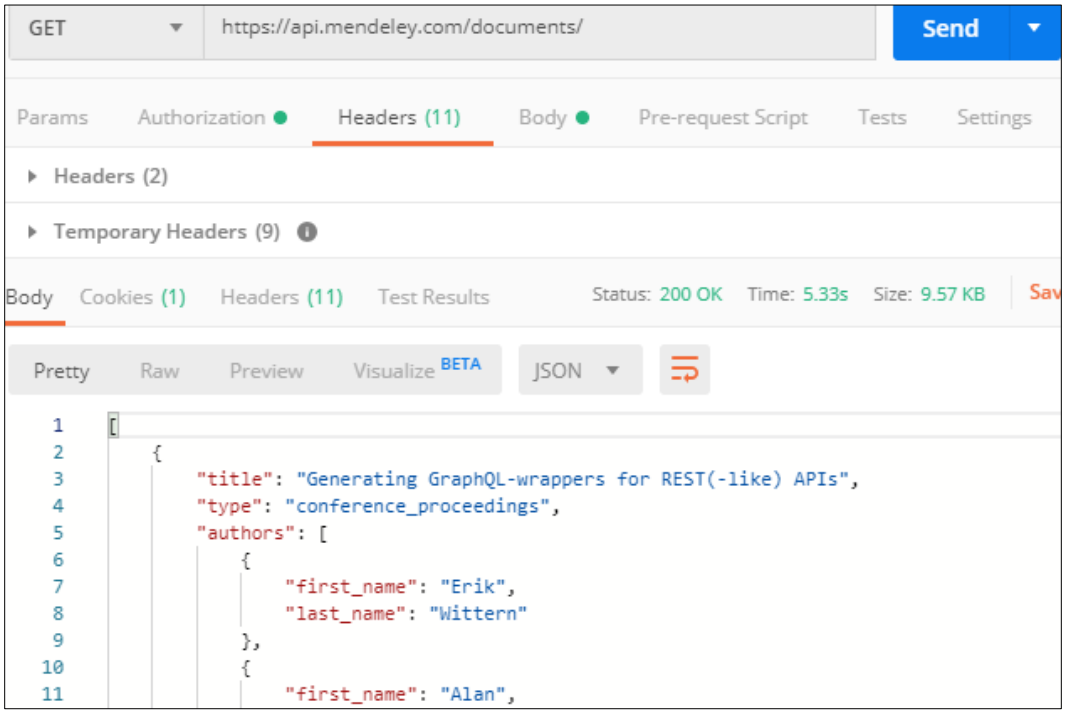


Fig. 23 Consumo del recurso Document en el software Postman

RECURSO CARPETAS

- **Documentación del recurso**: https://dev.mendeley.com/methods/#folders
- **URI:** /folders
- **Atributos**

id	string	ID de la carpeta.
name	string	Nombre de la carpeta
parent_id	string	ID de la carpeta padre.
group_id	string	ID del grupo que pertenece la carpeta.
created	string	Fecha de creación de la carpeta, establecido por el servidor.
modified	string	Fecha de última edición de la carpeta, establecido por el servidor

- **Operaciones**

GET	POST	PATCH	DELETE
Response Content Type: application/vnd.mendeley-folder.1+json **Parámetros:** ***limit*** (string) - de 1 a 500 (Default 20) ***group_id***: Documentos de la carpeta con el ID especificado (Default: si no especifica; se lista todos los documentos) ***id:*** ID para obtener una carpeta específica	**Response Content Type:** application/vnd.mendeley-folder.1+json **Parameter content type:** application/vnd.mendeley-folder.1+json **Parámetros:** ***body***: Cuerpo solicitud	**Response Content Type:** application/vnd.mendeley-folder.1+json **Parameter content type:** application/vnd.mendeley-folder.1+json **Parámetros:** ***body***: Cuerpo solicitud ***id:*** ID de la carpeta a modificar	**Parámetros:** ***id****:* ID de la carpeta a eliminar

A continuación, se muestra un ejemplo del consumo del recurso carpeta, ver Fig. 24

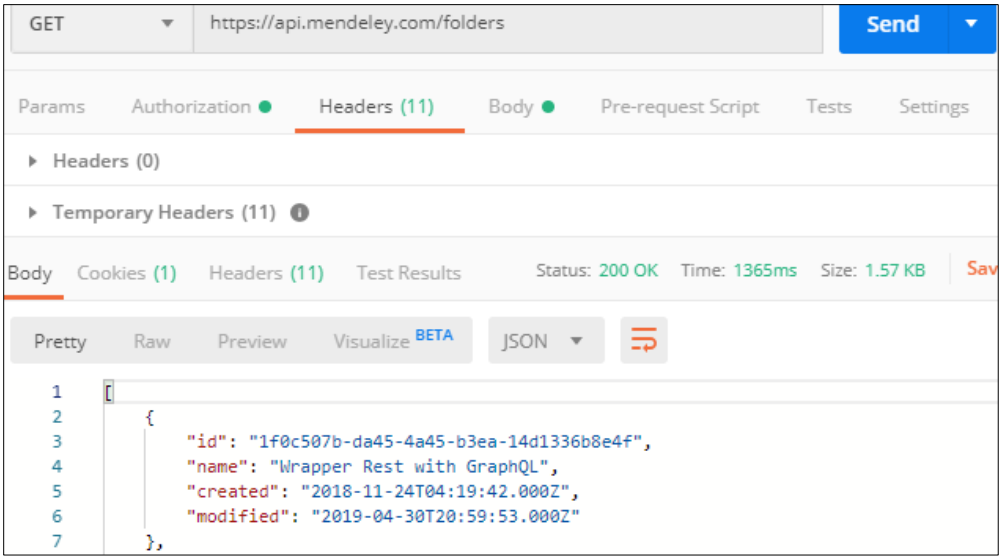


Fig. 24 Consumo del recurso Folder en el software Postman

RECURSO GRUPOS

- **Documentación del recurso:** https://dev.mendeley.com/methods/?shell#groups-v2
- **URI:** /groups/v2
- **Atributos**

id	string	Identificador del grupo, establecido por el servidor.
link	string	URL del grupo.
owning_profile_id	string	Id del dueño del grupo.
access_level	string	Tipo de acceso del grupo como public, private o invite_only.
created	string	Fecha de creación del grupo, establecido por el servidor.
name	string	Nombre del grupo.
description	string	Descripción del grupo.
photo	object	Foto del grupo que contiene una URL del recurso
webpage	string	Sitio web del grupo fuera de Mendeley
disciplines	array	Lista de las disciplinas del cual es parte el grupo.
tags	array	Lista de etiquetas del grup.
role	**string**	Rol del usuario autenticado: owner, admin, normal, follower

- **Operaciones**

GET	POST	PATCH	DELETE
Response Content Type: application/vnd.mendeley-group-list+json application/vnd.mendeley-group+json (Para un grupo) **Parámetros:** ***limit*** (string) - de 1 a 500 (Default 20) ***id:*** ID para obtener un grupo específico	**Response Content Type:** application/vnd.mendeley-group+json **Parameter content type:** application/vnd.mendeley-group+json **Parámetros:** ***body***: Cuerpo solicitud	**Response Content Type:** application/vnd.mendeley-group+json **Parameter content type:** application/vnd.mendeley-group+json **Parámetros:** ***body***: Cuerpo solicitud ***group_id:*** ID del grupo a modificar	**Parámetros:** ***group_id****:* ID del grupo a eliminar

A continuación, se muestra un ejemplo del consumo del recurso grupos, en este caso se observa que el endpoint de este recurso es la versión 2 (v2), por lo que la versión 1 se encuentra actualmente obsoleta, ver Fig. 25.

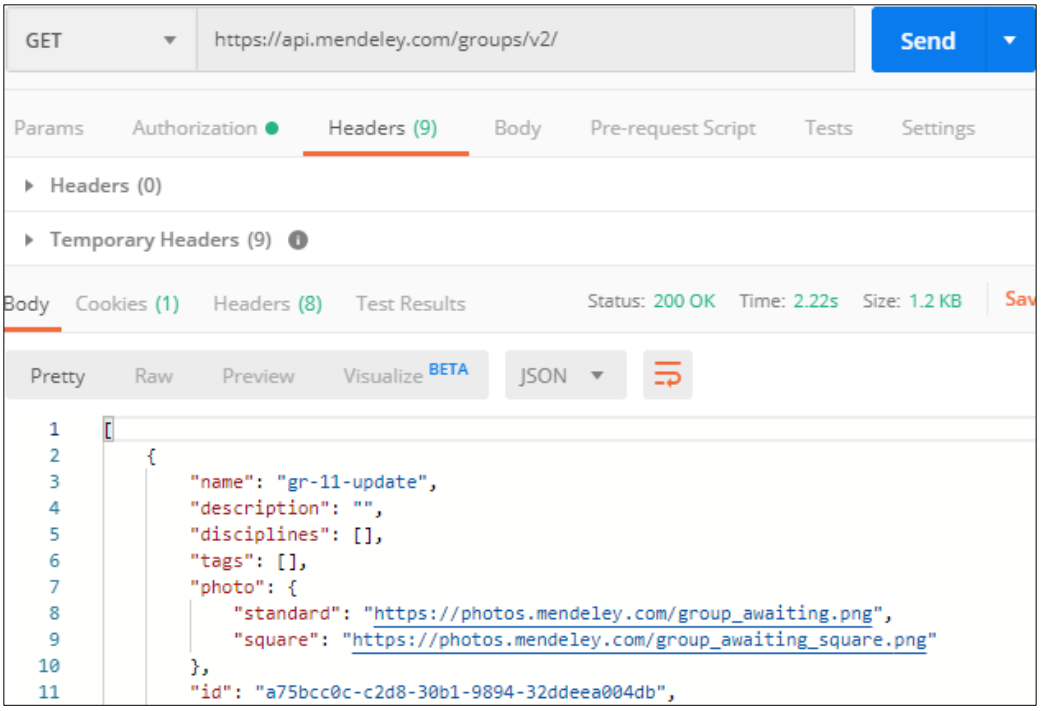


Fig. 25 Consumo del recurso Group en el software Postman

RECURSO ARCHIVOS

- **Documentación del recurso:** https://dev.mendeley.com/methods/ #files
- **URI:** /files
- **Atributos**

id	string	Identificador único del archivo.
file_name	string	Nombre del archivo, generado de los metadatos del doc enlazado
mime_type	string	Especifica la extensión del archivo
filehash	string	SHA1 hash del archivo. Usado para verificar la integridad del archivo.
document_id	string	Id del documento al cual pertenece el archivo
size	integer	Tamaño en bytes del archive.

- **Operaciones**

GET	DELETE
Response Content Type: application/vnd.mendeley-file.1+json **Parámetros:** ***limit*** (string) - de 1 a 500 (Default 20) ***document_id***: Archivos que estan enlazados al documento especificado ***id:*** ID para obtener un archivo específico (Descarga de archivo)	**Parámetros:** ***id:*** ID de la carpeta a eliminar

A continuación, se muestra un ejemplo del consumo del recurso archivo, ver Fig. 26

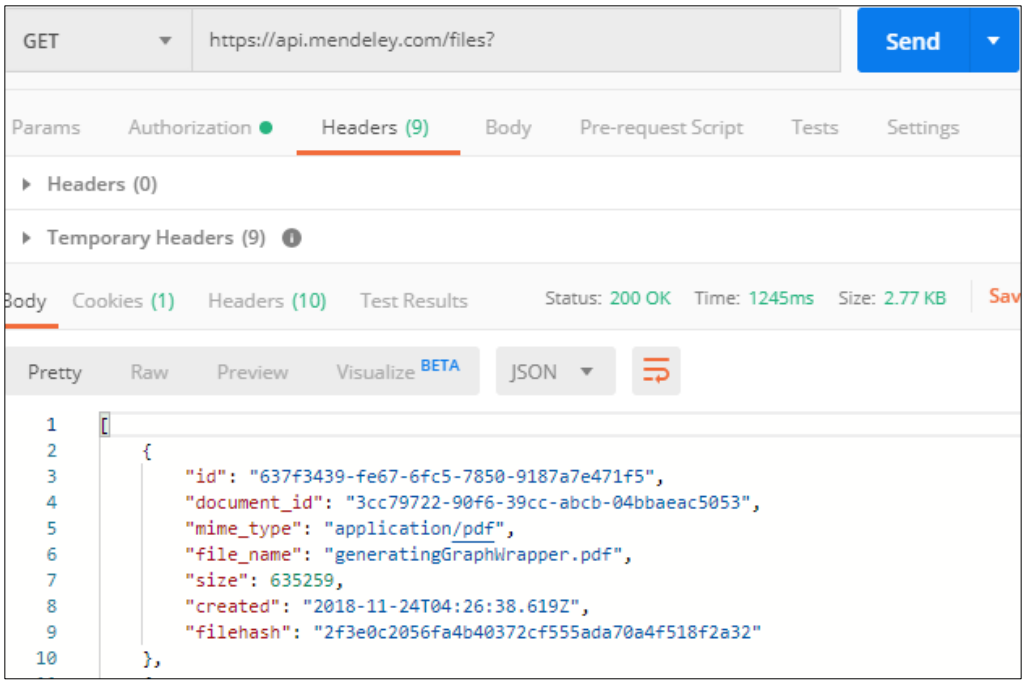


Fig. 26 Consumo del recurso File en el software Postman

- **Retrospectiva del Sprint 1**

a) Reunión de Retrospectiva

Fecha: 22 de julio de 2019

Asistentes: Scrum Master, Product Owner, Team Development

Objetivo: Análisis de aciertos, errores y mejoras

b) Resultado de Retrospectiva

TABLA 23 Retrospectiva Sprint 1

		RETROSPECTIVA
Aciertos (¿Qué salió bien del Sprint?)	-	Los insumos producto de la investigación de la API son claros y contribuyen a un mejor entendimiento para el desarrollo del envoltorio.
Errores (¿Qué no salió bien del Sprint?)	-	La extensa documentación del API limitan la generación de insumos para un completo entendimiento del API.
Mejoras (¿Qué mejoras se implementará?)	-	Estructurar mejor el proceso investigativo para obtener insumos de mejor calidad y entendimiento.

2.3.2. SPRINT 2

- **Planificación del Sprint 2**

a) Reunión de planificación

Fecha: 22 de julio de 2019

Asistentes: Scrum Master, Product Owner, Team Development

Objetivo: Planificación del Sprint 2 (Creación del Sprint backlog)

b) Sprint Backlog

TABLA 24 Sprint 2 Backlog

PLANIFICACIÓN

Historia de usuario	Fase de desarrollo	Tarea	Estimación de tiempo
OTROS	Desarrollo	Crear el proyecto Node, instalar las dependencias para el servidor, y para graphql; crear los archivos y estructura adecuada del proyecto.	2
	Desarrollo	Implementar el servidor	2
	Pruebas	Pruebas de funcionamiento del servidor	2
HU-02	Desarrollo	Crear un tipo Document con los campos definidos en el archivo de esquema graphql.	2
	Desarrollo	Crear un input de documento, también los tipos document types, document identifier types y el tipo person (tipos secundarios) en el esquema graphql.	2
	Desarrollo	Crear un tipo Query con la operación listar y un tipo Mutation con las operaciones crear, actualizar y eliminar; también para las operaciones de los tipos secundarios en el esquema graphql.	3
	Desarrollo	Implementar los resolutores para las operaciones del recurso documento (consumo del API-REST Mendeley).	5
	Pruebas	Pruebas de concepto de la implementación	2
HU-03	Desarrollo	Crear un tipo Folder con los campos definidos en el archivo de esquema graphql.	2
	Desarrollo	Crear un input de folder en el esquema graphql.	2
	Desarrollo	Crear un tipo Query con la operación listar y un tipo Mutation con las operaciones crear, actualizar y eliminar.	2
	Desarrollo	Implementar los resolutores para las operaciones del recurso carpeta (consumo del API-REST Mendeley).	5
	Pruebas	Pruebas de concepto de la implementación	1
Eventos	Planificación	Detallar las tareas a realizar en el Sprint actual	4
	Revisión	Revisar los resultados del desarrollo del Sprint	3
	Retrospectiva	Analizar los resultados del Sprint	1
		TOTAL HORAS	**40**

- **Revisión del Sprint 2**

a) Reunión de revisión

Fecha: 12 de agosto de 2019

Asistentes: Scrum Master, Product Owner, Team Development

Resultado: Revisión del desarrollo del incremento del producto.

b) Incremento del producto potencialmente entregable

OPERACIONES DE DOCUMENTOS

Crear Documento

Para crear un documento se realiza una mutación llamando a "*createDocument*", en donde recibe un parámetro tipo "*DocumentInput*"; y como salida puede obtener un tipo "*Document*"

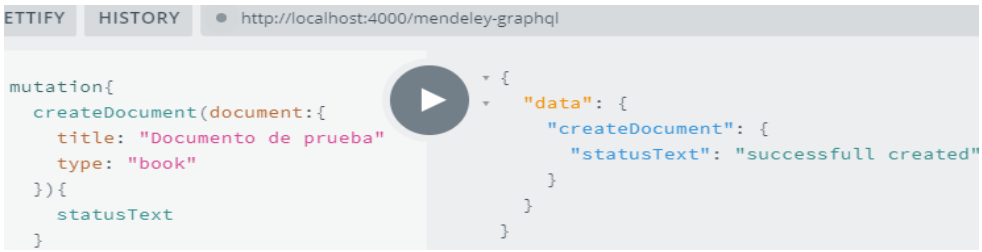


Fig. 27 Ejemplo de estructura para crear un documento

Listar Documentos

Para listar de documentos, permite especificar parámetros de entrada para obtener un listado personalizado; llamando a la consulta "*documents*", se obtiene como salida un tipo "*Document*". La consulta "*document*" devuelve un documento específico enviando un parámetro "*id*".

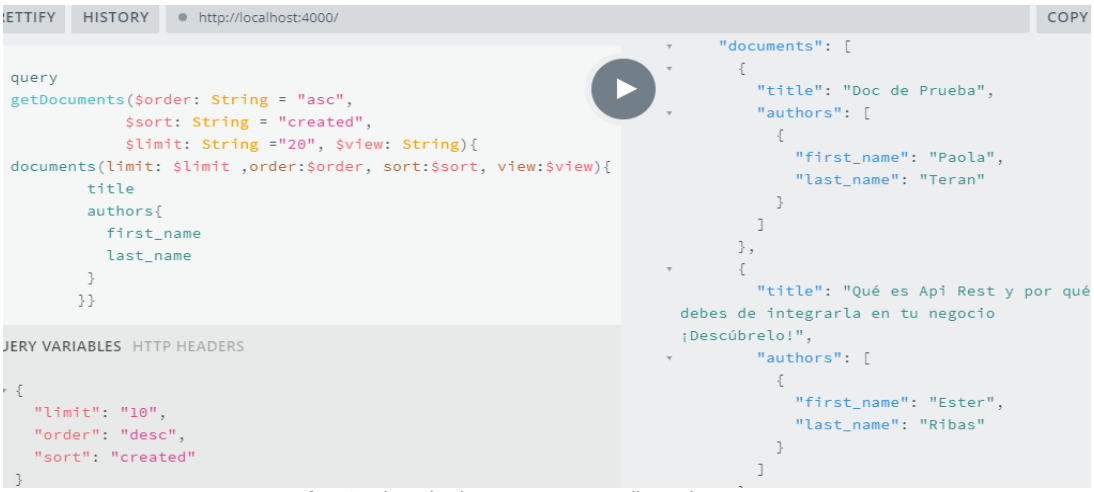


Fig. 28 Ejemplo de estructura para listar documentos

Actualizar Documento

La actualización de un documento requiere de un parámetro "*id*" y un tipo "*DocumentInput*" que contiene la información a actualizarse

Fig. 29 Ejemplo de estructura para actualizar un documento

Eliminar Documento

La eliminación de un documento requiere de un parámetro "*id*"; como salida se obtiene un "*status*" de verdadero o falso, según el éxito o fracaso de la operación.

Fig. 30 Ejemplo de estructura para eliminar un documento

Para las operaciones del recurso "*Carpetas*", las estructuras de petición son similares, con ciertas variaciones en los parámetros de entrada, y estructura de tipos de entrada y salida, etc.

- **Retrospectiva del Sprint 2**

a) Reunión de Retrospectiva

Fecha: 12 de agosto de 2019

Asistentes: Scrum Master, Product Owner, Team Development

Objetivo: Análisis de aciertos, errores y mejoras

b) Resultado de Retrospectiva

TABLA 25 Retrospectiva Sprint 2

		RETROSPECTIVA
Aciertos (¿Qué salió bien del Sprint?)	-	Facilidad de realización de pruebas de concepto
Errores (¿Qué no salió bien del Sprint?)	-	Subestimación del tiempo de desarrollo.

	- Comprensión limitada acerca de las tecnologías a utilizarse para el desarrollo.
Mejoras (¿Qué mejoras se implementará?)	- Optimización del tiempo en base a un software de control de tiempos. - Mayor comprobación y comprensión de tecnologías y herramientas a usar.

2.3.3. SPRINT 3

- **Planificación del Sprint 3**

a) Reunión de planificación

Fecha: 12 de agosto de 2019

Asistentes: Scrum Master, Product Owner, Team Development

Objetivo: Planificación del Sprint 3 (Creación del Sprint backlog)

b) Sprint Backlog

TABLA 26 Sprint 3 Backlog

PLANIFICACIÓN

Historia de usuario	Fase de desarrollo	Tarea	Estimación de tiempo
HU-04	Desarrollo	Crear un tipo Group y un tipo Photo en el archivo de esquema graphql con los campos definidos.	2
	Desarrollo	Crear un input de groups y photo en el esquema graphql.	2
	Desarrollo	Crear un tipo Query con la operación listar y un tipo Mutation con las operaciones crear, actualizar y eliminar.	3
	Desarrollo	Implementar los resolutores para las operaciones del recurso grupo (consumo del API-REST Mendeley).	5
	Pruebas	Pruebas de concepto de la implementación	2
HU-05	Desarrollo	Crear un tipo File en el archivo de esquema graphql con los campos definidos	2
	Desarrollo	Crear un tipo Query con la operación listar y un tipo Mutation con la operación eliminar.	3
	Desarrollo	Implementar los resolutores para las operaciones del recurso carpeta (consumo del API-REST Mendeley).	5
	Pruebas	Pruebas de concepto de la implementación	2
OTROS	Desarrollo	Manejo de errores de la API	4
	Pruebas	Pruebas del manejo de errores	2
Eventos	Planificación	Detallar las tareas a realizar en el Sprint actual	4
	Revisión	Revisar los resultados del desarrollo del Sprint	3
	Retrospectiva	Analizar los resultados del Sprint	1
		TOTAL HORAS	**40**

- **Revisión del Sprint 3**

a) Reunión de revisión

Fecha: 02 de septiembre de 2019

Asistentes: Scrum Master, Product Owner, Team Development

Resultado: Revisión del desarrollo del incremento del producto.

b) Incremento del producto potencialmente entregable

Crear Grupo

La creación de un grupo se hace mediante una mutación llamando a la función "*createGroup*", en donde recibe un parámetro tipo "*GroupInput*"; y como salida puede obtener un tipo "*Group*".

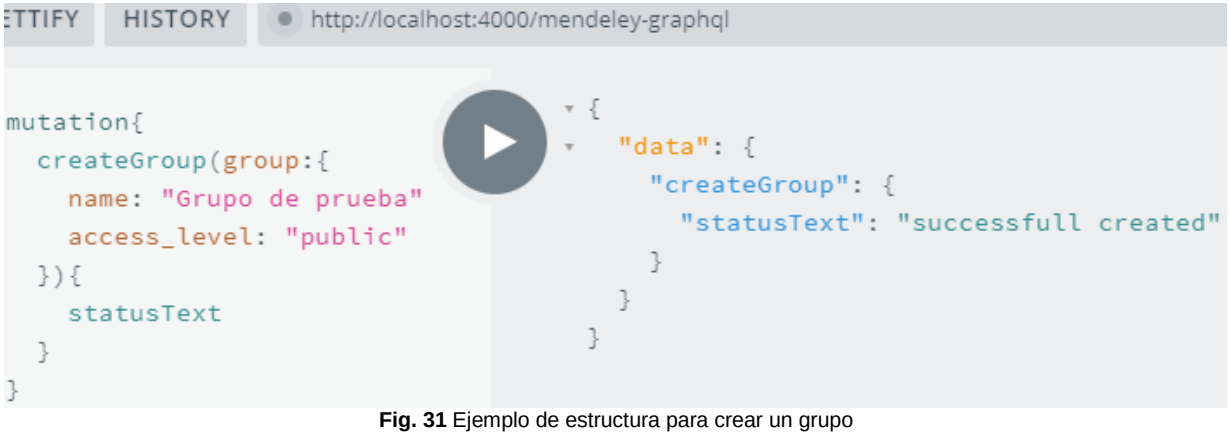


Fig. 31 Ejemplo de estructura para crear un grupo

Listar Grupos

Para el listado del recurso grupos, permite enviar un parámetro de límite de grupos; llamando a la consulta "*groups*", se obtiene como salida un tipo "*Group*". Para obtener un grupo específico, es necesario agregar un parámetro "*id*", de esta manera se puede filtrar el grupo perteneciente al id proporcionado.

Fig. 32 Ejemplo de estructura para listar grupos con una lista de carpetas de cada grupo

Actualizar Grupo

La actualización de un grupo requiere de un parámetro "*id*" y un tipo "*GroupInput*" que contiene la información a actualizarse

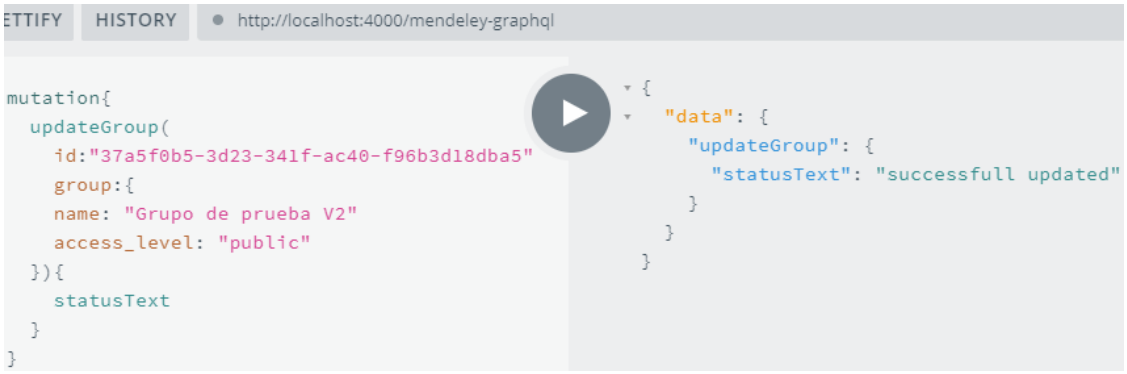


Fig. 33 Ejemplo de estructura para actualizar un grupo

Eliminar Grupo

La eliminación de un grupo requiere de un parámetro obligatorio "*id*" y obteniendo como salida se un campo llamado "*status*" con valor de verdadero o falso, según el éxito o fracaso de la operación.

Fig. 34 Ejemplo de estructura para eliminar un grupo

Descargar Archivo

La descarga de un archivo requiere de un parámetro "*id*"; como salida se obtiene un "*status*" con el mensaje de confirmación o fracaso de la operación, y un campo con la url del archivo.

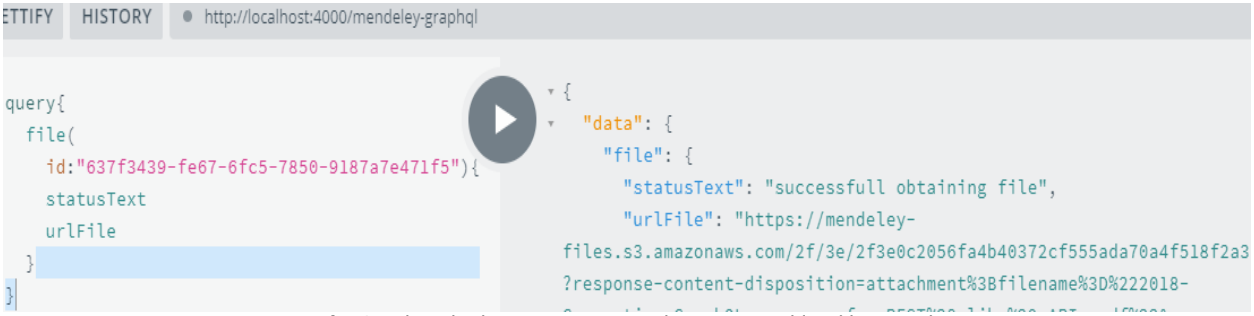


Fig. 35 Ejemplo de estructura para obtener un archivo (descarga)

- **Retrospectiva del Sprint 3**

a) Reunión de Retrospectiva

Fecha: 02 de septiembre de 2019

Asistentes: Scrum Master, Product Owner, Team Development

Objetivo: Análisis de aciertos, errores y mejoras

b) Resultado de Retrospectiva

TABLA 27 Retrospectiva Sprint 3

		RETROSPECTIVA
Aciertos (¿Qué salió bien del Sprint?)	-	Facilidad para demostrar funcionalidades del desarrollo.
Errores (¿Qué no salió bien del Sprint?)	-	Las operaciones para el recurso File requirieron una investigación adicional ya que se manejan diferente.
Mejoras (¿Qué mejoras se implementará?)	-	Investigación antecedente de funcionalidades difíciles de implementar en relación al recurso File.

2.3.4. SPRINT 4

- **Planificación del Sprint 4**

a) Reunión de planificación

Fecha: 02 de septiembre de 2019

Asistentes: Scrum Master, Product Owner, Team Development

Objetivo: Planificación del Sprint 4 (Creación del Sprint backlog)

b) Sprint Backlog

TABLA 28 Sprint 4 Backlog

PLANIFICACIÓN

Historia de usuario	Fase de desarrollo	Tarea	Estimación de tiempo
HU-10	Diseño	Diseñar la estructura html de la página de inicio (Portal de presentación del envoltorio).	5
	Diseño	Diseñar la página html de información de uso de token	3
	Pruebas	Realizar las pruebas de las implementaciones anteriores.	2
	Documentación	Realizar un manual de documentación en formato PDF	6
HU-06	Desarrollo	Crear una aplicación Mendeley en el portal de desarrollo del gestor bibliográfico.	1
	Diseño	Crear un enlace en la página de inicio del envoltorio hacia la página de inicio de sesión de Mendeley adjuntando las credenciales de la aplicación Mendeley.	1
	Desarrollo	Capturar el campo “code” de la URL de redirección que brinda el proceso de autorización de Mendeley y guardarlo en una variable.	2
	Diseño	Implementar una ventana de alerta en donde informe que se guardará el token generado en una cookie.	1
	Desarrollo	Intercambiar el valor “code” por un token de acceso si el usuario aceptó la ventana de alerta, realizando una solicitud al EndPoint de intercambio de Mendeley y guardarlo en una variable.	6
	Desarrollo	Crear y guardar una cookie con el valor del token de acceso generado.	2
	Desarrollo	Implementar el proceso de eliminación de la cookie mediante una ventana de alerta.	1
	Pruebas	Realizar pruebas de la implementación	2
Eventos	Planificación	Detallar las tareas a realizar en el Sprint actual	4
	Revisión	Revisar los resultados del desarrollo del Sprint	3
	Retrospectiva	Analizar los resultados del Sprint	1
		TOTAL HORAS	**40**

- **Revisión del Sprint 4**

a) Reunión de revisión

Fecha: 23 de septiembre de 2019

Asistentes: Scrum Master, Product Owner, Team Development

Resultado: Revisión del desarrollo del incremento del producto.

b) Incremento del producto potencialmente entregable

Portal de presentación del envoltorio

Página de inicio del envoltorio, donde consta el enlace al inicio de sesión de Mendeley, además de enlaces a GraphQL Playground y GraphiQL.

Nota: Los enlaces a los repositorios de GitHub no están funcionales, estos enlaces se los implementará en un posterior Sprint, además el manual de documentación de la API se encuentra disponible por medio de la descarga.

Fig. 36 Portal de desarrollo del API GraphQL Mendeley

Interfaz gráfica del inicio de sesión Mendeley

Página de inicio de sesión de Mendeley adjuntando credenciales de la app Mendeley

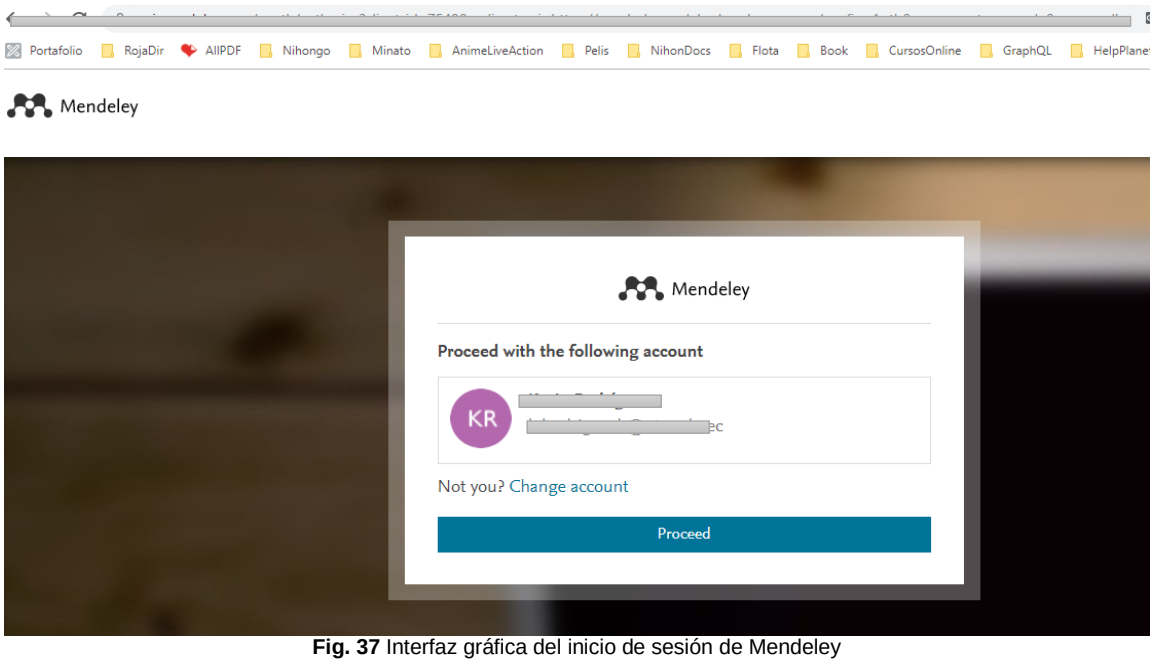


Fig. 37 Interfaz gráfica del inicio de sesión de Mendeley

Confirmación de uso de token.

Ventana de confirmación para crear y guardar una cookie con el valor del token generado, al confirmar este mensaje, la aplicación intercambia el valor *code* por un token de acceso generado por sus credenciales de Mendeley.

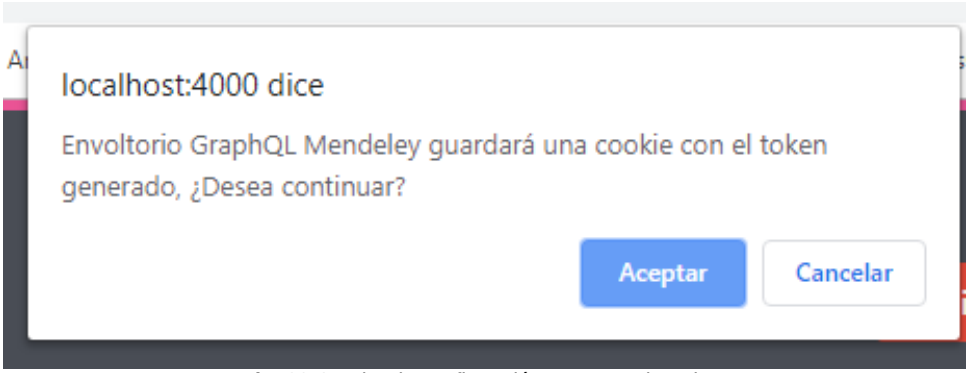


Fig. 38 Cuadro de confirmación para guardar token

Página de confirmación de uso de token

Página informativa que muestra cómo se usará su token, además tiene las acciones de eliminar el token u obtener uno nuevo. También le muestra las herramientas que puede usar para probar la API.

Fig. 39 Página de confirmación de uso de token

Eliminar token

Acción que elimina el token y la cookie que lo contiene.

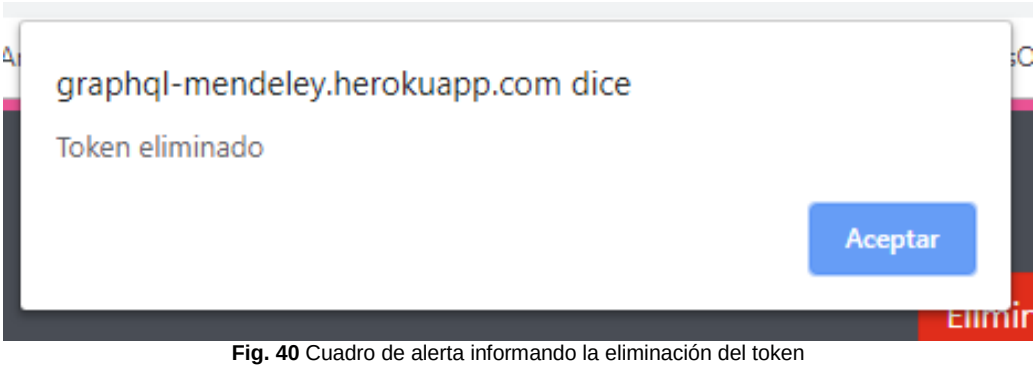


Fig. 40 Cuadro de alerta informando la eliminación del token

- **Retrospectiva del Sprint 4**

a) Reunión de Retrospectiva

Fecha: 23 de septiembre de 2019

Asistentes: Scrum Master, Product Owner, Team Development

Objetivo: Análisis de aciertos, errores y mejoras

b) Resultado de Retrospectiva

TABLA 29 Retrospectiva Sprint 4

RETROSPECTIVA		
Aciertos (¿Qué salió bien del Sprint?)	-	Satisfacción alta de los resultados obtenidos
Errores (¿Qué no salió bien del Sprint?)	-	Fue necesario una investigación adicional acerca de la generación de cookies.
	-	Complejidad para empatar el proceso de obtención del token de acceso y la generación de la cookie.
Mejoras (¿Qué mejoras se implementará?)	-	Realizar una investigación completa y profunda antes de empezar con el desarrollo de ciertas actividades complejas.
	-	Realizar pruebas de generación y obtención de cookies en base a los requisitos del proyecto.

2.3.5. SPRINT 5

- **Planificación del Sprint 5**
- **Reunión de planificación**

Fecha: 23 de septiembre de 2019

Asistentes: Scrum Master, Product Owner, Team Development

Objetivo: Planificación del Sprint 5 (Creación del Sprint backlog)

- **Sprint Backlog**

TABLA 30 Sprint 5 Backlog

PLANIFICACIÓN			
Historia de usuario	**Fase de desarrollo**	**Tarea**	**Estimación de tiempo**
OTROS	Desarrollo	Crear el proyecto cliente React, instalar las dependencias necesarias (front-end y graphql); crear los archivos y estructura adecuada del proyecto.	2
	Desarrollo	Crear las rutas respectivas de navegación	2
	Pruebas	Pruebas de funcionamiento del proyecto.	1
H7	Diseño	Crear una barra de navegación con el logo de Mendeley, un menú de navegación para mostrar usuario y eliminar token, links a la API GraphQL, repositorio GitHub.	2
	Diseño	Crear una ventana emergente que servirá para visualizar la información del usuario que inicio sesión.	2
	Desarrollo	Obtener la información del usuario que inicio sesión y desplegarla en la venta emergente.	2

	Diseño	Diseño de paneles (componente base) para los recursos de carpetas, grupos, documentos y detalle de documento.	3
	Diseño	Crear un pie de página que muestre texto del proyecto	1
	Pruebas	Realizar las pruebas de la implementación.	1
H8	Diseño	Crear el diseño del componente interno de grupos y carpetas.	2
	Desarrollo	Obtener el componente Query y estructurar la consulta para la obtención del listado de carpetas y grupos	1
	Desarrollo	Obtener la lista de carpetas y grupos con Apollo Client y conectar con el componente gráfico respectivo.	3
	Desarrollo	Desarrollar la inserción de una nueva carpeta y de un nuevo grupo en una ventana emergente, validando los tipos de datos correspondientes.	3
	Desarrollo	Implementar la actualización de carpetas y grupos en una ventana emergente	3
	Desarrollo	Implementar la eliminación de carpetas y grupos	2
	Pruebas	Pruebas de concepto de la implementación	2
Eventos	Planificación	Detallar las tareas a realizar en el Sprint actual	4
	Revisión	Revisar los resultados del desarrollo del Sprint	3
	Retrospectiva	Analizar los resultados del Sprint	1
		TOTAL HORAS	**40**

- **Revisión del Sprint 5**

a) Reunión de revisión

Fecha: 14 de octubre de 2019

Asistentes: Scrum Master, Product Owner, Team Development

Resultado: Revisión del desarrollo del incremento del producto.

b) Incremento del producto potencialmente entregable

Vista inicial de la prueba de concepto

Por defecto, en la pantalla inicial, la aplicación muestra la barra de navegación con el logo de Mendeley, links a herramientas de la API GraphQL como GraphQL Playground y GraphiQL, al repositorio GitHUb y opciones de ver perfil y eliminar token. También muestra el listado de carpetas y grupos y un cuadro de bienvenida

Fig. 41 Vista de inicio de la prueba de concepto

Cuadro emergente de información del usuario actual

Modal con información del usuario logueado, con campos de nombre, correo electrónico, estado académico y un enlace a la cuenta en la red social de Mendeley.

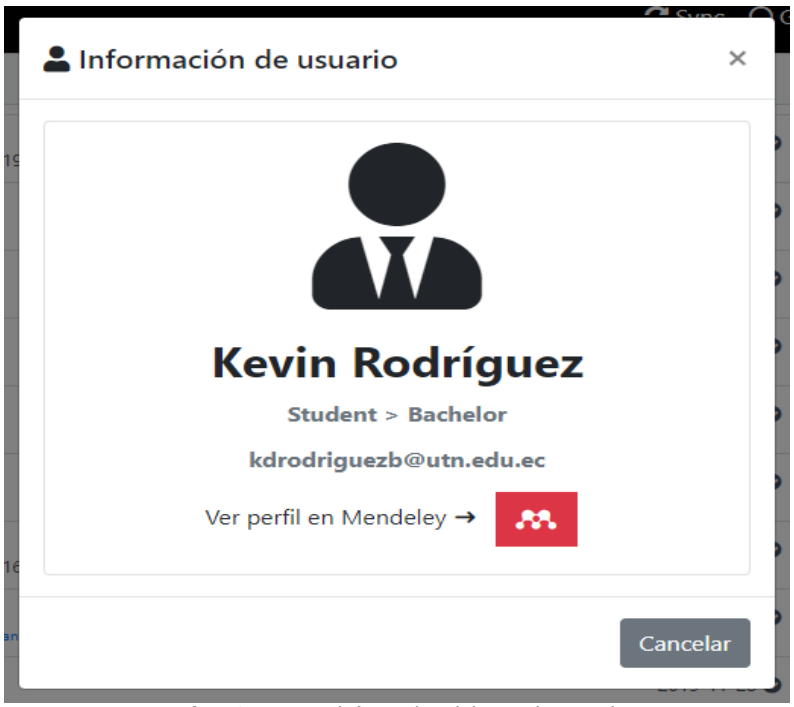


Fig. 42 Ventana informativa del usuario actual

Componentes de listado de carpetas y grupos

Diseño de los componentes de carpetas y grupos conectado con los componentes Query que obtienen las listas obtenidas de la API.

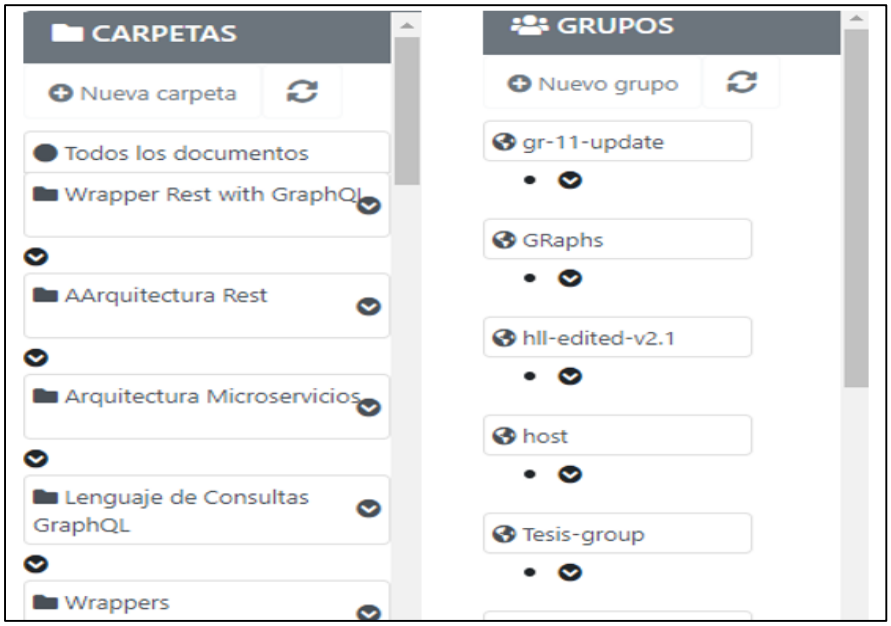


Fig. 43 Componentes carpetas y grupos

Componente de creación de nuevo grupo

Ventana emergente para la creación de un nuevo grupo, especificando los datos necesarios.

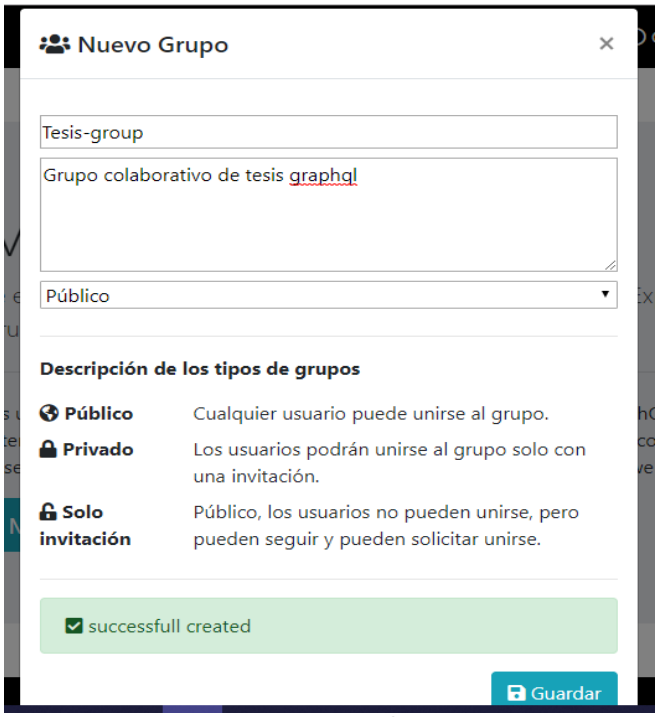


Fig. 44 Componente de creación de nuevo grupo

Componente de creación de nueva carpeta

La creación de una nueva carpeta solo se pide el nombre de la carpeta como dato de entrada.

Fig. 45 Componente de creación de nueva carpeta

Componente de edición de grupo

El componente de edición de grupo, permite actualizar el nombre y descripción del grupo, adicional le da un link para modificar el recurso desde la aplicación oficial de Mendeley.

Nota: La edición de carpeta muestra un proceso similar, pero solo permite editar el nombre.

Fig. 46 Componente de edición de grupo

Componente de eliminación de carpeta

El componente de eliminación de carpeta muestra un cuadro en donde le pide confirmar la eliminación del recurso.

Nota: La eliminación de un grupo tiene el mismo proceso y vista, adecuada al tipo de recurso grupo.

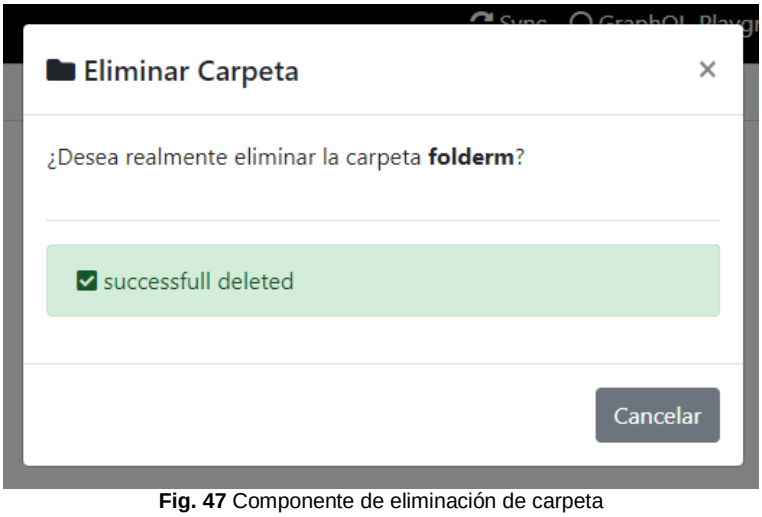


Fig. 47 Componente de eliminación de carpeta

- **Retrospectiva del Sprint 5**

a) Reunión de Retrospectiva

Fecha: 14 de octubre de 2019

Asistentes: Scrum Master, Product Owner, Team Development

Objetivo: Análisis de aciertos, errores y mejoras

b) Resultado de Retrospectiva

TABLA 31 Retrospectiva Sprint 5

	RETROSPECTIVA
Aciertos (¿Qué salió bien del Sprint?)	- La conexión del servidor con el cliente fue satisfactorio; se obtuvo todas las llamadas a la API satisfactoriamente.
Errores (¿Qué no salió bien del Sprint?)	- El poco conocimiento de desarrollo front-end complico un cierto grado la creación de las vistas de los componentes. - Dificultad de enlazar el listado de carpetas y grupos con un componente con estilo árbol.
Mejoras (¿Qué mejoras se implementará?)	- Para el desarrollo front-end, es necesario tener conocimientos básicos en esta sub-área, la capacitación permanente facilita a futuro el inconveniente de la falta de conocimiento. - Búsqueda de ejemplos de recursos que se muestren en un estilo de árbol.

2.3.6. SPRINT 6

- **Planificación del Sprint 6**

a) Reunión de planificación

Fecha: 14 de octubre de 2019

Asistentes: Scrum Master, Product Owner, Team Development

Objetivo: Planificación del Sprint 6 (Creación del Sprint backlog)

b) Sprint Backlog

TABLA 32 Sprint 6 Backlog

PLANIFICACIÓN

Historia de usuario	Fase de desarrollo	Tarea	Estimación de tiempo
HU-10	Desarrollo	Subir los proyectos servidor y cliente a repositorios GitHub.	2
	Diseño	Diseñar botones de enlace a los repositorios GitHub	1
	Desarrollo	Desplegar los proyectos de cliente y servidor en la Heroku. Validar ambiente de producción.	3
	Pruebas	Pruebas de funcionamiento de las implementaciones.	1
HU-9	Diseño	Crear el diseño del componente interno de documentos y detalle de documento.	2
	Desarrollo	Obtener el componente Query y estructurar la consulta para la obtención del listado de documentos.	1
	Desarrollo	Obtener la lista filtrada de documentos con Apollo Client y conectar con el componente gráfico respectivo.	2
	Desarrollo	Obtener la estructura de un documento con Apollo Client y conectar con el componente gráfico respectivo.	2
	Desarrollo	Desarrollar la inserción de un nuevo documento en una ventana emergente, validando los tipos de datos correspondientes.	2
	Desarrollo	Implementar la actualización de documentos en una ventana emergente.	2
	Desarrollo	Implementar la eliminación de un documento.	1
	Desarrollo	Obtener el componente Query y estructurar la consulta para la obtención de los archivos de un documento y conectar con un componente gráfico.	2
	Desarrollo	Implementar la visualización y descarga de archivos	2
	Desarrollo	Implementar la eliminación de un archivo	1
	Pruebas	Realizar las pruebas de la implementación.	1
HU-11	Arquitectura	Creación de base de datos de actividad basado en el modelo del Sprint 0 y realizar consultas de actividad.	2
	Diseño	Diseñar la página html de actividad de la API	3
	Desarrollo	Conectar el servidor con la base de datos para obtener las consultas base de la actividad	2
	Diseño	Mediante una motor de plantillas para nodejs, mostrar los resultados en tablas de la consulta general de la API y los usuarios que han usado la API.	3
	Diseño	Diseñar cartillas que muestren los resultados de las consultas de recursos, operaciones REST y operaciones GraphQL.	3

	Diseño	Crear componente de porcentaje, uso cronológico y gráfica de uso de las operaciones GraphQL basado en las consultas a la base de datos.	3
	Pruebas	Pruebas de la implementación	1
Eventos	Planificación	Detallar las tareas a realizar en el Sprint actual	4
	Revisión	Revisar los resultados del desarrollo del Sprint	3
	Retrospectiva	Analizar los resultados del Sprint	1
		TOTAL HORAS	**50**

- **Revisión del Sprint 6**

a) Reunión de revisión

Fecha: 04 de noviembre de 2019

Asistentes: Scrum Master, Product Owner, Team Development

Resultado: Revisión del desarrollo del incremento del producto.

b) Incremento del producto potencialmente entregable

Repositorios GitHub y despliegue de los proyectos

Proyecto	Nombre	GitHub	Heroku
Envoltorio	Graphql-mendeley-wrapper	https://github.com/kdrodriguez/graphql-mendeley-wrapper	https://graphql-mendeley.herokuapp.com/
Prueba de concepto	test-concept-gmw	https://github.com/kdrodriguez/test-concept-gmw	https://test-concept-gmw.herokuapp.com/

Componente de documentos y detalle de documento

Diseño de los componentes del recurso de documentos y el detalle de documento, el cual es una filtración de un documento seleccionado, estos dos componentes conectados con los componentes Query.

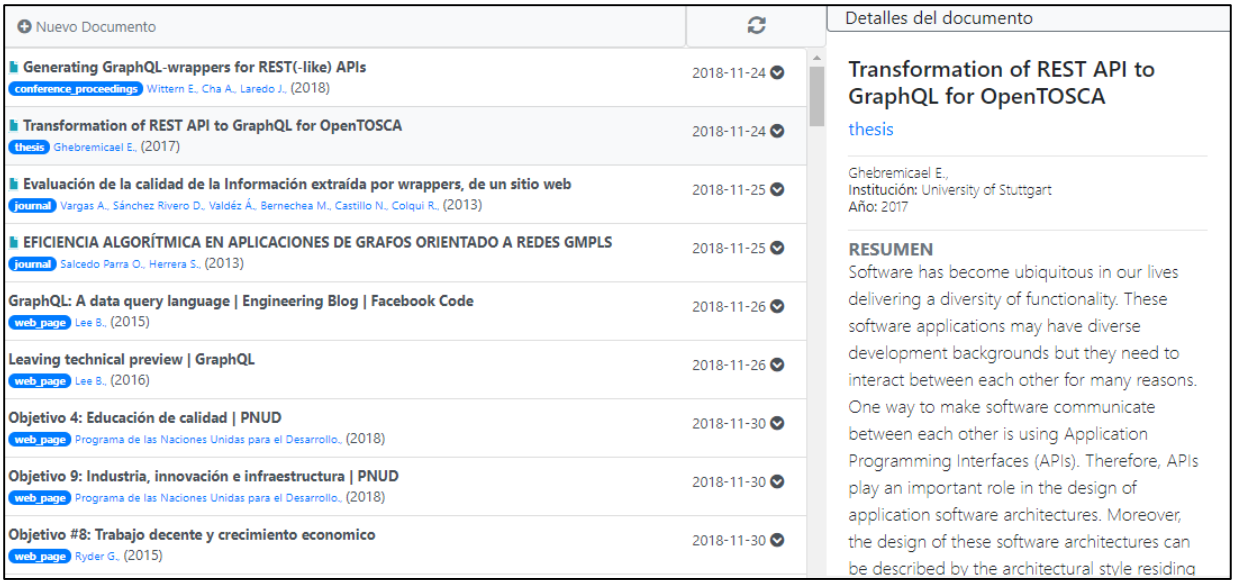


Fig. 48 Componente de listado de documentos y detalle de documento

Componente de archivo(s) de un documento

Filtrado de archivos de un documento específico y opciones de visualización, descarga o eliminación del mismo.

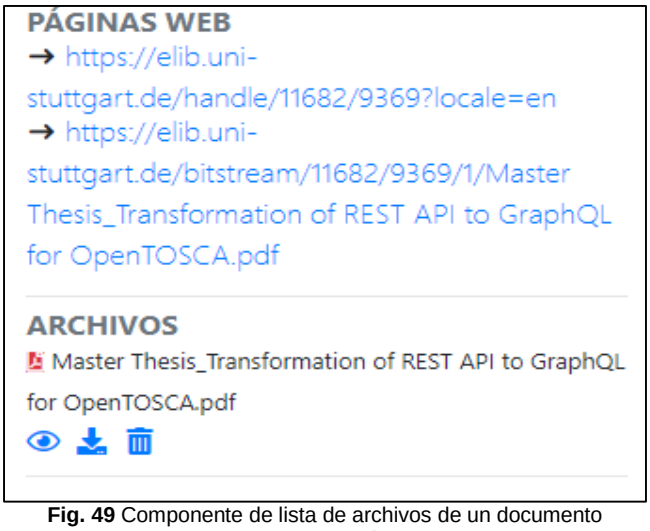


Fig. 49 Componente de lista de archivos de un documento
Fuente: Elaboración propia

Visualización en línea de un archivo

Implementación del lector de archivos de Mendeley

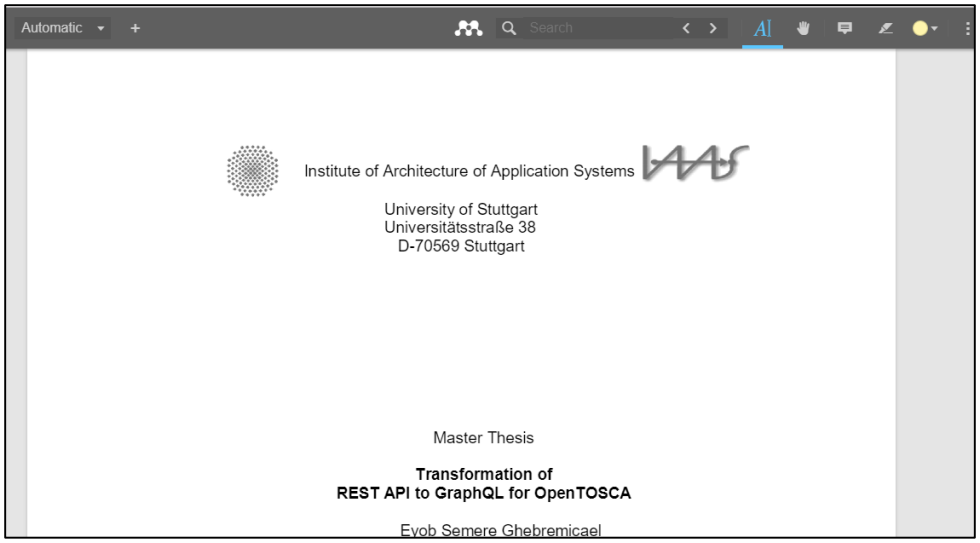


Fig. 50 Visualización en línea de un archivo

Componte de nuevo documento

Ventana emergente para la creación de un nuevo documento, especificando los datos necesarios.

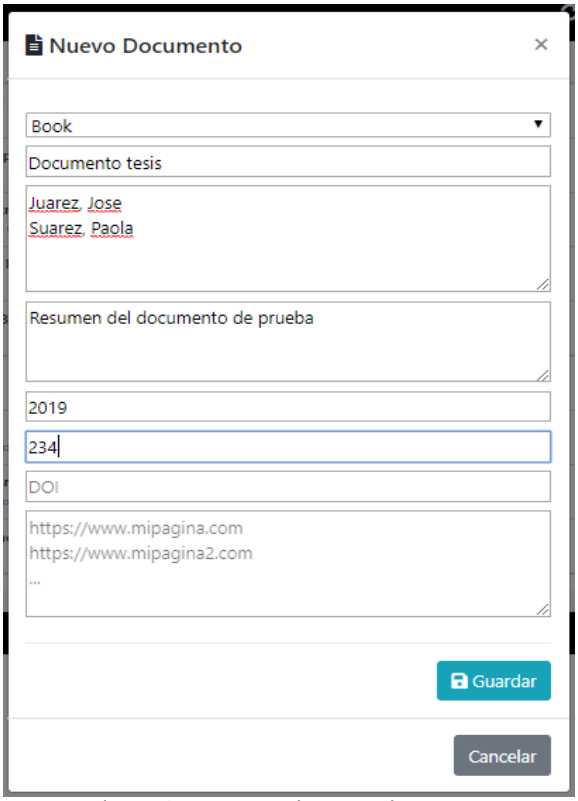


Fig. 51 Componente de nuevo documento

Componente de edición de un documento

Ventana emergente para la edición de un documento, especificando los datos necesarios.

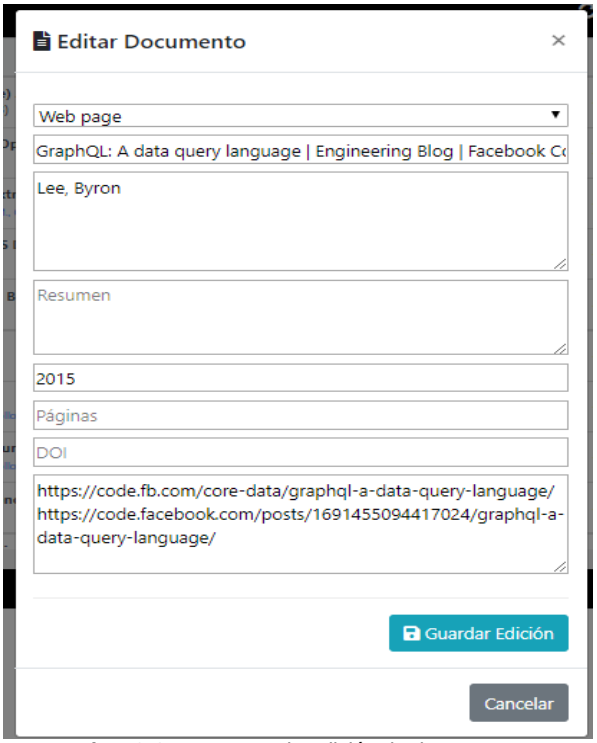


Fig. 52 Componente de edición de documento
Fuente: Elaboración propia

Componente de eliminación de un componente

El componente de eliminación de documento muestra un cuadro en donde le pide confirmar la eliminación del recurso.

Nota: La eliminación de un archivo tiene el mismo proceso y vista, adecuada al tipo de recurso archivo.

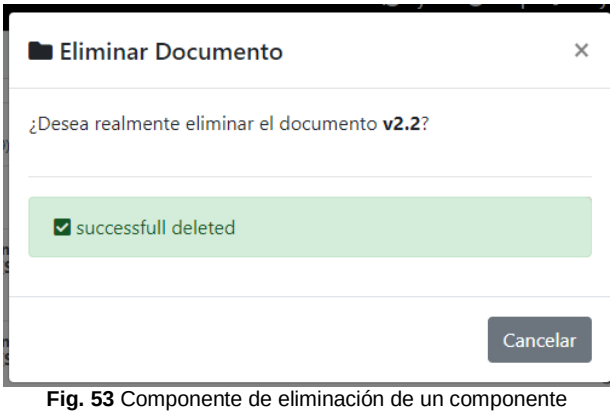


Fig. 53 Componente de eliminación de un componente

Actividad General uso envoltorio

Tabla que muestra cada una de las peticiones realizadas a la API.

Nota: La tabla de usuarios muestra que usuarios han realizado peticiones a la API.

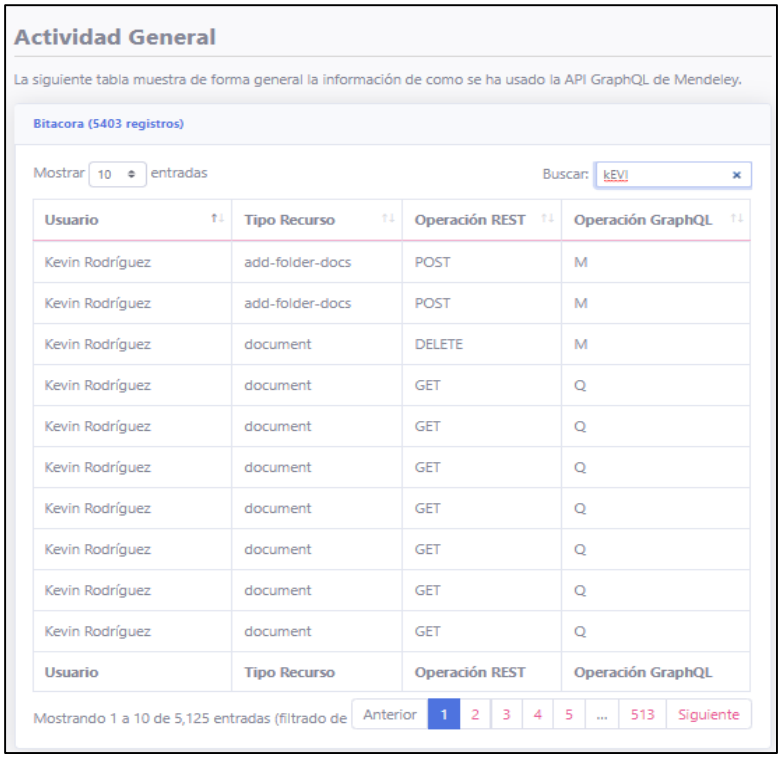


Fig. 54 Tabla de uso general de la API

Detalle de uso de recursos

Detalle de número de veces que se accedió al recurso por medio del envolvimiento a la API REST de Mendeley.

Fig. 55 Cartillas de detalle de uso de recursos de Mendeley

Detalle de operaciones REST envueltas.

Detalle del número de veces que se realizó una operación HTTP a la API REST de Mendeley.

Fig. 56 Cartillas de detalle de operaciones REST AL API Mendeley

Detalle de operaciones GraphQL realizadas.

Detalle del número de veces que se realizó una operación GraphQL y gráfica de porcentaje**.**

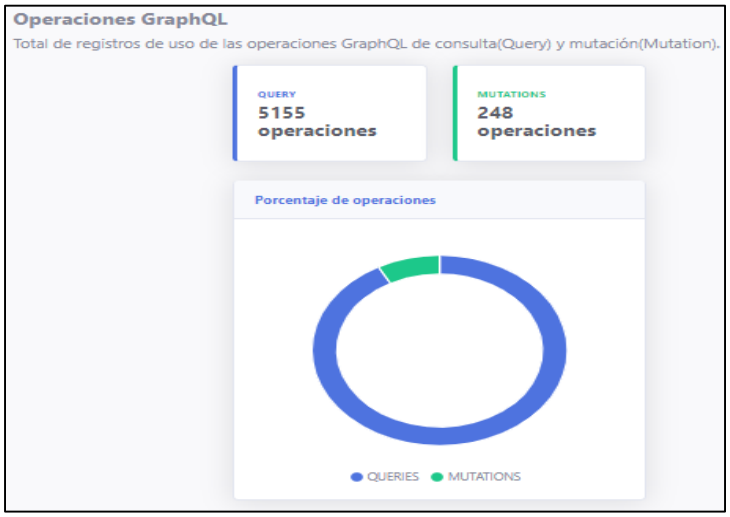


Fig. 57 Cartillas y gráfica de porcentaje de operaciones GraphQL

Cronología de uso de la API.

Tabla y gráfica de uso que muestra la cronología de uso de la API según el número de peticiones por mes del año actual

Fig. 58 Tabla y gráfica de uso cronológico de la API

- **Retrospectiva del Sprint 6**

a) Reunión de Retrospectiva

Fecha: 04 de noviembre de 2019

Asistentes: Scrum Master, Product Owner, Team Development

Objetivo: Análisis de aciertos, errores y mejoras

b) Resultado de Retrospectiva

TABLA 33 Retrospectiva Sprint 6

		RETROSPECTIVA
Aciertos (¿Qué salió bien del Sprint?)	-	Los resultados obtenidos fueron los esperados.
	-	Satisfacción del desarrollo de componentes estadísticos de actividad.
Errores (¿Qué no salió bien del Sprint?)	-	Dificultad en conectar las consultas de la base de datos con el motor de plantillas para pintar en pantalla.
Mejoras (¿Qué mejoras se implementará?)	-	Profundizar en la fusión de un servidor GraphQL y un modelo de navegación en HTML.

2.4. Pruebas de aceptación

Al finalizar el desarrollo del proyecto, es necesario verificar mediante pruebas de aceptación el cumplimiento de las funciones del software. Ver TABLA 34.

TABLA 34 Detalle de pruebas de aceptación

ID-HU	Descripción	Función	Aceptación	
			SI	NO
HU-01	Definición de estructura del software	Modelo documentado del proceso de autenticación del API Mendeley	X	
		Modelo documentado del consumo de los principales recursos Mendeley	X	
		Modelo documentado del consumo de los recursos secundarios de Mendeley	X	
HU-02	Gestión del recurso Documentos	Listar documento(s)	X	
		Crear documento	X	
		Actualizar documento	X	
		Eliminar documento	X	
HU-03	Gestión del recurso Carpetas	Listar carpeta(s)	X	
		Crear carpeta	X	
		Actualizar carpeta	X	
		Eliminar carpeta	X	

HU-04	Gestión del recurso Grupos	Listar grupo(s)	X
		Crear grupo	X
		Actualizar grupo	X
		Eliminar grupo	X
HU-05	Gestión del recurso Archivos	Obtener archivo	X
		Eliminar archivo	X
HU-06	Autenticación de API	Autenticar usuario	X
HU-07	Modelo de gestión de la aplicación web.	Mostrar el modelo del cliente web diferenciando los componentes	X
HU-08	Gestión de carpetas y grupos	Listar carpetas (cliente)	X
		Crear carpeta (cliente)	X
		Actualizar carpeta (cliente)	X
		Eliminar carpeta (cliente)	X
		Listar grupos (cliente)	X
		Crear grupo (cliente)	X
		Actualizar grupo (cliente)	X
		Eliminar grupo (cliente)	X
HU-09	Gestión de documentos y archivos	Listar documentos (cliente)	X
		Crear documento (cliente)	X
		Actualizar documento (cliente)	X
		Eliminar documento (cliente)	X
		Descargar archivo (cliente)	X
		Eliminar archivo (cliente)	X
HU-10	Portal de desarrollo del envoltorio	Mostrar presentación del envoltorio	X
		Descargar documentación del envoltorio	X
		Mostrar información de uso de token	X
HU-11	Actividad del envoltorio	Mostrar actividad de uso del API	X

Sr. Kevin Rodriguez
Scrum Master

Ing. Antonio Quiña
Product Owner

2.5. ACTA DE ENTREGA RECEPCIÓN

Una vez finalizada la etapa de desarrollo del envoltorio GraphQL de Mendeley y la prueba de concepto, se realizó la revisión final del funcionamiento del mismo y se realizó la entrega del producto al Product Owner.

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

ACTA DE ENTREGA RECEPCIÓN

En las instalaciones de la Universidad Técnica del Norte de la ciudad de Ibarra, a los 28 días del mes de febrero de 2020, el Sr. Kevin David Rodríguez Barahona hace la entrega formal del software "API GraphQL de Mendeley" y la prueba de concepto "GRAPHQL - MENDELEY", al Ing Antonio Quiña, Product Owner del proyecto.

El software "API GraphQL de Mendeley", ofrece los servicios de gestión API de Mendeley de los recursos de Documentos, Carpetas, Grupos y Archivos. El software "GRAPHQL - MENDELEY", ofrece las funcionalidades de gestón de Mendeley de los recursos de Documentos, Carpetas, Grupos y Archivos por medio de una aplicación web.

Productos entregados

- Código fuente de los proyectos desarrollados
- Implementación de los proyectos en Heroku (prototipo).
- Pruebas de funcionalidad, operatividad y aceptación con la tecnología asignada.
- Manual de usuario y técnico.

Atentamente, ENTREGA CONFORME

Sr. Kevin Rodríguez
Tesista
Universidad Técnica del Norte

RECIBI CONFORME:

MSc. Antonio Quiña
DOCENTE FICA UTN

MSc Antonio Quiña
Docente tutor,
Universidad Técnica del Norte

CAPÍTULO 3

Validación e interpretación de resultados

En este capítulo se presenta la validación e interpretación de los resultados de la calidad en uso del API-GraphQL desarrollado como envoltorio de los API-REST de Mendeley basado en las normas ISO/IEC 25000. La estructura del capítulo es la siguiente:

- Definición de la evaluación de calidad en uso.
- Medición del modelo de evaluación.
- Evaluación y resultados del modelo.

3.1. Definición de la evaluación de calidad en uso

En esta sección se definen los elementos necesarios para la evaluación del modelo de calidad en uso:

- Modelo de calidad en uso, basado en el estándar ISO/IEC 25010.
- Diseño de los artefactos de recolección de datos, con los que se medirá el modelo de calidad propuesto, basado en el estándar ISO/IEC 25022 y la encuesta SUS.
- Definición de la muestra que evaluará el modelo de calidad definido.

3.1.1. Modelo de calidad en uso

Los encargados de definir el modelo de calidad en uso fueron el Product Owner y el Scrum Master, los cuales escogieron las características y sub-características de usabilidad y asignaron ponderaciones a cada una de estas, conforme a los requisitos propios del contexto en el que se desenvolvió el presente proyecto, Ver TABLA 35.

TABLA 35 Modelo de calidad en uso

Características (C)	Sub-características (SC)	Peso (C)	Peso (SC)	Criterio de medición
Eficacia	Tareas completas	**40%**	**17%**	Rendimiento del usuario
	Objetivos logrados		**13%**	
	Tareas con errores		**10%**	
Eficiencia	Tiempo de tareas	**30%**	**10%**	Rendimiento del usuario
	Eficiencia del Tiempo		**20%**	
Satisfacción	Utilidad	**30%**	**10%**	Satisfacción del usuario
	Confianza		**11%**	
	Comodidad		**9%**	

3.1.2. Artefactos de recolección de datos

Una vez definido el modelo de calidad en uso, se obtuvo las métricas de medición de las sub-características establecidas en dicho modelo. Para obtener los elementos de medición se diseñó un taller práctico por medio del cual se obtenga información para medir las características de eficacia, eficiencia y la sub-característica de confianza perteneciente a la satisfacción del uso del API GraphQL. Para obtener los elementos de medición de las sub-características de comodidad y utilidad pertenecientes a la satisfacción de uso, se utilizó el cuestionario System Usability Scale (SUS).

- **Taller práctico**

a) Definición del taller

El presente taller recaba los elementos necesarios para medir las características de eficacia, eficiencia y la sub-característica de confianza del envoltorio GraphQL de Mendeley, estos factores correspondientes a un marco de trabajo de calidad en uso basado en la norma ISO/IEC 25000. El taller se aplicará a la muestra previamente seleccionada, los mismos que cuentan con el conocimiento técnico acerca de la tecnología GraphQL.

- **Objetivo**: Evaluar la eficacia, eficiencia y satisfacción del API-GraphQL de Mendeley, desde el punto de vista del investigador en el contexto de un grupo de estudiantes.
- **Variable independiente:** Desarrollo de un API-GraphQL de Mendeley (Envoltorio) que envuelva las API-REST de Mendeley.
- **Variables dependientes:** Eficacia, eficiencia y satisfacción de la calidad de uso.
- **Objetos:** Servicios que presenta el API-GraphQL (Envoltorio).
- **Sujetos:** Estudiantes de la Universidad Técnica del Norte de la Carrera de Ingeniería en Sistemas
- **Test:** Las personas (sujetos) usan técnicas de consumo de APIs (tratamiento) para consumir un servicio del API-GraphQL Mendeley (Objeto)

b) Planificación del taller

- **Formulación de la hipótesis:** En concordancia con la encuesta, la hipótesis se deriva del objetivo.

  H0: Envoltorio GraphQL tiene eficacia, eficiencia y confianza

  H1: Envoltorio GraphQL no tiene eficacia, eficiencia y confianza

- **Selección de los sujetos:** Muestra seleccionada

c) Fases del taller

El taller constará de dos fases las cuales son complementarias, la primera denominada sesión introductoria y la segunda denominada sesión práctica.

Fase 1:

- **Duración**: 15 min
- **Nombre**: Fase de introducción
- **Descripción**: Explicación de conceptos teóricos acerca de la base técnica del proyecto y una introducción acerca del proceso de evaluación.
- **Objetivo**: Dar a conocer e involucrar a los estudiantes sujetos a evaluación en el uso del envoltorio GraphQL de Mendeley.
- **Desarrollo**: Por medio de una presentación digital tipo exposición, se da a conocer el proceso de ejecución del taller, así como de requisitos previos para poder realizarlo, tales como disponer de una cuenta Mendeley que contenga documentos, carpetas, archivos y grupos.

Fase 2

- **Duración**: 50 – 70 min
- **Nombre**: Fase de práctica
- **Descripción**: Desarrollo de la sesión práctica por medio de la propuesta de múltiples tareas técnicas las cuales serán desarrolladas por el grupo de estudiantes.
- **Objetivo**: Obtener la experiencia del usuario evaluado por medio de datos cuantitativos
- **Desarrollo**: Los estudiantes deberán realizar tareas propuestas y registrar los tiempos que se demoran en realizarlas, además podrán registrar una observaciones y quejas si las tuvieren. Las evidencias de la realización de las tareas serán presentadas por medio de capturas de pantalla y su script correspondiente.

- **Encuesta**

La encuesta utilizada para obtener los elementos de medición de usabilidad está basada en System Usability Scale (SUS) que usa un sistema de escalas de respuesta en base a la experiencia del usuario (Kortum & Bangor, 2013). La encuesta utiliza la escala de Likert de 5 niveles, ver TABLA 36, en el caso del valor inverso, se lo usará para las preguntas consideradas negativas compuesta por 10 preguntas específicas que se muestran a continuación.

- Pregunta 1: Creo que usaría esta API frecuentemente
- Pregunta 2: Encontré el API innecesariamente complejo
- Pregunta 3: Creo que el API fue fácil de usar
- Pregunta 4: Creo que necesitaría ayuda de una persona con conocimientos técnicos para usar esta API
- Pregunta 5: Las funciones de este API están bien integradas
- Pregunta 6: Creo que el API es muy inconsistente
- Pregunta 7: Imagino que la mayoría de la gente aprendería a usar esta API en forma muy rápida
- Pregunta 8: Encuentro que el API es muy difícil de usar
- Pregunta 9: Me siento confiado al usar esta API
- Pregunta 10: Necesité aprender muchas cosas antes de ser capaz de usar esta API

TABLA 36 Escala numérica de calificación de encuesta

Criterio de calificación	Valor proporcional	Inversa
Totalmente de acuerdo	5	1
De acuerdo	4	2
Ni de acuerdo ni en desacuerdo	3	3
En desacuerdo	2	4
Totalmente en desacuerdo	1	5

3.1.3. Muestra

Para definir la muestra se utilizó una técnica de muestreo no probabilístico[3] debido a la dificultad de accesibilidad y proximidad que presenta la población objetivo. La técnica utilizada es el muestreo por conveniencia[4], mediante esta técnica se consideró a un grupo de 26 estudiantes de la carrera de sistemas computacionales de la Universidad Técnica del Norte los cuales contaban con el conocimiento técnico necesario para la realización de la evaluación.

3.2. Medición del modelo de evaluación

La especificación de medición de calidad en uso se basa en las normas ISO/IEC 25022, las cuales detallan medidas de calidad que se utilizaran junto con el modelo de calidad definido en la sección 3.1.1 para la evaluación del producto. Ver TABLA 37.

TABLA 37 Criterios de medición de las características de calidad de uso

Característica	Sub-característica	Medición
Eficacia	Tareas completas	***X= A/B**; A= Tareas únicas completadas, B = Tareas únicas intentadas.*
	Objetivos logrados	***X= [(ΣAi) * 0,25] / U**; Ai= Objetivos cumplidos por el usuario i, U= Total usuarios*
	Tareas con errores	***X= 1-(A/B**); A= Tareas con errores, B= Tareas propuestas*
Eficiencia	Tiempo de tareas	***X= ΣAi / U**; Ai = Relación del tiempo total del usuario i con el tiempo experto, U= Total usuarios*
	Eficiencia del Tiempo	***X= ΣAi / U**; Ai= Relación del tiempo de cumplimiento de los objetivos del usuario i con el tiempo experto, U=Total usuarios.*
Satisfacción	Utilidad	***X = S / U**; S= Usuarios satisfechos, U= Total Usuarios*
	Confianza	***X=A/T, C=1-X**; X=%Reclamos, C=% Confianza, A= Quejas presentadas, T= Total usuarios*
	Comodidad	***X=A+B+C+D+E;** A= Totalmente de acuerdo, B= De acuerdo, C= Ni de acuerdo ni en desacuerdo, D= En desacuerdo, E= Totalmente en desacuerdo*

[3] **Muestreo no probabilístico**: Técnica de muestreo que se basa en un proceso que no les permite a todos los sujetos pertenecientes a una población investigada tener las mismas oportunidades de ser seleccionados.

[4] **Muestreo por conveniencia**: Muestreo que permite seleccionar sujetos en base a su accesibilidad y proximidad en relación con el investigador.

El taller y la encuesta se llevaron a cabo exitosamente el día 04 de febrero de 2020 en donde participaron los integrantes de la muestra establecida en la sección 3.1.3 (26 estudiantes). Los resultados obtenidos de este proceso se detallan a continuación.

3.2.1. Resultados del taller práctico

Los resultados del taller proporcionan datos los cuales servirán para ejecutar la fase de evaluación del producto, en la TABLA 38 se observan los resultados generales del mismo.

TABLA 38 Resultados generales del taller

Criterio	Estimación
Tiempo experto de realización del taller	16 min.
Tiempo promedio de realización del taller	33,46 min
Tareas completadas	201 de 208
Objetivos alcanzados	95 de 104
Errores	13
Quejas	3

A continuación, se presenta un resumen basado en los resultados del taller concernientes al tiempo que se demoran los usuarios en realizar cada una de las tareas y objetivos propuestos y la relación de estos con el tiempo experto. Los tiempos y porcentajes de relación se basan en los promedios de los estudiantes de cada tarea y/u objetivo, ver TABLA 39. En algunas tareas y objetivos la relación con el tiempo experto tiene un margen amplio, sin embargo, este margen se puede reducir mediante la repetición aplicando el mismo taller a la misma muestra.

TABLA 39 Resumen de resultados de tareas y objetivos del taller

N° objetivo	N° tarea	Tarea			Objetivo		
		Tiempo	T. experto	% Rel. experto	Tiempo	T. experto	% Rel. experto
Obj. 1	T1	1 min.	1 min.	100%	2 min.	2 min.	100%
	T2	1 min.	1 min.	100%			
Obj. 2	T3	4,12 min	1 min.	24,2%	9,12 min.	3 min.	32,8%
	T4	5 min.	2 min.	40%			
Obj. 3	T5	2,58 min.	2 min.	77,5%	7,54 min.	5 min.	66,3%
	T6	4,96 min.	3 min.	60,4%			
Obj. 4	T7	6,46 min.	3 min.	46,4%	14,81 min	6 min.	40,5%
	T8	8,35 min.	3 min.	35,9%			
TOTAL		**36,46 min**			**36,46 min**		

- **Análisis estadístico de los resultados del taller**

Para medir la confiabilidad del taller práctico, se usó el cálculo del coeficiente alfa de Cronbach el cual mide el grado en el cual las variables están correlacionadas entre sí (Cronbach, 1951). Para este cálculo se utilizó las variables *tiempo de tareas* y *relación experto,* ya que estas dos

variables son cuantitativas a diferencia de las demás variables consideradas categóricas y además son las variables que teóricamente y a juicio propio se prestan para realizar este tipo de cálculo.

Antes de obtener el alfa de Cronbach del taller, se realizó un tratamiento de las variables consideradas para el cálculo. La variable *relación experto* estaba en relación inversa con el *tiempo de tareas*, lo cual consideraba a más tiempo de tareas, menos relación experta (1). La variable *tiempo de tareas* se la escalo dividendo para el máximo del *tiempo de tareas (2).*

$$\boldsymbol{RelacionExperto} = 1 - RelacionExperto \quad (1)$$

$$\boldsymbol{TiempoTareas} = TiempoTareas \,/\, \max(TiempoTareas) \quad (2)$$

Una vez realizado el tratamiento, se realizó el cálculo del coeficiente alfa de Cronbach por medio del cual se obtuvo un coeficiente de 0.97, ver Fig. 59, lo cual según (Taber, 2018), este valor se encuentra en con una valoración de excelente, cuanto más se acerca a 1, mayor confiabilidad. así se concluye que el instrumento es válido.

```
Reliability analysis
Call: psych::alpha(x = setValidacion)

  raw_alpha std.alpha G6(smc) average_r S/N   ase mean   sd median_r
      0.97      0.97    0.94      0.94  31 0.012 0.57 0.19     0.94

 lower alpha upper     95% confidence boundaries
0.94 0.97 0.99

 Reliability if an item is dropped:
                raw_alpha std.alpha G6(smc) average_r S/N alpha se var.r med.r
TiempoTareas         0.94      0.94    0.88      0.94  NA       NA  0.94  0.94
RelacionExperto      0.88      0.94      NA        NA  NA       NA  0.88  0.94

 Item statistics
                 n raw.r std.r r.cor r.drop mean   sd
TiempoTareas    26  0.99  0.98  0.95   0.94 0.67 0.20
RelacionExperto 26  0.98  0.98  0.95   0.94 0.47 0.18
```

Fig. 59 Resultados coeficiente alfa de Cronbach del taller en R Studio

3.2.2. Resultados de la encuesta

La encuesta propuesta obtuvo resultados que se muestran en la TABLA 40, los cuales permitieron medir algunas sub-características del modelo de evaluación. Los resultados de las preguntas pares sufrieron un tratamiento ya que se consideran preguntas negativas, el tratamiento consistió en dar su valor inverso, este valor inverso se muestra en la TABLA 36.

TABLA 40 Tabulación general de la encuesta

	PREGUNTAS									
ESCALA DE RESPUESTA	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**	**P10**
Totalmente de acuerdo	8	2	5	1	9	3	3	4	4	0
De acuerdo	16	7	14	4	13	11	11	11	14	5
Ni de acuerdo ni en desacuerdo	1	12	5	12	4	6	11	6	8	10
En desacuerdo	1	5	1	6	0	6	1	5	0	6
Totalmente en acuerdo	0	0	1	3	0	0	0	0	0	5

- **Análisis estadístico de los resultados de la encuesta**

Por medio de los resultados de la encuesta realizada al grupo de estudiantes integrantes de la muestra, se diseñó una estructura factorial mediante un Análisis Factorial Exploratorio (AFE) lo cual permitió identificar a dos factores, seguido de un Análisis Factorial Confirmatorio (AFC) que afirmó el ajuste de los factores comodidad y utilidad pertenecientes a la característica de Satisfacción con el modelo factorial.

Para realizar el análisis factorial AFE y AFC, es necesario verificar los supuestos de aditividad, linealidad, normalidad, homogeneidad y homocedasticidad, adicionalmente el AFE necesita la verificación de los supuestos de suficiencia de correlación y suficiencia de muestreo (Jácome, Caraguay, Herrera, & Herrera, 2020).

El supuesto de suficiencia de correlación, se lo realizo por medio de un test de Barlett, con lo cual se obtuvo un p-value de 5.804135e-07 por lo que se acepta este supuesto. Para el supuesto de suficiencia de muestreo se realizó una prueba Kaiser Meyer Olkin (KMO) lo cual verificó una insuficiencia de muestreo, sin embargo, se acepta este supuesto con la condicionante de eliminar reactivos que saturen el modelo propuesto por el AFE.

La verificación de la aditividad se realizó mediante una matriz de correlaciones, donde se mostró que ninguna de las combinaciones de preguntas logró un 0.95 de correlación, ver Fig. 60.

```
    P1 P2 P3 P4 P5 P6 P7 P8 P9 P10
P1  1
P2     1
P3  .     1
P4           1
P5  .     .     1
P6     .        ,  1
P7  .     .     .     1
P8                .  ,  .  1
P9  .     .     .  .  .  .  1
P10          .     .     .     1
attr(,"legend")
[1] 0 ' ' 0.3 '.' 0.6 ',' 0.8 '+' 0.9 '*' 0.95 'B' 1
```

Fig. 60 Matriz de correlaciones de los reactivos de la encuesta

Por medio de una regresión lineal falsa, se comprobó los supuestos de linealidad, normalidad, homogeneidad y homocedasticidad. La normalidad, se muestra y verifica en el histograma de la Fig. 61 en donde las frecuencias siguen una distribución normal entre 2 y -2.

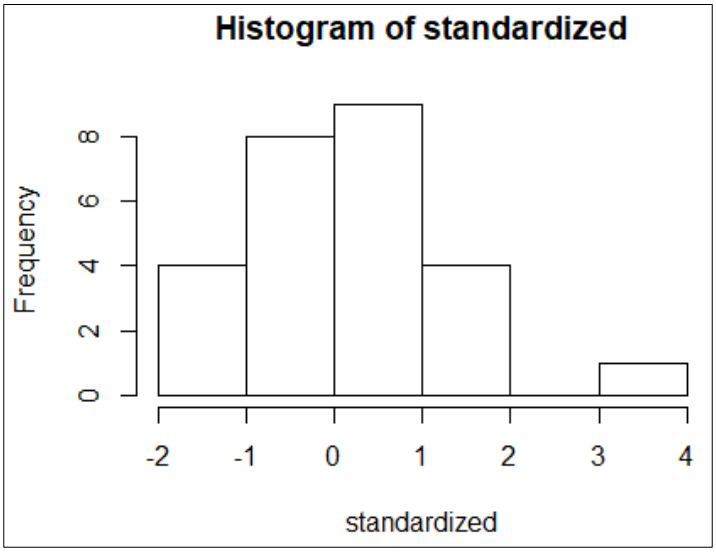


Fig. 61 Histograma de frecuencias

La tendencia lineal que se observa en la Fig. 62, verifica el cumplimiento de la linealidad con un ligero error de una distribución.

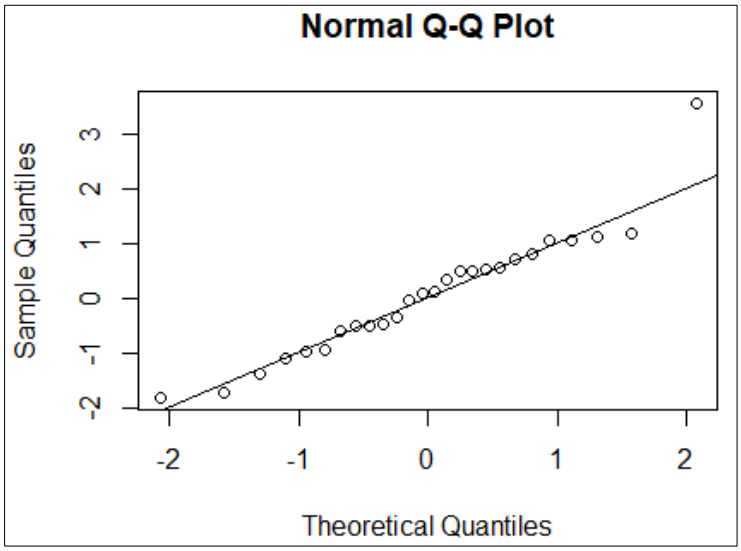


Fig. 62 Plot del supuesto de linealidad

La homogeneidad y homocedasticidad se verifica mediante el criterio de esfericidad; en la Fig. 63 se observa que la distribución es proporcional en los cuatro cuadrantes y así se aceptan estos supuestos.

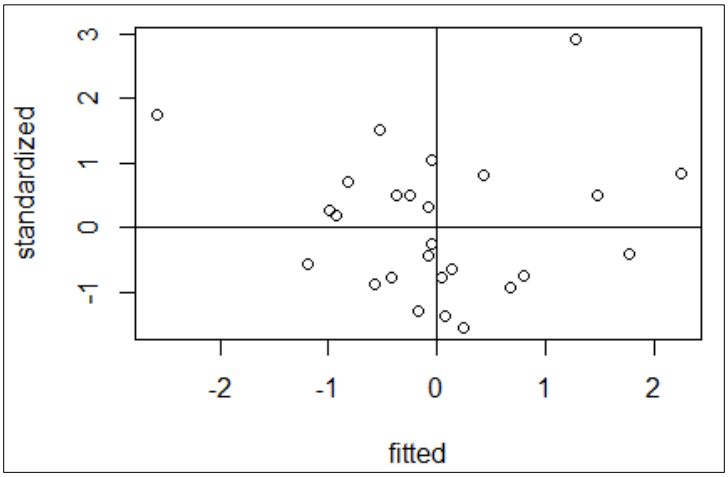


Fig. 63 Plot de los supuestos de homogeneidad y homocedasticdad

La estructura factorial propuesta por el AFE, indicó la presencia de dos factores que, por medio de una revisión teórica de las preguntas del cuestionario se detectó las sub-características de utilidad y comodidad. Debido al problema con el supuesto de suficiencia de muestreo, se realizó un tratamiento al modelo por lo que se eliminaron las preguntas 1, 7 y 9 del factor comodidad y la pregunta 4 del factor utilidad, las mismas que saturaban al modelo.

Una vez definido el modelo factorial, se realiza un AFC con los resultados del AFE, lo cual muestra las saturaciones de cada reactivo con su factor, como se observa en la Fig. 64. La pregunta 10 del factor utilidad no alcanza la mínima saturación con su factor, sin embargo, por juicio propio de investigación se la acepta para formar parte del modelo factorial.

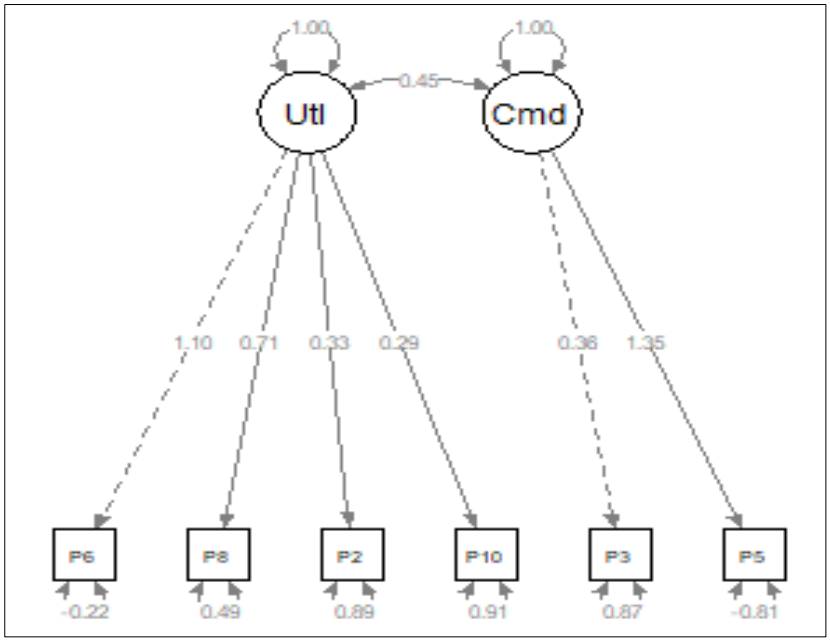


Fig. 64 Estructura factorial del AFC

En el AFC es necesario verificar si los datos se ajustan al modelo, por lo que existen algunas índices estadísticos que validan este ajuste, entre los indices más usados están: el Comparative Fit Index (CFI[5]), el Tucker y Lewis Index (TLI[6]), el Root Mean Square Error of Approximation (RMSEA[7]) y el Standardized Root Mean Residual (SRMR[8]) (Perry, Nicholls, Clough, & Crust, 2015). Para la aceptación de los índices, se han sugerido valores de corte: Para el TLI y CFI valores ≥0.95, para el RMSEA valores ≤0.6 y para el SRMR, valores ≤0.8 (Nye & Drasgow, 2011).

De acuerdo a los índices de ajuste mencionados, se obtuvo los siguientes resultados: el CFI con un valor de 0.989, el TLI con valor de 0.980, el RMSEA y el SRMR obtuvieron valores de 0.052 y 0.092 respectivamente, ver Fig. 65.

npar	fmin	chisq	df	pvalue
13.000	0.165	8.557	8.000	0.381
baseline.chisq	baseline.df	baseline.pvalue	cfi	tli
66.062	15.000	0.000	0.989	0.980
nnfi	rfi	nfi	pnfi	ifi
0.980	0.757	0.870	0.464	0.990
rni	logl	unrestricted.logl	aic	bic
0.989	-174.624	-170.346	375.248	391.603
ntotal	bic2	rmsea	rmsea.ci.lower	rmsea.ci.upper
26.000	351.252	0.052	0.000	0.240
rmsea.pvalue	rmr	rmr_nomean	srmr	srmr_bentler
0.430	0.085	0.085	0.092	0.092
srmr_bentler_nomean	crmr	crmr_nomean	srmr_mplus	srmr_mplus_nomean
0.092	0.109	0.109	0.092	0.092
cn_05	cn_01	gfi	agfi	pgfi
48.119	62.044	0.909	0.760	0.346
mfi	ecvi			
0.989	1.329			

Fig. 65 Índices de ajustes de confiabilidad y validez

Los resultados de los índices CFI y TLI obtenidos indican un ajuste aceptable al superar el 0.95 mínimo, el índice RMSEA, también muestra un valor aceptable al obtener menos de 0.06, sin embargo, el SRMR no cumplen el máximo de 0.05 pero en este caso se lo admite, por lo cual se acepta la validez del análisis factorial aplicado y del constructo obtenido.

3.3. Evaluación y resultados.

Después de obtener y validar los resultados del taller y encuesta, se procede a evaluar cada una de las características y sub-características de calidad en uso. Las características de *eficacia* y *eficiencia* pretenden medir el rendimiento del usuario, mientras que la característica de *satisfacción* mide el grado de satisfacción del usuario en base a la experiencia de uso del

[5] **CFI**: Analiza el ajuste del modelo examinando la discrepancia entre los datos y el modelo, al tiempo que se ajusta para los problemas de muestreo inherentes a la prueba de ajuste de chi-cuadrado y el índice de ajuste estándar.

[6] **TLI:** Analiza la discrepancia entre el valor chi cuadrado del modelo hipotético y el chi cuadrado del modelo nulo.

[7] **RMSEA:** Evita problemas de tamaño de la muestra mediante el análisis de la discrepancia entre el modelo hipotético, con estimaciones de parámetros elegidas de forma óptima, y la matriz de covarianza de población

[8] **SRMR:** Raíz cuadrada de la discrepancia entre la matriz de covarianza de la muestra y la matriz de covarianza del modelo.

producto. Cada sub-característica es medida por medio de una fórmula establecida según las métricas de la norma ISO/IEC 25022 que se especifican en la TABLA 37.

3.3.1. Evaluación característica de Eficacia

- **Sub-característica tareas completas**

Esta sub-característica es la proporción de las tareas que se han completado correctamente sin asistencia, ver TABLA 41.

TABLA 41 Especificaciones de la métrica Tareas completas

Métrica:	Tareas completas	
Elementos de datos	**Descripción**	**Valor**
A	Número de tareas únicas completadas	201
B	Número de tareas únicas intentadas	208

$$X = A / B = 201 / 208 = \mathbf{0,97}$$

- **Sub-característica objetivos logrados**

Esta sub-característica mide el grado proporcional del cumplimiento de cuatro objetivos por cada usuario, cada objetivo tiene un peso de 25% y su aceptación se lo hace por medio de dos tareas completadas, ver TABLA 42.

TABLA 42 Especificaciones de la métrica Objetivos logrados

Métrica:	Objetivos logrados	
Elementos de datos	**Descripción**	**Valor**
ΣAi	Sumatoria del número de objetivos cumplidos por cada usuario (i).	95
U	Total de usuarios	26

$$X = [(\Sigma Ai) * 0,25] / U = [(95) * 0,25] / 26 = \mathbf{0,91}$$

- **Sub-característica tareas con errores**

La sub-característica tareas con errores mide el grado proporcional de tareas con errores que se han completado, sin embargo, para la obtención del grado proporcional de tareas sin errores se aplica la f´romula 1 – tareas con errores , ver TABLA 43.

TABLA 43 Especificaciones de la métrica Tarea con errores

Métrica:	Tareas con errores	
Elementos de datos	**Descripción**	**Valor**
A	Número de tareas con errores	13
B	Número de tareas únicas propuestas	208

$$X = 1-(A / B) = 1- (13 / 208) = \mathbf{0,94}$$

3.3.2. Evaluación característica de Eficiencia

- **Sub-característica tiempo de tareas**

El tiempo en tareas mide la relación del tiempo empleado por cada usuario con el tiempo experto. Se obtiene la sumatoria de la relación del tiempo de cada usuario con el tiempo experto y a esto se divide para el total de usuarios, obteniendo así un promedio, ver TABLA 44.

TABLA 44 Especificaciones de la métrica Tiempo de tareas

Métrica:	Tiempo de tareas	
Elementos de datos	**Descripción**	**Valor**
*Σ*Ai	Relación del tiempo empleado de tareas con el tiempo experto por cada usuario (i)	13,67
U	Número total de usuarios	26

$$X = \Sigma Ai / U = 13,67 / 26 = \mathbf{0,526}$$

- **Sub-característica eficiencia del tiempo**

La eficiencia del tiempo mide la relación del tiempo empleado por cada usuario en el cumplimiento de los objetivos con el tiempo experto. Se obtiene la sumatoria de la relación del tiempo de cada usuario según los objetivos cumplidos con el tiempo experto y a esto se divide para el total de usuarios. Cabe aclarar que si una tarea del objetivo no se cumple, el objetivo no será válido, ver TABLA 45

TABLA 45 Especificaciones de la métrica Eficiencia del tiempo

Métrica:	Eficiencia del tiempo	
Elementos de datos	**Descripción**	**Valor**
*Σ*Ai	Relación del tiempo empleado de tareas de los objetivos con el tiempo experto por cada usuario (i)	15,95
U	Número total de usuarios	26

$$X = \Sigma Ai / U = 15,95 / 26 = \mathbf{0,61}$$

3.3.3. Evaluación característica de Satisfacción

- **Sub-característica utilidad**

La sub-característica de utilidad determina la relación entre el número de usuarios satisfechos con el uso del producto y el número total de usuarios, ver

Con el resultado final de los usuarios satisfechos, se puede realizar el cálculo de la sub-característica de utilidad.

TABLA 46. Las preguntas de la encuesta SUS que se ajustaron a esta sub-característica por medio de los resultados del análisis factorial fueron las preguntas 2, 6, 8 y 10.

Para obtener los usuarios satisfechos se realizó el cálculo por cada pregunta mediante la suma de todas las respuestas de los 26 estudiantes dividido para 5, que es el máximo valor numérico con el cual se obtiene el valor de usuarios satisfechos, al final se obtiene el promedio de las 4 preguntas (7).

$$Pregunta\ 2 = sumaRespuestas\ /\ 5\ = 84\ /\ 5 = 16{,}8 \quad (3)$$

$$Pregunta\ 6 = sumaRespuestas\ /\ 5 =\ 89\ /\ 5\ = 14{,}4 \quad (4)$$

$$Pregunta\ 8 = sumaRespuestas\ /\ 5\ = 92\ /\ 5\ = 18{,}4 \quad (5)$$

$$Pregunta\ 10 = sumaRespuestas\ /\ 5\ = 67\ /\ 5\ = 13{,}4 \quad (6)$$

$$Usuarios\ satisfechos = S = Prom(16.8, 14.4, 18.4, 13.4) = \mathbf{15{,}75} \quad (7)$$

Con el resultado final de los usuarios satisfechos, se puede realizar el cálculo de la sub-característica de utilidad.

TABLA 46 Especificaciones de la métrica Utilidad

Métrica:	Utilidad	
Elementos de datos	**Descripción**	**Valor**
S	Número de usuarios satisfechos	15,75
U	Número total de usuarios	26

X = S / U = 15,75 / 26 = ***0,61***

- **Sub-característica confianza**

Esta sub-característica determina el grado de confianza de los usuarios, basado en las quejas presentadas por la falla del producto, ver TABLA 47.

TABLA 47 Especificaciones de la métrica Confianza

Métrica:	Confianza	
Elementos de datos	**Descripción**	**Valor**
A	Número de quejas presentadas	4
T	Total de encuestados	26

X = % Reclamos ; C = % Confianza

$$X = A / T \; ; \; C = 1 - X$$
$$C = 1 - X = 1 - (A / T) = 1 - (4 / 26) = \mathbf{0,85}$$

- **Sub-característica comodidad**

La comodidad determina el grado de facilidad y poco esfuerzo en el uso del producto y se la obtiene por medio del cálculo de los valores ponderados de la escala de respuesta de las preguntas de la encuesta asociadas a esta sub-característica (Preguntas 3 y 5). Para obtener el valor de cada elemento de dato, se multiplica la ponderación por el total de respuestas y se divide para 52 que es el total de usuarios de las dos preguntas, ver TABLA 48.

TABLA 48 Especificaciones de la métrica Comodidad

Métrica:	Comodidad			
Elementos de datos	**Descripción**	**Ponderación**	**T. respuestas**	**Valor**
A	Totalmente de acuerdo	1	14	0,27
B	De acuerdo	0,8	27	0,42
C	Ni de acuerdo ni en desacuerdo	0,6	9	0,10
D	En desacuerdo	0,4	1	0,01
E	Totalmente en desacuerdo	0,2	1	0,00

$$X = A + B + C + D + E = 0,27 + 0,42 + 0,10 + 0,01 + 0,00 = \mathbf{0,80}$$

3.3.4. Resultados de la evaluación

Esta sección describe los resultados obtenidos de la evaluación de calidad en uso sobre el envoltorio GraphQL de Mendeley. En base al modelo presentado en la TABLA 35, se muestra los pesos, medición y resultado de cada característica (C) y sub-característica (SC), y al final se obtiene el porcentaje final el cual indica el grado de usabilidad del producto, ver TABLA 49.

TABLA 49 Resultado de la evaluación final de calidad en uso

Característica	**Sub-característica**	**Peso (C)**	**Peso (SC)**	**Medición**	**Resultado**	**Res.característica**
Eficacia	Tareas completas	**40%**	17%	0,97	16,43%	**37,68**
	Objetivos logrados		13%	0,91	11,88%	
	Tareas con errores		10%	0,94	9,38%	
Eficiencia	Tiempo de tareas	**30%**	10%	0,53	5,26%	**17,53**
	Eficiencia del Tiempo		20%	0,61	12,27%	
Satisfacción	Utilidad	**30%**	10%	0,61	6,06%	**22,57**
	Confianza		11%	0,85	9,31%	
	Comodidad		9%	0,80	7,20%	
	TOTAL	**100%**				**77,77%**

3.3.5. Análisis de los resultados de la evaluación

Para analizar los resultados, se tomó en cuenta la escala de evaluación descrita en la norma ISO/IEC 25040, la cual dice que la evaluación de un producto es realizada según el contexto en el que se desarrolla, por lo cual, la Fig 66 índica los niveles de puntuación para las métricas. En el caso de la evaluación realizada, todas las métricas están con la valoración de satisfactorio.

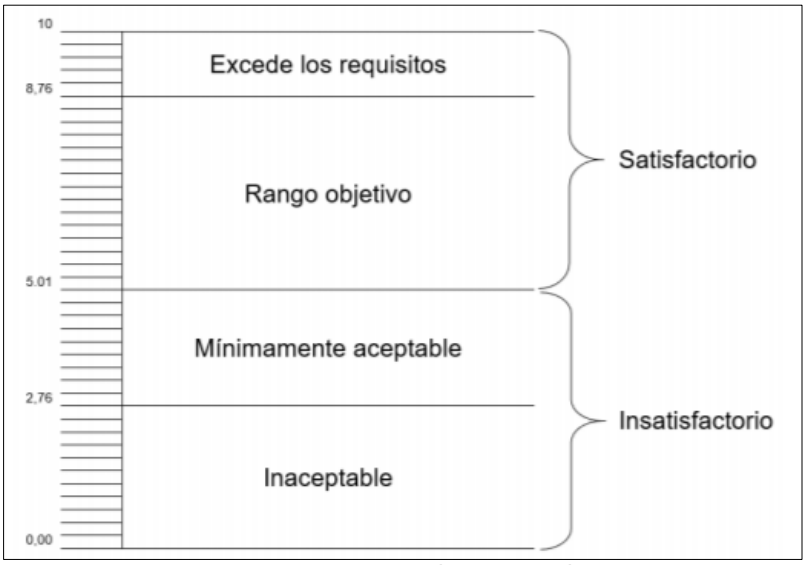


Fig 66 Niveles de puntuación para las métricas
Fuente: Adaptada de (ISO/IEC 14598-1, 1999)

La evaluación de la calidad en uso del envoltorio GraphQL fue satisfactoria, con un 77,77% de usabilidad, se puede concluir que el producto es adecuado en el contexto de uso. La Fig. 67 muestra resultados individuales del cumplimiento de las 3 características del modelo de calidad de uso con una escala porcentual.

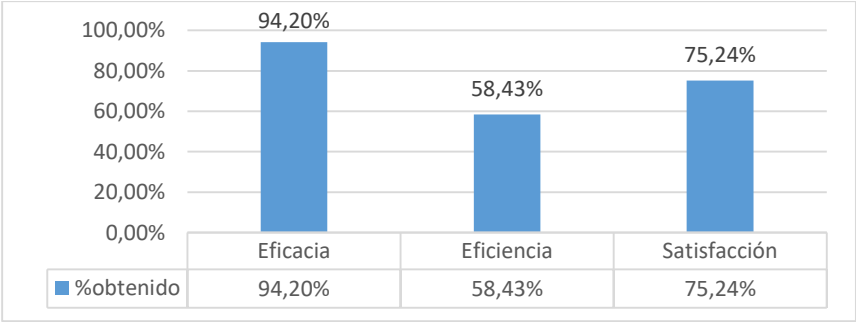


Fig. 67 Porcentaje obtenido de las características del modelo de calidad

A continuación, se analizará cada una de las características que componen el modelo de calidad en uso, según los resultados de la evaluación aplicada.

- **Análisis de la característica de Eficacia**

El resultado de la evaluación obtenido con respecto a la eficacia fue satisfacoria, con un cumplimento del 94,20%, es la característica con el valor porcentual más alto. La Fig. 68 muestra el cumplimiento de cada sub-característica en relación a lo especificado.

El resultado de la sub-característica de tareas completadas demuestra que la gran mayoría de tareas propuestas fueron ejecutadas, independientemente de si tuvieron o no errores. La sub-característica de objetivos logrados muestra que el grado de cumplimiento de los cuatro objetivos que cada participante debía cumplir, es aceptable (91% de cumplimiento). La sub-característica tareas con errores mide el porcentaje de errores cometidos en una tarea, sin embargo, para la medición se obtiene el grado de tareas sin errores por medio de la resta de 1 menos el resultado de tareas con errores (6% tarea con errores).

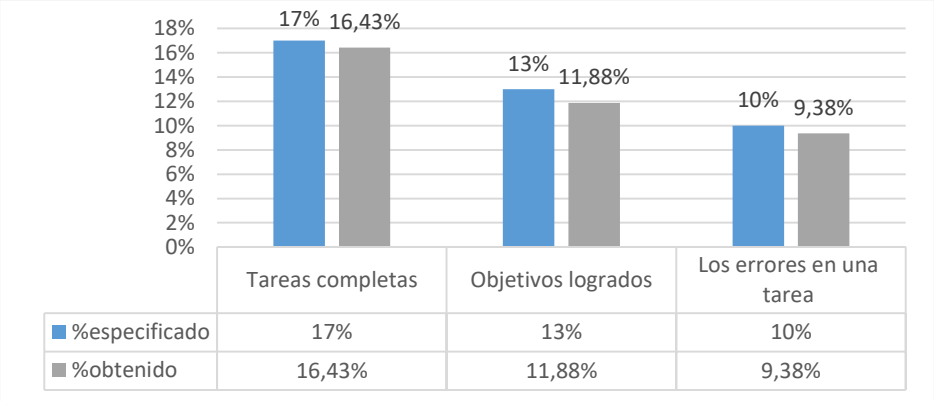


Fig. 68 Porcentaje especificado y obtenido de la característica de eficacia

- **Análisis de la característica de Eficiencia**

Esta característica tuvo el más bajo resultado de la evaluación, con un 58,43%, por lo cual se concluye que, en el taller, los estudiantes tuvieron que gastar más de recursos de lo esperado (tiempo, esfuerzo). La Fig. 69 muestra el cumplimiento de cada sub-característica en relación a lo especificado.

El resultado de tiempo de tareas que fue calculado por medio de la relación del tiempo experto con el tiempo de desarrollo de taller de cada uno de los participantes muestra una posible dificultad en el manejo del tiempo a la hora de ejecutar las tareas. La eficiencia del tiempo nos muestra el mismo criterio de relación de tiempo, pero por objetivos cumplidos, este resultado tiene un ligero aumento en relación al tiempo de tareas por lo cual se la entra en el rango de satisfactoria.

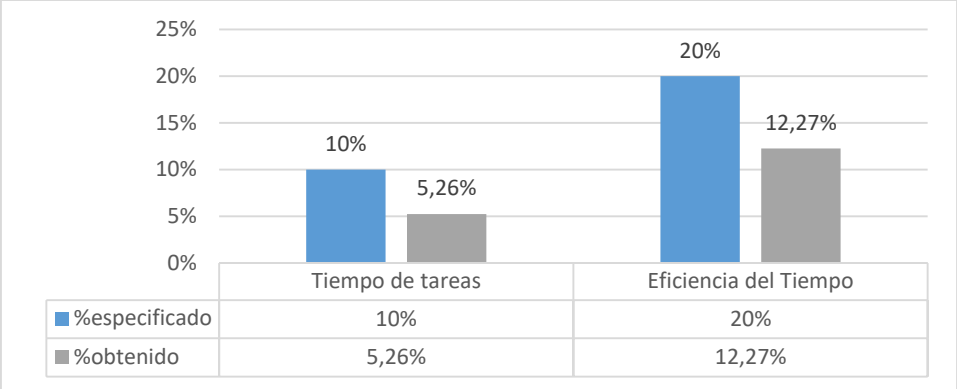


Fig. 69 Porcentaje especificado y obtenido de la característica de eficiencia

Los estudiantes que aportaron información para la evaluación, tenían conocimiento básico de GraphQL, sin embargo, al no estar inmiscuidos en esta tecnología, se especula que tuvo un efecto en la valoración mediana-baja de la eficiencia del modelo de calidad. A pesar de haber obtenido el resultado más bajo de la evaluación, el porcentaje de eficiencia se lo puede mejorar mediante la repetición del taller, no obstante, desde el contexto en el que se ejecutó la evaluación, no se realizó repeticiones.

- **Análisis de la característica de Satisfacción**

La satisfacción obtuvo un porcentaje satisfactorio, con un 75,24% se concluye que los estudiantes estuvieron cómodos al usar el API. La Fig. 70 muestra el cumplimiento de cada sub-característica en relación a lo especificado.

Con un grado de confianza individual de 85%, y una comodidad del 80%, muestra que la satisfacción de uso del producto se cumple, a esto sumado el porcentaje de utilidad del 61%, hace que el objetivo de la característica de satisfacción aporte en gran medida al modelo de calidad en uso.

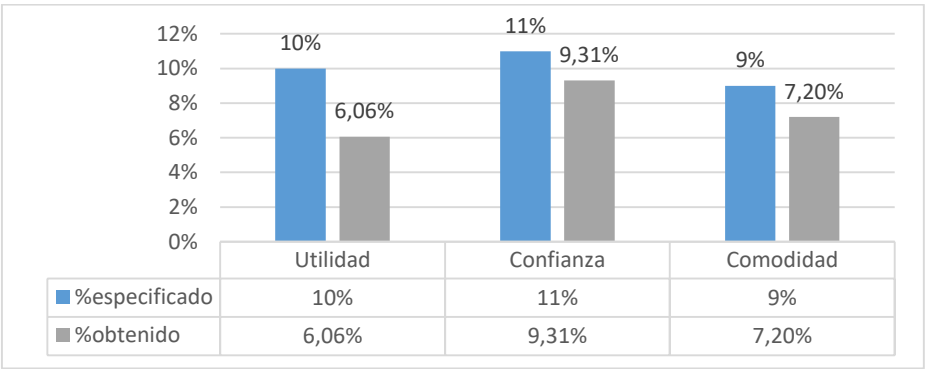


Fig. 70 Porcentaje especificado y obtenido de la característica de satisfacción

CONCLUSIONES

- Con el desarrollo de la investigación teórica, se definió la base conceptual que sirvió para determinar la arquitectura tecnológica y tecnologías para el desarrollo del API-GraphQL (servidor) como envoltorio de las APIs Mendeley, y la aplicación React (Cliente) para consumo del API-GraphQL. También se pudo establecer la base conceptual de la familia de normas ISO/IEC 25000, para validación de la calidad de uso del software.
- En el desarrollo del API-GraphQL, se pudo utilizar las tecnologías establecidas en la planificación del proyecto como: Node.js como framework base del lenguaje JavaScript, librería npm como gestor de dependencias, express como marco de aplicación web. Además, se complementó la implementación del servidor con la biblioteca graphql-yoga ya que facilitó el desarrollo de funcionalidades del envoltorio.
- La programación del Cliente (prueba de concepto) se lo realizó en React.js, el cual fue flexible para el desarrollo con GraphQL ya que se complementa de una manera adecuada porque expone un conjunto de librerías, recursos e información en la web importante que permiten elaborar trabajos con estas tecnologías. Además, que estas tecnologías al ser concebidas por una misma organización se complementan de manera eficiente.
- El uso de GraphQL para la generación del envoltorio demostró que se puede envolver APIs-REST, sin embargo, existe un mínimo porcentaje de aspectos que no se envolvieron por su dificultad de desarrollo, sin embargo esto no fue un limitante para que el software no cumpla con lo establecido, por lo que, la envoltura cumplió con las pruebas de aceptación realizadas.
- El uso de la metodología de desarrollo ágil SCRUM, facilitó la toma de decisiones en cada una de las revisiones de los Sprints, ya que se evidenciaban ciertas dificultades en el desarrollo del producto, las cuales se pudieron solventar de manera rápida y eficiente, con lo que se redujo generación un entregable defectuoso.
- Mediante la evaluación de calidad en uso de la norma ISO/IEC 25000 se determinó que el grado de usabilidad del envoltorio GraphQL es de 77,7% lo cual se considera satisfactorio, las características de eficacia y satisfacción aportaron un 37,68% y un 22,57% de un máximo de 40% y 30% respectivamente, sin embargo, la característica de eficiencia brindó el más bajo aporte con un 17,53% de un máximo de 30%, lo que evidenció que para evaluar de mejor manera esta característica sería necesario realizar varias repeticiones para que los usuarios mejoraren su eficiencia en el uso del software.

RECOMENDACIONES

- Leer artículos científicos de alto impacto de investigaciones que involucre la tecnología GraphQL y proponer una nueva línea de investigación que no haya sido explorada.
- Estudiar temáticas relacionadas con la creación de software back-end tales como protocolos para autenticación y autorización de solicitudes, estructuras de peticiones HTTP, buenas prácticas de desarrollo, lenguajes de programación específicos para servidores, etc.
- Se recomienda a la comunidad de desarrollo, analizar el código fuente del envoltorio y la prueba de concepto los cuales están alojados en GitHub (ver Anexos) y proponer mejoras ya sea en optimización de código, corrección de errores, ampliación de las funcionalidades del API o prueba de concepto, etc.
- Desarrollar una envoltura a una API-REST diferente a la utilizada en este trabajo y realizar un estudio comparativo entre el producto resultante del envolvimiento y el producto original, con la posibilidad de usar lenguajes de programación diferentes a los descritos en este trabajo.
- Como trabajo futuro se recomienda desarrollar un software cliente que tenga requisitos específicos y que permita usar las funcionalidades del envoltorio, utilizando diferentes plataformas de desarrollo como React.js, Angular.js, y clientes GraphQL tales como Apollo Client y Relay, esto con el fin de analizar las ventajas y desventajas que presentan estas soluciones.
- Profundizar en la rama de la estadística para tener conocimiento de cómo validar los resultados de la investigación, estudio o desarrollo por medio de técnicas, modelos o métodos estadísticos que se ajusten al trabajo realizado.

REFERENCIAS

Abdalkareem, R., Nourry, O., Wehaibi, S., Mujahid, S., & Shihab, E. (2017). Why do developers use trivial packages? an empirical case study on npm (pp. 385–395). Association for Computing Machinery (ACM). https://doi.org/10.1145/3106237.3106267

Aggarwal, S. (2018). Modern Web-Development using ReactJS. *International Journal of Recent Research Aspects*, *5*(1), 2349–7688. Retrieved from https://www.ijrra.net/Vol5issue1/IJRRA-05-01-27.pdf

Apollo-GraphQL. (2019). Apollo Docs. Retrieved May 12, 2019, from https://www.apollographql.com/docs/

Bryant, M. (2017). GraphQL for archival metadata: An overview of the EHRI GraphQL API. In *2017 IEEE International Conference on Big Data (Big Data)* (pp. 2225–2230). IEEE. https://doi.org/10.1109/BigData.2017.8258173

Casciaro, M. (2014). *Node.js Design Patterns*. Birmingham.

Cho, H., & Ryu, S. (2014). REST to JavaScript for better client-side development. In *Proceedings of the 23rd International Conference on World Wide Web - WWW '14 Companion* (pp. 937–942). New York, New York, USA: ACM Press. https://doi.org/10.1145/2567948.2579219

Cronbach, L. J. (1951). Coefficient alpha and the internal structure of tests. *Psychometrika*, *16*(3), 297–334. https://doi.org/10.1007/BF02310555

Dmitry Namiot, M. S.-S. (2014). On Micro-services Architecture. *International Journal of Open Information Technologies*, *2*(9), 24–27.

Elsevier. (2019). About Mendeley. Retrieved February 21, 2019, from https://www.elsevier.com/solutions/mendeley

Endrei, M., Ang, J., Arsanjani, A., Chua, S., Comte, P., Krogdahl, P., … Newling, T. (2004). *Understanding SOA with Web Services*.

Express.js. (2017). Express - Node.js web application framework. Retrieved from https://expressjs.com/

Fielding, R. T., & Taylor, R. N. (2002). Principled design of the modern Web architecture. *ACM Transactions on Internet Technology*, *2*(2), 115–150. https://doi.org/10.1145/514183.514185

GARLAN, D., & SHAW, M. (1993). AN INTRODUCTION TO SOFTWARE ARCHITECTURE (pp. 1–39). https://doi.org/10.1142/9789812798039_0001

Ghebremicael, E. S. (2017). *Transformation of REST API to GraphQL for OpenTOSCA*. University of Stuttgart. https://doi.org/10.18419/opus-9352

Greene, J., & Stellman, A. (2014). *Learning Agile*. O'Reilly Media, Inc. Retrieved from https://learning.oreilly.com/library/view/learning-agile/9781449363819/

IBM Knowledge Center. (2017). Arquitectura orientada a servicios. Retrieved June 2, 2019, from https://www.ibm.com/support/knowledgecenter/es/SSV2LR/com.ibm.wbpm.wid.main.doc/prodoverview/topics/csoa.html

ISO/IEC 14598-1. (1999). Information technology — Software product evaluation — Part 1: General overview.

ISO/IEC 25010. (2011). ISO/IEC 25010:2011(en), Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. Retrieved December 4, 2019, from https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-1:v1:en

ISO/IEC 25022. (2016). ISO/IEC 25022;2016, Systems and software engineering — Systems and software quality requirements and evaluation (SQuaRE) — Measurement of quality in use. Retrieved from https://www.iso.org/obp/ui/#iso:std:iso-iec:25022:ed-1:v1:en

ISO/IEC 25040. (2011). ISO/IEC 25040:2011, Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Evaluation process.
ISO 25000. (2019). PORTAL ISO 25000. Retrieved December 4, 2019, from https://iso25000.com/
Jacobson, D., Brail, G., & Woods, D. (2011). *APIs : a strategy guide*. O'Reilly. Retrieved from https://learning.oreilly.com/library/view/apis-a-strategy/9781449321628/
Jácome, A. E., Caraguay, J. A., Herrera, E. P., & Herrera, I. D. (2020). Confirmatory Factorial Analysis Applied on Teacher Evaluation Processes in Higher Education Institutions of Ecuador. In *Advances in Intelligent Systems and Computing* (Vol. 1110 AISC, pp. 157–170). Springer. https://doi.org/10.1007/978-3-030-37221-7_14
Kimokoti, B. (2018). *Beginning GraphQL : fetch data faster and more efficiently whilst improving the overall performance of your web application*.
Kobusińska, A., & Hsu, C. H. (2018). Towards increasing reliability of clouds environments with RESTful web services. *Future Generation Computer Systems*, *87*, 502–513. https://doi.org/10.1016/j.future.2017.10.050
Lee, B. (2015). GraphQL: A data query language | Engineering Blog | Facebook Code. Retrieved November 25, 2018, from https://code.fb.com/core-data/graphql-a-data-query-language/
Lee, B. (2016). Leaving technical preview | GraphQL. Retrieved November 25, 2018, from https://graphql.org/blog/production-ready/
Linux-Fundation. (2019). Introduction to GraphQL | GraphQL. Retrieved February 21, 2019, from https://graphql.org/learn/
MacLennan, E., & Van Belle, J. P. (2014). Factors affecting the organizational adoption of service-oriented architecture (SOA). *Information Systems and E-Business Management*, *12*(1), 71–100. https://doi.org/10.1007/s10257-012-0212-x
Mendeley. (2019a). API Reference | Mendeley. Retrieved from https://dev.mendeley.com/methods/#introduction
Mendeley. (2019b). Guides | Mendeley. Retrieved November 28, 2019, from https://www.mendeley.com/guides
Mendeley. (2019c). Mendeley API Portal. Retrieved from https://dev.mendeley.com/
Molina, B., Vite, H., & Dávila, J. (2018). Metodologías ágiles frente a las tradicionales en el proceso de desarrollo de software.
Mozilla developers. (2019). Introducción a Express/Node. Retrieved May 29, 2019, from https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction
Newman, S. (2017). *Building Microservices @ Squarespace*.
Node. (2019). Acerca de | Node.js. *Node.Js*.
Nye, C. D., & Drasgow, F. (2011). Assessing goodness of fit: Simple rules of thumb simply do not work. *Organizational Research Methods*, *14*(3), 548–570. https://doi.org/10.1177/1094428110368562
Oracle. (2019). Documentación Java. Retrieved from https://docs.oracle.com/en/java/javase/
Palacio, J., Menzinsky, A., & López, G. (2016). *Scrum Manager, Guía de información*.
Perry, J. L., Nicholls, A. R., Clough, P. J., & Crust, L. (2015). Assessing model fit: Caveats and recommendations for confirmatory factor analysis and exploratory structural equation modeling. *Measurement in Physical Education and Exercise Science*, *19*(1), 12–21. https://doi.org/10.1080/1091367X.2014.952370
Porcello, E., & Banks, A. (Software engineer). (2018). *Learning GraphQL : declarative data fetching for modern web apps*. Retrieved from https://www.amazon.es/Learning-GraphQL-Eve-Porcello/dp/1492030716
Porras, J. (2019). Primeros pasos scrum. Retrieved November 28, 2019, from https://synapptica.net/metodologia-scrum.html
Programa de las Naciones Unidas para el Desarrollo. (2018a). Objetivo 4: Educación de calidad

| PNUD. Retrieved November 29, 2018, from http://www.undp.org/content/undp/es/home/sustainable-development-goals/goal-4-quality-education.html
Programa de las Naciones Unidas para el Desarrollo. (2018b). Objetivo 9: Industria, innovación e infraestructura | PNUD. Retrieved November 29, 2018, from http://www.undp.org/content/undp/es/home/sustainable-development-goals/goal-9-industry-innovation-and-infrastructure.html
Ramdhani, Abdullah, Ramdhani, M. A., & Amin, A. S. (2014). Writing a Literature Review Research Paper: A step-by-step approach.
Ribas, E. (2018). Qué es Api Rest y por qué debes de integrarla en tu negocio ¡Descúbrelo! Retrieved June 3, 2019, from https://www.iebschool.com/blog/que-es-api-rest-integrar-negocio-business-tech/
Ryder, G. (2015). Objetivo #8: Trabajo decente y crecimiento economico. Retrieved November 29, 2018, from http://www.undp.org/content/undp/es/home/sustainable-development-goals/goal-8-decent-work-and-economic-growth.html
Salcedo Parra, O., & Herrera, S. (2013). EFICIENCIA ALGORÍTMICA EN APLICACIONES DE GRAFOS ORIENTADO A REDES GMPLS. *Redes de Ingeniería*, *2*(2), 107. https://doi.org/10.14483/2248762X.7173
Sandoval, J. (2009). *RESTful Java Web Services*. Mumbai: Packt Publishing Ltd.
Scrum. (2019). Recursos Scrum.org. Retrieved from https://www.scrum.org/resources
Snell, J., Tidwell, D., & Kulchenko, P. (2002). *Programming Web services with SOAP*. O'Reilly & Associates.
Taber, K. S. (2018). The Use of Cronbach's Alpha When Developing and Reporting Research Instruments in Science Education. *Research in Science Education*, *48*(6), 1273–1296. https://doi.org/10.1007/s11165-016-9602-2
Thiran, P., Hainaut, J., & Houben, G. (2005). Database Wrappers Development: Towards Automatic Generation. In *Ninth European Conference on Software Maintenance and Reengineering* (pp. 207–216). IEEE. https://doi.org/10.1109/CSMR.2005.22
Vargas, A., Sánchez Rivero, D., Valdéz, Á., Bernechea, M., Castillo, N., & Colqui, R. (2013). Evaluación de la calidad de la Información extraída por wrappers, de un sitio web. Retrieved from http://sedici.unlp.edu.ar/handle/10915/27136
Vázquez-Ingelmo, A., Cruz-Benito, J., & García-Peñalvo, F. J. (2017). Improving the OEEU's data-driven technological ecosystem's interoperability with GraphQL. In *Proceedings of the 5th International Conference on Technological Ecosystems for Enhancing Multiculturality - TEEM 2017* (pp. 1–8). New York, New York, USA: ACM Press. https://doi.org/10.1145/3144826.3145437
Villamizar, M., Garces, O., Castro, H., Verano, M., Salamanca, L., Casallas, R., & Gil, S. (2015). Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. In *2015 10th Computing Colombian Conference (10CCC)* (pp. 583–590). IEEE. https://doi.org/10.1109/ColumbianCC.2015.7333476
Vogel, M., Weber, S., & Zirpins, C. (2018). Experiences on Migrating RESTful Web Services to GraphQL. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 10797 LNCS, pp. 283–295). https://doi.org/10.1007/978-3-319-91764-1_23
Wittern, E., Cha, A., & Laredo, J. A. (2018). Generating GraphQL-wrappers for REST(-like) APIs. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 10845 LNCS, pp. 65–83). Springer, Cham. https://doi.org/10.1007/978-3-319-91662-0_5
Xinyang, F., Jianjing, S., & Ying, F. (2009). REST: An alternative to RPC for web services architecture. In *2009 1st International Conference on Future Information Networks, ICFIN 2009* (pp. 7–10). IEEE. https://doi.org/10.1109/ICFIN.2009.5339611

ANEXOS

Anexo A: Código fuente y despliegue de los proyectos desarrollados

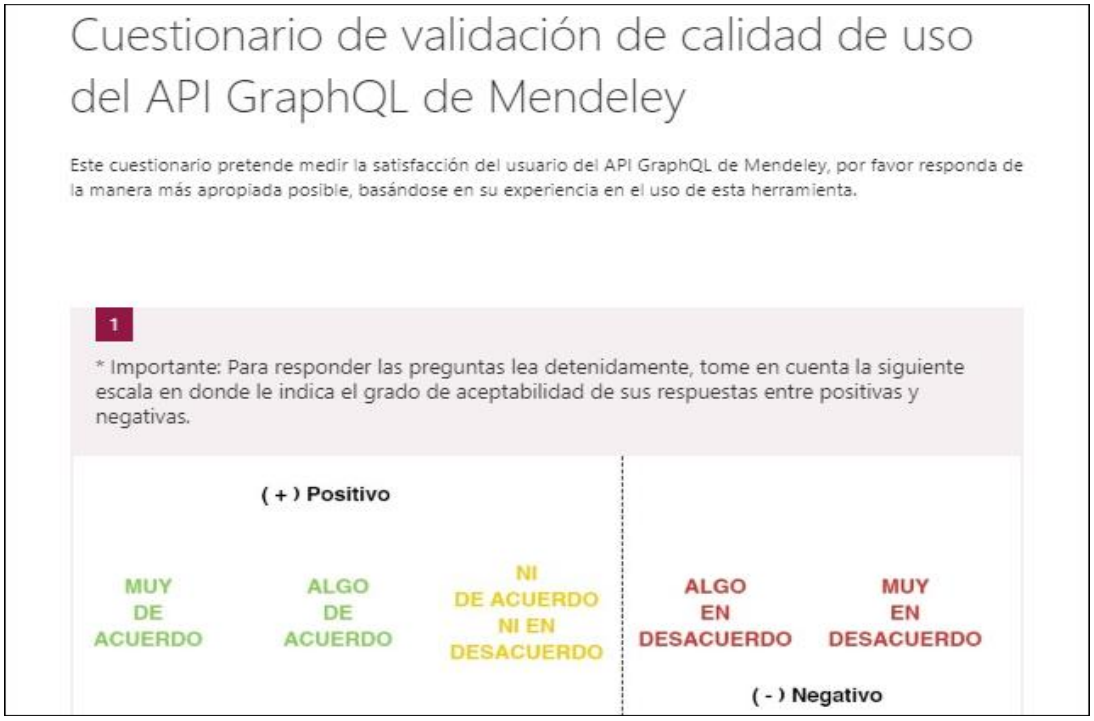
Proyecto	Repositorio	Url Producción
Envoltorio GraphQL del API-REST de Mendeley	https://github.com/kdrodriguez/graphql-mendeley-wrapper	https://graphql-mendeley.herokuapp.com/
Prueba de concepto	https://github.com/kdrodriguez/test-concept-gmw	https://test-concept-gmw.herokuapp.com/

Anexo B: Documentación del envoltorio para el usuario, disponible en el portal del envoltorio en la sección de “Como usar” o en la siguiente dirección.

- https://graphql-mendeley.herokuapp.com/Documentaci%C3%B3nAPI-GM.pdf

Anexo C: Desarrollo del taller y encuesta a un grupo de estudiantes de la Carrera de Ingeniería en Sistemas de la Universidad Técnica del Norte

Anexo D: Diseño de la encuesta SUS en los formularios de Office.



Cuestionario de validación de calidad de uso del API GraphQL de Mendeley

Este cuestionario pretende medir la satisfacción del usuario del API GraphQL de Mendeley, por favor responda de la manera más apropiada posible, basándose en su experiencia en el uso de esta herramienta.

1

* Importante: Para responder las preguntas lea detenidamente, tome en cuenta la siguiente escala en donde le indica el grado de aceptabilidad de sus respuestas entre positivas y negativas.

(+) Positivo

MUY DE ACUERDO

ALGO DE ACUERDO

NI DE ACUERDO NI EN DESACUERDO

ALGO EN DESACUERDO

MUY EN DESACUERDO

(-) Negativo

	Totalmente de acuerdo	De acuerdo	Ni de acuerdo ni en desacuerdo	En desacuerdo	Totalmente en desacuerdo
Creo que usaría esta API frecuentemente	○	○	○	○	○
Encontré el API innecesariamente complejo	○	○	○	○	○
Creo que el API fue fácil de usar	○	○	○	○	○
Creo que necesitaría ayuda de una persona con conocimientos técnicos para usar esta API	○	○	○	○	○
Las funciones de este API están bien integradas	○	○	○	○	○
Creo que el API es muy inconsistente	○	○	○	○	○
Imagino que la mayoría de la gente aprendería a usar esta API en forma muy rápida	○	○	○	○	○
Encuentro que el API es muy difícil de usar	○	○	○	○	○
Me siento confiado al usar esta API	○	○	○	○	○
Necesité aprender muchas cosas antes de ser capaz de usar este API	○	○	○	○	○

Anexo E: Resultados de la Encuesta SUS.

Totalmente de acuerdo	5
De acuerdo	4
Ni de acuerdo ni en desacuerdo	3
En desacuerdo	2
Totalmente en desacuerdo	1

Nombre	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
DIEGO ANDRES LEON VILLAVICENCIO	5	3	4	3	5	4	3	3	4	1
NEIDER FABRICIO REQUENE ESTACIO	4	2	2	5	4	5	3	5	4	4
JESUS ANDRES JACOME QUILUMBANGO	4	4	5	2	5	5	4	4	5	3
JAYLI DAVID DE LA TORRE RAMIREZ	4	4	5	3	5	4	4	4	3	4
JEFFERSON XAVIER LAPO MEDRANDA	4	3	4	3	4	2	3	2	3	2
SAMIR ALEXANDER ANDRADE LOZA	4	3	4	2	4	4	3	4	4	3
ADRIAN ROBAYO ORDOÑEZ	4	2	5	1	3	2	3	3	4	1
BRYAN ANDRE ALDAS LOPEZ	4	4	4	3	5	4	4	4	5	3
EDISON GEOVANNY PINANGO CUALCHI	5	3	3	4	4	3	2	2	3	3
ABRAHAM JACOBO VALLEJOS MALES	2	3	3	3	3	3	3	3	3	3
JHONY MARCELO MARTINEZ CHUGA	4	4	5	3	4	4	4	5	4	4
ALEJANDRO EMMANUEL ARIAS COLLAGUAZO	4	3	4	3	4	4	4	5	4	3
GUIDO ANTHONY CALDERON RAMIREZ	4	3	4	3	5	4	4	4	4	3
CARLOS SEBASTIAN PABON ANDRADE	5	4	4	4	4	4	4	4	4	2
FAUSTO RODRIGO NECPAS AMANTA	5	3	4	2	4	3	5	4	4	3
KEVIN FERNANDO CASTRO PONCE	4	3	3	4	4	3	3	3	4	1
ROBERT ALEXANDER PATIÑO CHALACAN	5	4	5	4	5	4	5	4	5	4
BRAYAN GUILLERMO PÉREZ PASPUEL	5	2	3	3	4	2	4	3	3	3
JUAN ANDRES DE LA CRUZ RON	5	2	4	1	5	4	5	4	4	1
CHRISTIAN ALEXANDER QUESPAZ ROSERO	4	2	4	2	4	2	4	2	4	2
CARLOS VICENTE TONTAQUIMBA QUINCHUQUI	5	5	4	1	5	4	3	3	5	1
LUIS ANDRES QUINCHE MORAN	4	3	4	2	4	2	4	2	4	2
KATERIN FERNANDA BENAVIDES DE LA CRUZ	4	3	4	3	4	3	4	4	3	2
JHONATAN DAVID HINOJOSA CHALAN	4	3	3	3	3	2	3	2	3	2
CARMEN PATRICIA TITUAÑA CORDOVA	4	4	4	3	5	5	3	5	4	4
EDISON DAVID CUASAPAS LOPEZ	3	5	1	2	3	3	3	4	3	3

Anexo F: Especificaciones de las tareas del taller de validación de calidad en uso del envoltorio GraphQL de Mendeley.

UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS
CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

TALLER DE VALIDACIÓN DE LA CALIDAD EN USO DEL ENVOLTORIO GRAPHQL DE MENDELEY

Objetivo: Utilizar el API GraphQL de Mendeley con el fin de verificar la calidad de uso del mismo por medio de la realización de las siguientes tareas:

Especificación de las tareas

- **Autorizar uso de recursos de la API.**
  1. Iniciar sesión con las credenciales de Mendeley.
  2. Brindar permiso de uso de token.
- **Realizar consultas simples**
  3. Obtener la lista de carpetas con los siguientes requisitos:
     Campos: id, name, parent_id, created, modified.
  4. Obtener la lista de documentos con los siguientes requisitos
     Parámetros: limit=5 y sort=title,
     Campos: id, title, abstract, authors (firs_name, last_name), year.
- **Realizar consultas complejas**
  5. Obtener la lista de carpetas con los siguientes requisitos:
     Campos: id, name, created,
     documents (id, title, year)
  6. Obtener un grupo de Mendeley por su id con los siguientes requisitos:
     Parámetros: id=bb0c19b2-4e61-3732-a7e1-96ebf78d623c
     Campos: id, name, created,
     folders (id, name,
     documents (id, title,
     files (id, file_name)
)
)
- **Realizar mutaciones**
  7. Crear un documento con los siguientes requisitos
     Parámetros: document= { title=Test_Document, type=book, abstract= Test abstract, authors= last_name=Perez, first_name=Juan }
     Campos: statusText
  8. Editar el documento creado en la tarea anterior con los siguientes requisitos:
     Parámetros: id (del documento creado), document= { title=Test_Document, type=journal,
     Campos: StatusText

Anexo G: Tabla de seguimiento de las tareas del taller de validación de calidad en uso del envoltorio GraphQL de Mendeley.

TALLER DE VALIDACIÓN DE LA CALIDAD EN USO DEL ENVOLTORIO GRAPHQL DE MENDELEY

Nombre: _____

Fecha: 04 de febrero de 2020

Tabla de seguimiento de las tareas

N°	Objetivo	N°	Tarea	Realizado (SI / No)	Hora inicio (hh:mm)	Hora fin (hh:mm)
1	Autorizar uso de recursos de la API.	1	Iniciar sesión con las credenciales de Mendeley.			
		2	Brindar permiso de uso de token.			
2	Realizar consultas simples	3	Obtener la lista de carpetas			
		4	Obtener la lista de documentos			
3	Realizar consultas complejas	5	Obtener la lista de carpetas			
		6	Obtener un grupo de Mendeley por su id			
4	Realizar mutaciones	7	Crear un documento			
		8	Editar el documento			

Tabla de quejas

N° queja	N° tarea	Descripción queja
1		
2		
3		
4		

Observaciones: _____

_____

_____

Anexo H: Reporte de análisis en Urkund.

URKUND

Urkund Analysis Result

Analysed Document: TESIS-DOCUMENT.pdf (D64857964)
Submitted: 3/4/2020 11:17:00 PM
Submitted By: kdrodriguezb@utn.edu.ec
Significance: 2 %

Sources included in the report:

TESIS TEO.docx (D48241428)
TRABAJO DE TITULACIÓN FINAL (Natalia Pilco).docx (D42788611)
https://dev.mendeley.com/methods/?shell#groups-v2.
https://www.amazon.es/Learning-GraphQL-Eve-Porcello/dp/1492030716
https://www.iebschool.com/blog/que-es-api-rest-integrar-negocio-business-tech/
http://sedici.unlp.edu.ar/handle/10915/27136
https://graphql-mendeley.herokuapp.com/Documentaci%C3%B3nAPI-GM.pdf

Instances where selected sources appear:

16