

**UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO**

**CARRERA:
INGENIERÍA DE SISTEMAS**

**Trabajo de titulación previo a la obtención del título de:
Ingenieros de Sistemas**

**TEMA:
CREACIÓN DE UN REPOSITORIO WEB PARA DATOS DE ESTACIONES
METEOROLÓGICAS DE LA CIUDAD DE QUITO**

**AUTORES:
JONATHAN ANDRÉS CASTILLO VIRE
CRISTHIAN FERNANDO SALAZAR CEVALLOS**

**TUTOR:
RODRIGO EFRAÍN TUFÍÑO CÁRDENAS**

Quito, julio del 2021

CESIÓN DE DERECHOS DE AUTOR

Nosotros Jonathan Andrés Castillo Vire, Cristhian Fernando Salazar Cevallos, con documento de identificación N° 1719555029 y N° 1717721177, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de titulación intitulado: CREACIÓN DE UN REPOSITORIO WEB PARA DATOS DE ESTACIONES METEOROLÓGICAS DE LA CIUDAD DE QUITO, mismo que ha sido desarrollado para optar por el título de: INGENIEROS DE SISTEMAS, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en nuestra condición de autores nos reservamos los derechos morales de la obra antes citada. En concordancia, suscribo este documento en el momento que hacemos entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

.....
Jonathan Andrés Castillo Vire
1719555029

.....
Cristhian Fernando Salazar Cevallos
1717721177

Quito, julio del 2021

DECLARATORIO DE COAUTORÍA DEL DOCENTE TUTOR

Yo declaro que bajo mi dirección y asesoría fue desarrollado el proyecto técnico, con el tema: **CREACIÓN DE UN REPOSITORIO WEB PARA DATOS DE ESTACIONES METEOROLÓGICAS DE LA CIUDAD DE QUITO**, realizado por Jonathan Andrés Castillo Vire y Cristhian Fernando Salazar Cevallos, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana para ser considerado como trabajo final de titulación.

Quito, julio 2021

A handwritten signature in blue ink, appearing to read 'R. E. T. C.', with a long horizontal stroke extending to the right.

Rodrigo Efraín Tufiño Cárdenas

C.I.: 1717646390

DEDICATORIA

De Jonathan Castillo

Dedico este trabajo a mis padres y mi abuelita, quienes siempre han dado su esfuerzo, sacrificio, dedicación y cariño para seguir adelante y lograr culminar esta etapa de mi vida, siempre me supieron motivar en el día a día para seguir adelante y poder llegar hasta el final. A mi hermanita por el apoyo y animo que me ha ofrecido durante todo este tiempo.

De Cristhian Salazar

Este trabajo está dedicado a mis padres Fernando y Fanny, quienes con su esfuerzo, cariño y apoyo incondicional han significado la mayor motivación para alcanzar todos los objetivos que me propongo, pero sobre todo por los valores que me inculcaron desde pequeño para ser una persona honesta y responsable. A mis hermanos Joselyn y Matheo por poner siempre una sonrisa en mi rostro y ser mi soporte en los momentos más complicados de este camino, sin duda alguna ustedes complementan cada uno de mis días.

AGRADECIMIENTO

De Jonathan Castillo

Al finalizar este trabajo quiero agradecer a Dios por todas sus bendiciones, a mi mamá y abuelita quienes han sabido darme su ejemplo, dedicación, esfuerzo y finalmente a mi hermana por su apoyo incondicional.

Un agradecimiento especial al Ing. Rodrigo Tufiño, principal colaborador durante todo este proceso, quien con su conocimiento, orientación, enseñanza y apoyo permitió el desarrollo de este trabajo.

De Cristhian Salazar

Quiero agradecer a Dios por permitirme tener a mi familia a mi lado al culminar esta etapa de mi vida, a mis padres por la confianza y el apoyo para permitir que me desarrolle profesionalmente, a la Universidad Politécnica Salesiana por mostrarme el carisma y alegría de los salesianos, a los docentes que me educaron a lo largo de la carrera por los conocimientos y experiencias que me ayudaron a crecer profesionalmente, a mi compañero Jonathan por ser un excelente compañero y amigo durante este proceso.

Finalmente agradecer al Ingeniero Rodrigo Tufiño por compartir su experiencia y conocimientos para ser nuestra guía a lo largo del desarrollo de este proyecto.

ÍNDICE

INTRODUCCIÓN.....	1
ANTECEDENTES	1
PROBLEMA DE ESTUDIO	1
JUSTIFICACIÓN.....	2
OBJETIVOS.....	3
Objetivo General.....	3
Objetivos específicos	3
METODOLOGÍA.....	3
CAPÍTULO I.....	5
MARCO TEÓRICO	5
1.1 OPEN ACCESS.....	5
1.2 REPOSITARIOS	6
1.2.1 Tipos de repositorios.....	6
1.3 HERRAMIENTAS DE DESARROLLO	7
1.3.1 Node.js	7
1.3.2 JavaScript	7
1.3.3 Handlebars.....	7
1.3.4 MongoDB	8
1.3.5 Oauth2	8
1.3.6 Heroku	8
1.3.7 Git.....	8

1.4 HERRAMIENTAS E INSTRUMENTOS PARA LA GESTIÓN DE PROYECTOS.....	9
1.4.1 Cuestionarios abiertos	9
1.4.2 Taiga.....	9
1.5 METEOROLOGÍA	9
1.5.1 Estaciones Meteorológicas	12
1.5.1.1 Estaciones meteorológicas convencionales.....	13
1.5.1.2 Estaciones meteorológicas automáticas.	13
1.6 METODOLOGÍAS ÁGILES	13
1.6.1 SCRUM	14
CAPÍTULO II.....	16
ANÁLISIS Y REQUERIMIENTOS.....	16
2.1 ANÁLISIS DE FACTIBILIDAD.....	16
2.1.1 Viabilidad Técnica.....	16
2.1.2 Viabilidad Económica	17
2.1.3 Viabilidad Operacional.....	18
2.2 ANÁLISIS DE REQUERIMIENTOS.....	19
2.2.1 Alcance	19
2.2.2 Definiciones acrónimos	20
2.2.3 Descripción General	20
2.2.4 Perspectiva del producto.....	20
2.2.4.1 Funciones del producto.....	20

2.2.4.2 Características de usuarios.....	21
2.2.4.3 Restricciones.....	22
2.2.5 Requerimientos Funcionales	22
2.2.6 Requisitos no funcionales.....	27
2.3 PROCESO ACTUAL	28
2.3.1. Proceso en INAMHI	28
2.3.2 Proceso en REMMAQ.....	30
2.4 DIAGRAMAS DE CASOS DE USO.	30
2.4.1 Caso de uso del usuario Invitado.....	30
2.4.2 Casos de uso del Investigador	31
2.4.3 Casos de Uso de Subida de datos	32
2.5 DIAGRAMAS DE SECUENCIA	33
2.5.1 Diagrama de secuencia de creación de cuenta.....	33
2.5.2 Diagrama de Secuencia de recuperación de contraseña	34
2.5.3 Diagrama de Secuencia de subida de información.....	35
2.5.4 Diagrama de secuencia para la búsqueda y descarga de información.....	36
2.6 DIAGRAMA DE CLASES	37
2.7 DIAGRAMA CONCEPTUAL DE BASE	41
2.8 DIAGRAMA NAVEGACIONAL	42
2.9 DISEÑO DE INTERFACES ABSTRACTAS	45
2.9.1 Interfaz abstracta invitado	45
2.9.2 Interfaz abstracta de opciones para el investigador.....	45

2.9.3	Interfaz abstracta para la subida de información de la secretaria del ambiente	46
2.9.4	Interfaz abstracta para subida de información del INAMHI	47
2.9.5	Interfaz abstracta para la edición de perfil	48
CAPÍTULO III		50
CONSTRUCCIÓN		50
3.1	ARQUITECTURA DE LA SOLUCIÓN	50
3.1.2	Modelo	50
3.1.3	Vista	50
3.1.4	Controlador	50
3.2	PRODUCT BACKLOG	52
3.3	SPRINTS	53
3.3.1	Sprint 1	53
3.3.1.1	Preparación del entorno de desarrollo.	55
3.3.1.2	Preparación del entorno Node.js	55
3.3.1.3	Preparación de Handlebars	55
3.3.2	Sprint 2	56
3.3.2.1	Preparación del gestor de base de datos MongoDB.	58
3.3.2.2	Conexión con MongoDB.	58
3.3.3	Sprint 3	59
3.3.4	Sprint 4	61
3.3.4.1	Autenticación	62

3.3.4.2 Registro.....	62
3.3.4.3 Creación del cliente Oauth2.	64
3.3.4.4 Generar token con JWT.....	65
3.3.4.5 Implementación de Nodemailer.	65
3.3.4.6 Activación de cuenta.	67
3.3.5 Sprint 5	68
3.3.5.1 Inicio de Sesión	69
3.3.5.2 Cierre de sesión.	70
3.3.5.3 Cambio de contraseña.....	71
3.3.5.4 Edición de perfil.	73
3.3.6 Sprint 6	73
3.3.6.1 Carga de archivos.	74
3.3.7 Sprint 7	75
3.3.7.1 Lectura archivo INAMHI.	76
3.3.7.2 Lectura archivo REMMAQ.....	77
3.3.8 Sprint 8	80
3.3.9 Sprint 9	80
3.3.9.1 Generación de archivos y descarga.	81
3.4 PRUEBAS	85
3.4.1 Pruebas de Carga.....	85
3.4.2 Prueba de carga con 400 usuarios	85

CAPÍTULO IV	92
IMPLEMENTACIÓN	92
4.1 ENTORNO DE IMPLEMENTACIÓN.....	92
4.2 VARIABLES DE ENTORNO	92
4.3 CREDENCIALES	93
4.4 CONEXIÓN.....	93
CONCLUSIONES.....	96
RECOMENDACIONES	98
REFERENCIAS BIBLIOGRÁFICAS	99

ÍNDICE DE TABLAS

Tabla 1 Características del hardware disponible	16
Tabla 2 Herramientas de desarrollo.....	17
Tabla 3 Tabla de consideraciones de costos para el desarrollo del proyecto.	18
Tabla 4 Tabla de Características de Usuarios.....	21
Tabla 5 RF01: Creación de Usuario.	22
Tabla 6 RF02: Inicio de Sesión.	23
Tabla 7 RF03: Logout.....	24
Tabla 8 RF04: Recuperación de contraseña.	24
Tabla 9 RF05: Subida de información.....	25
Tabla 10 RF06: Listar información subida recientemente.	26
Tabla 11 RF07: Descarga de información.....	27
Tabla 12 Requisitos no funcionales.....	27
Tabla 13 Prueba de funcionalidad 1.	87
Tabla 14 Prueba funcional 2.....	88
Tabla 15 Prueba funcional 3.....	89
Tabla 16 Prueba funcional 4.....	90

ÍNDICE DE FIGURAS

Figura 1 Diagrama del proceso actual.....	29
Figura 2 Caso de uso del usuario Invitado.	31
Figura 3 Casos de uso del usuario investigador.	32
Figura 4 Casos de uso del proceso de subida de datos.....	33
Figura 5 Diagrama de secuencia de creación de cuenta.....	34
Figura 6 Diagrama de secuencia recuperación de contraseña.	35
Figura 7 Diagrama de secuencia subida de información.....	36
Figura 8 Diagrama de secuencia búsqueda y descarga de información.....	37
Figura 9 Diagrama de clase.....	40
Figura 10 Diagrama conceptual de base de datos.	42
Figura 11 Diagrama navegacional.....	44
Figura 12 Interfaz abstracta invitado.....	45
Figura 13 Interfaz abstracta opciones para el investigador.	46
Figura 14 Interfaz abstracta para subida de información REMMAQ.	47
Figura 15 Interfaz abstracta para subida de información INAMHI.	48
Figura 16 Interfaz abstracta edición de perfil.....	49
Figura 17 Diagrama de componentes del repositorio.....	51
Figura 18 Product Backlog.....	52
Figura 19 Sprint 1.....	54
Figura 20 Configuración Handlebars.	56
Figura 21 Sprint 2.....	57
Figura 22 Conexión con MongoDB.	59
Figura 23 Sprint 3.....	59
Figura 24 Pantalla principal.	60

Figura 25 Sprint 4.....	61
Figura 26 Manejo de registro.	62
Figura 27 Validación contraseña.	63
Figura 28 Configuración de Oauth2.....	64
Figura 29 Generar token JWT.....	65
Figura 30 Creación y uso de transporter de nodemailer.....	66
Figura 31 Mensaje de Correo electrónico.	66
Figura 32 Validación de activación de cuenta.....	67
Figura 33 Sprint 5.....	68
Figura 34 Definición de Passport para autenticación.....	69
Figura 35 Manejo de Login.....	70
Figura 36 Función de cierre de sesión.....	70
Figura 37 Validación campo correo.	71
Figura 38 Función para cambiar contraseña.....	72
Figura 39 Ruta y función para edición de perfil.....	73
Figura 40 Sprint 6.....	74
Figura 41 Librería multer	74
Figura 42 Sprint 7.....	75
Figura 43 Formato archivo INAMHI.	76
Figura 44 Librería fs.....	77
Figura 45 Formato de archivo REMMAQ.	78
Figura 46 Lectura XLS.....	79
Figura 47 Extracción Información del archivo xlsx.....	79
Figura 48 Sprint 8.....	80
Figura 49 Sprint 9.....	81

Figura 50 Extracción de información.....	82
Figura 51 Generar archivo CSV.....	83
Figura 52 Parámetros para la descarga.....	84
Figura 53 Construcción txt.....	84
Figura 54 Prueba de carga con 100 peticiones.....	85
Figura 55 Prueba con 400 peticiones.....	86
Figura 56 Prueba de rendimiento.....	86
Figura 57 Variables de entorno.....	92
Figura 58 Credenciales.....	93
Figura 59 Conexión remota al servidor.....	93
Figura 60 Clonar repositorio.....	94
Figura 61 Panel de administración Pm2.....	94
Figura 62 Aplicación desplegada.....	95

RESUMEN

El sistema desarrollado en el presente proyecto previo a la obtención del título de Ingenieros de Sistemas de la Universidad Politécnica Salesiana es un repositorio Web de archivos de datos meteorológicos para datos de las estaciones pertenecientes a la ciudad de Quito. El repositorio tiene como objetivo agilizar la divulgación de archivos de datos de tipo meteorológico, brindando a los usuarios la posibilidad compartir y obtener archivos en formatos específicos que se adapten a sus necesidades.

Aquí se expone el ciclo de desarrollo del sistema, desde la etapa inicial de análisis, continuando con el diseño, para posteriormente entrar en la etapa de construcción bajo la metodología de desarrollo ágil SCRUM. Para culminar se realizan pruebas y la implementación del sistema en un servidor Web.

A través de este proceso se obtuvo como producto final un repositorio que permite la carga de archivos manejados por INAMHI y REMMAQ en sus formatos originales, brindando la opción de obtenerlos en formato xls, CSV, txt y el archivo original.

ABSTRACT

The system developed in this Project is a web meteorologic file repository for data of the stations that belong to Quito. The main purpose of the repository is to speed up the divulgation of meteorologic files, allowing users to share and get files in specific formats according to their needs.

In this document you can find repository's development cycle, from the analysis stage, moving to design and then through the build stage under the SCRUM agile development methodology. Finally, the system is tested and deployed on a web server.

The result of this process is a repository that allows users to upload files used by INAMHI and REMMAQ in their original formats, allowing to download them in xls, CSV, txt and the original format.

INTRODUCCIÓN

ANTECEDENTES

El Instituto Nacional de Hidrología y Meteorología (INAMHI) es el encargado de proveer datos de las distintas estaciones meteorológicas que se encuentran distribuidas en las regiones del Ecuador las cuales toman medidas como: humedad, precipitación, velocidad del viento, temperatura media, etc.

Los datos generados por las distintas estaciones meteorológicas de la ciudad de Quito son utilizados por distintos profesionales y organizaciones para el cumplimiento de sus actividades de investigación por lo cual buscan técnicas y herramientas que les permita gestionar de forma correcta dicho volumen de información.

La divulgación de estos datos es adecuada debido al largo y burocrático proceso que debe realizarse para obtener datos o información específica. Los investigadores no cuentan con un espacio en el cual puedan adquirir y compartir información meteorológica, con el que puedan simplificar el proceso de adquisición de datos actual.

Los repositorios se han convertido en herramientas robustas para divulgar información dentro de distintas comunidades, volviéndose fundamentales en el área de la investigación donde las personas involucradas en un tema específico puedan compartir información de forma directa, omitiendo los procesos complicados y aumentando la productividad del investigador o equipo de investigación.

PROBLEMA DE ESTUDIO

Actualmente, el volumen de datos que se genera obliga a las organizaciones a administrarlos de una manera inteligente, utilizando herramientas de última tecnología que les permitan un correcto procesamiento y representación de estos (Moreno, 2020).

En la ciudad de Quito existen diferentes estaciones que se encargan diariamente de la producción de datos meteorológicos, los mismos que son empleados por usuarios que se

desempeñan en campos como la investigación y la educación (Moscoso, Serrano Vincenti, Jácome, Palacios, & Villacís, 2012).

El problema radica en la variedad de formatos (texto, hojas de cálculo, CSV) de los archivos generados por las distintas fuentes, provocando que la información se encuentre desordenada. Por esta razón el proceso estandarización y limpieza de los datos es una etapa compleja que requiere esfuerzos adicionales, costosos y redundantes para lograr conseguir documentos ordenados y comprensibles (Ayala, 2019).

Este inconveniente afecta a las personas que requieren datos medioambientales para sus investigaciones, pues al momento de manipular la información los formatos son incompatibles y demandan que los usuarios los modifiquen manualmente antes de poder utilizarlos.

Para solventar esta situación es necesaria la construcción de un repositorio donde los usuarios puedan acceder a la información con una estructura estandarizada, brindándoles la posibilidad de compartirla, modificarla y descargarla en un formato que se ajuste a sus necesidades.

JUSTIFICACIÓN

La utilización de sistemas para gestionar la información se ha convertido en una parte fundamental en las organizaciones, debido a que la principal fuente de riqueza y prosperidad en la economía actual es la producción y distribución tanto de información como de conocimiento. Las empresas han visto necesario emplear internamente tecnologías Web para sus colaboradores como repositorios, blogs y wikis, incentivando el trabajo colaborativo y el intercambio de información entre distintos usuarios y terminales (McLeod Jr. & Raymond, 2000).

Los repositorios brindan una mayor visibilidad y capacidad de contribución, aumentando considerablemente el impacto de la producción científica. Además, destacan por lograr la preservación a largo plazo de sus contenidos y el libre acceso a los mismos (Medina, 2006). Este proyecto tiene la finalidad de crear un repositorio Web que le permita a las personas que

requieren archivos con datos meteorológicos tener acceso a información que pueda ser compartida y descargada en formatos preestablecidos que se ajusten a sus requerimientos.

OBJETIVOS

Objetivo General

Crear un repositorio Web para datos de estaciones meteorológicas de la ciudad de Quito.

Objetivos específicos

Analizar los requerimientos planteados por el usuario que manipula datos meteorológicos para la creación del repositorio.

Diseñar un repositorio Web, para almacenar y compartir archivos en múltiples formatos (xls, CSV, txt), que contengan información meteorológica.

Construir el repositorio Web empleando un modelo de desarrollo de software incremental, a través de la metodología ágil SCRUM.

Poner en producción el repositorio construido como producto del proyecto de titulación.

METODOLOGÍA

Se establecen los parámetros de la metodología ágil para el desarrollo de proyectos de software SCRUM. Adicionalmente se documentan las técnicas y herramientas involucradas en la investigación para efectuar la construcción del repositorio.

Para el proceso de investigación se emplearon herramientas como la reunión mediante la cual se pudo recolectar los requerimientos, realizar el análisis y planificación respectiva para el diseño del sistema. Se utilizaron las herramientas y procedimientos planteados por el marco de trabajo de SCRUM, esto permitió llevar una mayor organización en cuanto a tiempo y gestión de las tareas establecidas. Se determinó la utilización de esta metodología de desarrollo debido a la importancia de llevar un control y seguimiento sobre el proceso de construcción, teniendo en cuenta que cada incremento al sistema debe tener una funcionalidad tangible que permita evaluarlo y realizar la depuración de errores respectiva que permita avanzar a etapas posteriores.

La técnica de recolección utilizada a lo largo del proyecto fue el análisis documental, debido a la vasta información existente en fuentes de datos secundarias como libros, artículos académicos y tesis relacionados al tema del proyecto que son los repositorios. Adicionalmente se investigaron fuentes de capacitación y aprendizaje para la adquisición de conocimiento sobre el lenguaje de programación JavaScript con el framework Node.js específicamente para la realización del presente proyecto.

Roles SCRUM

La asignación de roles dentro del proyecto se estableció de la siguiente forma:

- **Product Owner:** el encargado de este rol fue Rodrigo Efraín Tufiño, tutor del proyecto, quien supo expresar las necesidades que debían ser cubiertas por este proyecto.
- **Scrum Máster:** Rodrigo Efraín Tufiño, tutor del proyecto, fue quien planteó la idea general, de esta forma se definieron los requerimientos que componen el Product Backlog que sirvió como guía para el equipo de desarrollo a lo largo del proyecto. Adicionalmente fue el responsable de gestionar las reuniones para la revisión de los incrementos del producto.
- **Equipo de desarrollo:** El equipo desempeñó su trabajo bajo el paradigma propuesto por SCRUM, entregando incrementos con mejoras tangibles en cada uno de los Sprint reviews. Esto aportó a la detección de errores y a la mejora de aspectos tanto funcionales como visuales del producto.
- **Usuarios:** Se establecieron dos tipos de usuario: invitados e investigadores, quienes van a tener interacción con el sistema.

CAPÍTULO I

MARCO TEÓRICO

Este capítulo describe conceptos fundamentales para que el lector tenga un acercamiento a temas como el paradigma del acceso abierto a la información, el objetivo de los repositorios en el área de la investigación, las herramientas que se emplean en el desarrollo del proyecto, así como también los instrumentos de administración del proyecto y una noción sobre la meteorología.

1.1 OPEN ACCESS

Es un paradigma que permite el libre acceso a recursos producidos por la comunidad científica o académica, omitiendo las barreras de todo tipo de derecho de autor que exista sobre ellos. El tipo de elementos que pueden ser compartidos bajo este paradigma incluye documentos, datos en bruto, documentos audiovisuales, etc. Este tipo de iniciativas aparecen a principios de los años noventa, más específicamente en el año 1991 con la publicación de Arvix, un repositorio de física, matemáticas y computación (Soares Guimaraes et al., 2012).

Open Access se fundamenta en el control sobre los derechos de autor de las publicaciones, haciendo un énfasis en el factor económico debido al aumento de los precios de las revistas científicas a principios de la década de los 80, siendo considerado una respuesta de la sociedad en contra de un abuso editorial que establecía una brecha para quienes no podían costear el acceso a dichas publicaciones (Suber, 2014).

Son 3 los compromisos que avalan y respaldan la existencia de Open Access internacionalmente, estos son la Declaración de Budapest de 2002 (Budapest Open Access Initiative, 2002), la Declaración de Bethesda de 2003 (Bethesda Statement on Open Access Publishing, 2003) y en el mismo año la declaración de Berlín. Todas ellas han motivado a que múltiples grupos y organizaciones reconozcan a Open Access no solo como un medio de difusión de conocimiento, sino también como una manera de preservar la información y que

trascienda en el tiempo, esto se puede lograr a través del uso de repositorios que puedan ser gestionados por las propias organizaciones que adopten el paradigma.

1.2 REPOSITARIOS

Hoy en día la cantidad de información que se genera es significativamente alta, el desarrollo de las tecnologías de la información y comunicación (TICS) ha marcado un hito en la manera en la cual los conocimientos y la información es transmitida, por ello una de las soluciones más implementadas son los repositorios, donde la gente puede guardar y obtener la información desde un solo lugar de forma centralizada, con mayor visibilidad y generando mayor impacto (Álvarez et al., 2011).

El objetivo primordial de un repositorio es el de obtener, almacenar, ordenar, preservar y difundir adecuadamente la información histórica obtenida de la investigación tanto científica como empírica, realizada por profesionales de múltiples áreas del conocimiento (Barton & Waters, 2005). Esto representa una mejor utilización de un recurso tan valioso como la información para instituciones educativas como gubernamentales.

1.2.1 Tipos de repositorios

Hoy en día con la gran variedad de información que se encuentra dispersa en el Internet, los repositorios digitales suelen ser diversos dependiendo de la temática y el entorno en que se los necesite (Corral et al., 2014). A pesar de esto existen distintas categorías como:

- **Repositorios institucionales:** son los encargados de almacenar y difundir información generada dentro de instituciones educativas públicas o privadas en sus distintas áreas.
- **Repositorios disciplinares:** son los encargados de almacenar y difundir información generada dentro de instituciones educativas públicas o privadas pero enfocada a una sola área.
- **Repositorios temáticos:** son los encargados de almacenar y difundir información generada dentro de instituciones educativas públicas o privadas, en un tema específico.

- **Repositorios de datos:** son los encargados de almacenar y difundir información generada al interior de alguna entidad pública o privada.

1.3 HERRAMIENTAS DE DESARROLLO

1.3.1 Node.js

Es un entorno de servidor manejado por eventos que utiliza un modelo sin bloqueo de entradas y salidas basado en JavaScript, destacando su rendimiento y bajo consumo de recursos, lo que lo hace adecuado para aplicaciones de procesamiento de datos en tiempo real que funcionen a través de múltiples dispositivos (Lei et al., 2014). El sitio oficial del proyecto es (<https://nodejs.org/es/>).

1.3.2 JavaScript

JavaScript es un lenguaje de programación interpretado, lo que quiere decir que no necesariamente se debe utilizar algún interprete para realizar la ejecución de un programa, sino que basta con probar directamente en cualquier navegador, ya que se encuentra integrado dentro del motor de los navegadores más populares (Luna, 2019). Las últimas versiones de JS permiten que sea un lenguaje tanto para el lado del cliente como del servidor, donde se puede potenciar las interfaces de usuario y páginas Web dinámicas (Comunidad MDN, 2020). Información y recursos referentes al proyecto se encuentran en el sitio (<https://developer.mozilla.org/es/docs/Web/JavaScript>)

1.3.3 Handlebars

Es un lenguaje de plantillas de JavaScript que permite la creación de sentencias en lenguaje HTML, brindando la posibilidad de formatearlo y mostrarlo en bloques utilizando archivos JSON como origen de los bloques mencionados. Es esencial a la hora de separar el frontend del backend, de esta forma se mantiene organizada la estructura del código (Katz, 2011).

1.3.4 MongoDB

Mongo es una base de datos no estructurada que ofrece a sus usuarios escalabilidad y particionamiento. Utiliza documentos para hacerla flexible, escalable y rápida pues brinda alta productividad al momento de manipular datos, destacando su dinamismo para el manejo de esquemas (González Valiente, 2020). Las herramientas y soluciones que ofrece MongoDB están presentes en el portal oficial del proyecto (<https://www.mongodb.com/what-is-mongodb>).

1.3.5 OAuth2

Se define como un protocolo utilizado en aplicaciones web, cuyo objetivo es autorizar al usuario final la utilización o empleo de interfaces de programación de aplicaciones a través de la implementación de credenciales únicas que se generan en forma de tokens, garantizando la seguridad y permitiendo al sistema conocer quién está utilizándolo (Sanso & Richer, 2017).

1.3.6 Heroku

Heroku es una plataforma como servicio la cual brinda la facilidad de desplegar, administrar y escalar aplicaciones en Internet (<https://www.heroku.com/>). Una de sus grandes ventajas es la diversidad de lenguajes con la que es compatible como son: Node.js, Python, go, entre otros (Heroku, s.f.).

1.3.7 Git

Git es un sistema de control de versiones tanto de documentos planos como de software. Es de código abierto y está diseñado principalmente para la colaboración entre desarrolladores, permitiendo tener un registro de los cambios en el código que se va desarrollando sin importar la escala del proyecto (GIT, s.f.).

1.4 HERRAMIENTAS E INSTRUMENTOS PARA LA GESTIÓN DE PROYECTOS

Es importante seleccionar los instrumentos y herramientas acorde a las necesidades del proyecto con la finalidad de gestionarlo correctamente además de tener un control organizado sobre el mismo.

1.4.1 Cuestionarios abiertos

Los cuestionarios abiertos son usados cuando se realiza una entrevista a una persona, estos contienen un conjunto de preguntas abiertas para que puedan ser contestados con total libertad y sin limitaciones. Facilitan el levantamiento de información, por lo general son utilizados cuando se quiere conocer sobre un tema específico. En la gestión de proyectos este instrumento es uno de los principales para definir los requerimientos y para determinar el alcance del proyecto.

1.4.2 Taiga

Es una plataforma de código abierto para la gestión de proyectos bajo el paradigma SCRUM que permite la utilización de plantillas adaptables a las necesidades del proyecto (<https://www.taiga.io/es>).

1.5 METEOROLOGÍA

La meteorología o también llamada física de la atmósfera, es una ciencia exacta encargada de la predicción de los múltiples fenómenos atmosféricos, cuyos resultados aportan al desarrollo de actividades como la agricultura, estrategias militares, prevención de desastres naturales, etc. Adicionalmente se encarga de la investigación acerca del funcionamiento del sistema climático (García, 2012). Se fundamenta en teorías y estadísticas que sustentan la aparición de dichos fenómenos, a la vez que busca interpretar las causas exactas que los provocan. Para lograrlo se debe realizar el análisis de magnitudes o conocidas también como variables meteorológicas las cuales varían en función del tiempo y el espacio (Rodríguez Jiménez et al., 2004).

Las magnitudes más destacadas se listan a continuación:

➤ **Temperatura:** Magnitud que está relacionada al movimiento aleatorio medio de las moléculas físicas, como en el caso de la atmósfera, el aire. Esta magnitud es expresada en grados centígrados. De entre todas las magnitudes, la temperatura es la que mayor exposición tiene en medios de comunicación y difusión en donde se utilizan términos como temperaturas máximas y mínimas para el entendimiento de un público común. La rapidez del movimiento de las partículas es directamente proporcional a la temperatura, por lo tanto, mientras más movimiento presenten mayor será la temperatura (Rodríguez Jiménez et al., 2004).

La temperatura puede ser comprendida de forma conceptual por el ser humano, sin embargo, la percepción física a través de los sentidos de esta magnitud puede variar dependiendo del individuo, por lo que la medición se torna demasiado simple y con poca precisión. Para la medición de la temperatura se debe contar con instrumentos especializados y con una calibración adecuada que asegure la máxima precisión de los resultados. El instrumento empleado para efectuar mediciones de temperatura es conocido como termómetro, cuya invención se atribuye a Galileo Galilei en 1593 (Rodríguez Jiménez et al., 2004). Existen diferentes tipos de termómetros, pero la estructura del más sencillo está constituida por un tubo de vidrio en cuyo interior se aloja un líquido que se expande o se contrae a través del tubo permitiendo realizar la medición. Con el fin de medir esta magnitud, se deben tener a consideración ciertas propiedades que se ven afectadas cuando esta se modifica: el volumen, el material, el color del objeto, entre otras (Paucar Quinteros, 2017).

➤ **Presión atmosférica:** Magnitud que representa la fuerza ejercida por los gases que constituyen la atmósfera sobre un cuerpo. La presión atmosférica se ve afectada directamente por la altitud a la que se encuentre un cuerpo, es decir, que mientras se encuentre a más altura, la cantidad de aire será menor por consecuencia la presión atmosférica disminuirá. La unidad de medida empleada para la medición de esta magnitud es el Pascal, la cual es registrada a

través de un instrumento especializado conocido como barómetro (Rodríguez Jiménez et al., 2004).

- **Velocidad del viento:** Se define viento como el movimiento de aire de un lugar a otro, para que exista viento deben intervenir múltiples factores como la temperatura y la presión de los lugares en cuestión. La diferencia de presión y la diferencia de temperatura producen cambios tanto en la densidad como en el volumen del viento. Esta magnitud obtiene medidas referentes al movimiento del aire con respecto a la superficie de la tierra en una dirección y velocidad determinadas y es expresada en metro por segundo (m/s). Las medidas son obtenidas a través un artefacto constituido por una rueda que se une a brazos con terminales cóncavos que giran en un eje vertical, las cuales giran en función de la intensidad del viento. El instrumento en cuestión es llamado anemómetro de copas (Oliva, 2012).
- **Dirección del viento:** Es la dirección desde la cual sopla el viento, esta es expresada en grados a partir del norte geográfico. Para su medición se emplean veletas, las cuales se encargan de determinar el origen del viento. El tipo de viento se determina por la referencia geográfica de donde se origine (Rodríguez Jiménez et al., 2004).
- **Humedad:** Expresa la cantidad de vapor de agua que se encuentra contenido en la atmósfera. Dicha cantidad depende de la ubicación geográfica, lluvias recientes, vegetación en el entorno, etc. El contenido de humedad de la atmósfera se define de las siguientes formas:
 - **Humedad específica:** Cantidad vapor de agua que se encuentra dentro de un kilogramo de aire.
 - **Humedad absoluta:** Cantidad de vapor de agua que se encuentra dentro de un metro cúbico de aire.
 - **Razón de mezcla:** Cantidad de vapor de agua que se encuentra dentro de un kilogramo de aire seco.

- **Humedad relativa:** Cantidad de vapor de agua que se encuentra dentro de una masa de aire antes de cambiar de estado gaseoso a líquido.

De entre todas, la más destacada es la humedad relativa expresada en porcentaje (%) la cual brinda una noción de cuán cerca se encuentra una masa de aire en cambiar el estado del vapor de agua que contiene. El instrumento empleado para la medición de la humedad es conocido como psicómetro (Rodríguez Jiménez et al., 2004).

➤ **Precipitación:** en meteorología esta magnitud tiene relación a la caída de agua sólida o líquida por la condensación del vapor sobre la superficie terrestre. Esta magnitud es expresada en milímetros de agua (mm) (Ayarzagüena Porras et al., s.f.).

1.5.1 Estaciones Meteorológicas

Son instalaciones cuya función principal es la medición y registro constante de magnitudes climáticas, a través de distintos sensores específicos que captan las diferentes variables del entorno. Los instrumentos encargados de la medición de las variables y que se incluyen en una estación meteorológica son: termómetro, barómetro, piranómetro, pluviómetro, anemómetro y psicómetro (Bravo et al., 2012).

Las condiciones ambientales se encuentran dadas por distintos parámetros como lo son: la medición y determinación de variables atmosféricas en un determinado tiempo. Con el fin de brindar información que sea útil con respecto a los distintos parámetros que pueden influir en el clima (Talavera, 2014).

Las estaciones meteorológicas se encuentran distribuidas a lo largo del Ecuador en zonas estratégicas para poder contar con un adecuado monitoreo del comportamiento climatológico.

1.5.1.1 Estaciones meteorológicas convencionales. En este tipo de estaciones por lo general se realizan mediciones de variables como: temperatura, nubosidad, precipitación, tormentas, nieblas, etc. Donde las mediciones son asistidas por un observador meteorológico, el cual a su vez se encarga del respectivo mantenimiento de los instrumentos y del lugar de observación de tal forma que se obtengan unas buenas mediciones (Gattinoni et al., 2011).

1.5.1.2 Estaciones meteorológicas automáticas. Las estaciones meteorológicas automáticas según la Organización Mundial Meteorológica (OMM) son aquellas donde “las observaciones son realizadas y transmitidas automáticamente” (WMO, 2011). Sin embargo, aún se requiere que exista personal supervisando posibles fallas de comunicación, de instrumentación y poder actuar de manera rápida para evitar que se produzcan pérdidas masivas de mediciones.

1.6 METODOLOGÍAS ÁGILES

Surgen con el objetivo de reducir la probabilidad de fracaso relacionada con la planificación de costos, tiempo y funcionalidades dentro de un proyecto de desarrollo de software, permitiendo que exista mayor rapidez en la construcción y que el equipo de desarrollo tenga una mejor respuesta ante cambios que puedan presentarse a lo largo del proceso. A diferencia de las metodologías tradicionales, las metodologías ágiles presentan mayor flexibilidad ante los cambios y no están regidas estrictamente por la documentación.

- El Manifiesto ágil es documento que contiene la filosofía “ágil”, en el mismo se destacan las siguientes características (Canós et al., 2012).
- La interacción y afinidad de los miembros del equipo de desarrollo es un factor determinante para obtener un producto de software exitoso. Es recomendable definir el equipo y permitir que este construya un entorno a la medida de sus requerimientos.
- La documentación es importante, pero no indispensable. A diferencia de las metodologías tradicionales, la producción de documentación no es una prioridad dentro de la filosofía

ágil. Solamente si el documento es un factor decisivo para la toma de decisiones se procede con su creación y debe caracterizarse por ser lo más concreto posible.

- La colaboración entre el cliente y el equipo de desarrollo es fundamental para la actualización constante de requerimientos que permitan generar un proyecto de software exitoso y de calidad.
- Flexibilidad y adaptabilidad ante cambios de múltiples tipos que pueden surgir en el transcurso del proyecto.

1.6.1 SCRUM

Se constituye como una metodología de desarrollo iterativo e incremental, con cerca de 10 años de trayectoria, es adecuado para proyectos donde los cambios de requisitos sean recurrentes (Canavid et al., 2013).

Se constituye en dos características principales:

- El proceso de desarrollo ocurre a través de iteraciones denominadas Sprint. Se debe exponer al cliente el resultado de cada sprint, es decir, un avance que incrementa la funcionalidad del proyecto.
- Se programan reuniones diarias a lo largo del proceso de desarrollo las cuales deben ser planificadas y coordinadas con los miembros del equipo de desarrollo, en donde se gestionan responsabilidades y distribución de tareas. El equipo de trabajo se encuentra constituido por:
 - **Product owner:** persona que conoce del negocio y los requerimientos del cliente.
 - **Equipo de desarrollo:** personas profesionales autoorganizadas, encargadas de cumplir con los objetivos definidos en el Sprint.
 - **Scrum máster:** encargado de asegurar que el marco metodológico se está implementando correctamente.
 - **Usuarios:** destinatarios finales del producto.

Los componentes y elementos adoptados por Scrum son:

- **Reuniones diarias:** se realizan reuniones diarias con una duración de 10-15 min donde se exponen los avances, problemas o inconvenientes que surgieron y así se va realizando un seguimiento de las tareas.
- **Scrum board:** es una pizarra la cual contiene las tareas en distintas etapas las cuales son: tareas por hacer, tareas en proceso y tareas realizadas.
- **Sprint:** son las listas de tareas que el equipo de desarrollo debe elaborar las cuales tienen una duración determinada.
- **Product backlog:** es el inventario de requisitos estructurales para el desarrollo del proyecto.

CAPÍTULO II

ANÁLISIS Y REQUERIMIENTOS

El presente capítulo contiene el análisis realizado para el levantamiento de requerimientos del sistema donde se tendrá en cuenta el análisis de factibilidad.

2.1 ANÁLISIS DE FACTIBILIDAD

El presente análisis involucra tres aspectos fundamentales para la realización del proyecto, estos son la viabilidad técnica, económica y operacional, entre los cuales debe existir un balance y coherencia que permita el desenvolvimiento correcto del proyecto.

2.1.1 Viabilidad Técnica

Para el desarrollo del repositorio se optó por tecnologías nuevas y código abierto que permitan al sistema rendir a su máxima capacidad, a su vez esto le brinda una escalabilidad considerable en caso de que se requiera aumentar mejoras o módulos nuevos en un futuro.

Tabla 1

Características del hardware disponible.

Tipo de computadora	Procesador	Almacenamiento	Memoria RAM	Función
Laptop HP	i5 7th gen	256Gb SSD	4gb	Investigación, desarrollo, pruebas
PC	i7 2nd gen	1Tb HD	6gb	Investigación, desarrollo, pruebas

Nota. Características técnicas de los equipos de cómputo disponibles para el desarrollo del aplicativo.

Elaborado por: Los autores.

Tabla 2*Herramientas de desarrollo.*

Nombre	Tipo de Herramienta	Función
MongoDb	Motor de base de datos	Almacenamiento de información
Visual Studio Code	Entorno de desarrollo (IDE)	Editor de código
Github	Repositorio de código de proyectos de software	Repositorio de código fuente
Heroku	Alojamiento Web	Plataforma de despliegue de aplicaciones.

Nota. Lista de herramientas de desarrollo disponibles para el desarrollo del aplicativo.

Elaborado por: Los autores.

2.1.2 Viabilidad Económica

Económicamente el proyecto demuestra ser rentable. Los costos de hardware están cubiertos en su totalidad debido a la previa adquisición de los equipos de cómputo. Las soluciones de software listadas son todas de código abierto por lo que su adquisición y utilización no involucran ningún tipo de inversión. Los costos de los servicios listados son establecidos por los proveedores de Internet contratados por los autores, cuya facturación es mensual. La remuneración listada para los programadores del proyecto hace referencia al sueldo promedio de un desarrollador junior dentro del ámbito laboral de Ecuador. Los gastos académicos listados hacen referencia al pago individual de cada uno de los autores para acceder a la Unidad de Titulación de la carrera de Ingeniería de Sistemas de la Universidad Politécnica Salesiana.

Tabla 3

Tabla de consideraciones de costos para el desarrollo del proyecto.

Tipo	Descripción	Precio unitario	TOTAL, Semestral
Hardware	Laptop HP	\$800	\$800
	PC	\$650	\$650
Software	MongoDB	Open Source	\$0
	Visual Studio Code	Open Source	\$0
	Github	Open Source	\$0
	Heroku	Open Source	\$0
Servicios	Servicio de Internet 100 Mbps	\$40	\$240
	Servicio de Internet 5 Mbps	\$20	\$120
Recursos humanos	Programador	\$480	\$2880
	Programador	\$480	\$2880
Gastos Académicos	Pago ingreso unidad de titulación	\$549	\$1098
TOTAL			\$8668

Nota. Costos de equipos y programas disponibles para el desarrollo del aplicativo, los costos descritos son cubiertos por los autores.

Elaborado por: Los autores.

2.1.3 Viabilidad Operacional

La implementación del repositorio brinda a los usuarios un entorno donde el principal objetivo es compartir datos meteorológicos de una forma óptima, evitando procesos complicados que retrasen el cumplimiento de sus trabajos de investigación. Los usuarios que operan con el repositorio son profesionales, investigadores y personas en general que se desenvuelven en el

área de la meteorología, los cuales cuentan con conocimientos técnicos de nivel básico/intermedio, es decir, que no tienen amplios conocimientos en cuanto a la utilización de una aplicación Web, pero tienen la capacidad de aprender a manejarla y realizar las tareas esenciales para las cuales fue desarrollada, adicionalmente la interfaz del sistema se presenta como un entorno amigable para el usuario, lo que le permite desenvolverse sin inconvenientes dentro del mismo.

2.2 ANÁLISIS DE REQUERIMIENTOS

2.2.1 Alcance

Este proyecto se centra en la creación de un repositorio de datos meteorológicos, que almacene información generada por las estaciones meteorológicas de la ciudad de Quito, la cual debe ser cargada al repositorio por un usuario registrado, permitiendo que individuos y organizaciones cuyas actividades dependen de la manipulación de esta información, puedan cumplirlas de forma exitosa, en menor tiempo y de una manera más sencilla en comparación al proceso tradicional.

El repositorio debe simplificar el acceso y divulgación de la información, esto implica:

- Que posea una interfaz amigable que le permita al usuario interactuar con el sistema de forma intuitiva.
- Que permita la descarga de datos meteorológicos en formatos como hojas de cálculo (xls, xlsx), archivos separados por comas (CSV) y el formato original del archivo subido por el usuario.
- Que permita el registro y autenticación de usuarios para una experiencia personalizada.
- Que permita la búsqueda de contenido en función de las necesidades del usuario.

2.2.2 Definiciones acrónimos

Repositorio: Depósito de documentos digitales, que tiene como fin organizarlos, almacenarlos y compartirlos de forma abierta para todas aquellas personas que forman parte de una comunidad o grupo determinado.

Datos meteorológicos: Datos obtenidos por estaciones meteorológicas a través de múltiples sensores. Algunas variables meteorológicas comunes son temperatura, humedad, velocidad del viento, dirección del viento, radiación solar y precipitación.

Open Access: acceso inmediato a la información sin necesidad de un registro o un pago.

Aplicación Web: Programa de computador que se encuentra alojado en Internet y al cual se accede por medio de un navegador Web.

Base de datos: Sistema informático que permite el almacenamiento de grandes volúmenes de datos para su procesamiento y visualización.

2.2.3 Descripción General

En este apartado se mencionan los aspectos generales del sistema como su perspectiva, funciones y características.

2.2.4 Perspectiva del producto

El sistema que se desarrolla es un repositorio permite la divulgación y adquisición de datos meteorológicos de una forma rápida y confiable.

2.2.4.1 Funciones del producto. El sistema que se desarrolla para un ambiente Web, para lo cual los usuarios que deseen publicar datos meteorológicos lo realizarán a través de la carga de archivos al sistema. Para esto deberán registrarse previamente en la plataforma.

Este tipo de usuarios dispondrá de los siguientes módulos:

Autenticación de usuario: permite al usuario registrarse y obtener un perfil dentro de la aplicación para que pueda publicar información en la aplicación.

Edición de perfil: el usuario puede añadir o actualizar información personal de su perfil.

Subir información: se procesa el archivo proporcionado por el usuario y se almacena en la base de datos.

La plataforma permite el ingreso de usuarios “visitantes” al repositorio. Este tipo de usuarios pueden hacer uso de los siguientes módulos:

Búsqueda: este módulo permite realizar una búsqueda específica por estación y magnitud

Descarga de archivos: en este módulo se procesa y pone a disponibilidad la descarga de información en 3 tipos de formatos específicos (xls, CSV, archivo original).

2.2.4.2 Características de usuarios. A continuación, se describen las características con las que cuentan los usuarios identificados que van a hacer uso del repositorio.

Tabla 4

Tabla de Características de Usuarios.

Tipo de Usuario Características	Investigador	Invitado
Formación	-Usuario con estudios superiores relacionados a la investigación científica, climatología y meteorología.	Usuario con una formación básica/intermedia con interés en estudiar el clima para fines de investigación, educativos y didácticos.
Habilidades	-Interpretar datos meteorológicos. -Predecir fenómenos climatológicos en base a los datos. - Manejo básico de aplicaciones Web.	-Manejo Básico de aplicaciones Web. -Facilidad para aprender y descubrir nuevas temáticas.

Actividades	Subir datos. Agregar descripción de los datos. Editar perfil. Descargar datos Visualizar historial de datos.	Buscar datos. Visualizar datos. Descargar datos
--------------------	--	---

Nota. Descripción de las características de los usuarios que van a interactuar con el repositorio.

Elaborado por: Los autores.

2.2.4.3 Restricciones.

- El usuario investigador no puede subir más de un archivo a la vez.
- El usuario invitado no tiene los privilegios para subir datos.
- Los formatos de los archivos meteorológicos de subida que serán admitidos por el sistema serán dos: Archivos de texto (txt), archivos de hojas de cálculo (xls, xlsx).
- Los formatos disponibles para la descarga de datos serán tres: xls, CSV, archivo original.

2.2.5 Requerimientos Funcionales

Tabla 5

RF01: Creación de Usuario.

Código:	RF01	Módulo:	Investigador	Prioridad:	ALTA
Descripción	El usuario ingresa su información para obtener credenciales de acceso al sistema y poder subir y publicar un archivo.		Precondición	Ninguna	
Función	Crear un nuevo usuario en el sistema				
Entrada	Nombre, correo, contraseña, confirmación de contraseña				

Proceso	Abrir página principal. Ingresar a la página de registro. Llenar los campos que se muestran en pantalla (Nombre, correo, contraseña, confirmación de contraseña). Confirmar la información para guardarla.	
Salida Exitosa	Salida Conflicto	
Creación de usuario exitoso	No coinciden las contraseñas	

Nota. Tabla descriptiva del requerimiento funcional creación de usuario.

Elaborado por: Los autores.

Tabla 6

RF02: Inicio de Sesión.

Código:	RF02	Módulo:	Investigador	Prioridad:	ALTA
Descripción	El usuario investigador ingresa las credenciales proporcionadas en el proceso de registro para identificarse en el sistema.		Precondición	RF01	
Función	Ingresar al sistema con credenciales válidas.				
Entrada	Correo, contraseña				
Proceso	Abrir página principal. Ingresar a la página de inicio de sesión. Llenar los campos con las credenciales. Confirmar inicio de sesión.				
Salida Exitosa	Salida Conflicto				
Mostrar página de investigador	Credenciales incorrectas				

Nota. Tabla descriptiva del requerimiento funcional inicio de sesión.

Elaborado por: Los autores

Tabla 7

RF03: Logout.

Código:	RF03	Módulo:	Investigador	Prioridad:	BAJA
Descripción	El usuario investigador cierra sesión		Precondición	RF01, RF02	
Función	Salir de la página de investigador.				
Entrada	Ninguna				
Proceso	Dirigirse al enlace de cerrar sesión.				
Salida Exitosa			Salida Conflicto		
Mostrar página principal del sistema			Error inesperado		

Nota. Tabla descriptiva del requerimiento funcional Logout.

Elaborado por: Los autores.

Tabla 8

RF04: Recuperación de contraseña.

Código:	RF04	Módulo:	Investigador	Prioridad:	ALTA
Descripción	El usuario investigador ingresa en el sistema el correo electrónico para recuperar contraseña.		Precondición	RF01	
Función	Recuperar contraseña				
Entrada	Correo, contraseña, confirmación de contraseña				

Proceso	Abrir página principal. Ingresar a la página de inicio de sesión. Dar clic en Olvide contraseña. Ingresar correo electrónico Ingresar contraseña nueva	
Salida Exitosa	Salida Conflicto	
Cambio de contraseña exitoso	Las contraseñas no coinciden	

Nota. Tabla descriptiva del requerimiento funcional recuperación de contraseña.

Elaborado por: Los autores.

Tabla 9

RF05: Subida de información.

Código:	RF05	Módulo:	Investigador	Prioridad:	ALTA
Descripción	El usuario investigador selecciona un archivo de su dispositivo para cargarlo al repositorio.		Precondición	RF01, RF02	
Función	Cargar un solo archivo y publicarlo				
Entrada	Correo, contraseña, confirmación de contraseña				
Proceso	Buscar archivo en la computadora Subir el archivo Establecer el nombre del archivo Elegir la estación a la que pertenecen los datos Agregar descripción. Previsualizar la información del archivo. Publicar o guardar información.				
Salida Exitosa			Salida Conflicto		

Datos almacenados correctamente	Vuelva a cargar la información
---------------------------------	--------------------------------

Nota. Tabla descriptiva del requerimiento funcional subida de información.

Elaborado por: Los autores

Tabla 10

RF06: Listar información subida recientemente.

Código:	RF06	Módulo:	Invitado	Prioridad:	MEDIA
Descripción	Se muestran los datos que se han subido recientemente al repositorio.			Precondición	Ninguna
Función	Mostrar una lista de la información cargada recientemente al repositorio.				
Entrada	Ninguna				
Proceso	Abrir página principal				
Salida Exitosa			Salida Conflicto		
Lista de información subida recientemente			No se muestra información.		

Nota. Tabla descriptiva del requerimiento funcional “listar información subida recientemente”.

Elaborado por: Los autores.

Tabla 11*RF07: Descarga de información.*

Código:	RF07	Módulo:	Invitado	Prioridad:	ALTA
Descripción	El usuario selecciona un formato para descargar el archivo deseado.			Precondición	Ninguna
Función	Descargar archivo en formato específico.				
Entrada	Ninguna				
Proceso	Seleccionar el archivo deseado para desplegar los formatos disponibles. Seleccionar el formato deseado para proceder con la descarga.				
Salida Exitosa			Salida Conflicto		
Información descargada			Información no disponible		

*Nota. Tabla descriptiva del requerimiento funcional “descarga de información”.**Elaborado por: Los autores*

2.2.6 Requisitos no funcionales

En la tabla 12 se presentan los requerimientos no funcionales, que describen aspectos relacionados al rendimiento, disponibilidad y accesibilidad del sistema.

Tabla 12*Requisitos no funcionales.*

Código	Descripción
RNF1	El sistema únicamente funciona en navegadores Web.
RNF2	Únicamente el perfil de investigador puede subir archivos en el sistema.

RNF3	El sistema debe encontrarse disponible cuando el usuario necesite acceder.
RNF4	El sistema debe ser intuitivo permitiendo la rápida adaptación del usuario.
RNF5	El sistema debe ser modular, lo que le brinda mayor escalabilidad y flexibilidad ante cambios.

Nota. Tabla que enumera los requerimientos no funcionales del sistema.

Elaborado por: Los autores.

2.3 PROCESO ACTUAL

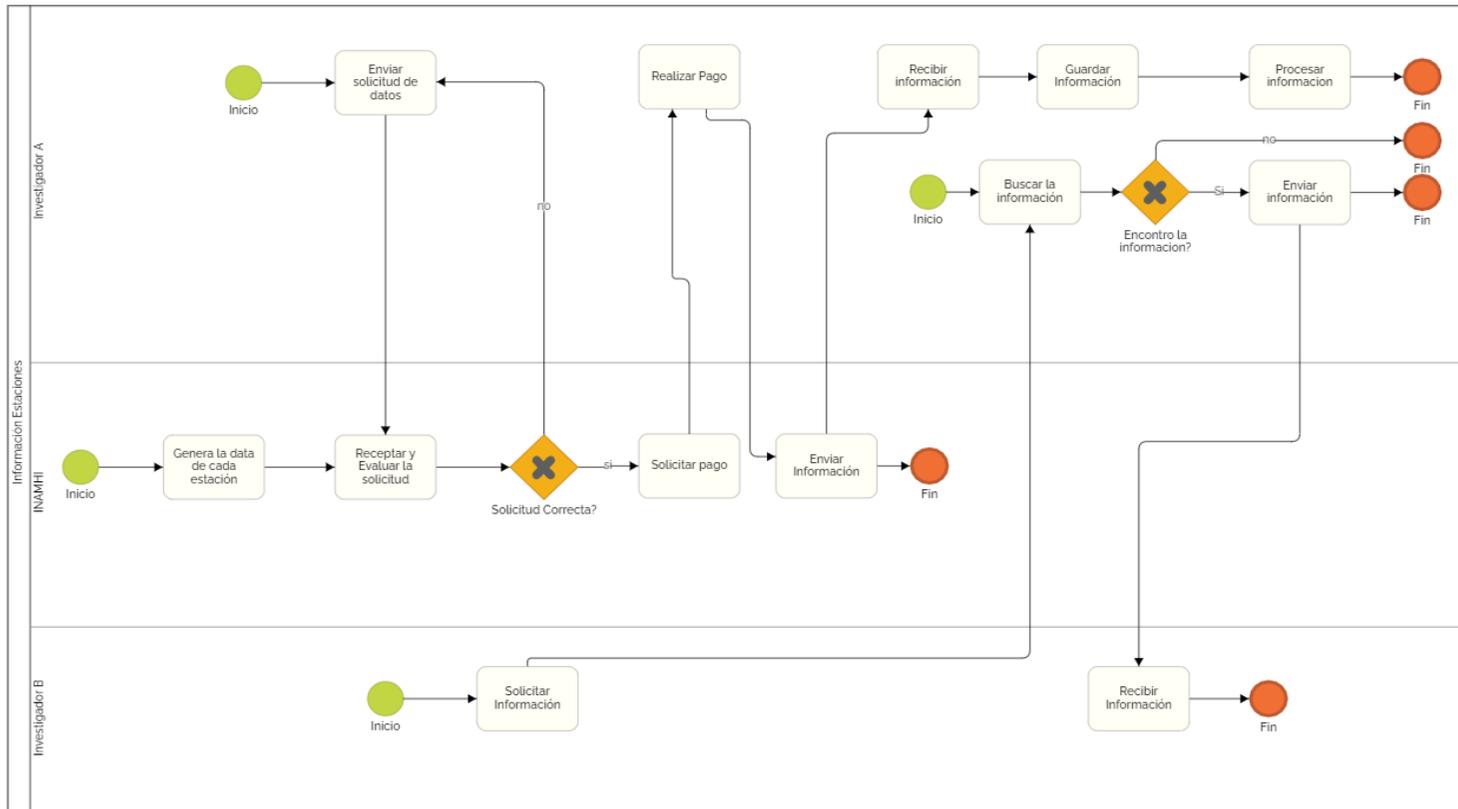
El proceso actual es un procedimiento manual que involucra al investigador y las diferentes instancias. Dependiendo de la instancia a la cual se requiere acceder a los datos tienen procesos diferentes que se detalla a continuación:

2.3.1 Proceso en INAMHI

El proceso actualmente funciona a través del envío de una solicitud formal para la obtención de datos la cual no tiene ningún costo y debe ser aprobada para continuar con el proceso. Una vez aprobada esta solicitud y dependiendo del tipo de datos requeridos, el investigador debe realizar un pago adicional para poder acceder a la información. Efectuado el pago respectivo se procede al envío de la información solicitada como lo muestra la Figura 1 (INAMHI, 2016).

Figura 1

Diagrama del proceso actual.



Nota. Proceso actual para la obtención de datos meteorológicos.

Elaborado por: Los autores.

El proceso de aprobación de la solicitud no tiene un tiempo de duración establecido y puede variar dependiendo de factores como la disponibilidad de los datos, el investigador que envía la solicitud, la determinación del organismo al evaluar la solicitud, etc.

Esto afecta directamente al desarrollo de las actividades de los investigadores, quienes deben adaptar sus cronogramas a la disponibilidad de la institución. Este proceso en adición a la manipulación y adaptación del formato de los datos adquiridos, resultan en esfuerzos adicionales y retrasos en las investigaciones.

2.3.2 Proceso en REMMAQ

El proceso en la red metropolitana de monitoreo atmosférico de Quito es mucho más sencillo debido a que sus datos son públicos y se encuentran disponibles en la siguiente dirección web <http://www.quitoambiente.gob.ec/ambiente/index.php> .

2.4 DIAGRAMAS DE CASOS DE USO.

A continuación, se muestran diagramas de casos de uso que muestran cuáles son las acciones que van a realizar los usuarios en el sistema.

2.4.1 Caso de uso del usuario Invitado

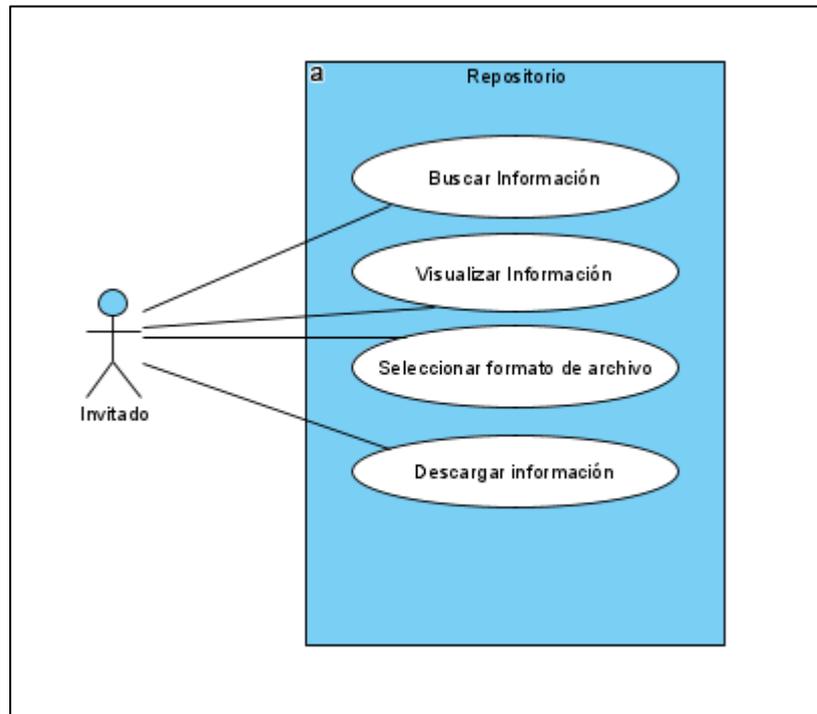
El diagrama de casos de uso de la Figura 2 describe las tareas que el usuario invitado puede realizar cuando interactúa con el repositorio. El mismo que cuenta con 4 tareas principales que se describen a continuación:

- **Buscar información:** Buscar la información requerida por el investigador dentro del repositorio.
- **Visualizar información:** Navegar y visualizar los datos meteorológicos disponibles en el repositorio.
- **Seleccionar formato de archivo:** Selección del formato de descarga disponible que mejor le convenga al usuario para su investigación.

- **Descargar información:** Descarga de datos meteorológicos en el formato seleccionado en el caso de uso anterior.

Figura 2

Caso de uso del usuario Invitado.



Nota. Diagrama de caso de uso del usuario invitado.

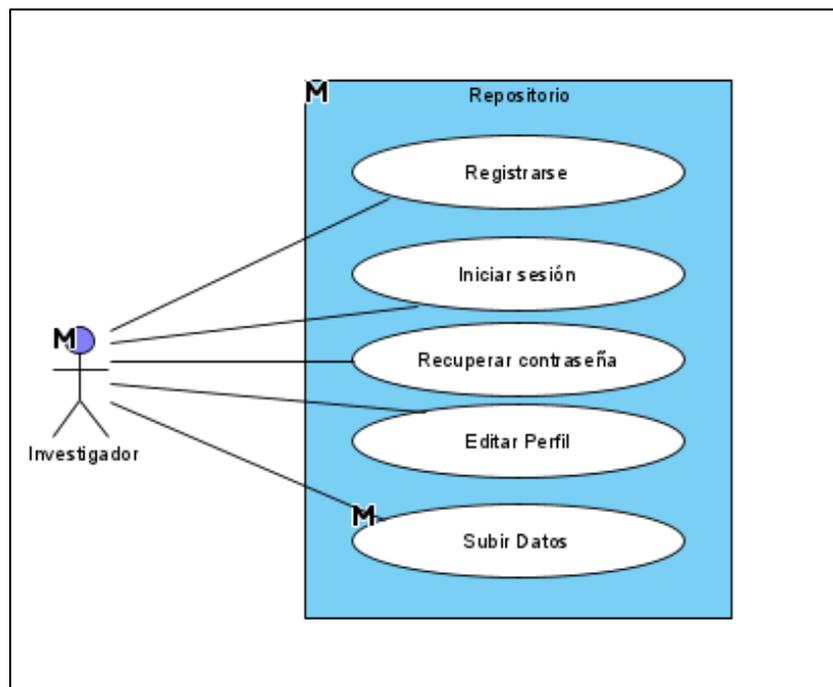
Elaborado por: Los autores.

2.4.2 Casos de uso del Investigador

A continuación, se presenta en la Figura 3 el diagrama de casos de uso del actor “Investigador” quien tiene la capacidad de realizar las mismas tareas que el actor “Invitado”, añadiendo tareas adicionales que requieren de un proceso de registro e inicio de sesión en el sistema.

Figura 3

Casos de uso del usuario investigador.



Nota. Diagrama de casos de uso del usuario Investigador.

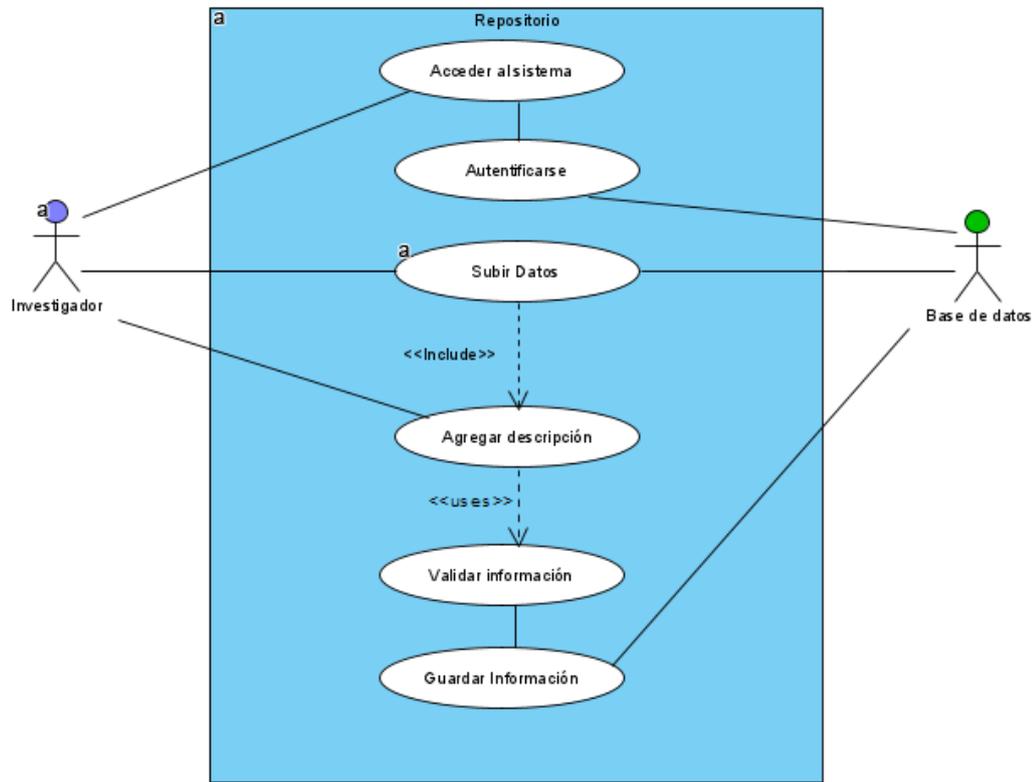
Elaborado por: Los autores.

2.4.3 Casos de Uso de Subida de datos

El siguiente diagrama de la Figura 4 representa de forma detallada la interacción del actor “Investigador” con el repositorio al subir datos, mostrando la secuencia del proceso y las tareas adicionales que se involucran para completarlo. Se muestra también a la base de datos como un actor que interactúa con el sistema.

Figura 4

Casos de uso del proceso de subida de datos.



Nota. Diagrama de casos de uso del proceso de subida de datos.

Elaborado por: Los autores.

2.5 DIAGRAMAS DE SECUENCIA

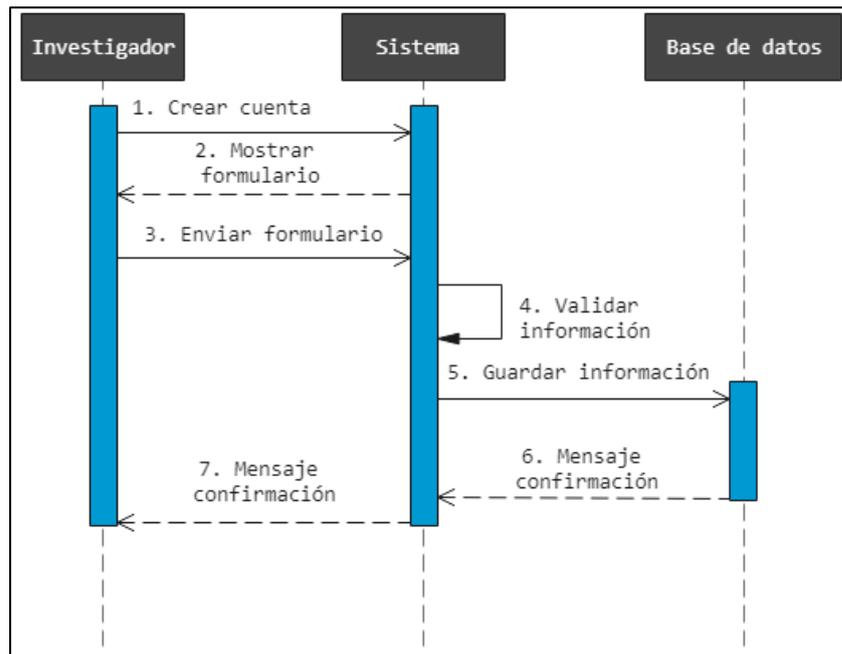
Los diagramas de secuencia muestran cómo interactúan los diferentes objetos del sistema y se empezará a detallar los diagramas del usuario investigador.

2.5.1 Diagrama de secuencia de creación de cuenta

En la Figura 5 se muestra el diagrama de secuencia de la creación de cuenta del investigador en el sistema, proceso necesario para poder subir información al repositorio. Donde se realizará la verificación de los datos ingresados por el usuario (investigador) antes de ser almacenados.

Figura 5

Diagrama de secuencia de creación de cuenta.



Nota. Diagrama de casos de uso del proceso de creación de cuenta.

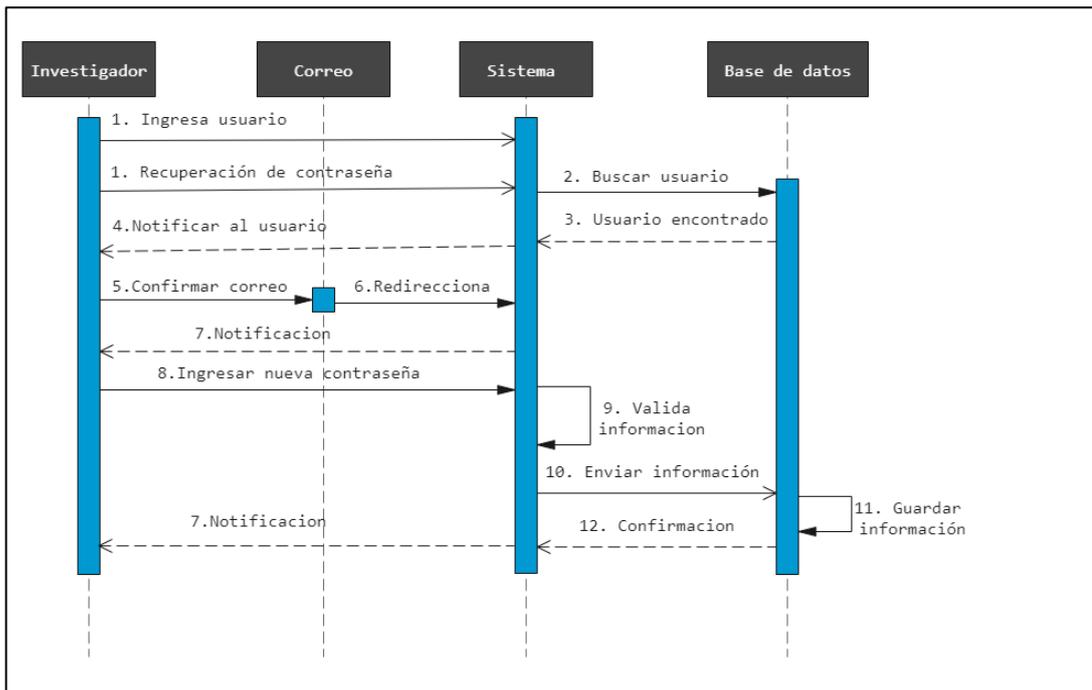
Elaborado por: Los autores.

2.5.2 Diagrama de Secuencia de recuperación de contraseña

En la Figura 6 se presenta el diagrama de secuencia para la recuperación de contraseña del investigador. El usuario investigador podrá actualizar su contraseña a través del enlace de restablecimiento enviado a su correo.

Figura 6

Diagrama de secuencia recuperación de contraseña.



Nota. Secuencia del proceso de recuperación de contraseña.

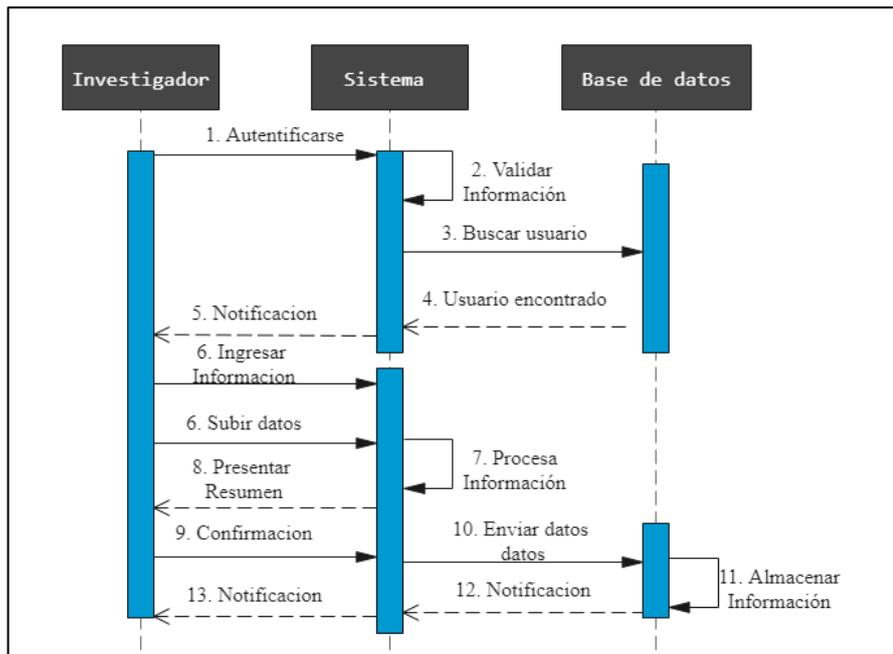
Elaborado por: Los autores.

2.5.3 Diagrama de Secuencia de subida de información

En la Figura 7 se observa el diagrama de secuencia para la subida de información al sistema. Cuando el investigador carga el archivo con la información el sistema deberá leerla y almacenarla en la base de datos.

Figura 7

Diagrama de secuencia subida de información.



Nota. Secuencia del proceso de subida de información.

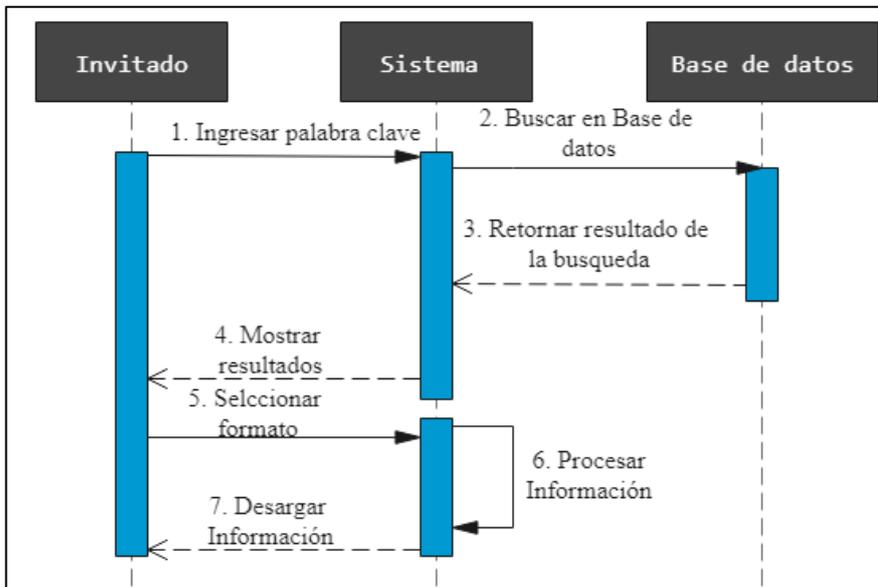
Elaborado por: Los autores.

2.5.4 Diagrama de secuencia para la búsqueda y descarga de información

En la Figura 8 se observa el diagrama de secuencia del invitado para la búsqueda y descarga de información. El usuario(invitado) podrá buscar la información por magnitud o por estación y tendrá tres formatos disponibles para la descarga.

Figura 8

Diagrama de secuencia búsqueda y descarga de información.



Nota. Secuencia de búsqueda y descarga de información.

Elaborado por: Los autores.

2.6 DIAGRAMA DE CLASES

En la Figura 9 se muestra el diagrama de clases, el mismo que representa los distintos tipos de clases involucrados en el funcionamiento del repositorio, así como también sus métodos, tipo de datos de los campos y las relaciones de agregación o composición existentes entre ellas.

Las clases que conforman el diagrama se presentan a continuación:

Investigador: Esta clase contiene atributos pertenecientes al usuario Investigador, así como los métodos específicos que realiza. Tiene una relación de agregación de uno a varios con la clase Datos.

Invitado: Esta clase no contiene atributos debido a que el usuario invitado no se registra en el sistema y puede utilizarlo con solo ingresar en la página de inicio. Se incluyen también los métodos que utiliza los cuales son búsqueda y descarga de datos. Tiene una relación de agregación de uno a varios con la clase Datos.

Encabezado: Clase que contiene atributos referentes a la información descriptiva de los datos meteorológicos. Tiene una relación de composición con de uno a varios con la tabla Datos. Se relaciona a través de composición con las tablas “encabezado” y “magnitud” de varios a uno respectivamente. Finalmente tiene una relación de agregación con la tabla motor de búsqueda.

Motor de búsqueda: Contiene el atributo palabra clave, argumento en base al cual se realizará la búsqueda de datos meteorológicos dentro del repositorio.

Datos: Clase que dispone de los atributos propios de los datos meteorológicos. Tiene relaciones de agregación de varios a uno con las tablas investigador, invitado y motor de búsqueda. Adicionalmente cuenta con relaciones de composición de varios a uno con las clases encabezado y magnitud.

Estación: Clase con atributos referentes a las distintas estaciones meteorológicas, donde se presentan su código único de identificación, el nombre de la estación, así como también atributos geográficos como latitud, longitud y elevación.

Magnitud: Clase con los atributos id y descripción, los cuales representan el número único de identificación de cada magnitud, así como su respectiva descripción. Tiene una relación de composición del tipo uno a varios con la clase Datos.

Institución: Clase cuyos atributos representan el identificador de cada institución y el nombre de estas. Tiene una relación de agregación con la clase estación del tipo uno a varios.

También se presentan las clases de tipo interfaz, las cuales emplean a las clases mencionadas anteriormente para el funcionamiento del repositorio. Se observan también los distintos métodos de cada clase interfaz.

Registro: Clase de tipo interfaz con los atributos utilizados por el usuario investigador a registrarse en el repositorio. Dispone de los métodos registrar usuario y comparar contraseña.

Iniciar_sesión: Clase de tipo interfaz que usa los datos registrados por el usuario investigador para el inicio de sesión en el sistema. Dispone del método iniciar sesión.

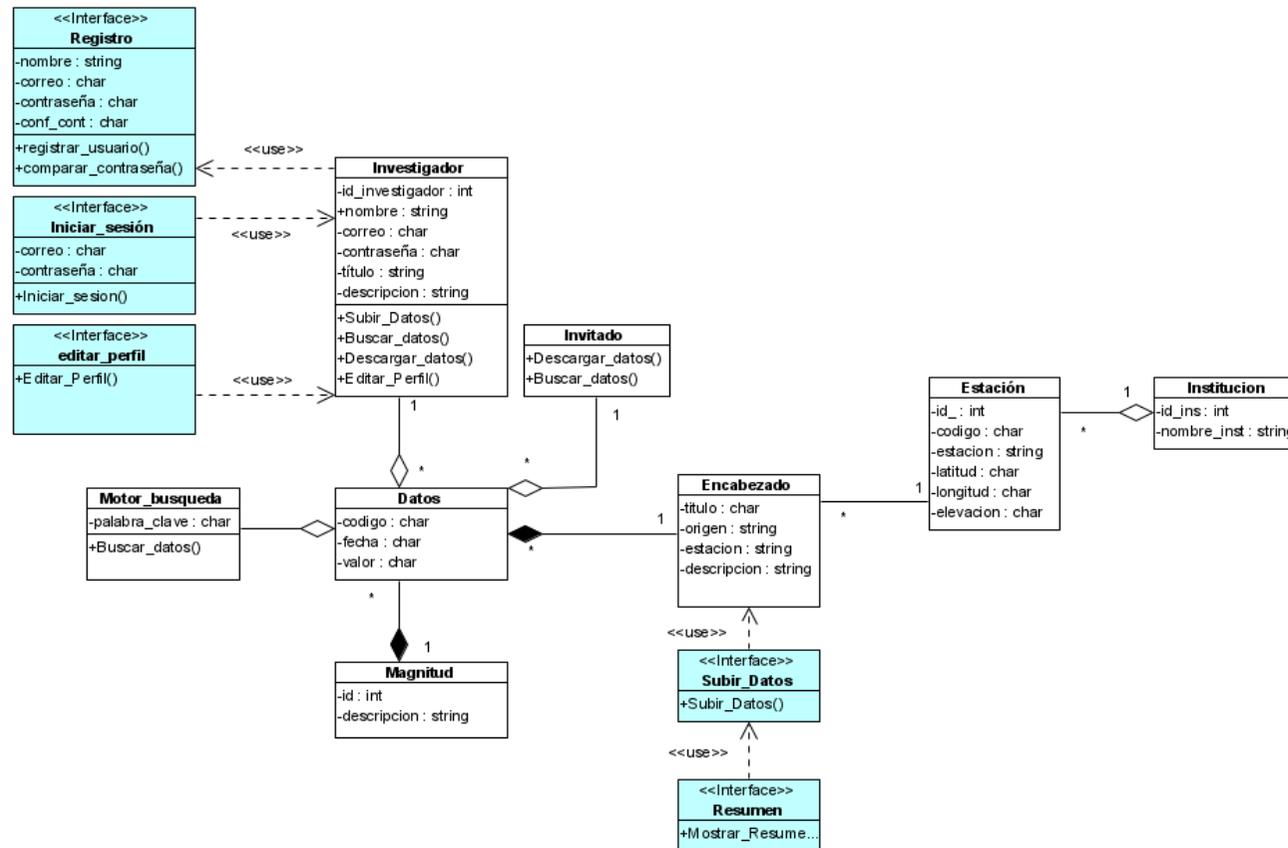
Editar_perfil: Clase de tipo interfaz que usa los datos registrados por el usuario investigador para la edición de información del perfil del usuario. Dispone del método editar perfil.

Subir_Datos: Clase de tipo interfaz que utiliza la clase encabezado para realizar la carga de datos meteorológicos al sistema. Dispone del método subir datos.

Resumen: Clase de tipo interfaz que utiliza la clase subir datos para mostrar un recuento de los datos cargados al sistema. Dispone del método mostrar resumen.

Figura 9

Diagrama de clase.



Nota. Diagrama de las distintas clases que intervienen en el repositorio.

Elaborado por: Los autores.

2.7 DIAGRAMA CONCEPTUAL DE BASE

El diagrama conceptual de la base de datos de la Figura 10 representa la estructura de cómo se almacenan los datos del repositorio, siendo la tabla “Datos” la que unifica la información subida independientemente del formato, origen, estación, etc. Las tablas mostradas en el diagrama conceptual son las siguientes:

Investigador: esta tabla contiene las columnas nombre, correo, contraseña, título y descripción. Los tres primeros campos son obligatorios, están conectados al formulario de registro. Los dos restantes son campos opcionales que estarán conectados con la interfaz de edición de perfil. Esta tabla se relaciona con la tabla encabezados a través de la clave principal código.

Encabezados: Tabla central en donde se encuentran las columnas título, origen, estación, descripción y fecha_de_subida en donde convergen las claves. Tiene una relación de varios a uno con la tabla investigador y contiene la clave foránea “código_investigador”. Adicionalmente tiene una relación de uno a varios con la tabla “Datos” y finalmente cuenta con una relación de varios a uno con la tabla “Estación” mediante la clave foránea “código_estación”.

Magnitud: Tabla que contiene las columnas código y descripción y se relaciona con la tabla “Datos”.

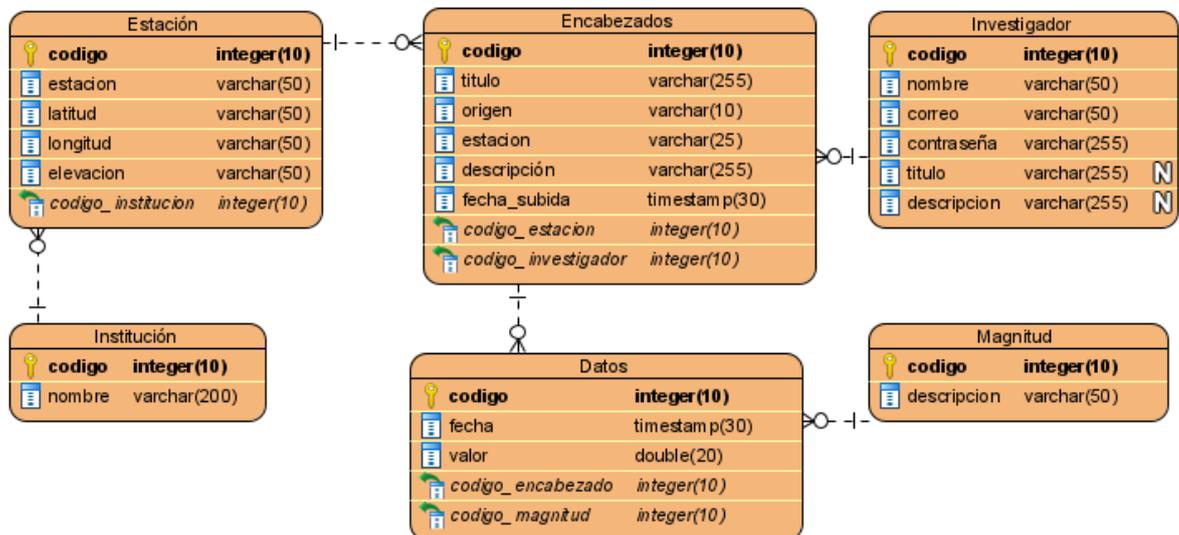
Datos: Tabla que contiene los campos código, fecha y valor. Se relaciona con la tabla encabezados mediante una relación de varios a uno y contiene la clave foránea “código_encabezado”. También se relaciona con la tabla Magnitud mediante una relación de varios a uno y contiene la clave foránea “código_magnitud”.

Estación: Tabla que incluye las columnas código, estación, latitud, longitud y elevación. Se relaciona con la tabla Institución mediante una relación de varios a uno y contiene la clave foránea “código_institución”.

Institución: Tabla que contiene las columnas código y nombre. Se relaciona con la tabla Estación mediante una relación de uno a varios.

Figura 10

Diagrama conceptual de base de datos.



Nota. Diagrama conceptual de base de datos.

Elaborado por: Los autores.

2.8 DIAGRAMA NAVEGACIONAL

El diagrama de la Figura 11 representa la estructura navegacional del repositorio, es decir, las distintas páginas que se pueden visitar por el usuario y los caminos que puede tomar a partir de la página principal. Se muestran cada una de las páginas a las que el usuario puede acceder, algunas de manera directa y otras de forma secuencial tras completar pasos previos.

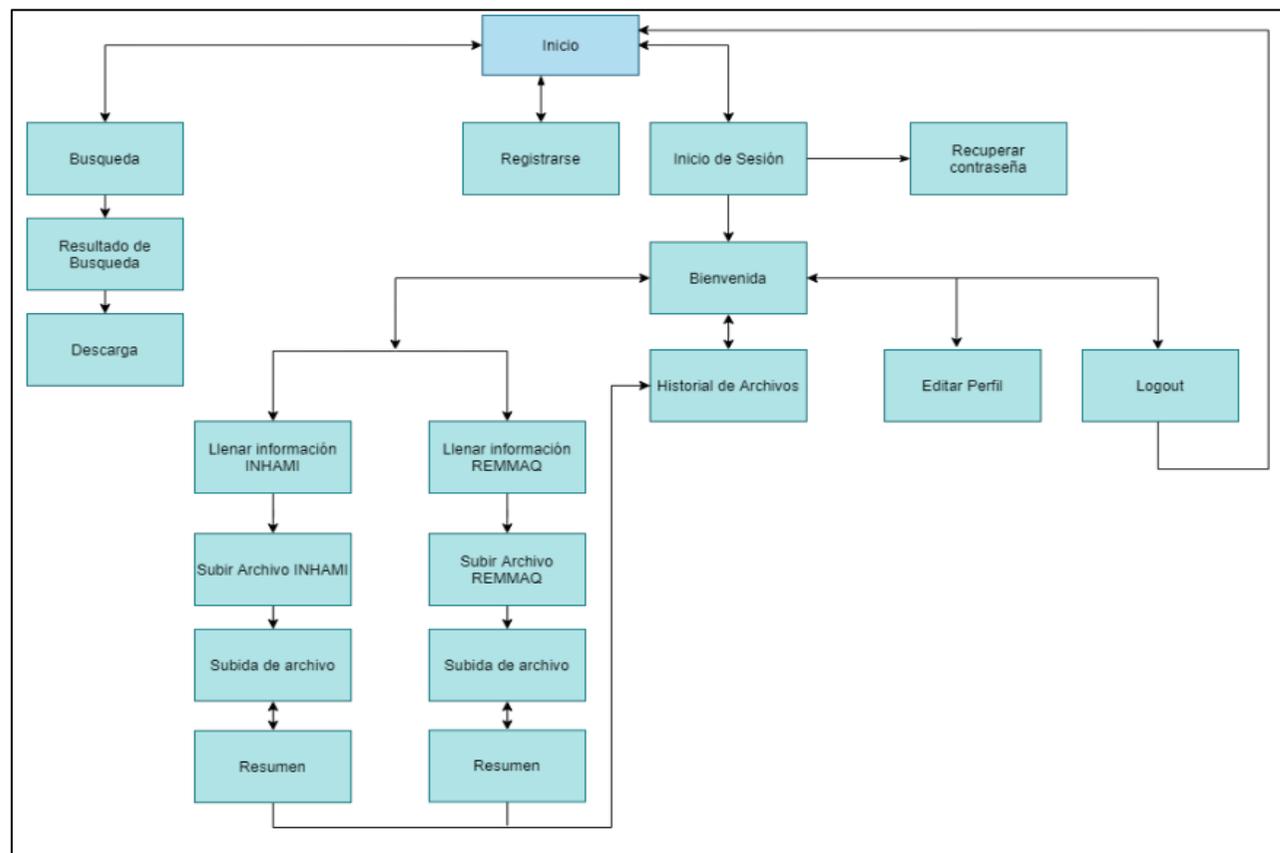
A continuación, se describen las páginas disponibles para la navegación del usuario:

- **Inicio:** página principal donde se muestran los archivos recientes y los respectivos enlaces para inicio de sesión y registro del sistema.
- **Búsqueda:** muestra los resultados de una búsqueda a partir de la palabra clave ingresada en la barra.

- **Registro:** Página donde el usuario ingresa datos personales para obtener una cuenta en el sistema.
- **Editar perfil:** Página en la cual el usuario podrá añadir información personal adicional a la que puso en un inicio en la página de registro.
- **Inicio de sesión:** Página en la cual el usuario ingresa su correo electrónico y contraseña para autenticarse en el sistema.
- **Recuperar contraseña:** Página en donde se ingresa el correo electrónico a donde se envía el enlace de restablecimiento de contraseña.
- **Bienvenida:** Página que se muestra cuando el usuario “Investigador” inicia sesión en el sistema, mostrando las tareas que puede realizar.
- **Formulario de subida de información INAMHI:** Página donde el usuario investigador ingresa los datos descriptivos de la información meteorológica proveniente de las estaciones del INAMHI que cuentan con un formato único.
- **Formulario de subida de información Secretaría del Ambiente:** Página donde el usuario investigador ingresa los datos descriptivos de la información meteorológica proveniente de las estaciones de la Secretaría del Ambiente que cuentan con un formato único.
- **Resumen:** Página que muestra un recuento de la información subida al repositorio.

Figura 11

Diagrama navegacional.



Nota. Estructura navegacional del repositorio.

Elaborado por: Los autores

2.9 DISEÑO DE INTERFACES ABSTRACTAS

2.9.1 Interfaz abstracta invitado

La pantalla que se muestra en la Figura 12 presenta la interfaz de un usuario invitado (usuario sin autenticación en el sistema) donde podrá descargar la información que necesite, se puede observar como Inicio en el diagrama de navegación.

Figura 12

Interfaz abstracta “invitado”.



Nota. Diseño de la Interfaz abstracta: Página principal.

Elaborado por: Los autores.

2.9.2 Interfaz abstracta de opciones para el investigador.

En la interfaz que se presenta en la Figura 13 se muestran las distintas tareas que pueden ser realizadas por el usuario investigador después de iniciar sesión en el repositorio.

Figura 13

Interfaz abstracta opciones para el investigador.



Nota. Diseño de la Interfaz Abstracta: Pantalla de usuario Investigador.

Elaborado por: Los autores.

2.9.3 Interfaz abstracta para la subida de información de la secretaría del ambiente

En la pantalla que se muestra en la Figura 14, el investigador podrá cargar el archivo con la información perteneciente a la Red metropolitana de monitoreo atmosférico de Quito y deberá añadir una breve descripción de la información, se puede ver en el mapa navegacional-subida.

Figura 14

Interfaz abstracta para subida de información REMMAQ.

Repositorio Meteorológico

Subir archivo

Subir

Título: M008 - Nubosidad [mes/año inicio - mes/año fin]

Origen: REMAQ

Magnitud:

Descripción:

Archivo:

Nota. Diseño de la Interfaz Abstracta: Pantalla de subida de datos Secretaría del Ambiente.

Elaborado por: Los autores.

2.9.4 Interfaz abstracta para subida de información del INAMHI

En la interfaz de la Figura 15 se muestran los campos con información descriptiva sobre los datos meteorológicos, previo a la carga de información al repositorio.

Figura 15

Interfaz abstracta para subida de información INAMHI.

Repositorio Meteorológico

Subir archivo

Subir

Título: M008 - Nubosidad [mes/año inicio - mes/año fin]

Origen: INAMHI

Estación:

Descripción:

Archivo:

Nota. Diseño de la Interfaz Abstracta: Pantalla subida de datos INAMHI.

Elaborado por: Los autores.

2.9.5 Interfaz abstracta para la edición de perfil

En la pantalla que se presenta en la Figura 16, se puede añadir información del usuario investigador que se registró, con la presencia de los campos título académico y descripción, campos que pueden ser editados para presentar más información referente al usuario. Se puede ver en el mapa navegacional-Editar.

Figura 16

Interfaz abstracta edición de perfil.



The image shows a user interface for editing a profile. At the top left, there is a logo of a sun and rain clouds next to the text "Repositorio Meteorológico". At the top right, there is a green square icon with a white person silhouette. Below this is a dark green horizontal bar with the word "Editar" in white. The main content area is white and contains four input fields, each with a blue pencil icon to its right: "Nombre", "Correo", "Título Académico:", and "Descripción". The "Descripción" field is a larger rounded rectangle. At the bottom center, there is a green rounded button with the word "Aceptar" in white.

Nota. Diseño de la Interfaz Abstracta: Pantalla edición perfil.

Elaborado por: Los autores.

CAPÍTULO III

CONSTRUCCIÓN

El presente capítulo expone el proceso de desarrollo de la aplicación bajo los parámetros establecidos en las etapas de análisis y diseño, empleando las herramientas y paradigmas pertinentes para generar un producto cuyo objetivo es cumplir con las necesidades planteadas por el usuario. Adicionalmente se presentan los bloques de código principales de cada uno de los módulos.

3.1 ARQUITECTURA DE LA SOLUCIÓN

La arquitectura seleccionada para la construcción de la solución es la planteada por el modelo MVC que se muestra en la Figura 17, la cual se describe de la siguiente forma:

3.1.2 Modelo

Donde se alojan los archivos que contienen los esquemas de los datos que va a manipular el repositorio.

3.1.3 Vista

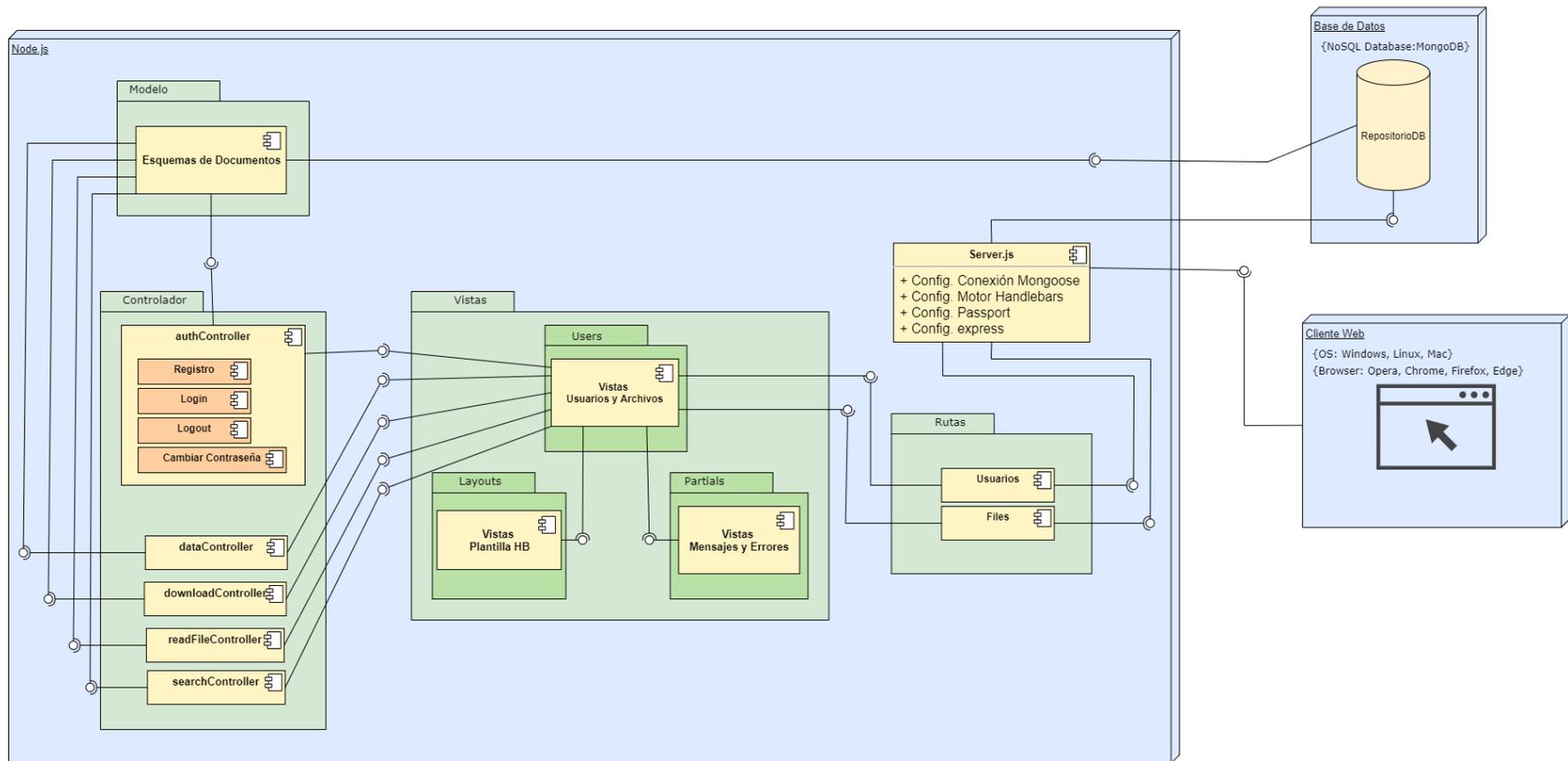
Se optó por el uso del motor de plantillas Handlebars, por consecuencia los archivos que se alojan en la vista tienen la extensión “.hbs” y son los encargados de conectarse con los respectivos controladores que obtienen datos del modelo y los devuelve a la vista para presentarlos al usuario.

3.1.4 Controlador

Son aquellos bloques encargados de enlazar el modelo con la vista y actúan como un intérprete de datos para que estos puedan ser comprendidos por los componentes antes mencionados.

Figura 17

Diagrama de componentes del repositorio.



Nota. Diagrama que describe la arquitectura y componentes del repositorio.

Elaborado por: Los autores.

3.2 PRODUCT BACKLOG

Dentro de la metodología de desarrollo ágil SCRUM, el producto backlog que se presenta en la Figura 18, lista los requerimientos con los que debe contar el sistema. Es la guía definitiva para el desarrollo del proyecto. Para la gestión del proyecto con el mencionado enfoque se utilizó la herramienta Taiga, que proporciona una interfaz robusta e intuitiva que expone el proceso desarrollo de una forma gráfica y atractiva.

Figura 18

Product Backlog.



Nota. Product Backlog con las historias de usuario para el desarrollo del proyecto bajo SCRUM.

Elaborado por: Los autores, a través de Taiga.

3.3 SPRINTS

El desarrollo del proyecto se completó en un total de 9 sprints. Al final de cada sprint se produjo el sprint review donde el tutor del proyecto evaluó el incremento correspondiente y brindaba la retroalimentación necesaria para la corrección o mejora del producto.

El seguimiento de las tareas realizadas en cada sprint se realizó con taiga, que tiene la función de arrastrar y soltar las tareas de acuerdo con el estado en el que se encuentren a medida que avanza el proyecto.

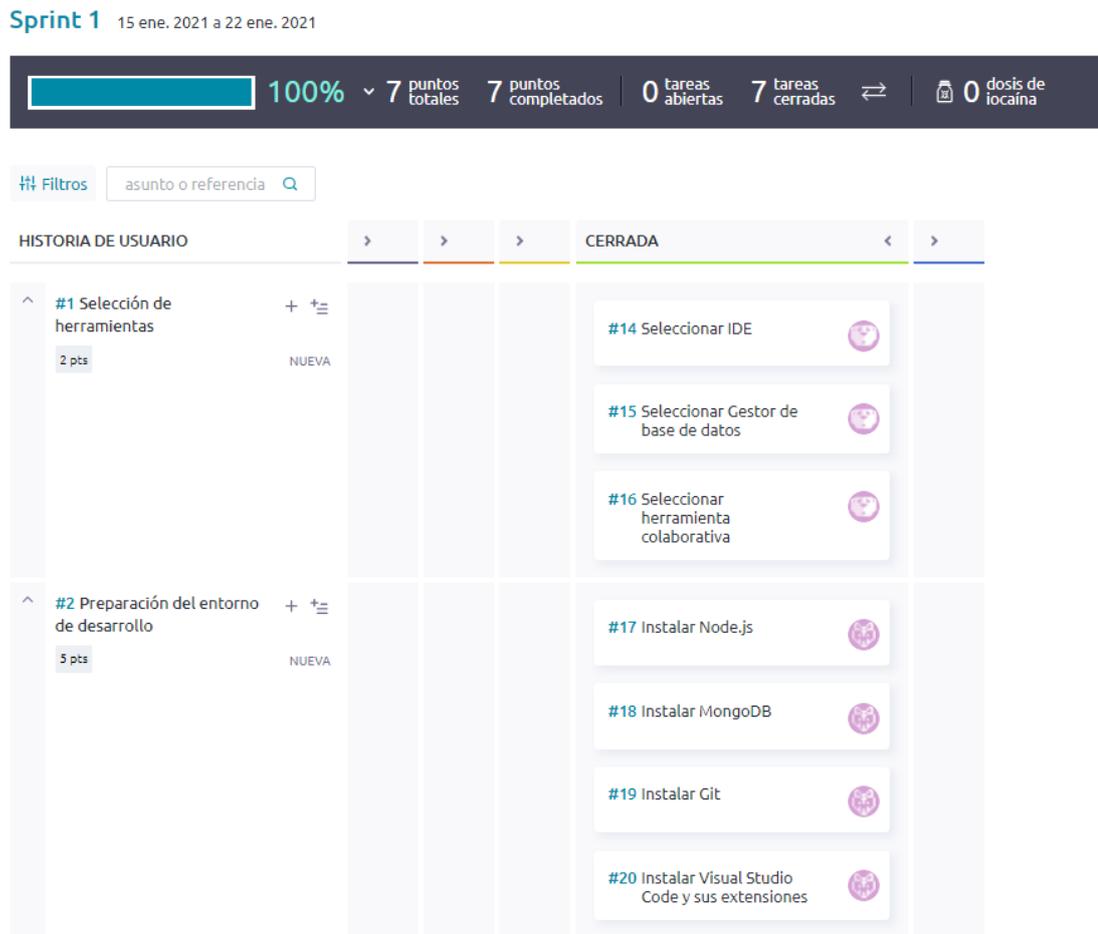
El sistema de puntos de historias que utiliza taiga cumple con la finalidad de medir el nivel de esfuerzo necesario para cumplir con una historia. Cabe mencionar que este sistema es subjetivo dependiendo del equipo de desarrollo. En el caso del presente proyecto se optó por establecer el tope de 20 puntos como el nivel más alto de esfuerzo requerido para cumplir con el requerimiento.

3.3.1 Sprint 1

En el sprint inicial las actividades fueron la selección de herramientas de desarrollo del proyecto y la preparación del entorno de desarrollo como se muestra en el tablero de la Figura 19. Se seleccionó a Visual Studio Code como IDE, MongoDB como gestor de base de datos y Github como repositorio de datos colaborativo. El repositorio donde se aloja el proyecto se encuentra en el siguiente enlace <https://github.com/crizgd221116/Repo-Met>.

Figura 19

Sprint 1.



Nota. Tablero del Sprint 1.

Elaborado por: Los autores, a través de Taiga.

3.3.1.1 Preparación del entorno de desarrollo. Un aspecto esencial previo al inicio del desarrollo del aplicativo es la preparación del entorno, realizando la instalación respectiva de las herramientas establecidas en la etapa de planificación y configurándose de manera que la construcción se realice de forma efectiva. En el presente capítulo se describe cómo fue adecuado y configurado el entorno de Node.js junto el gestor de bases de datos MongoDB.

3.3.1.2 Preparación del entorno Node.js. Existen múltiples formas de adquirir los paquetes y librerías necesarias del entorno de Node.js, en caso del presente proyecto se optó por el instalador pre-compilado para Windows en su versión 14.16. Una vez instalado se gestiona la configuración a través de la consola del editor de código Visual Studio Code mediante comandos que facilitan la creación correcta de la estructura de un proyecto. Se emplea npm, el administrador de paquetes de Node.js en la versión 6.14.10, que facilita la adquisición de librerías adicionales a través de líneas de código relativamente sencillas, encargándose a la vez de las dependencias que existen entre cada una de las librerías mencionadas. Dentro de la estructura de Node.js se encuentra el archivo server.js donde se realiza definición y configuración de las distintas dependencias que van a ser utilizadas para la ejecución del código.

3.3.1.3 Preparación de Handlebars. Para la utilización del motor de plantillas Handlebars en su versión 5.2.0 es necesario configurarlo en el archivo server.js como se muestra en la Figura 20, creando una función donde se especifican la extensión de archivo que interpreta el motor, la plantilla predeterminada para todas las pantallas, los directorios donde se alojan los archivos con extensión “.hbs” y los helpers los cuales son estructuras de código escritas en JavaScript que pueden ser llamadas de una forma más simple mediante la sintaxis de Handlebars.

Figura 20

Configuración Handlebars.

```
app.engine('.hbs', exphbs({
  defaultLayout: 'main',
  layoutsDir: path.join(app.get('views'), 'layouts'),
  partialsDir: path.join(app.get('views'), 'partials')
  extname: '.hbs',
  helpers: {
    eq: function(v1, v2) {
      return v1 === v2;
    },
  },
});
```

Nota. Bloque de configuración del motor de plantillas Handlebars.

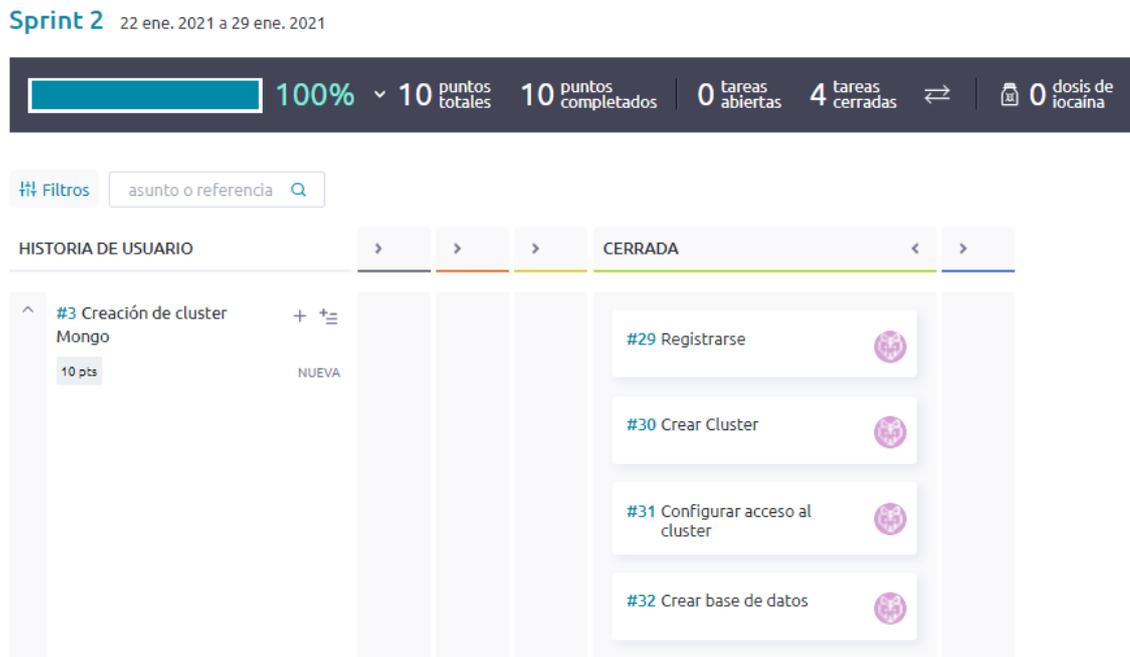
Elaborado por: Los autores, disponible en <https://github.com/crizgd221116/Repo-Met/blob/main/src/server.js>.

3.3.2 Sprint 2

El segundo sprint se enfoca en la creación y configuración del clúster en MongoDB como se presenta en el tablero de la Figura 21. Se empleó un sprint para este apartado con la finalidad de investigar y entender de manera correcta el funcionamiento de las bases de datos NoSQL. Cabe mencionar que el proceso de creación se realizó a través de MongoDB Atlas y una vez concluido se utilizó la aplicación de escritorio para Windows conocida como MongoDB Compass.

Figura 21

Sprint 2.



Nota. Tablero del Sprint 2.

Elaborado por: Los autores, a través de Taiga.

3.3.2.1 Preparación del gestor de base de datos MongoDB. La preparación del gestor de base de datos MongoDB se realizó a través de la interfaz de administración en línea conocida como MongoDB Atlas, donde después de un proceso de registro para la obtención de una cuenta se presenta un asistente que facilita la creación de un clúster. El asistente permite seleccionar aspectos referentes al proveedor de servicios en la nube, el tipo de clúster según en la capacidad de cómputo requerida, la ubicación geográfica del clúster, todos estos aspectos englobados en distintos planes que pueden ser gratuitos o requieren de una inversión para su uso. Para el proyecto se seleccionó la opción M0 que cuenta con memoria RAM compartida, 512 MB de almacenamiento y procesamiento de CPU compartido, siendo la única opción disponible de forma gratuita. Una vez creado se procede a configurar las credenciales de acceso al mismo, mediante la creación del usuario y el establecimiento de una contraseña, estableciendo también los privilegios con los que va a contar, siendo la opción “leer y escribir en cualquier base de datos” la seleccionada por el equipo de desarrollo. Atlas facilita la opción de conectarse al clúster desde cualquier plataforma mediante la cadena de conexión única, por lo que se utilizó el driver diseñado para proyectos en Node.js.

3.3.2.2 Conexión con MongoDB. Para realizar la conexión con la base de datos desde el aplicativo, se realiza la configuración en el archivo server.js como se muestra en la Figura 22, donde se coloca la cadena única de conexión proporcionada por Atlas y se emplea la librería mongoose, diseñada especialmente para conectar proyectos desarrollados en Node.js con una base de datos MongoDB.

Figura 22

Conexión con MongoDB.

```
//----- Conexión Mongo -----//
mongoose.connect('mongodb://localhost:27020/?authSource=admin&ssl=false')
  { useNewUrlParser: true, useUnifiedTopology: true, useFindAndModify: false }
  .then(db => console.log('Base de datos conectada'))
  .catch(err => console.error(err));
```

Nota. Bloque de configuración de mongoose para conectar base de datos MongoDB.

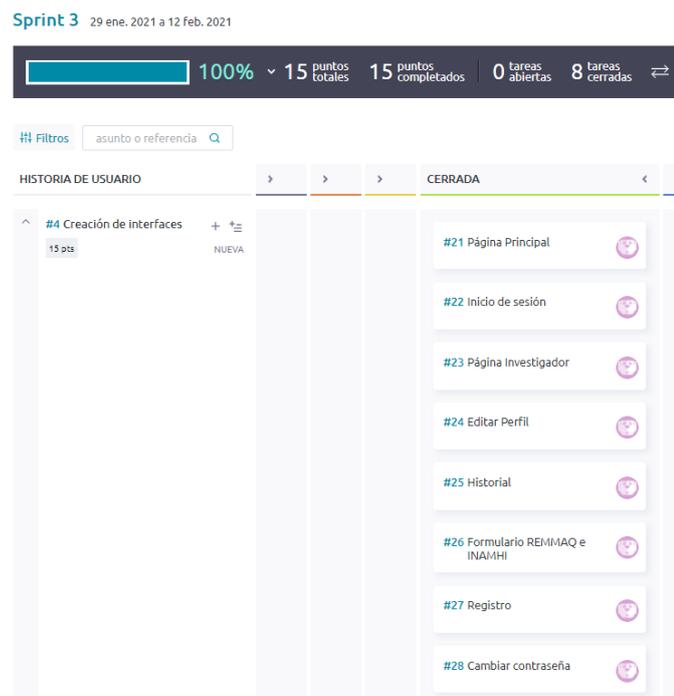
Elaborado por Los autores, disponible en <https://github.com/crizgd221116/Repo-Met/blob/main/src/server.js>.

3.3.3 Sprint 3

El tercer sprint se enfocó en la codificación de las interfaces abstractas presentadas en la etapa de diseño como se muestra en el tablero de la Figura 23.

Figura 23

Sprint 3.

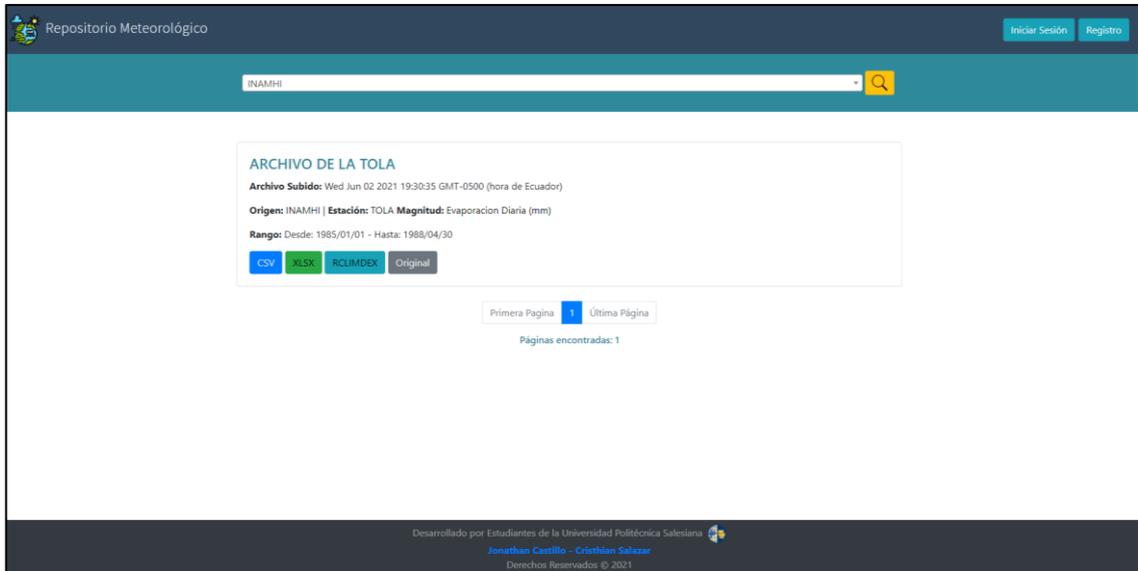


Nota. Tablero del Sprint 3.

Elaborado por: Los autores a través de Taiga.

Figura 24

Pantalla principal.



Nota. Interfaz principal con el diseño y esquema de colores final del repositorio.

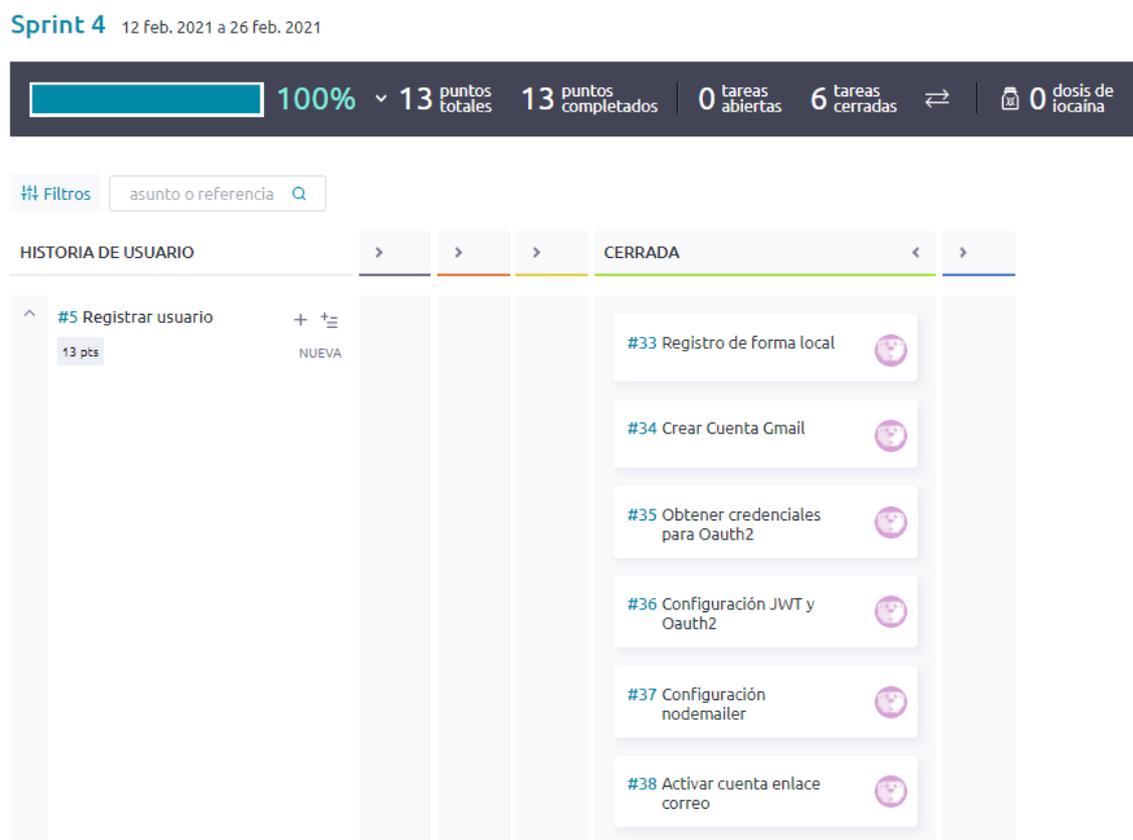
Elaborado por: Los autores.

3.3.4 Sprint 4

En este sprint se realizó la codificación del módulo de registro del sistema con la verificación y activación de la cuenta a través de un enlace enviado por correo electrónico como se muestra en el tablero de la Figura 25.

Figura 25

Sprint 4.



Nota. Tablero del Sprint 4.

Elaborado por: Los autores, a través de Taiga.

3.3.4.1 Autenticación. La autenticación de usuarios dentro del aplicativo tiene como objetivo administrar el acceso al sistema, por consecuencia el acceso a la información que se genera y almacena en el mismo, a través la gestión de credenciales utilizadas para garantizar la seguridad y la correcta manipulación de los archivos que son cargados en el sistema. De acuerdo con lo establecido en las etapas previas a la construcción, el usuario que va a cumplir el rol de investigador es el único que debe registrarse en el portal para poder realizar sus respectivas tareas.

3.3.4.2 Registro. El módulo de registro es donde el usuario ingresa su información personal para la creación de una cuenta nueva. Para la construcción de los distintos formularios del proyecto se utilizaron los componentes de Bootstrap en su versión 4.1.1. El formulario de registro cuenta con los siguientes campos: Nombres, correo, contraseña y confirmar contraseña. Se crea una función en el controlador de autenticación `authController.js` como se presenta en la Figura 26, en donde se utilizan los campos antes mencionados.

Figura 26

Manejo de registro.

```
//----- Manejo del registro -----//
exports.registerHandle = (req, res) => {
  const { name, email, password, password2 } = req.body;
  let errors = [];
  console.log(req.body)
```

Nota. Bloque de código del manejo de registro donde se especifican las variables utilizadas.

Elaborado por: Los autores, disponible en <https://github.com/crizgd221116/Repo-Met/blob/main/src/controllers/authController.js>.

Se realiza la comparación de los campos contraseña y confirmar contraseña como se presenta en la Figura 27 con la finalidad de que no existan inconsistencias. Adicionalmente se realiza la validación de la longitud de la contraseña que debe ser mayor a 8 caracteres y se verifica la existencia de un usuario ya existente con el correo ingresado.

Figura 27

Validación de contraseña.

```
//----- Validación de contraseña -----//
if (password !== password2) {
  console.log("las contraseñas no coinciden")
  req.flash("error_msg", "las contraseñas no coinciden");
  res.redirect('/users/register');
  //----- Número de caracteres de la contraseña-----//
}
if (password.length < 8) {
  console.log("Las contraseñas deben tener por lo menos 8 caracteres")
  req.flash("error_msg", "Las contraseñas deben tener por lo menos 8 caracteres");
  res.redirect('/users/register');
} else {
  //----- Validación usuario existente -----//
  User.findOne({ email: email }).then(user => {
    if (user) {
      console.log("Correo ya Existe")
      req.flash("error_msg", "Correo ya Existe");
      res.redirect('/users/register');
    }
  });
}
```

Nota. Bloques de código de la validación de contraseña y correo para el registro de usuario.

Elaborada por: Los autores, disponible en <https://github.com/crizgd221116/Repo-Met/blob/main/src/controllers/authController.js> .

3.3.4.3 Creación del cliente OAuth2. Para la utilización del protocolo OAuth2 es necesario registrar la aplicación en Google Cloud Platform, así se obtienen dos parámetros importantes para el proceso que son Client Id y Client secret. Una vez obtenidos los parámetros antes mencionados es posible dirigirse a OAuth2 playground, donde se selecciona la API de Gmail para generar nuevos tokens que son Refresh Token y Access token, este último se actualiza automáticamente a través de la función `getAccessToken()` como se presenta en la Figura 28.

Figura 28

Configuración de OAuth2.

```
const oauth2Client = new OAuth2(  
  // Client ID  
  // Client secret  
  // Redirect URI  
);  
console.log(1)  
  
oauth2Client.setCredentials({  
  refresh_token:   
});  
console.log(2)  
  
const accessToken = oauth2Client.getAccessToken()
```

Nota. Configuración de parámetros para la utilización de OAuth2.

Elaborado por: Los autores, disponible en <https://github.com/crizgd221116/Repo-Met/blob/main/src/controllers/authController.js> .

3.3.4.4 Generar token con JWT. En este apartado se genera el token que va a ser enviado a través de correo electrónico al usuario para la activación de su cuenta y cuenta con un tiempo de expiración de 5 minutos como se presenta en la Figura 29. Una vez que el tiempo se termine el token pierde validez y el usuario debe volver a realizar el proceso para generar nuevamente el token.

Figura 29

Generar token JWT.

```
oauth2Client.setCredentials({
  refresh_token: '...'
});

console.log(2)

const accessToken = oauth2Client.getAccessToken()

const token = jwt.sign({ name, email, password }, JWT_KEY, { expiresIn: '5m' });
const CLIENT_URL = 'http://' + req.headers.host;
const output = ''
```

Nota. Generación del token JWT con sus respectivos parámetros y tiempo de expiración.

Elaborado por: Los autores, disponible en <https://github.com/crizgd221116/Repo-Met/blob/main/src/controllers/authController.js>

3.3.4.5 Implementación de Nodemailer. Se crea un objeto transporter especificando el método de autenticación que es OAuth2 e incluyendo las credenciales obtenidas en la plataforma Cloud de Google como se muestra en la Figura 30. A continuación, se configuran los aspectos propios del correo electrónico como la dirección de origen, el destinatario, el asunto y el parámetro “html” donde se incluirá el contenido del mensaje como se presenta en la Figura 31.

Nota. Cuerpo de correo electrónico renderizado en la bandeja de entrada del usuario para la activación de la cuenta.

Elaborado por: Los autores.

3.3.4.6 Activación de cuenta. Se realizan las respectivas validaciones para confirmar si el enlace enviado por correo tiene vigencia o es erróneo, de igual manera se verifica si el correo ingresado ya existe en el sistema y finalmente se realiza el registro y activación de la cuenta de usuario como se muestra en la Figura 32.

Figura 32

Validación de activación de cuenta.

Algorithm : Activación de cuenta

```
Result: Cuenta activada correctamente
initialization;
instructions;
if token then
  if token caducado then
    Imprimir ("Enlace Caducado");
    Redirigir(página: "Registro");
  else
    (nombre, email, contraseña) ← Decodifica(token)
    usuario ← BuscarBDD(usuario)
    if email then
      Imprimir ("Usuario ya está registrado, iniciar sesión");
      Redirigir(página: "Login");
    else
      Registrar(usuario);
      Imprimir("Cuenta activada inicie sesión");
    end
  end
end
else
  Imprimir ("Error registro");
end
```

Nota. Manejo de la activación de cuenta de usuario en el repositorio.

Elaborado por: Los autores, disponible en <https://github.com/crizgd221116/Repo->

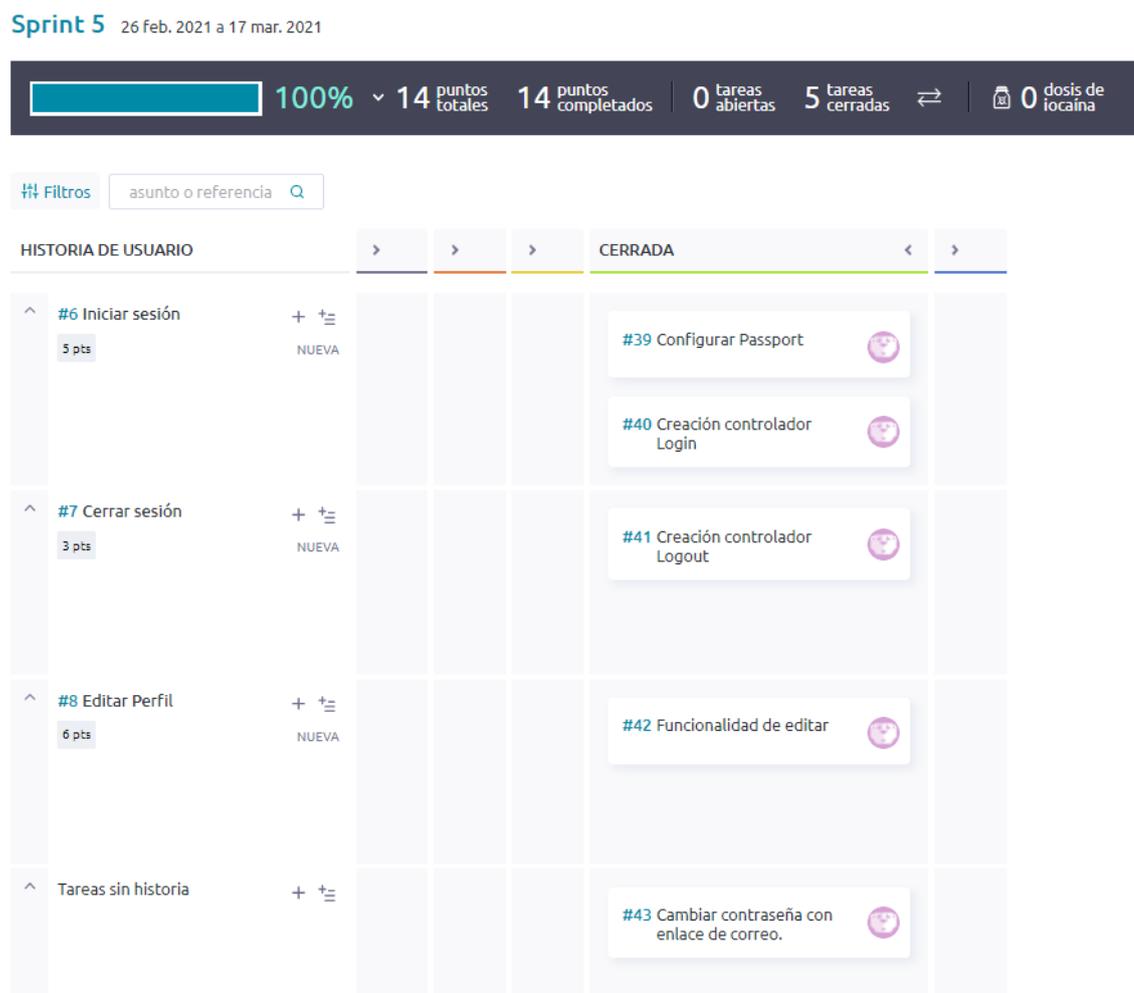
[Met/blob/main/src/controllers/authController.js](https://github.com/crizgd221116/Repo-Met/blob/main/src/controllers/authController.js)

3.3.5 Sprint 5

El quinto sprint tuvo como enfoque las codificaciones de login, logout, la edición de perfil del usuario investigador y el cambio de contraseña utilizando la funcionalidad de Nodemailer de la misma forma que en el registro como se muestra en el tablero de la Figura 33.

Figura 33

Sprint 5.



Nota. Tablero del Sprint 5.

Elaborado por: Los autores, a través de Taiga.

3.3.5.1 Inicio de Sesión. La funcionalidad del inicio de sesión se construye en el archivo Passport.js donde se define una función que determina si el usuario se encuentra registrado en el sistema en base al campo email. Posteriormente se verifica si la contraseña ingresada coincide con la establecida por el usuario en el proceso de registro como se muestra en la Figura 34.

Figura 34

Definición de Passport para autenticación.

```
//----- Modelo Usuario -----//
const User = require('../models/User');

module.exports = function(passport) {
  passport.use(
    new LocalStrategy({ usernameField: 'email' }, (email, password, done) => {
      //----- Usuario existe -----//
      User.findOne({
        email: email
      }).then(user => {
        if (!user) {
          return done(null, false, { message: 'Este correo electrónico no está registrado' });
        }

        //----- Contraseñas coinciden -----//
        bcrypt.compare(password, user.password, (err, isMatch) => {
          if (err) throw err;
          if (isMatch) {
            return done(null, user);
          } else {
            return done(null, false, { message: 'Contraseña Incorrecta, intente nuevamente' });
          }
        });
      });
    })
  );
};
```

Nota. Función de autenticación utilizando Passport.js.

Elaborado por: Los autores, disponible en

<https://github.com/crizgd221116/Repo-Met/blob/main/src/controllers/authController.js>.

En el controlador authController.js se llama a la función de Passport authenticate y se establecen las rutas en caso de que la autenticación sea exitosa o fallida como se presenta en la Figura 35.

Figura 35

Manejo de Login.

```
//----- Manejo login -----//
exports.loginHandle = (req, res, next) => {
  passport.authenticate('local', {
    successRedirect: '/users/invest',
    failureRedirect: '/users/login',
    failureFlash: true
  })(req, res, next);
}
```

Nota. Función para el manejo de inicio de sesión con sus salidas tanto exitosa como fallida.

Elaborado por: Los autores, disponible en <https://github.com/crizgd221116/Repo-Met/blob/main/src/controllers/authController.js> .

3.3.5.2 Cierre de sesión. En el archivo authController.js se realiza una función en donde se establece el cierre de sesión, se muestra una notificación de cierre de sesión y se redirecciona a la página principal del repositorio como se presenta en la Figura 36.

Figura 36

Función de cierre de sesión.

```
//----- Manejo cerrar sesión -----//
exports.logoutHandle = (req, res) => {
  req.logout();
  req.flash('success_msg', 'se ha cerrado sesión');
  res.redirect('/index/1');
}
```

Nota. Función de cierre de sesión en el controlador de autenticación.

Elaborado por: Los autores, disponible en <https://github.com/crizgd221116/Repo-Met/blob/main/src/controllers/authController.js> .

3.3.5.3 Cambio de contraseña. La recuperación o restablecimiento de contraseña se realiza en función del campo email, una vez ingresado se valida la información como se presenta en la Figura 37 y posteriormente se genera un token único de identificación del usuario con un tiempo de vigencia de 5 minutos. Al igual que en el proceso de activación de cuenta, el enlace de cambio de contraseña es enviado a través de correo electrónico como se presenta en la Figura 38.

Figura 37

Validación campo correo.

```
Algorithm : Validación de email
```

```
Result: Email Validado  
initialization;  
instructions;  
if email then  
| email ← BuscarBDD(email)  
else  
| Imprimir ("Ingrese un email");  
| Redirigir(página: "recuperar contraseña");  
end
```

Nota. Función de validación del campo correo para cambio de contraseña.

Elaborado por: Los autores, disponible en <https://github.com/crizgd221116/Repo-Met/blob/main/src/controllers/authController.js>

Figura 38

Función para cambiar contraseña.

Algorithm : Función para cambiar de contraseña

```
Result: Cambio de contraseña exitoso
instructions;
if token then
  | if token caducado then
  | | Imprimir ("Enlace Caducado");
  | | Redirigir(página: "recuperar contraseña");
  | else
  | | idusuario ← Decodifica(token)
  | | usuario ← BuscarBDD(usuario)
  | end
else
  | Imprimir ("Error Reset");
end
```

Nota. Función que realiza el proceso de cambio de contraseña verificando la validez del token generado.

Elaborado por: Los autores, disponible en

<https://github.com/crizgd221116/Repo-Met/blob/main/src/controllers/authController.js>

3.3.5.4 Edición de perfil. En la edición de perfil el usuario tiene la posibilidad de añadir información adicional a la proporcionada al momento de registrarse. Estos campos adicionales son: género, título académico, ocupación y bio que es una descripción personal del usuario como se presenta en la Figura 39. Se emplea la función `isAuthenticated` que solamente permite el acceso a un usuario registrado y con sesión iniciada en el repositorio.

Figura 39

Ruta y función para edición de perfil.

```
router.get("/users/editinfo/:id", isAuthenticated, async(req, res) => {
  const userAuth = await User.findById(req.params.id);
  res.render("users/editinfo.hbs", { userAuth });
});

router.put("/users/editinfo/:id", async(req, res) => {
  const {
    /*genero,titulo,ocupacion,description*/
    name,
    email,
    genero,
    titulo,
    ocupacion,
    description,
  } = req.body;
  console.log(req.body);
  await User.findOneAndUpdate(
    { _id: req.params.id }, { $set: req.body }, { new: true }
  );
  const userAuth = await User.findById(req.params.id);
  res.render("users/editinfo.hbs", { userAuth });
});
```

Nota. Función para la edición de datos en el perfil investigador.

Elaborado por: Los autores, disponible en

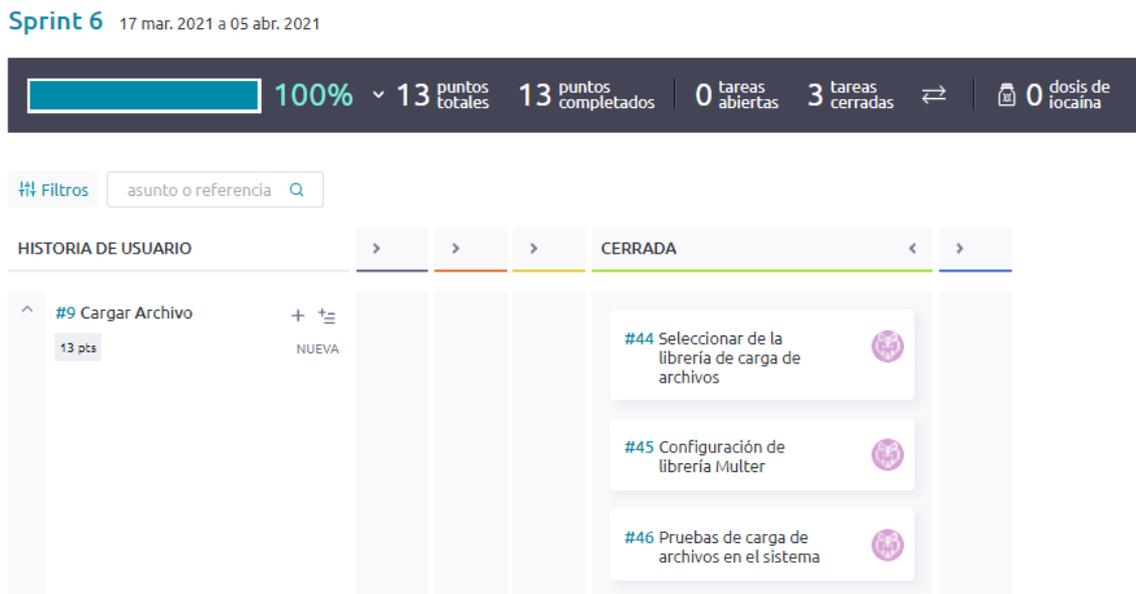
<https://github.com/crizgd221116/Repo-Met/blob/main/src/routes/usuarios.js>.

3.3.6 Sprint 6

En este sprint se trabajó en la carga de los archivos al sistema para posteriormente realizar la respectiva lectura como se muestra en el tablero de la Figura 40.

Figura 40

Sprint 6.



Nota. Tablero del Sprint 6.

Elaborado por: Los autores, a través de Taiga.

3.3.6.1 Carga de archivos. Para realizar la carga de un archivo se implementó la librería Multer que actúa como un middleware. Su configuración se realiza en el archivo `server.js` como se muestra en Figura 41, donde se establecen parámetros como la ruta de almacenamiento, el nombre del archivo y su respectiva extensión. El nombre del archivo tendrá un identificador único dentro del sistema.

Figura 41

Librería Multer.

```
//subir archivos
const multer = require('multer');

const storage = multer.diskStorage({
  destination: path.join(__dirname, 'public/uploads'),
  filename: (req, file, cb) => {
    cb(null, uuid() + path.extname(file.originalname));
  }
});
```

Nota. Parámetros de la librería Multer para la carga de archivos.

Elaborado por: Los autores, disponible en

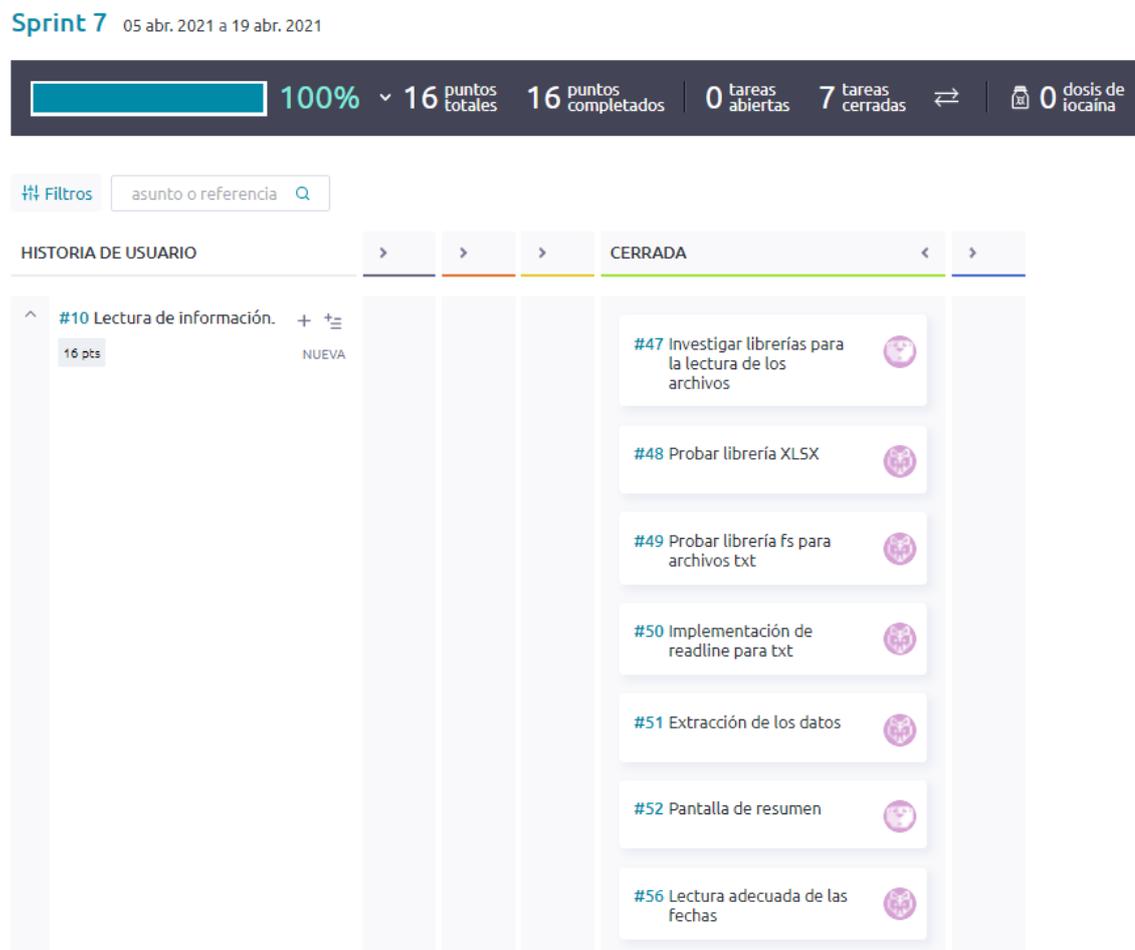
<https://github.com/crizgd221116/Repo-Met/blob/main/src/server.js>.

3.3.7 Sprint 7

El séptimo sprint se focalizó en la codificación de la lectura y extracción de datos de los archivos en sus formatos originales como se presenta en el tablero de la Figura 42. De esta forma se obtiene información para presentarla en la pantalla de resumen.

Figura 42

Sprint 7.



Nota. Tablero del Sprint 7.

Elaborado por: Los autores, a través de Taiga.

3.3.7.1 Lectura archivo INAMHI. La lectura se construyó en base al archivo original con el formato utilizado por INAMHI, que cuenta con una extensión “.txt” y un encabezado como lo muestra la Figura 43. En el caso de que el archivo no cuente con un encabezado la aplicación será capaz de leer los valores diarios y darle la posibilidad al usuario de ingresar el nombre de la estación y la magnitud a la que pertenecen los datos.

Figura 43

Formato archivo INAMHI.

INSTITUTO NACIONAL DE METEOROLOGIA E HIDROLOGIA

Evaporación Diaria (mm) 02/07/2019

SERIES DE DATOS METEOROLOGICOS

NOMBRE: TOLA CODIGO: M0009
 PERIODO: 1985 - 2019 LATITUD: 0G 21' 57.33" S LONGITUD: 78G 33' 18.46"W ELEVACION:

VALORES DIARIOS

ANIO DIA	1	2	3	4	5	6	7	8	9	10	11
1985 1	3.3	5.1	3.3	3.7	1.6	2.3	2.9	3.2	2.3	3.8	3.8
1985 2	2.4	6.1	3.8	4.9	4.3	4.3	6.4	4.1	1.1	4.1	4.1
1985 3	4.4	3.6	2.7	3.7	3.2	1.6	3.8	3.3	3.5	4.1	4.1
1985 4	0.5	3.0	3.6	4.2	4.3	1.2	2.7	1.7	2.8	3.7	3.7
1985 5	4.2	4.2	5.1	4.5	3.3	4.7	5.6	5.6	6.0	0.8	0.8
1985 6	2.1	2.8	4.3	3.0	3.4	3.4	2.9	3.8	4.8	6.1	6.1
1985 7	3.9	4.2	2.6	4.5	3.8	4.4	4.6	3.7	6.4	5.1	5.1
1985 8	3.5	5.2	4.7	6.2	4.1	3.8	0.9	2.9	2.7	4.7	4.7
1985 9	6.0	4.9	3.6	5.7	5.7	4.6	5.6	3.0	2.4	3.1	3.1
1985 10	2.1	4.3	3.3	1.7	2.6	4.3	4.4	4.3	2.8	3.8	3.8
1985 11	5.9	6.1	5.9	4.8	5.5	3.2	2.2	3.7	4.1	4.8	4.8

Nota. Estructura del archivo original de INAMHI.

Fuente: INAMHI.

En el controlador readFileController se creó una función para la lectura de los archivos INAMHI que tendrán extensión .txt y codificación UTF-8. La librería implementada para leer el contenido del archivo de forma global es fs, por su parte readline cumple con el mismo objetivo, pero leyéndolo línea por línea con la finalidad de extraer información como: Magnitud, nombre de la estación y los registros. Todos los datos obtenidos del archivo pasan al modelo para posteriormente ser almacenados en la base de datos.

Figura 44

Librería fs.

```
ReadContentTxtFile(filePath, process) {
  let file = new FileModel();
  file.origen = ORIGEN_INAMHI;
  fs.readFile(`${filePath}`, 'utf8', function (err, data) {
    if (err) {
      console.log(err);
    } else {
      let lector = readline.createInterface({
        input: fs.createReadStream(`${filePath}`)
      });

      let i = 0;
      const registros = [];
      lector.on("line", linea => {

        //magnitud
        if (i == 2) {
          const index = linea.indexOf(" ");
          file.magnitud = linea.substring(0, index + 1);
        }
      });
    }
  });
}
```

Nota. Parámetros de la librería fs para la lectura de archivos txt.

Elaborado por: Los autores, disponible en

<https://github.com/crizgd221116/Repo->

[Met/blob/main/src/controllers/readFileController.js](https://github.com/crizgd221116/Repo-Met/blob/main/src/controllers/readFileController.js)

3.3.7.2 Lectura archivo REMMAQ. La lectura se construyó en base al archivo original con el formato utilizado por REMMAQ que se presenta en la Figura 45, que cuenta con una extensión “.xlsx”. Cabe mencionar que este tipo de archivos cuenta con gran cantidad de datos ya que tiene información desde los 80 hasta la actualidad y de 6 o más estaciones y existe un valor por hora.

Figura 45

Formato de archivo REMMAQ.

	A	B	C	D	E	F	G	H	I	J
1		Belisario	Carapungo	Centro	Cotocollao	EI Camal	Guamaní	Los Chillos	San Antonio	Tumbaco
2	FECHA \ UNIDAD	%	%	%	%	%	%	%	%	%
3	1/1/2004 00:00	98.06	74.78		98.49	76.89		93.17		98.56
4	1/1/2004 01:00	98.47	72.42		100	75.94		94.62		99.61
5	1/1/2004 02:00	98.65	74.43		100	76.08		97.13		99.14
6	1/1/2004 03:00	99.03	76.16		100	78.94		100		98.45
7	1/1/2004 04:00	86.85	76.63		100	79.83		99.92		98.73
8	1/1/2004 05:00	83.43	77.66		100	80.36		100		99.83
9	1/1/2004 06:00	79.54	73.67		99.61	75.4		100		99.92
10	1/1/2004 07:00	74.52	66.93		78.44	70.73		94.12		93.55
11	1/1/2004 08:00	68.83	60.07		55.79	64.73		69.35		74.37
12	1/1/2004 09:00	61.58	53.37		50.47	54.78		48.88		60.77
13	1/1/2004 10:00	57.15	45.73		47.6	48.45		41.13		45.8
14	1/1/2004 11:00	53.5	39.35		44.39	44.49		37.09		43.5
15	1/1/2004 12:00	50.31	36.78		41.31	41.76		34.97		42.62
16	1/1/2004 13:00	50.36	37.77		39.72	40.62		34.88		43.28
17	1/1/2004 14:00	49.12	35.4		42.49	40.37		33.59		42.4
18	1/1/2004 15:00	48.62	34.24		52.17	41.35		34.5		42.39

Nota. Formato de archivo de REMMAQ.

Fuente: REMMAQ.

En el controlador readFileController se creó una función para la lectura de los archivos REMMAQ que tendrán extensión .xlsx. La librería implementada para leer el contenido del archivo es XLSX, instalada a través npm, el gestor de paquetes de Node.js. La librería cuenta con la propiedad llamada sheet_to_json que utiliza como parámetro la hoja de cálculo donde se localizan los datos. Esto permitirá transformar el contenido de la hoja de cálculo al formato JSON para la manipulación y extracción de la información deseada como se muestra en la Figura 47. Todos los datos obtenidos del archivo pasan al modelo para posteriormente ser almacenados en la base de datos.

Figura 46

Lectura XLS.

```
ReadContentXlsFile(filePath, process) {  
  
  let file = new FileModel();  
  file.origen = ORIGEN_REMMAQ;  
  var workbook = XLSX.readFile(`${filePath}`, {  
    type: "binary",  
    cellText: false,  
    cellDates: true,  
  });  
  
  var sheet_name_list = workbook.SheetNames;  
  var xlData = XLSX.utils.sheet_to_json(workbook.Sheets[sheet_name_list[0]], {  
    header: 1,  
    raw: false,  
    dateNF: "yyyy-mm-dd HH:mm:ss",  
  });  
});
```

Nota. Parámetros de la librería XLSX para la lectura de archivos REMMAQ.

Elaborado por: Los autores, disponible en <https://github.com/crizgd221116/Repo-Met/blob/main/src/controllers/readFileController.js>.

Figura 47

Extracción Información del archivo xlsx.

```
//Fecha de inicio del archivo  
var fechainicio = xlData[3] + "";  
file.fechaInicio = fechainicio.split(",", 1).toString();  
  
//Fecha de fin del archivo  
var fechafin = xlData[xlData.length - 1] + "";  
file.fechafin = fechafin.split(",", 1).toString();  
  
// Nombre de estaciones  
var estaciones = xlData[0].filter((estacion) => estacion != null) + "";  
file.nombreEstaciones = estaciones.split(",").toString();  
  
//Num registros  
file.numeroRegistros = xlData.length;  
  
file.path = filePath;
```

Nota. Extracción de datos del archivo REMMAQ.

Elaborado por: Los autores, disponible en

<https://github.com/crizgd221116/Repo-Met/blob/main/src/controllers/readFileController.js>

3.3.8 Sprint 8

En el octavo sprint se trabajó en el almacenamiento de los datos leídos de los archivos como se presenta en el tablero de la Figura 48.

Figura 48

Sprint 8.



Nota. Tablero del Sprint 8.

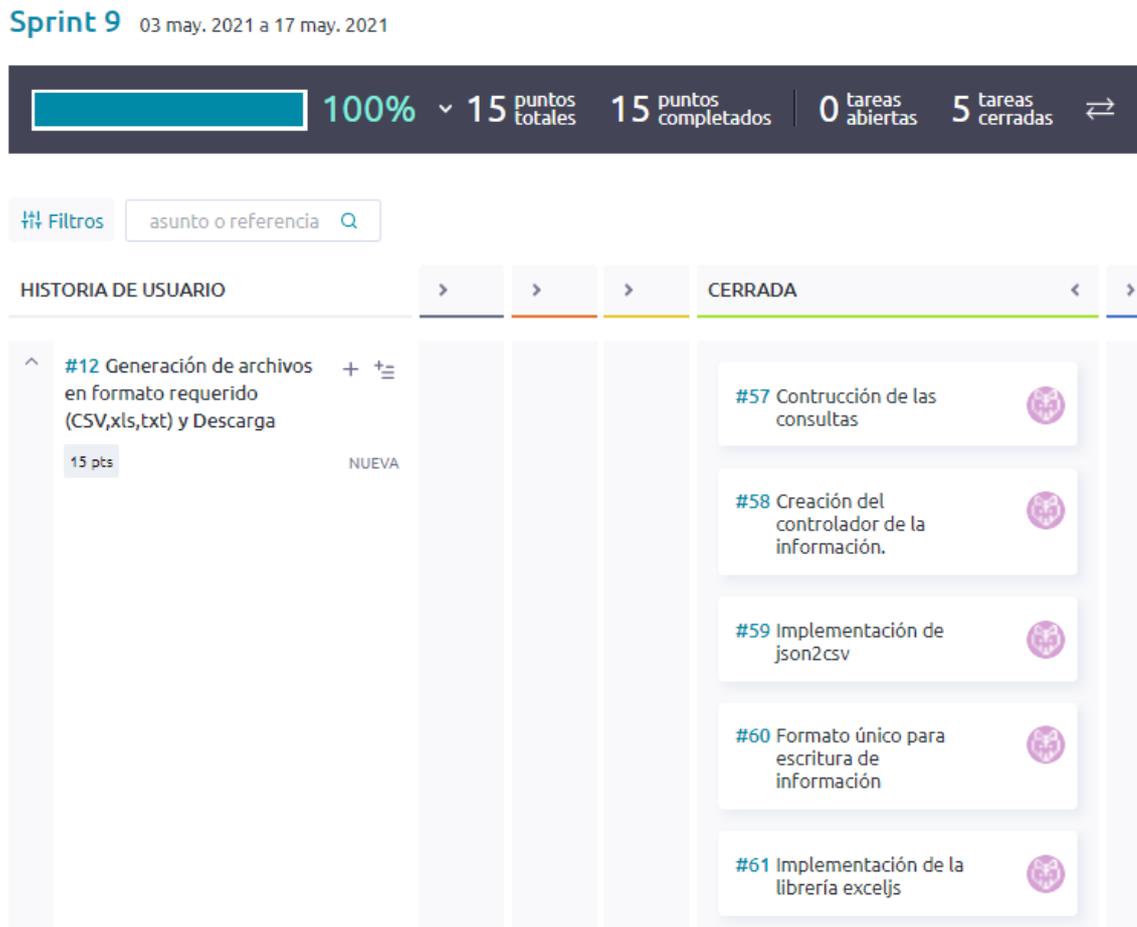
Elaborado por: Los autores, a través de Taiga.

3.3.9 Sprint 9

En el último sprint se trabajó la generación de archivos con los formatos de descarga establecidos como se muestra en el tablero de la Figura 49.

Figura 49

Sprint 9.



Nota. Tablero del Sprint 9.

Elaborado por: Los autores, a través de Taiga.

3.3.9.1 Generación de archivos y descarga. En el controlador `dataController` se manipula la información almacenada en la base para poder extraerla, lo que se realiza es una consulta a la colección de datos por el id, mediante la propiedad "populate" extraemos cada valor con el respectivo encabezado.

Se realiza un bucle `for` para iterar la información de la colección datos donde se encuentra la fecha, el valor y la estación, para esto se creó el objeto "dato" y se añaden todos

los valores al arreglo result. Finalmente se extrae la información necesaria para construir los archivos con distintas extensiones como se presenta en la Figura 50.

Figura 50

Extracción de información.

```
async GetDatos(idEncabezado,cb){
  const archivos = await Encabezado.find({ _id:idEncabezado });
  await Datos.find({ encabezado:idEncabezado }).populate('encabezado').then(data=>{
    let result = [];
    for (let index = 0; index < data.length; index++) {
      const element = data[index];
      const dato ={
        codigoEstacion: element.encabezado.codigoEstacion,
        estacion: element.estacion,
        fecha: element.fecha,
        ano: element.fecha.split("/")[0],
        mes: element.fecha.split("/")[1],
        dia: element.fecha.split("/")[2],
        valor: element.valor,
        magnitudArchivo: element.encabezado.magnitud
      }
      result.push(dato);
    }
  });
  cb(result);
}
```

Nota. Extracción de datos previo a la construcción del archivo de salida.

Elaborado por: Los autores, disponible en <https://github.com/crizgd221116/Repo-Met/blob/main/src/controllers/dataController.js>.

La librería json2csv permite exportar los datos en un archivo con extensión CSV, los parámetros que necesita son: nombre del archivo que en este caso será la magnitud, los fields que serán los títulos de los datos y finalmente la data. Los fields y la data son pasados por la opción parser la cual delimitará al contenido por comas. Mientras que en header se define el tipo de archivo de salida, que será CSV tal como se presenta en la Figura 51.

Figura 51

Generar archivo CSV.

```
const { Parser } = require('json2csv');

const downloadResource = (res, fileName, fields, data) => {
  const opts = { fields };
  const parser = new Parser(opts);
  const csv = parser.parse(data);
  res.header('Content-Type', 'text/csv');
  res.attachment(fileName);
  return res.send(csv);
}
```

Nota. Construcción del archivo CSV.

Elaborado por: Los autores, disponible en

<https://github.com/crizgd221116/Repo-Met/blob/main/src/utilities/util.js>.

En el controlador de descarga de archivo se establecen los nombres de los títulos de los datos y se realiza una instancia del controlador de los datos que se definió en la imagen GETDATOS para poder obtener el id del encabezado y qué datos pertenecen a este, finalmente se retorna en el método downloadResource el nombre del archivo, títulos de los datos y la data como muestra la Figura 52.

Figura 52

Parámetros para la descarga.

```
controller.download = async (req, res) => {
  const fields = ['fecha', 'estacion', 'valor'];
  const dc = new DataController();
  const data2 = await dc.GetDatos(req.params.id, data => {
    return downloadResource(res, `${data[0].magnitudArchivo}` + '.csv', fields, data);
  })
}
```

Nota. Descarga del archivo en formato CSV.

Elaborado por: Los autores, disponible en <https://github.com/crizgd221116/Repo-Met/blob/main/src/controllers/downloaderController.js>.

En la función de la Figura 53 se establecen los parámetros para construir la información del txt ya que este archivo contendrá 5 columnas (código, año, mes, día y valor) estos serán los títulos de los datos y se almacenarán en la variable “contenido”. Luego se realizará un bucle for para recorrer la información y darle el formato que requiere como se presenta en la Figura 53. Finalmente, como respuesta se envía en las cabeceras el tipo de archivo que es “text”, el nombre del archivo que será la magnitud y se envía la variable “contenido” que es la que contiene los valores del archivo.

Figura 53

Construcción txt.

```
const downloadResourceTxt = (res, fileName, data) => {
  let contenido = `codigo\tanio\tmes\tdia\tvalor\n`;
  for (let index = 0; index < data.length; index++) {
    const element = data[index];
    contenido = `${contenido}${element.codigoEstacion}\t${element.anio}\t${element.mes}\t${element.dia}\t${element.valor}\n`;
  }
  res.header('Content-Type', 'text');
  res.attachment(fileName);
  return res.send(contenido);
}
```

Nota. Construcción de archivo txt para su descarga.

Elaborado por Los autores, disponible en

<https://github.com/crizgd221116/Repo-Met/blob/main/src/utilities/utilTxt.js>

3.4 PRUEBAS

3.4.1 Pruebas de Carga.

Las pruebas de carga sirven para estudiar el comportamiento de la aplicación web en un entorno simulado que permita identificar la capacidad de respuesta y la carga que puede tolerar la aplicación. En este caso se evaluó la ruta de “Login” para que se realicen peticiones de 100 usuarios en 1 segundo, dando como resultado la respuesta de ok en 100/100 de las peticiones requeridas.

Figura 54

Prueba de carga con 100 peticiones.

Etiqueta	Tiempo de Mu...	Estado	Bytes	Sent Bytes	Latency	Connect Time...
Login	1207	✓	4499	130	1203	66
Login	1196	✓	4499	130	1192	55
Login	1166	✓	4497	130	1153	16
Login	1216	✓	4499	130	1216	65
Login	1513	✓	4499	130	1513	66
Login	1512	✓	4497	130	1512	64
Login	1484	✓	4505	130	1484	31
Login	1514	✓	4497	130	1514	61
Login	1620	✓	4497	130	1620	63
Login	1633	✓	4503	130	1633	63
Login	1671	✓	4499	130	1671	61
Login	1781	✓	4499	130	1781	63
Login	1781	✓	4501	130	1781	61
Login	2003	✓	4499	130	2003	64
Login	2001	✓	4501	130	2001	62

Nota. Resultados de la prueba de carga con 100 peticiones, siendo todas ellas exitosas.

Elaborado por: Los autores, a través de JMeter.

3.4.2 Prueba de carga con 400 usuarios

Como se puede observar a continuación en la Figura 55 la aplicación se cayó con la petición del usuario 324, es decir que solamente soporta 323 peticiones por segundo.

Figura 55

Prueba con 400 peticiones.

Etiqueta	Tiempo de Mu...	Estado	Bytes	Sent Bytes	Latency	Connect Time(...)
Login 400	9633	✓	4501	130	9618	1
Login 400	9634	✓	4499	130	9634	0
Login 400	9636	✓	4503	130	9636	0
Login 400	1502099	✗	2556	0	0	1
Login 400	1502050	✗	2556	0	0	0
Login 400	1502076	✗	2556	0	0	1
Login 400	1502045	✗	2556	0	0	1
Login 400	1501904	✗	2556	0	0	1
Login 400	1502101	✗	2556	0	0	0
Login 400	1502064	✗	2556	0	0	1

Nota. Resultados de la prueba de carga con 400 peticiones, de las cuales solo resultaron exitosas 323.

Elaborado por: Los autores, a través de JMeter.

En la Figura 56 se muestra las 300 peticiones simultáneas a la ruta de autenticación, una vez autenticado como investigador se realizan otras 300 peticiones hacia la ruta del investigador donde las muestras fueron completadas satisfactoriamente con un rendimiento aceptable dando un total entre los dos recorridos de 59.3 segundos.

Figura 56

Prueba de rendimiento.

Etiqueta	# Muestras	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento
Autenticacion	300	1176	151	2268	620,94	0,00%	40,9/sec
Investigador	300	2884	324	4034	1078,54	0,00%	30,1/sec
Total	600	2030	151	4034	1226,32	0,00%	59,3/sec

Nota. Resumen de resultados de la prueba de rendimiento.

Elaborado por: Los autores, a través de JMeter.

3.4.3 Pruebas de funcionalidad

Tabla 13

Prueba de funcionalidad 1.

Prueba Funcional	1	Módulo	Investigador
Descripción	Comprobar el registro del usuario investigador	Prioridad	Alta
Casos de prueba			
N.º	Descripción	Resultado esperado	Resultado obtenido
1	Clic en el botón enviar con los campos vacíos	Mensaje de advertencia que llene todos los campos	Mensaje de advertencia “llene todos los campos”
2	Ingresar texto en el correo	Debe ingresar un correo valido	Mensaje “Ingrese un correo valido”
3	Ingresar contraseñas que no coinciden	Mensaje de advertencia de claves erróneas	Mensaje “contraseñas no coinciden”
4	Campos correctamente llenados	Envió de enlace de	Correo de confirmación de registro al sistema

		confirmación al correo	
--	--	---------------------------	--

Nota. Comprobar el registro de usuario investigador.

Elaborado por: Los autores.

Tabla 14

Prueba funcional 2.

Prueba Funcional	2	Módulo	Investigador
Descripción	Recuperar contraseña	Prioridad	Alta
Casos de prueba			
N.º	Descripción	Resultado esperado	Resultado obtenido
1	Correo no registrado	Mensaje de advertencia que el usuario no existe	Mensaje de advertencia “usuario no existe”
2	Correo registrado	Enlace de recuperación enviado al correo	Recepción de correo con el enlace de recuperación de la contraseña-

Nota. Prueba de funcionalidad recuperación de contraseña.

Elaborado por: Los autores.

Tabla 15*Prueba funcional 3.*

Prueba Funcional	3	Módulo	Investigador
Descripción	Subir Archivo REMMAQ	Prioridad	Alta
Casos de prueba			
N.º	Descripción	Resultado esperado	Resultado obtenido
1	Archivo con el formato REMMAQ	Resumen de los datos del archivo que presenta el título, cantidad de registros, estaciones, fecha de inicio del primer registro y fecha fin del último registro	Resumen de la información que tiene el archivo
2	Archivo sin el formato REMMAQ	No se presenta información de	Resumen del archivo vacío, debe encontrarse en el

		resumen del archivo	formato de REMMAQ
--	--	------------------------	----------------------

Nota. Prueba de funcionalidad subir archivos REMMAQ.

Elaborado por: Los autores.

Tabla 16

Prueba funcional 4.

Prueba Funcional	4	Módulo	Investigador
Descripción	Subir Archivo INAMHI	Prioridad	Alta
Casos de prueba			
N.º	Descripción	Resultado esperado	Resultado obtenido
1	Archivo con el formato de INAMHI	Resumen de los datos del archivo que presenta el título, cantidad de registros, estaciones, fecha de inicio del primer registro y fecha	Resumen de la información que tiene el archivo

		fin del último registro	
2	Archivo sin el formato INAMHI	No se presenta información de resumen del archivo	Resumen del archivo vacío, debe encontrarse en el formato de INAMHI

Nota. Prueba de funcionalidad subir archivo INAMHI.

Elaborado por: Los autores.

CAPÍTULO IV

IMPLEMENTACIÓN

En el presente capítulo se documenta el proceso de despliegue de la aplicación en un servidor web para que pueda ser utilizado a través de internet.

4.1 ENTORNO DE IMPLEMENTACIÓN.

Se gestionó el pedido de un espacio en los servidores de la línea de inteligencia artificial del grupo de investigación IDEAIAGEOCA para la implementación del aplicativo. Se entregaron los requerimientos sobre versiones tanto de Node.js como de MongoDB a los administradores del servidor para que preparen el entorno de despliegue. Una vez realizado el pedido se obtuvieron las credenciales para acceder al servidor AWS de forma remota y desplegar el proyecto.

4.2 VARIABLES DE ENTORNO

Con fines de seguridad se creó un archivo con variables de entorno, en el cual se definió el puerto del servidor donde se encontrará levantada la aplicación y la cadena de conexión a la base de datos de Mongo. Con estas variables de entorno se logra facilitar el despliegue de la aplicación sin necesidad de tener estas variables presentes en el código.

Figura 57

Variables de entorno.



Nota. Configuración de variables de entorno en archivo .env.

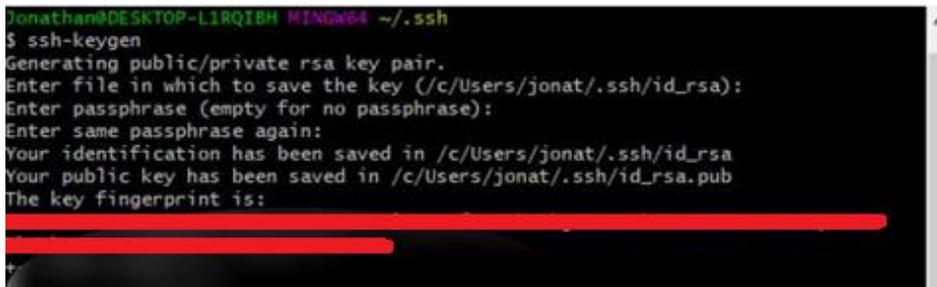
Elaborado por: Los autores.

4.3 CREDENCIALES

Se generó un archivo con la private-key de AWS, el cual se copia en el directorio .ssh. Posteriormente se generó la private y public key a través de la terminal de Git junto con la huella digital como se muestra a continuación.

Figura 58

Credenciales.



```
Jonathan@DESKTOP-LIRQIBH MINGW64 ~/.ssh
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/jonat/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/jonat/.ssh/id_rsa
Your public key has been saved in /c/Users/jonat/.ssh/id_rsa.pub
The key fingerprint is:
```

Nota. Implementación de credenciales para realizar la conexión.

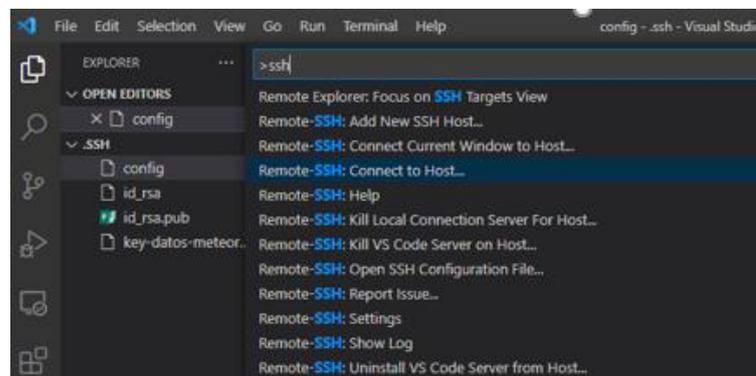
Elaborado por: Los autores.

4.4 CONEXIÓN

Una vez generada las llaves se procedió a utilizar la herramienta de Visual Studio Code para establecer la conexión mediante SSH con el servidor

Figura 59

Conexión remota al servidor.



Nota. Conexión a través de SSH con el servidor remoto utilizando Visual Studio.

Elaborado por: Los autores.

A continuación, se procede a clonar el repositorio para verificar el funcionamiento y habilitar el puerto respectivo para la correcta funcionalidad.

Figura 60

Clonar repositorio.

```
datos-meteorologicos@ [redacted]:~$ git clone https://github.com/c
Cloning into [redacted]
remote: Enumerating objects: 597, done.
remote: Counting objects: 100% (597/597), done.
remote: Compressing objects: 100% (355/355), done.
remote: Total 597 (delta 374), reused 442 (delta 222), pack-reused 0
Receiving objects: 100% (597/597), 1.46 MiB | 13.48 MiB/s, done.
Resolving deltas: 100% (374/374), done.
```

Nota. Descarga del repositorio de GitHub en el servidor.

Elaborado por: Los autores.

Para que la aplicación se mantenga en ejecución sin la necesidad de recurrir un comando de forma repetitiva, se utilizó el gestor de procesos de Node.js “pm2” que facilita la administración de la aplicación.

Figura 61

Panel de administración Pm2.

```
[PM2] Applying action restartProcessId on app [datos-metereologicos](ids: [ 0 ])
[PM2] [datos-metereologicos](0) ✓
[PM2] Process successfully started
```

id	name	mode	♻	status	cpu	memory
0	datos-metereologi...	Fork	59	online	0%	15.7mb

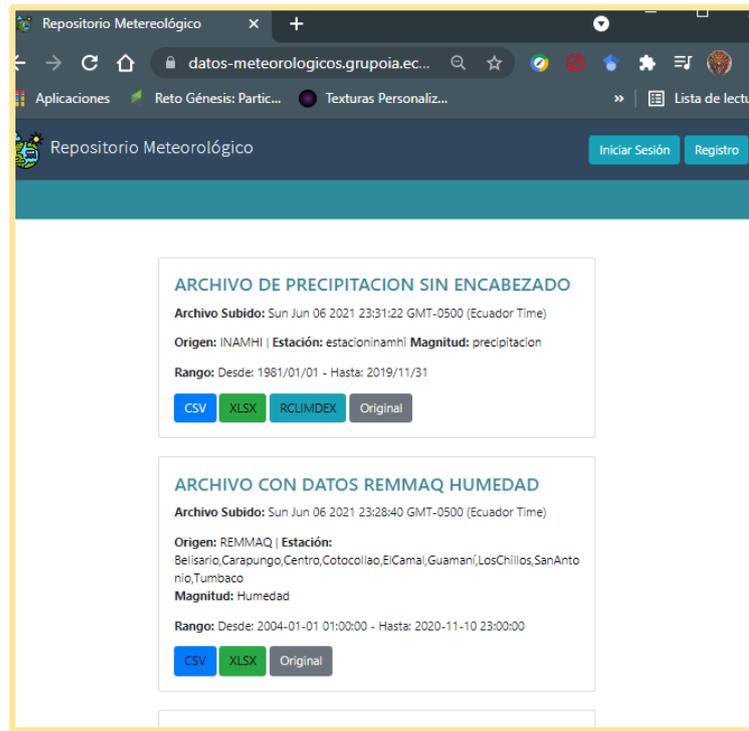
Nota. Cuadro de resumen de pm2 para la administración de la ejecución del proyecto.

Elaborado por: Los autores.

Finalmente, con la ayuda del gestor de procesos “pm2” se ejecuta la aplicación y se habilita el dominio para visualizarla desde internet como se muestra a continuación. Además, se puede ver el formato de los archivos admitidos por el sistema como los formatos de salida en el apartado de Anexos.

Figura 62

Aplicación desplegada.



Nota. Repositorio implementado en el servidor AWS y funcionando en internet.

Elaborado por: Los autores.

CONCLUSIONES

Se logró la construcción del aplicativo acorde a lo establecido en las etapas de planificación y análisis del proyecto, teniendo como resultado un repositorio de datos en donde se pueden compartir archivos de tipo meteorológico en el formato entregado por instituciones como INAMHI y REMMAQ para que tanto investigadores como personas interesadas en el estudio de este tipo de información los puedan tener a su alcance y agilizar el proceso de divulgación de estos.

La utilización de la metodología SCRUM para el desarrollo del aplicativo permitió realizar el seguimiento del proyecto de forma controlada y organizada. Esto se logró con la utilización de la herramienta Taiga la cual cuenta una interfaz dedicada para emplear esta metodología, razón por la cual el equipo de desarrollo pudo darle prioridad a la codificación de la solución. Adicionalmente esto permitió corregir errores y mejorar aspectos fundamentales del sistema para poder presentarlos dentro de los plazos acordados.

Una correcta investigación del área de interés del proyecto ayuda a tener una noción del tipo de terminología que va a ser manipulada por el equipo de desarrollo. En el presente proyecto fue necesario adquirir conocimientos sobre meteorología, lo cual resultó retador debido a la aparición de nuevos conceptos, en su mayoría ajenos a la carrera de sistemas, pero que a su vez resultaron esenciales para entender el escenario en el cual será empleada la solución.

El paradigma de acceso abierto a la información permite agilizar el desarrollo de la investigación científica y tecnológica pues brinda a los investigadores la facilidad de obtener los datos necesarios para desempeñar sus tareas sin involucrarse en procesos de solicitud demorosos.

Se logró mejorar la lectura de archivos INAMHI pues al iniciar el proyecto se mantenía un estándar, es decir, existía un encabezado al inicio del archivo. Sin embargo, tiempo después se entregó al equipo de desarrollo archivos que ya no contenían dicho encabezado por lo cual no

se realizaba la lectura adecuada de estos últimos. Se logró adaptar el sistema para que pueda leer ambos tipos de archivo.

Se realizó el cambio de plataforma para el despliegue del repositorio. Inicialmente se definió a Heroku como la plataforma para la implementación en Internet del sistema, pero debido a las limitaciones técnicas como el no permitir la subida de archivos al servidor, además de una lenta velocidad de respuesta, se decidió cambiar por un servidor de AWS que fue entregado por la línea de inteligencia artificial del grupo de investigación IDEAIAGEOCA.

RECOMENDACIONES

Se recomienda para futuras versiones del proyecto la implementación de filtros que permitan realizar una búsqueda más robusta de información y la construcción de archivos bajo los parámetros definidos por los filtros.

El repositorio actualmente permite la subida de un solo archivo, para futuros trabajos se recomienda realizar un estudio previo del middleware Multer para poder mejorar el módulo de carga de archivo y así poder subir más de un archivo a la vez.

Para el desarrollo de nuevos módulos se recomienda tener conocimientos de JavaScript del lado del servidor y sobre Handlebars que es el motor de plantillas para así mantener una estructura ordenada en el manejo del código fuente.

Para futuros cambios se recomienda realizar un módulo de administración donde se pueda crear estaciones y asociarlas con las respectivas medidas de los datos para tener un mejor control de la información que se sube.

Es recomendable la utilización de variables de entorno con la finalidad de proteger información sensible del proyecto al momento de subir cambios al repositorio de GitHub, esto con la finalidad de evitar la vulneración del sistema.

REFERENCIAS BIBLIOGRÁFICAS

Artículos académicos o científicos

- Álvarez, J. A., Álvarez, M. M., Gallegos, V., & Polanco, I. (2011). La importancia de los REPOSITARIOS INSTITUCIONALES PARA LA EDUCACIÓN Y LA INVESTIGACIÓN. *Synthesis*, 43-57.
http://www.uach.mx/extension_y_difusion/synthesis/2011/08/18/la_importancia_de_los_repositorios_institucionales_para_la_educacion_y_la_investigacion.pdf
- Barton, M. R., & Waters, M. M. (2005). Cómo crear un repositorio institucional: Manual LEADIRS II. *MIT Libraries, Cmi*, 169.
<http://www.recolecta.net/buscador/documentos/mit.pdf>
- Canós, J.H., & Letelier, M. (2012). Metodologías Ágiles en el Desarrollo de Software.
- García, F. F. (2012). Meteorología y climatología. Aspectos generales. *Índice: Revista De Estadística y Sociedad*, (50), 6-9.
- Gattinoni, N., Boca, T., C, R., & Di Bella, C. (2011, marzo 17). Comparación entre observaciones meteorológicas obtenidas de estaciones convencionales y automáticas a partir de la estimación de parámetros estadísticos. *37(1)*, 75-85.
- Soares Guimarães, M.^a C., Silva, C. H. da, & Horsth Noronha, I.. (2012). Los repositorios temáticos en la estrategia de la iniciativa Open Access. *Nutrición Hospitalaria*, 27(Supl. 2), 34-40. <https://dx.doi.org/10.3305/nh.2012.27.sup2.6271>
- Navarro Cadavid, A., Fernández Martínez, J. & Morales Vélez, J. (2013). Revisión de metodologías ágiles para el desarrollo de software. *PROSPECTIVA*, 11(2), 30-39.
<https://www.redalyc.org/articulo.oa?id=496250736004>

Conferencias

- Rodríguez Jiménez, R. M., Capa, Á. B., & Portela Lozano, A. (2004). Meteorología y Climatología. *Semana de Ciencia y la Tecnología 2004*.

Libros

Corral, G., Borrego, J., & Galán, J. (2014). *Extracción y organización del conocimiento de etiquetados. Aplicación a etiquetados en repositorios digitales sobre Arte*.

Fontela, C. (2011). *UML Modelado de Software para profesionales 2da Edición*. Alfaomega Grupo Editor.

Luna, F. (2019). *JAVASCRIPT aprende a programar en el lenguaje de la web*. Six Ediciones.

Sanso, A., & Richer, J. (2017). *Oauth2 in action*. Manning Publications.

García, F. F. (2012). Meteorología y climatología. Aspectos generales. *Índice: Revista De Estadística y Sociedad*, (50), 6-9.

Páginas Web

Budapestopenaccessinitiative.org. (2002). *Budapest Open Access Initiative | Read The Budapest Open Access Initiative*. <https://www.budapestopenaccessinitiative.org/read>

Comunidad MDN. (2020, noviembre 23). *JavaScript | MDN*. <https://developer.mozilla.org/es/docs/Web/JavaScript>

GIT. (s.f.). *git--fast-version-control*. <https://git-scm.com>

INAMHI. (2016, mayo). *INAMHI*. http://www.serviciometeorologico.gob.ec/wp-content/uploads/downloads/2016/05/d_servicios_que_ofrece_y_las_formas_de_accederlos.pdf

Joyent. (s.f.). *Node.js*. <https://nodejs.org/es/>

Katz, Y. (2011). <https://devdocs.io>. Obtenido de Handlebars.js: <https://devdocs.io/handlebars/>

Legacy.earlham.edu. (2003). *Bethesda Statement On Open Access Publishing*. <http://legacy.earlham.edu/~peters/fos/bethesda.htm>

MongoDB. (s.f.). *What is MongoDB?*. <https://www.mongodb.com/what-is-mongodb>

Toro, L. (s.f.). *Desde Linux*. Taiga, la Mejor Herramienta para la Gestión de Proyectos Ágiles + Caso Práctico. <https://blog.desdelinux.net/66643-2/>

WMO. (2011). Guide to Climatological Practices.

Suber, P. (2014). *Open Access Overview (definition, introduction)*. Earlham.edu.

<http://www.earlham.edu/~peters/fos/overview.htm>.

Revistas

Talavera, A. (2014). Lo que debemos saber sobre meteorología. Ecuador.

Repositorio

Castillo, J. & Salazar, C., 2021. crizgd221116/Repo-Met. GitHub.

<https://github.com/crizgd221116/Repo-Met.git>

Talleres

Ayarzagüena Porras, B., Yagüe Anguís, C., & Zubiaurre Molina, I. (s.f.). *Precipitación y Nubes*. Obtenido de Taller Virtual de Meteorología y Clima:

<http://meteolab.fis.ucm.es/meteorologia/precipitacion-y-nubes>

Tesis

Ayala, R. A. (2019). APLICACIÓN WEB PARA VISUALIZACIÓN DE INFORMACIÓN METEOROLÓGICA [Tesis de Pregrado, Universidad de las Américas]. Dspace UDLA
<http://dspace.udla.edu.ec/handle/33000/10751>

Bravo, D., García, A., & Muñoz, W. (2012, 15 de octubre). *Diseño e Implementación de un Prototipo de Estación Meteorológica* [Tesis de pregrado, Escuela Politécnica Nacional]

Bibdigital EPN. <http://bibdigital.epn.edu.ec/handle/15000/4240>

Moreno, J. V. (2020, 20 de enero). Herramienta web para la extracción y procesamiento de información a partir de ficheros CSV. Repositorio Institucional de la Universidad de Málaga. <https://hdl.handle.net/10630/19198>

Moscoso, V., Serrano Vincenti, S., Jácome, P., Palacios, E., & Villacís, M. (2012, diciembre). ANÁLISIS ESTADÍSTICO DE DATOS METEOROLÓGICOS MENSUALES Y DIARIOS PARA LA DETERMINACIÓN DE VARIABILIDAD CLIMÁTICA Y

CAMBIO CLIMÁTICO EN EL DISTRITO METROPOLITANO DE QUITO. Dspace
Universidad Politécnica Salesiana. <http://dspace.ups.edu.ec/handle/123456789/8828>

Oliva, R. B. (2012). ESTACIÓN METEOROLÓGICA DE CONSTRUCCION MODULAR
ORIENTADA A LA PROSPECCION EÓLICA EN ARGENTINA. [Tesis de
posgrado. Universidad Nacional de Salta de Argentina]. Archivo Digital.
https://www.researchgate.net/profile/Rafael-Oliva-3/publication/339354332_ESTACION_METEOROLOGICA_DE_CONSTRUCCION_MODULAR_ORIENTADA_A_LA_PROSPECCION_EOLICA_EN_ARGENTINA/links/5e4d30f3a6fdccd965b0debb/ESTACION-METEOROLOGICA-DE-CONSTRUCCION-MODULAR-ORIENTADA-A-LA-PROSPECCION-EOLICA-EN-ARGENTINA.pdf

Paucar Quinteros, W. D. (2017, septiembre). *Estudio e implementación del punto triple del mercurio en la calibración de instrumentos que miden la magnitud temperatura* [Tesis de pregrado, Escuela Politécnica Nacional] Bibdigital EPN.
<http://bibdigital.epn.edu.ec/handle/15000/19080>