UNIVERSIDAD TECNOLÓGICA DE PANAMÁ
FACULTAD DE INGENIERÍA DE SISTEMAS COMPUTACIONALES

AUTOMATIZACIÓN PARA LA DESCARGA DE IMÁGENES SATELITALES NOCTURNAS PARA EL MONITOREO DE LA CONTAMINACIÓN LUMÍNICA EN LA CUENCA DEL CANAL DE PANAMÁ

ASESOR:

M.Sc RAFAEL VEJARANO

CO-ASESOR:

Dr. JOSÉ ROBLES

INTEGRANTE:

JOSÉ DE LOS SANTOS JAÉN JAÉN

TRABAJO DE GRADUACIÓN PARA OPTAR AL TÍTULO DE LICENCIADO EN DESARROLLO DE SOFTWARE

2025

Resumen

La contaminación lumínica a lo largo del recorrido del Canal de Panamá representa un desafío ambiental significativo, con efectos adversos en los ecosistemas nocturnos, emisiones de CO_2, consumo eléctrico y la salud humana. Este proyecto, realizado en conjunto con la Universidad Tecnológica de Panamá (UTP) y el Instituto Nacional de Investigaciones Científicas Avanzadas en Tecnologías de Información y Comunicación (INDICATIC AIP), propone un sistema automatizado para el monitoreo de la contaminación lumínica mediante el uso de imágenes multiespectrales nocturnas, obtenidas tanto de la Estación Espacial Internacional (ISS, por sus siglas en inglés), como de sensores satelitales especializados como DMSP-OLS y VIIRS.

Los objetivos principales incluyen la configuración de una interfaz de programación de aplicaciones (API) para la descarga automatizada de imágenes satelitales, el desarrollo de scripts en Python para procesarlas, y su integración en un sistema de almacenamiento masivo (NAS) con capacidad de 144 TB. La incorporación de datos de DMSP-OLS, con cobertura histórica desde 1992, y de VIIRS, con alta resolución desde 2011, complementa las imágenes RGB de la ISS, permitiendo un análisis multitemporal y espacial más preciso de la contaminación lumínica en la cuenca del Canal de Panamá. El resultado esperado es un sistema eficiente que apoye la gestión, geoclasificación y análisis automatizado de imágenes satelitales nocturnas.

Palabras clave: Contaminación lumínica, APIs, imágenes satelitales, DMSP-OLS, VIIRS, Canal de Panamá.

Dedicatoria

A mis padres y hermanas, por su apoyo incondicional a lo largo de esta etapa académica.

A los docentes y mentores que, con su guía, me ayudaron a desarrollar las habilidades necesarias para completar este trabajo.

Y a todas las personas que, de una forma u otra, contribuyeron con palabras de aliento, consejos técnicos o compañía en el proceso.

Agradecimientos

Deseo expresar mi más sincero agradecimiento a la Universidad Tecnológica de Panamá por brindarme la formación académica y los recursos necesarios para llevar a cabo este trabajo.

Agradezco especialmente al M.Sc. Rafael Vejarano y al Dr. José Robles por su orientación, conocimientos compartidos y constante acompañamiento durante el desarrollo de este proyecto.

Extiendo mi gratitud al equipo del Instituto Nacional de Investigaciones Científicas Avanzadas en Tecnologías de Información y Comunicación (INDICATIC AIP) por su colaboración y por haber proporcionado el entorno técnico e institucional que permitió la implementación de este sistema.

También reconozco el valioso aporte de la Secretaría Nacional de Ciencia, Tecnología e Innovación (SENACYT), cuyo financiamiento hizo posible la ejecución del proyecto bajo el código FID2024-072.

Finalmente, agradezco a todas las personas que de forma directa o indirecta contribuyeron al desarrollo de este trabajo, ya sea mediante apoyo técnico, académico o moral.

Proyecto co-financiado por: INDICATIC AIP y la SENACYT bajo los fondos de investigación FID2024-072.

Índice general

Índice de Figuras

Índice de Tablas

Introducción

La contaminación lumínica es un fenómeno ambiental derivado del uso excesivo e inadecuado de luz artificial durante la noche. Este tipo de contaminación altera la oscuridad natural del cielo nocturno y se ha convertido en un problema creciente en zonas rurales, peri-urbanas y urbanas. Aunque existe un amplio conocimiento sobre sus efectos en la biodiversidad, la salud humana y el consumo energético, la capacidad para estudiar este fenómeno depende en gran medida de la disponibilidad y análisis de datos específicos, tales como imágenes satelitales nocturnas [3].

En regiones estratégicas como la cuenca hidrográfica del Canal de Panamá, caracterizada por una intensa actividad económica y una gran diversidad ecológica, la contaminación lumínica plantea desafíos importantes. Sin embargo, el monitoreo efectivo de esta problemática requiere herramientas tecnológicas que permitan obtener y procesar datos de forma eficiente. Actualmente, la descarga de imágenes satelitales nocturnas desde misiones como la Estación Espacial Internacional (ISS, por sus siglas en inglés), el Sistema Operativo de Barrido de Líneas del Programa Meteorológico de Defensa (DMSP-OLS, por sus siglas en inglés) y el Conjunto de radiómetros de imágenes infrarrojas visibles (VIIRS, por sus siglas en inglés) representa un proceso de descarga manual, propenso a errores y poco escalable para estudios a largo plazo, cabe destacar que las imágenes capturadas son de acceso libre y gratuito para uso científico, educativo y de desarrollo.

Además, las plataformas disponibles para la descarga de estos datos no siempre ofrecen interfaces amigables o mecanismos de automatización accesibles, lo que limita su uso por parte de investigadores y técnicos que necesitan grandes volúmenes de información para análisis detallados. Esta situación evidencia una carencia tecnológica: la falta de un sistema automatizado que permita gestionar de manera continua y eficiente la descarga, organización y almacenamiento de imágenes satelitales específicas para el estudio de la contaminación lumínica.

El presente trabajo de tesis de grado propone desarrollar un sistema automatizado, que facilite la descarga y gestión de imágenes satelitales nocturnas. Este sistema utiliza interfaces y scripts personalizados que

interactúan con APIs públicas de acceso a datos espaciales, permitiendo optimizar el proceso de recolección de datos. El sistema está diseñado para operar en entornos Linux, específicamente mediante la capa de compatibilidad canónica Windows Subsystem for Linux (WSL), lo que permite una integración eficiente con herramientas de automatización como aria2c (utilidad de descarga por línea de comandos de múltiples fuentes) y manejo de archivos.

El sistema automatizado propuesto no solo gestiona la descarga masiva de archivos, sino que también realiza el procesamiento básico de metadatos, la verificación de integridad de los archivos y su organización en un entorno de almacenamiento conectado en red (NAS, por sus siglas en inglés), con capacidad para manejar volúmenes de datos superiores a los 25 GB, dependiendo de la misión espacial. Esta infraestructura busca proporcionar una base sólida para futuras investigaciones científicas, facilitando análisis más precisos sobre la contaminación lumínica en la cuenca del Canal de Panamá, así como en otras regiones que carezcan de sistemas de medición del brillo y color del cielo nocturno desde estaciones terrestres. Además, en aquellas regiones que ya cuentan con estaciones de monitoreo, las herramientas desarrolladas en esta tesis permiten realizar análisis comparativos entre mediciones terrestres y observaciones por teledetección, enriqueciendo así la interpretación multimodal de los datos.

Desde una perspectiva técnica, este desarrollo contribuye a cerrar una brecha existente entre la disponibilidad de datos satelitales y la capacidad de los investigadores para utilizarlos de forma ágil y sistemática. Automatizar estos procesos no solo reduce significativamente el tiempo requerido para obtener grandes conjuntos de datos, sino que también minimiza errores humanos y mejora la trazabilidad de la información recolectada. La metodología empleada en el desarrollo de este sistema se basa en principios del desarrollo ágil de software, adoptando un enfoque iterativo e incremental que permite ajustar funcionalidades de acuerdo con los requerimientos del proyecto. Se hace uso de herramientas de código abierto, lo cual garantiza la flexibilidad y escalabilidad

del sistema para futuras mejoras o adaptaciones a otros contextos de estudio. Este documento se organiza de la siguiente manera: el primer capítulo introduce el problema, el contexto, los objetivos de la investigación y la revisión de la literatura. El segundo capítulo describe la arquitectura del sistema automatizado y las principales herramientas empleadas en el proyecto. El tercer capítulo detalla el desarrollo e implementación del sistema. El cuarto capítulo analiza el contexto y el contenido científico al que contribuye la API, preparando el terreno para el estudio posterior. Finalmente, el quinto capítulo presenta las conclusiones del estudio y ofrece recomendaciones para la mitigación del problema.

Glosario de Términos

Término	Definición
DMSP-OLS	*Defense Meteorological Satellite Program - Operational Linescan System*. Sensor óptico desarrollado por el Departamento de Defensa de EE. UU. para captar imágenes nocturnas de la Tierra. Aunque fue diseñado con fines meteorológicos, ha sido ampliamente utilizado en estudios de contaminación lumínica y crecimiento urbano. Su resolución es baja (2.7 km) y presenta limitaciones por saturación de brillo. Las imágenes capturadas son de acceso libre y gratuito para uso científico, educativo y de desarrollo.
VIIRS	*Visible Infrared Imaging Radiometer Suite*. Sensor satelital moderno que capta imágenes tanto en el espectro visible como en el infrarrojo. Posee una banda especial para capturar imágenes nocturnas de alta calidad (Day/Night Band). Es el sucesor del DMSP-OLS y ofrece mejor resolución, calibración y detección de fuentes luminosas en la superficie terrestre. Utilizado en satélites como Suomi NPP y NOAA-20. Las imágenes capturadas son de acceso libre y gratuito para uso científico, educativo y de desarrollo.
Estación Espacial Internacional (EEI)	Plataforma orbital que permite observaciones de la Tierra en tiempo real, incluyendo imágenes de emisiones nocturnas. Es operada por múltiples agencias espaciales (NASA, ESA, Roscosmos, JAXA, etc.) y ha sido fuente de imágenes relevantes para estudios de contaminación lumínica.
Electron	Framework para el desarrollo de aplicaciones de escritorio multiplataforma utilizando tecnologías web como HTML, CSS y JavaScript. Aunque no es un sensor ni API satelital, puede ser utilizado como parte de sistemas de visualización o interfaces para el procesamiento de imágenes en algunos proyectos.

Término	Definición
API (Application Programming Interface)	Conjunto de definiciones y protocolos que permiten que dos sistemas de software se comuniquen entre sí. En el contexto del trabajo, las API permiten acceder automáticamente a bases de datos satelitales para descargar imágenes o metadatos.
Contaminación lumínica	Alteración de la oscuridad natural del cielo nocturno debido al uso excesivo o mal direccionado de luz artificial. Afecta a la biodiversidad, la salud humana y dificulta la observación astronómica. Es un tema central del documento.

Capítulo 1: Antecedentes de misiones espaciales con tecnología de imágenes de observación terrestre

1.1 Introducción

La contaminación lumínica es un problema ambiental causado por el uso excesivo e inadecuado de luz artificial durante la noche. Adicionalmente, se considera como la alteración de la oscuridad del medio nocturno debido a la emisión de luz artificial que no solo es innecesaria, sino que también está mal direccionada o es desproporcionada [4]. Este fenómeno afecta tanto a áreas urbanas como rurales, incluidas zonas protegidas, alterando los ciclos naturales de luz y oscuridad, lo que genera impactos en la biodiversidad, la salud humana y procesos ecológicos fundamentales [5].

En este contexto, resulta indispensable desarrollar soluciones computacionales que permitan monitorear y analizar de manera sistemática, según la frecuencia orbital sobre el área de interés, las emisiones lumínicas. Este trabajo propone como eje central la creación de un sistema automatizado, orientado a fortalecer los estudios sobre la contaminación lumínica en la cuenca hidrográfica del Canal de Panamá. El sistema está diseñado para facilitar la recopilación, organización y gestión de imágenes satelitales nocturnas, georreferenciadas o no, provenientes de la ISS, DMSP-OLS y VIIRS, proporcionando una herramienta eficiente para el análisis ambiental.

La cuenca hidrográfica del Canal de Panamá constituye un caso de estudio relevante debido a la interacción de la luz nocturna entre zonas urbanas, industriales y ecosistemas naturales, lo que podría afectar eventualmente el funcionamiento del propio canal. Las actividades económicas intensivas, la urbanización acelerada y el tránsito marítimo constante contribuyen a la diversidad de fuentes de luz artificial, impactando tanto el entorno ecológico

como la calidad de vida de las comunidades locales. Sin embargo, a pesar de su importancia estratégica, el monitoreo de la contaminación lumínica en esta región enfrenta desafíos sustanciales, especialmente en cuanto al acceso a datos geoespaciales precisos y actualizados.

1.2 Marco Teórico

El monitoreo de la contaminación lumínica ha evolucionado significativamente gracias al acceso creciente a datos satelitales y al desarrollo de herramientas computacionales avanzadas, en especial en lugares donde no se cuenta con medidas de brillo, color y espectro hechas en estaciones terrestres. Este proyecto se centra en el diseño de un sistema automatizado para la descarga y organización de imágenes nocturnas en la cuenca hidrográfica del Canal de Panamá, una región donde el impacto de la luz artificial ha sido históricamente poco estudiado.

Tradicionalmente, existen misiones como el DMSP-OLS y el VIIRS lo cuales son fundamentales para el estudio de la iluminación nocturna de forma remota. El sensor que se encuentra incorporado en el DMSP-OLS ha proporcionado datos desde 1992, lo que lo convierte en una fuente valiosa plataforma para estudios evolutivos, aunque presenta limitaciones importantes en resolución espacial y espectral (no capta imágenes en color). El VIIRS, se encuentra operativo desde 2011, este instrumento, mejora la resolución temporal y ofrece imágenes calibradas, pero aún conserva una resolución espacial limitada y tampoco proporciona imágenes en color real [6].

Por su parte, las imágenes capturadas desde la ISS ofrecen una perspectiva geoespacial única gracias a su alta resolución (algunas hasta 3 metros por píxel) y a su riqueza cromática en formato RGB, con especial sensibilidad en la banda azul, lo cual favorece la detección de ciertas fuentes de luz artificial, figura 1.1 [7]. No obstante, presentan desafíos importantes: más del 93 % de estas imágenes carecen de georreferenciación y su ángulo oblicuo dificulta la

realización de estudios comparativos y evolutivos consistentes.

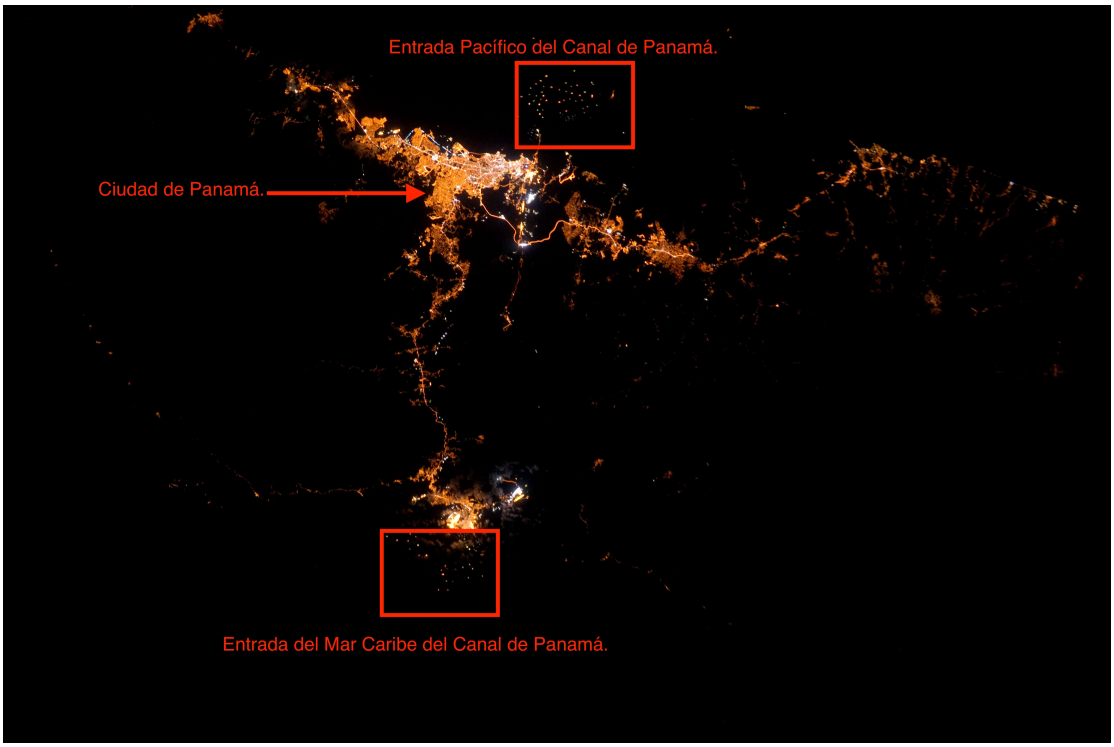

Figura 1.1: Imagen de la ciudad de Panamá capturada a bordo de la ISS mostrando la contribución lumínica de la ciudad y buques a través del Canal de Panamá. ISS030-E-167985-2 del 2012. Imágen cortesía de José Robles.

Para superar las limitaciones de georreferenciación de las imágenes ISS, estudios recientes como el de Stoken han desarrollado métodos automatizados. Su sistema *NightMatch* emplea datos de redes viales globales junto con imágenes diurnas de Sentinel para alinear y validar imágenes nocturnas mediante algoritmos de emparejamiento, logrando precisiones de geolocalización cercanas al 82 % [7].

La validación de imágenes satelitales mediante comparación con imágenes terrestres permite mejorar la precisión analítica. La identificación de puntos de referencia comunes en ambas fuentes posibilita ajustes finos, cruciales cuando se trabaja con imágenes oblicuas y no estandarizadas como las de la ISS. Aunque los sensores DMSP-OLS y VIIRS no ofrecen información de colores, su consistencia orbital permite realizar estudios fotométricos evolutivos. En este proyecto, el enfoque se centra en aprovechar la gran riqueza de imágenes disponibles a todo color de la ISS para crear la colección de imágenes satelitales

más grande a través del recorrido por el canal de Panamá.

1.3 Planteamiento del Problema

El monitoreo efectivo de la contaminación lumínica mediante imágenes satelitales nocturnas enfrenta desafíos técnicos significativos, especialmente cuando se utilizan fuentes como la ISS. Entre los principales obstáculos se encuentra la limitada georreferenciación de estas imágenes, lo que complica su integración en bases de datos espaciales y dificulta el estudio de radiance promedio por latitudes. Además, la alta oblicuidad en las tomas y la falta de metadatos consistentes limitan la utilización científica de las imágenes.

Por otro lado, el volumen creciente de imágenes provenientes de diferentes sensores como la ISS, VIIRS y DMSP-OLS demanda herramientas que permitan una gestión automatizada de datos satelitales. Actualmente, este proceso es mayormente manual, lo que lo hace ineficiente y poco viable para estudios que requieren un seguimiento temporal sostenido o cobertura geográfica extensa. Ante esta situación, se hace necesario el desarrollo de un sistema automatizado que permita no solo la recopilación sistemática de imágenes nocturnas, sino también su estructuración coherente para facilitar el análisis espacial y temporal de las fuentes de luz artificial por misiones espaciales. Este tipo de herramienta es clave para establecer en el futuro una metodología la cual permita estudiar la evolución de la contaminación lumínica en regiones estratégicas como la cuenca hidrográfica del Canal de Panamá.

1.4 Justificación

La falta de herramientas automatizadas para la recopilación, organización y gestión de imágenes satelitales nocturnas representa una limitación significativa en el estudio de la contaminación lumínica. Actualmente, investigadores y técnicos deben realizar procesos manuales para acceder a estos datos, lo que ralentiza los estudios, incrementa la posibilidad de errores y limita el alcance

de los análisis espaciales y temporales. Esta carencia se vuelve aún más crítica en regiones de alta sensibilidad ambiental y estratégica, como la cuenca hidrográfica del Canal de Panamá.

El presente proyecto busca atender esta necesidad mediante el desarrollo de un sistema automatizado que optimice la obtención y organización de imágenes nocturnas provenientes de sensores como la ISS, DMSP-OLS y VIIRS. Este sistema facilita la construcción de una base de datos estructurada, diseñada para estudios científicos que analicen patrones de iluminación artificial y su impacto ambiental a lo largo del tiempo.

Desde una perspectiva técnica, el sistema está diseñado con un enfoque modular y escalable, lo cual permite su adaptación a distintos entornos y volúmenes de datos. Su arquitectura contempla componentes para la descarga automática de archivos, validación de metadatos y almacenamiento organizado en un entorno NAS, garantizando la integridad y disponibilidad de la información.

A mediano plazo, este desarrollo puede evolucionar hacia una plataforma pública de acceso abierto, que permita compartir información de forma colaborativa entre investigadores, instituciones gubernamentales y organizaciones interesadas en la gestión ambiental. De esta manera, se promueve el uso de tecnologías de automatización aplicadas al monitoreo ambiental, generando un impacto positivo en la toma de decisiones informadas y en la construcción de políticas públicas basadas en evidencia.

1.5 Objetivos

1.5.1 Objetivo General

Desarrollar un sistema automatizado para la descarga, organización y almacenamiento de imágenes satelitales nocturnas, enfocándose en la cuenca hidrográfica del Canal de Panamá.

1.5.2 Objetivos Específicos

- Diseñar un esquema eficiente para la descarga masiva de imágenes satelitales utilizando herramientas como `aria2c` y mecanismos de scraping o acceso vía API.
- Implementar mecanismos de adquisición que permitan obtener imágenes satelitales nocturnas en la mejor resolución disponible, priorizando formatos como `.tif` o `.jpg`, junto con sus metadatos técnicos y espaciales, independientemente de su estado de georreferenciación.
- Configurar el sistema para realizar descargas automáticas a través de tareas programadas, sin necesidad de intervención manual.
- Desarrollar herramientas de automatización para el manejo de grandes volúmenes de datos provenientes de sensores como ISS, DMSP-OLS y VIIRS.
- Estructurar una base de datos relacional para organizar los metadatos mediante entidades normalizadas que vinculen información geográfica, atmosférica, técnica de cámara y archivos complementarios, garantizando la integridad referencial de cada imagen satelital.
- Integrar el sistema con un entorno de almacenamiento conectado en red (NAS), garantizando la integridad, disponibilidad y escalabilidad de los datos recolectados.
- Desplegar el sistema en una estación de trabajo local equipada con `WSL2`, utilizando conexiones remotas seguras vía `SSH` para su gestión y operación integrada con el entorno NAS.

1.6 Metodología de Investigación

El desarrollo del sistema siguió un enfoque iterativo basado en principios de metodologías ágiles, priorizando la automatización, escalabilidad y eficiencia en el manejo de datos. Las herramientas principales utilizadas incluyen `Python`

y `Shell` para los procesos automatizados, `Electron` para la interfaz gráfica de usuario, y un entorno de almacenamiento en red (NAS) para la gestión segura de archivos.

Durante la fase exploratoria del proyecto, se implementaron técnicas de extracción de datos desde sitios web (conocidas como *web scraping*) para acceder a imágenes nocturnas disponibles públicamente, especialmente del portal de la NASA relacionado con la Estación Espacial Internacional (ISS). Esta aproximación permitió comprobar la viabilidad del acceso automatizado, pero también evidenció limitaciones relacionadas con la dependencia de la estructura cambiante de los sitios web y la ausencia de interfaces estables.

Por ello, se migró a una solución más robusta basada en la utilización de la NASA Gateway PhotosDatabaseAPI del sistema de imágenes ISS de la Tierra. Esta API permite realizar búsquedas automatizadas y descargar imágenes junto con sus metadatos en formato estructurado. Para los sensores VIIRS y DMSP-OLS, se utilizó la plataforma Google Earth Engine (GEE) como fuente principal para la extracción de datos nocturnos.

Por ejemplo, el usuario puede programar el sistema para que ejecute descargas todos los lunes a las 6:00 a.m., pero el sistema buscará y descargará únicamente las imágenes que fueron capturadas durante horarios nocturnos y si no se encuentran almacenadas ya en el NAS, independientemente de cuándo se almacenaron en los servidores.

El sistema está configurado para descargar automáticamente imágenes que fueron capturadas entre las 6:45 p.m. y las 5:30 a.m., correspondiente al período nocturno de interés para el estudio de contaminación lumínica. Las descargas se realizan desde material previamente almacenado en los servidores de la NASA y NOAA, aplicando filtros temporales para seleccionar únicamente las imágenes nocturnas. El usuario puede programar estas descargas automáticas en cualquier momento del día, independientemente de la hora de captura original de las imágenes.

La interfaz de usuario permite visualizar los archivos disponibles, monitorear el estado del sistema y realizar búsquedas internas. Actualmente, el sistema gestiona más de 25 GB de imágenes y ha procesado más de 3,000 archivos

provenientes de distintas misiones.

Durante su desarrollo, el sistema fue sometido a fases de calibración y pruebas, ajustando los intervalos de descarga, validación de metadatos y organización jerárquica por fuente, misión y año. Esta metodología permitió consolidar una plataforma robusta, capaz de integrarse a nuevos flujos de datos y adaptable a futuros requisitos de análisis espacial (Ver diagrama de arquitectura del sistema en el Capítulo 3, Figura 3.2).

1.7 Limitaciones

El desarrollo del sistema enfrentó diversas limitaciones técnicas y operativas. La principal fue la dependencia de servicios públicos de datos, como la API de la ISS y la plataforma (GEE), cuya disponibilidad puede verse afectada por cambios en políticas de acceso, reestructuración de portales o interrupciones en los servicios. Otra limitación relevante fue la falta de estandarización en los formatos de imágenes y metadatos entre distintas fuentes. Esta variabilidad obligó al diseño de scripts personalizados para la extracción, validación y organización de datos, lo cual incrementó la complejidad del desarrollo y los tiempos de implementación.

En el caso específico de las imágenes de la ISS, la ausencia de georreferenciación en la mayoría de los archivos y la alta oblicuidad de las tomas representan un desafío persistente. Aunque el sistema organiza y almacena eficientemente estas imágenes, su uso en análisis espaciales más avanzados requiere procesos adicionales de alineación y validación radiativa. Además, el manejo de grandes volúmenes de datos implicó la adopción de estrategias de compresión, categorización y almacenamiento estructurado en el NAS. A pesar de estas medidas, la escalabilidad futura del sistema dependerá de la capacidad de procesamiento, almacenamiento y mantenimiento de las bases de datos.

Finalmente, el sistema está diseñado para operar bajo entornos específicos (Linux vía WSL y Windows), lo que puede limitar su portabilidad inmediata a otras plataformas sin ajustes técnicos adicionales.

1.8 Alcance

El sistema desarrollado permite la automatización completa de la descarga, organización y almacenamiento de imágenes satelitales nocturnas, con un enfoque específico en la cuenca hidrográfica del Canal de Panamá. Actualmente, el sistema gestiona datos provenientes de tres fuentes principales: la Estación Espacial Internacional (ISS), el sensor VIIRS y el sensor DMSP-OLS.

Entre sus funcionalidades clave se incluye la descarga programada de imágenes capturadas entre las 6:45 P.M y las 5:30 A.M, la validación de metadatos, su clasificación según misión, sensor y año, así como el almacenamiento estructurado en un entorno NAS.

El sistema fue diseñado como una solución de operación local y monousuario, en concordancia con los objetivos funcionales y el alcance del proyecto. Esta decisión responde a requerimientos específicos del cliente que utilizará la aplicación, quien requiere control individual sobre los procesos de descarga y análisis dentro de equipos de investigación especializados. Aunque las aplicaciones multiusuario distribuidas representan tendencias actuales, este proyecto priorizó la eficiencia operativa y los requerimientos del contexto de investigación científica sobre arquitecturas más complejas.

El diseño modular del sistema permite su escalabilidad y adaptación a otras regiones o fuentes de datos satelitales con características similares. Aunque el enfoque actual está centrado en la gestión automatizada de datos, la arquitectura propuesta facilita su futura integración con herramientas de análisis espacial y visualización, lo cual podría fortalecer aún más su aplicación científica y técnica. Este desarrollo ofrece una base sólida para estudios ambientales relacionados con la contaminación lumínica y otras problemáticas geoespaciales, y contribuye a cerrar la brecha entre la disponibilidad de datos satelitales y su aprovechamiento efectivo por parte de la comunidad científica y técnica.

Capítulo 2: Selección de API y Enfoque para la Automatización de Descargas

2.1 Introducción

La contaminación lumínica representa una problemática ambiental creciente que afecta la biodiversidad, la salud humana y el consumo energético en las ciudades contemporáneas [3]. Su monitoreo sistemático resulta esencial para el diseño de políticas públicas sostenibles y acciones de mitigación. No obstante, uno de los principales retos para su estudio es la cobertura espacial [8].

Tradicionalmente, este fenómeno se ha abordado mediante sensores terrestres, los cuales, aunque precisos a nivel local, presentan limitaciones importantes en cobertura geográfica. En contraste, las imágenes satelitales nocturnas permiten una visión global, con registros históricos útiles para el análisis lumínico espacial. En particular, los sensores *DMSP-OLS* y *VIIRS* han sido ampliamente utilizados para estudios de emisiones nocturnas. Estos datos, accesibles a través de plataformas como (GEE), se han convertido en herramientas esenciales para el análisis geoespacial automatizado. A su vez, las imágenes capturadas desde la (ISS), aportan datos en color real que pueden ser descompuestos por canal para estimaciones multi espectrales, cuando están adecuadamente georreferenciadas.

Este capítulo presenta los fundamentos técnicos para el acceso y automatización de datos satelitales mediante (APIs), analiza las principales fuentes disponibles y presenta el análisis de distintas fuentes de datos, con base en criterios técnicos alineados con los objetivos del estudio.

2.2 Fundamentos técnicos: APIs y plataformas para acceso satelital.

Una *Interfaz de Programación de Aplicaciones* (API) es una herramienta fundamental para la interacción entre sistemas de software. Su estructura permite realizar solicitudes automatizadas para acceder a recursos remotos, ejecutar funciones o consultar bases de datos, todo ello de manera programática y estandarizada [9]. En el campo del monitoreo ambiental por satélite, las APIs facilitan el acceso estructurado a catálogos de imágenes y metadatos, además permiten aplicar filtros espaciales y temporales sin necesidad de procesos manuales extensos.

Uno de los formatos más utilizados son las APIs RESTful, que operan sobre el protocolo HTTP y entregan respuestas en estructuras como JSON o XML. La Figura 2.1 ilustra el esquema básico de comunicación entre un cliente y un servidor mediante este tipo de interfaz. Estas interfaces son compatibles con lenguajes como Python, facilitando su integración en flujos de procesamiento geoespacial [10].

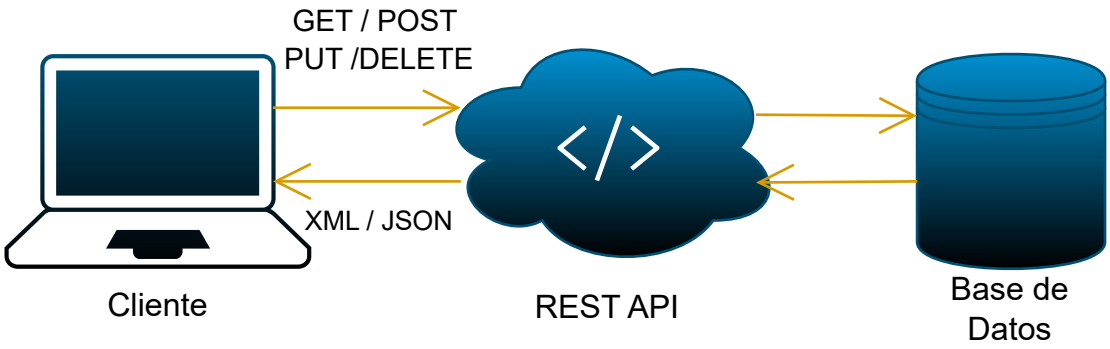

Figura 2.1: Esquema básico de comunicación mediante una API RESTful. Elaborado por: autor.

Un caso específico es el del catálogo de imágenes de la Estación Espacial Internacional (ISS), mantenido por la NASA. Este recurso cuenta con una API que permite realizar búsquedas filtradas a partir de metadatos, aunque su acceso requiere una clave previamente solicitada. Para acceder a datos que no están disponibles directamente a través de esta interfaz, se pueden

utilizar técnicas de extracción automatizada desde el sitio web (*web scraping*). Este enfoque se esquematiza en la Figura 2.2, y aunque permite recuperar información útil, es más susceptible a fallos si la estructura de la página cambia [11].

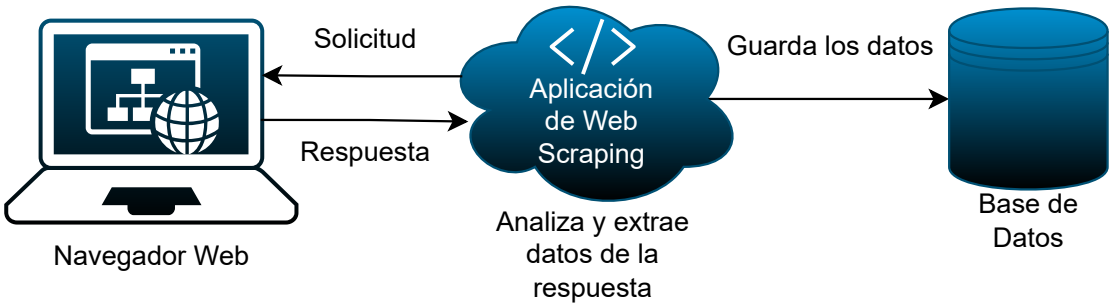

Figura 2.2: Proceso de extracción de datos mediante técnicas de web scraping. Elaboración propia basada en Richard (2022)[1].

Una tercera vía es el uso de plataformas en la nube como *Google Earth Engine* (GEE), que permite acceder, procesar y exportar imágenes satelitales sin necesidad de descarga local. Esta herramienta ofrece colecciones históricas de sensores como VIIRS y DMSP-OLS, con scripts personalizables en JavaScript o Python. GEE se ha consolidado como una plataforma de referencia por su escalabilidad, facilidad de uso y compatibilidad con múltiples fuentes [8]. Su funcionamiento general se muestra en la Figura 2.3, donde se esquematizan los componentes clave del procesamiento y análisis satelital en la nube.

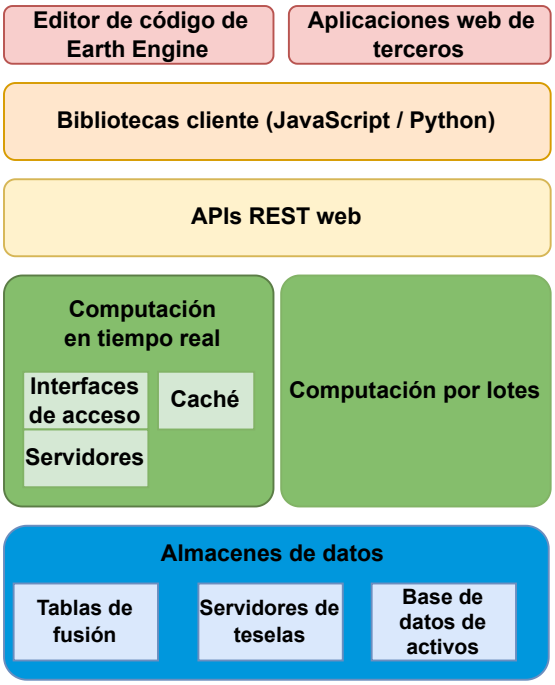

Figura 2.3: Arquitectura general de Google Earth Engine para análisis satelital en la nube. Elaboración propia basada en Gomes et al. (2020)[2].

Además, el sistema implementa una arquitectura de almacenamiento híbrida donde las imágenes se conservan de forma íntegra en el servidor NAS, mientras que la base de datos relacional mantiene tablas especializadas que almacenan los metadatos y atributos técnicos de cada imagen. Esta separación permite realizar consultas eficientes mediante filtros específicos para localizar rápidamente las imágenes requeridas en el almacenamiento físico sin necesidad de examinar los archivos individuales.

El sistema de metadatos permite realizar búsquedas avanzadas utilizando múltiples criterios simultáneos, tales como año de captura, hora local, elevación solar, inclinación de la cámara, tipo de sensor, coordenadas geográficas, cobertura nubosa y resolución de la imagen. Esta funcionalidad facilita la selección precisa de imágenes que cumplan con parámetros específicos para análisis científicos o estudios comparativos.

2.3 Análisis comparativo de fuentes y métodos de acceso

Para seleccionar las fuentes más adecuadas destinadas al estudio de la contaminación lumínica en la cuenca del Canal de Panamá, se realizó un análisis comparativo considerando resolución espacial, cobertura geográfica, disponibilidad histórica, accesibilidad técnica y grado de automatización posible.

2.3.1 NASA Gateway to Astronaut Photography of Earth

La plataforma de la NASA que alberga imágenes capturadas desde la Estación Espacial Internacional (ISS) contiene un amplio archivo de imágenes nocturnas, muchas de las cuales tienen una resolución espacial inferior a 3 metros por píxel. Estas imágenes, en formato RGB, permiten ser descompuestas por canal para estimar niveles relativos de radiancia, siempre que estén georreferenciadas. La Figura 2.4 ilustra una imagen utilizada para estimar radiancia relativa por canal RGB en el área metropolitana de Ciudad de Panamá. Su análisis permitió identificar zonas de mayor intensidad lumínica y establecer coordenadas específicas por píxel.

Figura 2.4: Ciudad de Panamá, imagen nocturna capturada desde la ISS. Fuente: NASA Earth Observatory.

Cuando las imágenes están georreferenciadas, es posible asignar coordenadas específicas (latitud y longitud) a cada píxel del canal correspondiente, lo cual permite realizar análisis detallados de luminancia en regiones clave como la cuenca del Canal de Panamá.

El acceso a estas imágenes se puede realizar mediante navegación manual o a través de su API. Esta fuente fue seleccionada como base principal del estudio por su resolución, cobertura histórica y potencial para estimar radiancia por canal en puntos geográficos específicos. El código de ejemplo correspondiente se encuentra detallado en el Apéndice 1.

Este conjunto de imágenes constituye la base para la estimación de radiancia relativa por canal y la evaluación de continuidad espacial con los sensores históricos.

2.3.2 DMSP-OLS y VIIRS a través de Google Earth Engine

El sensor DMSP-OLS proporciona cobertura global desde 1992 hasta 2013 y ha sido ampliamente utilizado para establecer líneas base históricas de luminancia

artificial. Sin embargo, presenta limitaciones importantes, como baja resolución espacial (aproximadamente 2.7 km por píxel) y saturación en zonas altamente iluminadas.

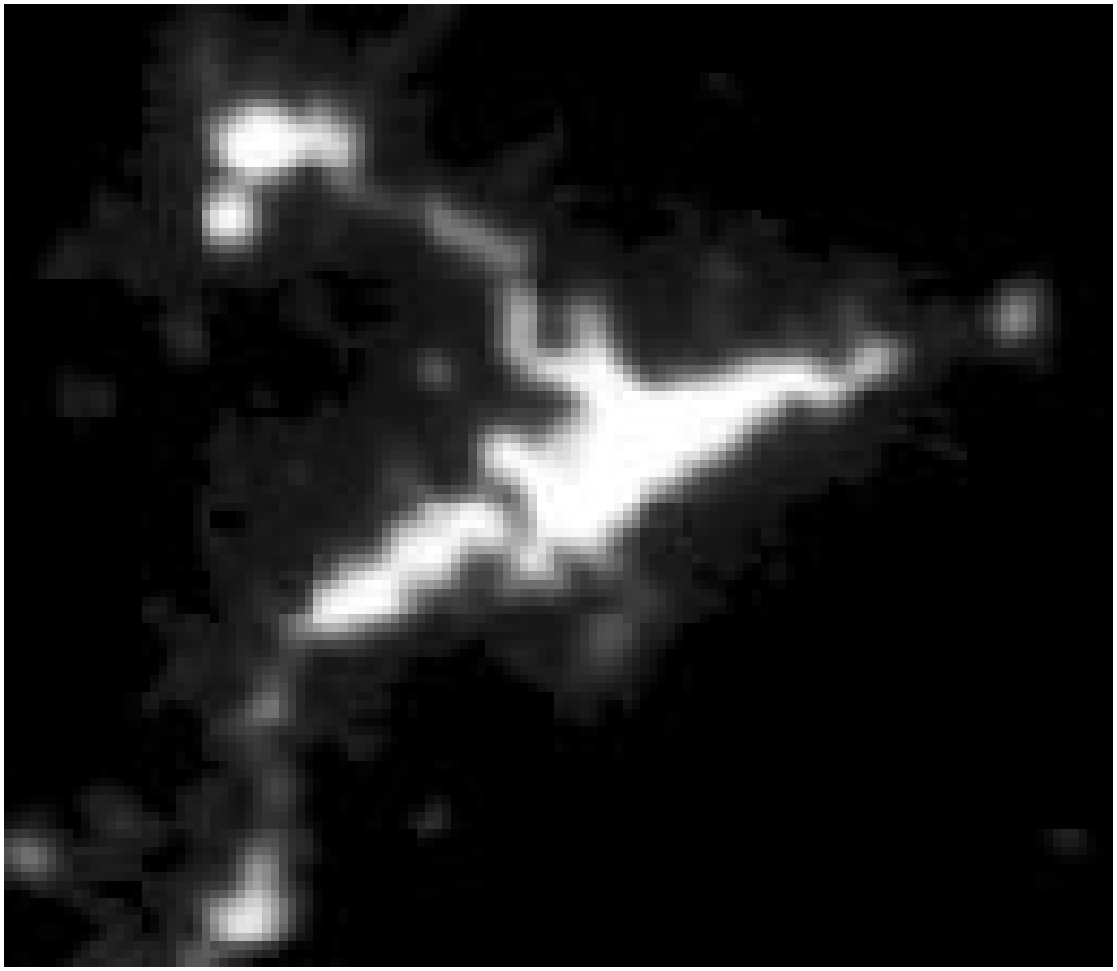

Figura 2.5: Datos DMSP-OLS originales en escala de grises para la cuenca del Canal de Panamá (2013). Fuente: DMSP Operational Linescan System.

Como continuidad operativa de DMSP-OLS, el sensor VIIRS ofrece datos diarios desde 2011 con mayor resolución (500 m por píxel), mejor calibración radiométrica y menor saturación.

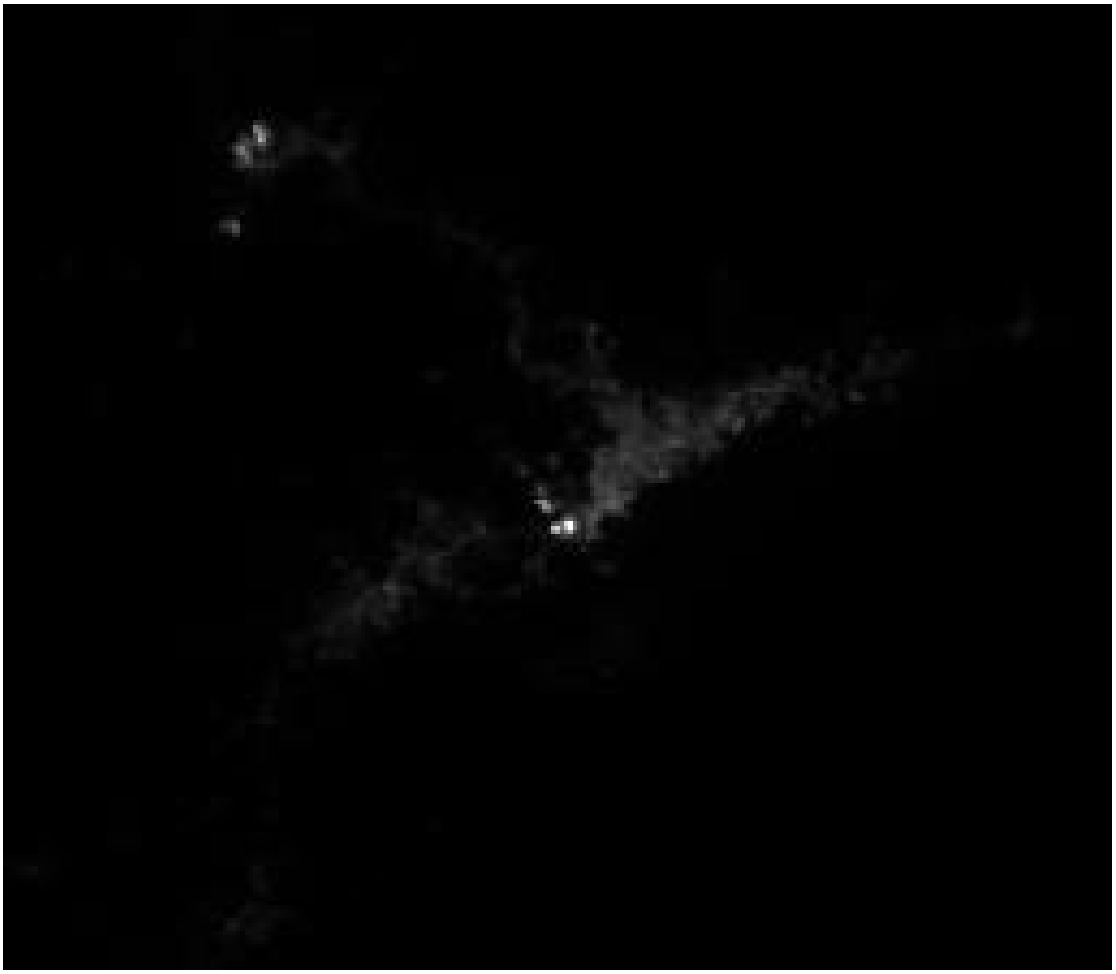

Figura 2.6: Datos VIIRS originales en escala de grises para la cuenca del Canal de Panamá (2023). Fuente: NOAA/VIIRS.

Limitaciones instrumentales

Es fundamental comprender que tanto DMSP-OLS como VIIRS presentan limitaciones instrumentales significativas. Estos sensores capturan únicamente intensidad lumínica en escala de grises y no registran información cromática real, a diferencia de las cámaras RGB de la Estación Espacial Internacional. Específicamente, ambos sensores presentan limitaciones en el espectro azul, donde muchas fuentes de iluminación LED moderna emiten significativamente. Ambos conjuntos de datos fueron accedidos y procesados mediante Google Earth Engine, lo que permitió automatizar la selección de fechas, aplicación de filtros espaciales y exportación de imágenes. Aunque su resolución es menor en comparación con las imágenes de la ISS, los datos históricos de DMSP-OLS y VIIRS proporcionan continuidad temporal esencial para estudios evolutivos.

Estos datos fueron utilizados como referencia para orientar la subclasificación de píxeles en las imágenes de la ISS, especialmente en el área del Canal de Panamá, donde se aplicaron rangos relativos de luminancia (baja, media y alta) para mejorar la interpretación de los valores obtenidos por canal (RGB) en imágenes georreferenciadas. El código de ejemplo utilizado se encuentra detallado en el Apéndice 2.

Tabla 2.1: Comparación de fuentes de datos satelitales utilizadas en este estudio.

Fuente	**Resolución Espacial**	**Cobertura Geográfica**	**Disponibilidad Histórica**	**Notas Técnicas**
NASA ISS	Algunas hasta 3 m/píxel	Limitada (según órbita)	Desde 1960	Imágenes reales, permiten separación por canales si están georreferenciadas.
DMSP-OLS	2.7 km/píxel	Global	1992–2013	Saturación en zonas brillantes, útil para líneas base históricas
VIIRS DNB	500 m/píxel	Global	Desde 2011	Alta frecuencia, buena calibración

2.4 Justificación de la selección de fuentes

La combinación de datos históricos (DMSP-OLS), contemporáneos (VIIRS) y multiespectrales en color real (NASA ISS) permite un enfoque multiescalar y complementario. GEE fue clave para automatizar el acceso a VIIRS y DMSP-OLS, garantizando eficiencia y reproducibilidad. Las imágenes de la ISS, por su parte, permiten descomposición por canales para análisis cuantitativo de radiancia en zonas georreferenciadas.

2.5 Fuentes adicionales y consideraciones

Se exploraron otras fuentes como Black Marble, Luojia 1-01, SkySat y Sentinel-2, pero fueron descartadas por razones de acceso, costo, formato complejo o falta de sensibilidad nocturna. Una tabla comparativa con sus ventajas y limitaciones se presenta a continuación.

Tabla 2.2: Resumen de fuentes satelitales adicionales para estudios de contaminación lumínica.

Fuente	**Resolución**	**Acceso**	**Ventajas**	**Limitaciones**
Black Marble (NASA)	500 m	Earthdata (registro)	Datos corregidos, listos para análisis	Formato complejo (`HDF5`)
Luojia 1-01	130 m	Solicitud directa	Alta resolución nocturna	Acceso restringido
Jilin-1 / SkySat	0.5–1 m	Comercial	Alta resolución urbana	Costoso, sin API pública
Sentinel-2	10–20 m	API Copernicus	Alta resolución diurna, gratuito	No diseñado para luminancia nocturna

En el siguiente capítulo se describe la metodología de desarrollo, arquitectura técnica del sistema automatizado, detallando los módulos que integran el flujo de descarga, clasificación y almacenamiento de imágenes satelitales.

Capítulo 3: Metodología de Desarrollo y Diseño del Sistema

3.1 Introducción

Este capítulo documenta de forma estructurada la metodología de desarrollo adoptada para la construcción del sistema automatizado de adquisición, procesamiento y organización de imágenes satelitales nocturnas, con aplicación específica en la cuenca hidrográfica del Canal de Panamá. El diseño se fundamentó en principios clave de la ingeniería de software moderna: modularidad, escalabilidad, automatización y mantenibilidad.

Se detallan las decisiones metodológicas, tecnológicas y arquitectónicas que guiaron la construcción del sistema, así como la planificación basada en incrementos funcionales. Esta estrategia permitió validar tempranamente cada módulo, incorporar mejoras derivadas de pruebas reales y asegurar la cohesión entre componentes desde las etapas iniciales del desarrollo.

El capítulo incluye los requerimientos funcionales y no funcionales, el modelo de datos, el diseño arquitectónico, las herramientas utilizadas, los procesos automatizados clave y la descripción detallada de los incrementos realizados. También se presentan diagramas explicativos que ilustran la estructura y la interacción entre los distintos niveles del sistema, facilitando su comprensión técnica e implementación futura.

3.2 Modelo de Desarrollo Adoptado

Para guiar el proceso de construcción del sistema, se seleccionó el modelo incremental como metodología de desarrollo. Este enfoque permite construir el software en fases funcionales consecutivas, de manera que cada incremento representa una versión parcialmente operativa del sistema, susceptible de ser evaluada, corregida y ampliada. Esta estrategia fue especialmente adecuada para un entorno con tecnologías heterogéneas, incertidumbre técnica inicial y la necesidad de validar funcionalidades interdependientes desde etapas tempranas

del proyecto.

Según González y Pérez [12], el modelo incremental proporciona ventajas significativas frente a modelos más rígidos, como la detección temprana de errores, entrega anticipada de funcionalidades clave y posibilidad de retroalimentación constante.

3.2.1 Comparación con otros modelos

Previo a la adopción del modelo incremental, se consideraron otros enfoques tradicionales y ágiles. La Tabla 3.1 resume las principales características de cada uno y justifica la elección final:

Tabla 3.1: Comparativa entre modelos de desarrollo de software

Modelo	**Ventajas**	**Limitaciones**
Cascada	Secuencia clara de etapas; útil en entornos controlados.	Rígido ante cambios; requiere requisitos completamente definidos desde el inicio.
Espiral	Integra gestión de riesgos y mejora continua.	Complejo de aplicar en proyectos individuales; mayor carga de documentación.
Ágil (Scrum, XP)	Alta flexibilidad y adaptabilidad; entregas frecuentes.	Requiere colaboración constante de un equipo; sobrecarga innecesaria en proyectos individuales.
Incremental	Permite construir el sistema por módulos funcionales; favorece la validación temprana.	Requiere planificación coherente para evitar inconsistencias entre iteraciones.

3.2.2 Aplicación práctica del modelo incremental

El desarrollo del sistema siguió una estrategia incremental adaptativa, permitiendo construir funcionalidades clave de forma progresiva. A diferencia de

un enfoque lineal, cada incremento no solo agregaba nuevos módulos, sino que también refinaba y consolidaba los componentes anteriores, particularmente las interfaces de usuario, procesamiento de datos y lógica.
A continuación, se describen los cinco incrementos principales que definieron la construcción del sistema:

- **Incremento 1: Web Scrapper NASA ISS + base de datos inicial.** Se desarrolló un scraper en Python para obtener imágenes nocturnas desde el portal NASA Gateway mediante técnicas de web scraping. En paralelo, se diseñó una base de datos relacional en SQLite para almacenar metadatos estructurados.

- **Incremento 2: Explorador de imágenes NAS + visualización por consola.** Se implementó un módulo que permite navegar el entorno NAS organizado por año, sensor y misión, y consultar metadatos desde la terminal. Esto permitió verificar la organización de archivos y evaluar la consistencia del almacenamiento.

- **Incremento 3: Consulta avanzada vía API NASA + mejoras en organización.** Se integró el uso de la API de NASA Gateway para realizar consultas programáticas y estructuradas. También se implementó el renombrado automático de archivos y un sistema de control de duplicados.

- **Incremento 4: Descarga de imágenes desde Google Earth Engine.** Se desarrollaron scripts para obtener imágenes calibradas VIIRS y DMSP-OLS desde GEE.

La figura 3.1 indica claramente el proceso y las fases definidas para el desarrollo del sistema.

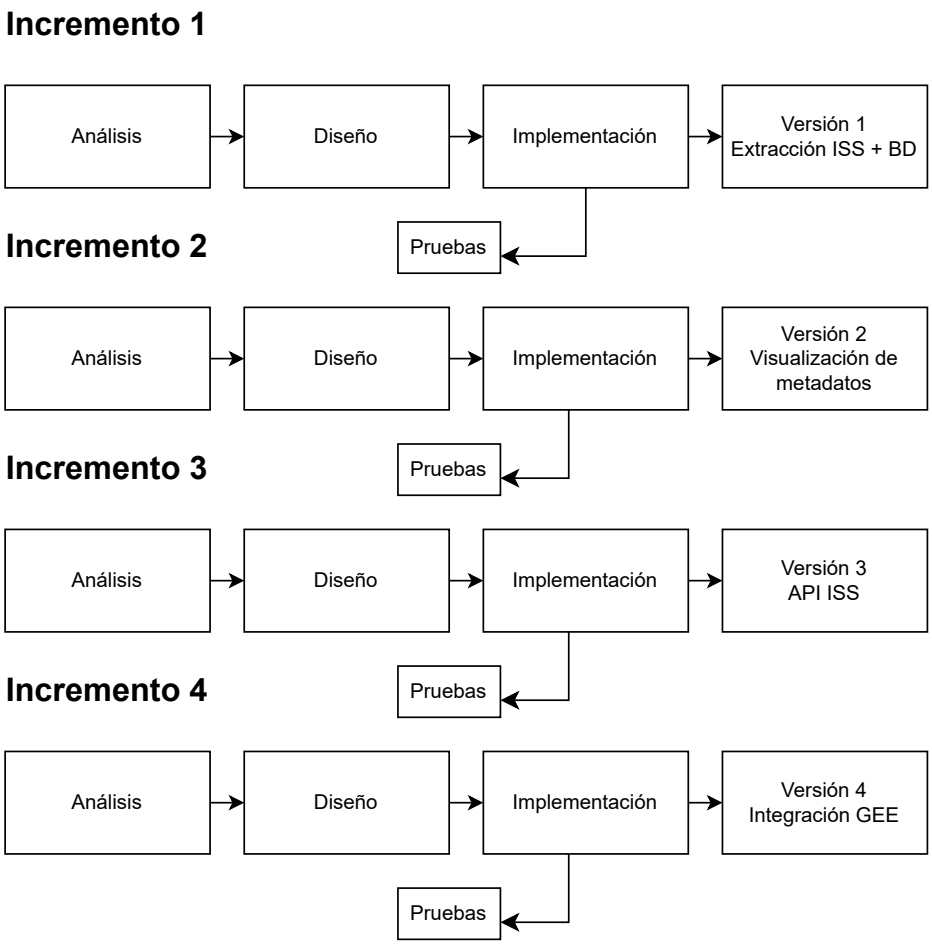

Figura 3.1: Aplicación del modelo incremental al desarrollo del sistema.

3.2.3 Ventajas observadas durante el desarrollo

Entre los beneficios más destacados del enfoque incremental en este proyecto se encuentran:

- **Reducción de errores**: los módulos fueron probados individualmente antes de ser integrados.
- **Flexibilidad**: fue posible ajustar funcionalidades tras cada entrega parcial.
- **Reutilización**: componentes del scraper y del sistema de almacenamiento fueron reutilizados en múltiples etapas.
- **Trazabilidad**: cada avance fue documentado y controlado mediante herramientas de versionado.

Estas ventajas validan la elección del enfoque incremental como metodología de desarrollo, al permitir iteraciones controladas que fortalecieron la estabilidad y calidad del sistema antes de su integración final.

3.3 Arquitectura General del Sistema

La arquitectura del sistema se diseñó bajo un enfoque modular, en el que cada componente asume una responsabilidad específica y se comunica con los demás mediante interfaces claramente definidas. Esta estructuración favorece el mantenimiento del sistema, incrementa su escalabilidad y permite la evolución independiente de los módulos sin comprometer la integridad general.

El diseño arquitectónico se fundamenta en los criterios técnicos establecidos en el Capítulo 2, donde se analizaron y seleccionaron las fuentes de datos satelitales y las herramientas de acceso automatizado más adecuadas. A partir de estas decisiones, se integraron componentes como APIs especializadas, scripts de procesamiento y esquemas de almacenamiento distribuido, todos alineados con los objetivos funcionales y de automatización del sistema.

La Figura 3.2 ilustra la organización general del sistema, destacando los bloques funcionales principales y sus interacciones.

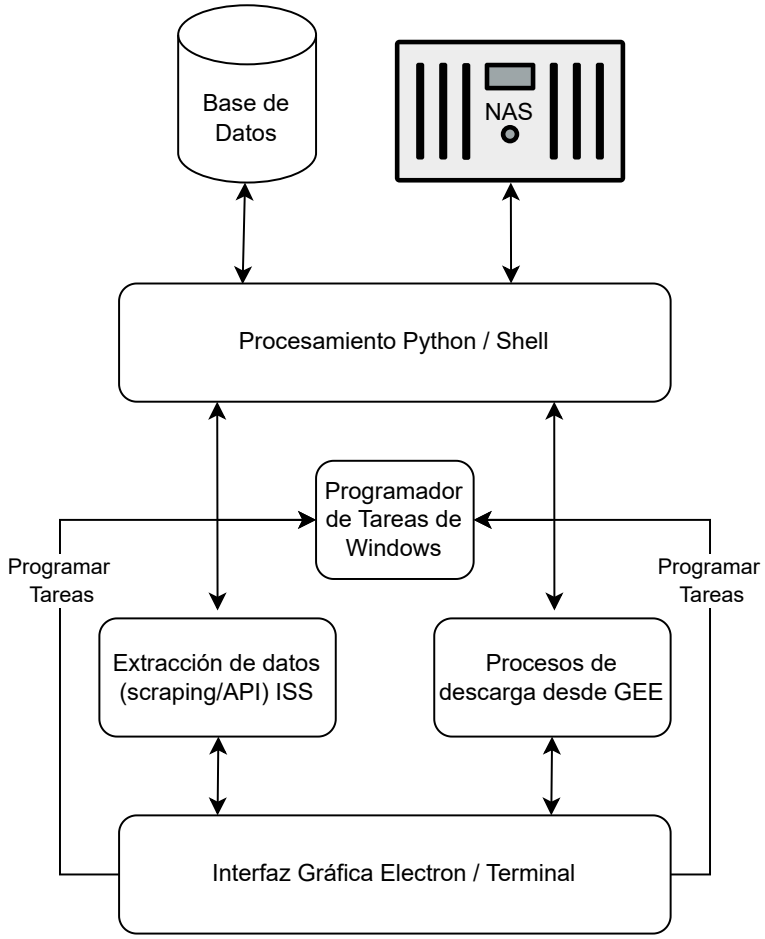

Figura 3.2: Arquitectura modular del sistema.

- **Interfaz gráfica / terminal:** Facilita la ejecución y programación de tareas, visualización de resultados y navegación de archivos. Además, permite agendar tareas directamente mediante la integración con el Programador de Tareas de Windows, facilitando la automatización.
- **Extracción de datos:** Compuesta por scripts desarrollados en Python para realizar web scraping sobre el portal NASA Gateway y para interactuar con la API de Google Earth Engine. Permite recuperar imágenes satelitales y metadatos asociados de manera automatizada.
- **Programador de tareas:** Encargado de ejecutar periódicamente scripts clave mediante el Programador de Tareas de Windows. Esta funcionalidad garantiza una operación desatendida y continua.

- **Procesamiento (Python/Shell):** Núcleo funcional encargado del procesamiento de datos, organización de archivos, renombrado, verificación de duplicados y conexión con la base de datos.
- **Base de datos:** Gestionada mediante SQLite. Contiene los metadatos asociados a las imágenes descargadas, permitiendo su consulta eficiente por atributos como sensor, fecha, misión y tipo de cámara.
- **Almacenamiento en red (NAS):** Sistema de archivos remoto donde se almacenan las imágenes descargadas, organizadas jerárquicamente. Asegura persistencia, disponibilidad y escalabilidad.

3.4 Comparativa y Justificación Tecnológica por Módulo

La selección de tecnologías se realizó conforme a los requerimientos funcionales del sistema, la compatibilidad con entornos mixtos y la facilidad de integración. En cada módulo arquitectónico se consideraron alternativas, priorizando aquellas con mejor soporte, documentación y estabilidad comprobada.

Extracción de datos. Se empleó Python 3.12 junto con `requests`, `BeautifulSoup` y la API oficial de Google Earth Engine. Esta combinación ofrece un entorno simple, flexible y con amplias bibliotecas orientadas al procesamiento web y científico. Tecnologías como Java o R fueron descartadas por su mayor complejidad y menor idoneidad para flujos automatizados.

Automatización de tareas. Se integraron scripts Bash con el Programador de Tareas de Windows, en conjunto con WSL2. Esta solución permite cronogramar procesos recurrentes sin requerir herramientas externas complejas como Apache Airflow o cron puro en servidores Linux. Esto es útil debido a que si wsl2 se encuentra apagado se puede ejecutar los scripts de descarga desde Windows.

Procesamiento y lógica. El backend emplea Python como lenguaje principal, centralizando las operaciones de verificación, renombrado y manejo de archivos. Esta elección responde a la necesidad de integración fluida con SQLite y Electron, y favorece una curva de desarrollo ágil.

Base de datos. Se seleccionó SQLite como motor relacional embebido, por su ligereza, portabilidad y capacidad de integrarse directamente con SQLAlchemy. Otras opciones como PostgreSQL o MySQL fueron descartadas por su complejidad innecesaria en un entorno monousuario local.

Interfaz gráfica. La GUI fue implementada con Electron y Node.js, permitiendo el desarrollo de una aplicación de escritorio basada en tecnologías web (HTML, CSS, JS). Este enfoque asegura portabilidad y un diseño moderno sin requerir interfaces nativas específicas.

Almacenamiento. El almacenamiento se resolvió mediante un entorno NAS, montado a través de `sshfs` para conexión remota y `cifs` para conexión local en WSL2. Esta solución ofrece control total sobre el entorno de archivos, sin depender de servicios externos como Google Drive o Amazon S3.

Descarga de archivos. Se utilizó `aria2c` como herramienta de descarga concurrente, ideal para archivos grandes provenientes de múltiples fuentes. Presenta ventajas frente a `wget` y `curl` por su capacidad de reanudar transferencias y optimizar ancho de banda.

La Tabla 3.2 sintetiza las tecnologías seleccionadas por módulo:

Tabla 3.2: Resumen de tecnologías seleccionadas por módulo

Módulo	**Tecnología seleccionada**
Extracción de datos	Python 3.12 + requests + BeautifulSoup + GEE API
Automatización	Shell script + Programador de Tareas (WSL2/Windows)
Procesamiento	Python (scripts de lógica y organización)
Base de datos	SQLite (relacional, embebida) + SQLAlchemy
Interfaz gráfica	Electron + Node.js + HTML/CSS/JS
Almacenamiento	NAS (CIFS + SSHFS)
Descarga de archivos	`aria2c` (concurrente y resumible)

3.5 Principios de Diseño Aplicados

Durante la implementación del sistema se aplicaron principios de diseño que favorecen la mantenibilidad, la escalabilidad y la claridad estructural del código. La arquitectura se basa en una adaptación del patrón Modelo–Vista–Controlador (MVC), distribuido entre tecnologías web (Electron) y backend en Python.

Patrón MVC adaptado al entorno híbrido

El sistema sigue una estructura lógica que replica el patrón MVC, distribuido de la siguiente manera:

- **Vista (View):** desarrollada en Electron mediante archivos HTML/CSS y scripts JavaScript en `renderer.js`. Proporciona al usuario una interfaz visual e interactiva.
- **Controlador (Controller):** implementado en `main.js`, que actúa como intermediario entre la vista y los procesos Python del backend, mediante llamadas a procesos hijo.
- **Modelo (Model):** compuesto por las clases `ImageProcessor` y `MetadataCRUD`, escritas en Python. Estas encapsulan la lógica de negocio y acceso a datos, incluyendo persistencia vía SQLAlchemy sobre SQLite.

Automatización y coordinación de procesos

El diseño también incorpora una capa de automatización que opera de forma independiente a la interfaz gráfica. Scripts Bash y Python son ejecutados de forma programada por el sistema operativo, permitiendo tareas como adquisición masiva de imágenes, limpieza periódica o generación de reportes sin intervención del usuario.

Uso de patrones adicionales

Se utilizó el patrón **DAO (Data Access Object)** en el módulo `MetadataCRUD`, con el fin de separar la lógica de acceso a datos del resto del sistema. Esto mejora la cohesión, facilita pruebas unitarias y permite modificar el sistema de almacenamiento sin reescribir la lógica de negocio.
Adicionalmente, se respetaron principios como bajo acoplamiento y alta cohesión, y se diseñaron los scripts en bloques reutilizables con entrada parametrizada.

3.6 Estrategia progresiva de adquisición de datos: de extracción web a interfaces de programación

Una decisión metodológica fundamental durante el desarrollo del sistema fue la adopción de un enfoque progresivo para la adquisición de datos, que comenzó con técnicas de extracción automática de información web (*web scraping*) y evolucionó hacia la integración de interfaces de programación de aplicaciones (APIs). Esta estrategia se diseñó para obtener resultados tempranos en un entorno caracterizado por documentación limitada y estructuras web no estandarizadas.

3.6.1 Etapa inicial: Extracción automática de información web

En las primeras fases del proyecto, la extracción automática de información web permitió explorar y recolectar imágenes satelitales desde el portal de NASA Gateway sin requerir credenciales complejas ni configuraciones avanzadas. Esta aproximación inicial facilitó tres aspectos críticos del desarrollo:

- **Validación del flujo de adquisición**: Se comprobó que el proceso de obtención de datos funcionaba correctamente desde las primeras etapas.
- **Inspección de formatos de archivo**: Se identificaron los tipos de datos disponibles y sus características técnicas.
- **Reconocimiento de patrones organizacionales**: Se descubrió cómo estaban estructurados y organizados los conjuntos de datos en las plataformas fuente.

3.6.2 Evolución hacia interfaces de programación oficiales

Una vez establecido un conocimiento sólido sobre la estructura de los recursos disponibles, se implementaron consultas estructuradas mediante las interfaces de programación oficiales (PhotosDatabaseAPI) del sitio web de imágenes ISS. Posteriormente, se incorporaron imágenes provenientes de Google Earth Engine específicamente de los conjuntos de datos de NOAA con los sensores DMSP-OLS y VIIRS.
Estas interfaces oficiales proporcionaron ventajas sustanciales:

- **Mayor estabilidad**: Reducción significativa de errores e interrupciones en el servicio.
- **Acceso optimizado**: Conexión directa a conjuntos de datos calibrados y validados científicamente.
- **Automatización robusta**: Menor riesgo de fallos ante cambios en las interfaces web de las plataformas.

- **Diversidad de datos**: La incorporación de NOAA amplió las fuentes disponibles con datos de sensores DMSP-OLS y VIIRS para análisis de iluminación nocturna con cobertura temporal más amplia.

3.6.3 Justificación metodológica del enfoque progresivo

La transición de técnicas de extracción web hacia interfaces de programación no respondió únicamente a consideraciones técnicas, sino que se alineó estratégicamente con el enfoque incremental adoptado en todo el proyecto. Este proceso metodológico permitió:

- **Validación temprana**: Confirmar la viabilidad de las funcionalidades básicas desde el inicio del desarrollo.
- **Refinamiento progresivo**: Evolucionar gradualmente hacia soluciones más robustas y escalables.
- **Gestión de riesgos**: Minimizar la incertidumbre mediante la obtención de resultados parciales verificables.

Esta evolución metodológica refleja una aplicación coherente de los principios de exploración temprana, mejora continua y consolidación progresiva que guiaron todo el proceso de desarrollo del sistema.

La arquitectura resultante permitió construir un sistema robusto, claro y fácilmente extensible, adaptable a futuros requerimientos de visualización, procesamiento o distribución de imágenes satelitales y datos ambientales.

En el siguiente capítulo se presentan los resultados obtenidos para cada uno de los incrementos funcionales definidos. Cada sección documenta la implementación práctica de las soluciones desarrolladas por incremento, como extensión directa del modelo incremental adoptado en este capítulo. De esta manera, se valida la estrategia metodológica empleada y se evidencia su efectividad para lograr un sistema robusto, escalable y automatizado.

Capítulo 4: Resultados

4.1 Introducción

Este capítulo presenta los resultados obtenidos durante la implementación progresiva del sistema automatizado, siguiendo el modelo incremental descrito en el capítulo 3. Cada incremento funcional corresponde a una unidad de desarrollo autónoma que incorpora nuevas capacidades al sistema, evaluadas mediante pruebas prácticas y retroalimentación directa.

Los resultados se organizan por incremento, detallando para cada uno las fases de análisis de requisitos, diseño, implementación, verificación y revisión. Esta estructura permite evidenciar cómo se logró construir una solución robusta, flexible y alineada con los objetivos técnicos definidos en capítulos anteriores.

A lo largo de este capítulo se incluyen diagramas, registros de ejecución, capturas de interfaz y descripciones de comportamiento que validan la efectividad del sistema en entornos reales. Además, se documentan las observaciones obtenidas tras las pruebas de cada módulo y las mejoras planificadas con base en la experiencia del desarrollo.

4.2 Incremento 1: Extracción Automatizada de Imágenes desde NASA Gateway y Base de Datos Inicial

4.2.1 Descripción General

Este primer incremento constituye la fase exploratoria y base estructural del sistema, enfocado en establecer un mecanismo eficiente para la adquisición automatizada de imágenes satelitales nocturnas. El repositorio seleccionado, *Gateway to Astronaut Photography of Earth* de la NASA, ofrece una valiosa fuente de datos para la investigación propuesta. Durante esta etapa inicial, se desarrolló un prototipo operativo que integra tres componentes fundamentales: un sistema de extracción automatizada, un modelo relacional para la persistencia

de datos y una estructura organizada para el almacenamiento físico de imágenes.

El entorno técnico utilizado combina Windows 11 con WSL2 (Ubuntu 24.04), configuración que permite aprovechar la robustez de los servicios Linux para procesamiento y scripts automatizados, mientras se mantiene la compatibilidad con herramientas de visualización en Windows. Este incremento sienta las bases tecnológicas para las funcionalidades avanzadas que se incorporarán en fases posteriores.

4.2.2 Fase 1: Análisis de Requisitos

La fase inicial de análisis permitió delimitar el alcance funcional del incremento y establecer las especificaciones técnicas necesarias para garantizar un desarrollo coherente. Se identificaron los requisitos prioritarios para este incremento y se definieron criterios específicos para su validación.

Requisitos Funcionales Los requerimientos funcionales priorizados para este incremento se centraron en establecer un flujo de trabajo automatizado completo para la extracción y almacenamiento de datos. La Tabla 4.1 detalla estos requisitos fundamentales.

Tabla 4.1: Requerimientos funcionales abordados en el incremento 1

ID	Descripción
RF1	Consultar y filtrar imágenes desde el portal NASA Gateway mediante técnicas de extracción automatizada que respeten los términos de uso del sitio.
RF2	Descargar imágenes en alta resolución junto con sus metadatos técnicos, garantizando la integridad de ambos.
RF3	Registrar los metadatos en una base de datos relacional local con mecanismos de validación para evitar duplicados e inconsistencias.
RF4	Establecer el flujo de trabajo integral que incluya descarga, filtrado, organización y capacidades preliminares de visualización.
RF5	Desarrollar un prototipo de interfaz técnica para interactuar con el sistema desde la consola, permitiendo ejecuciones parametrizadas.

Requisitos No Funcionales Complementando los requerimientos funcionales, se identificaron características técnicas esenciales para garantizar la calidad y mantenibilidad del sistema, como se muestra en la Tabla 4.2.

Tabla 4.2: Requerimientos no funcionales abordados en el incremento 1

ID	Descripción
RNF1	Compatibilidad con entornos mixtos: Windows 11 + WSL2 (Ubuntu 24.04), garantizando portabilidad en diferentes configuraciones.
RNF2	Arquitectura modular orientada a facilitar la integración progresiva de componentes y la extensibilidad del sistema.
RNF3	Persistencia fiable y eficiente de los datos extraídos, con mecanismos de integridad y recuperación.
RNF4	Interfaz técnica por consola que permita ejecutar tareas específicas de forma ágil y facilite la automatización mediante scripts.

La identificación temprana de estos requisitos permitió establecer una base sólida para las fases subsecuentes del incremento, orientando las decisiones de diseño e implementación hacia objetivos claramente definidos.

4.2.3 Fase 2: Diseño

La fase de diseño tradujo los requisitos identificados en modelos estructurales y de comportamiento que sirvieron como guía para la implementación. Se emplearon técnicas de modelado orientado a objetos, diagramas de casos de uso, modelos de datos relacionales y diagramas de actividad y secuencia, con el objetivo de garantizar la coherencia, escalabilidad y mantenibilidad del sistema.

Casos de Uso Los diagramas de casos de uso representaron las principales interacciones entre el usuario técnico y el sistema. La Figura 4.1 muestra el conjunto de funcionalidades asociadas al módulo de extracción web, destacando tareas como la selección de área, la aplicación de filtros, la obtención de metadatos y la gestión de descargas. Este modelo permitió delimitar los límites del sistema y sirvió como punto de partida para la definición detallada de cada módulo funcional.

Cabe destacar que el caso de uso “Obtener metadatos de imágenes” cumple un rol fundamental, ya que introduce la capa semántica que habilita el procesamiento posterior de los datos. Además, se utilizan relaciones «include» y «extend» para indicar dependencias entre funcionalidades, como la inclusión obligatoria del filtrado previo o la posibilidad de extender el flujo para visualizar detalles de imágenes específicas.

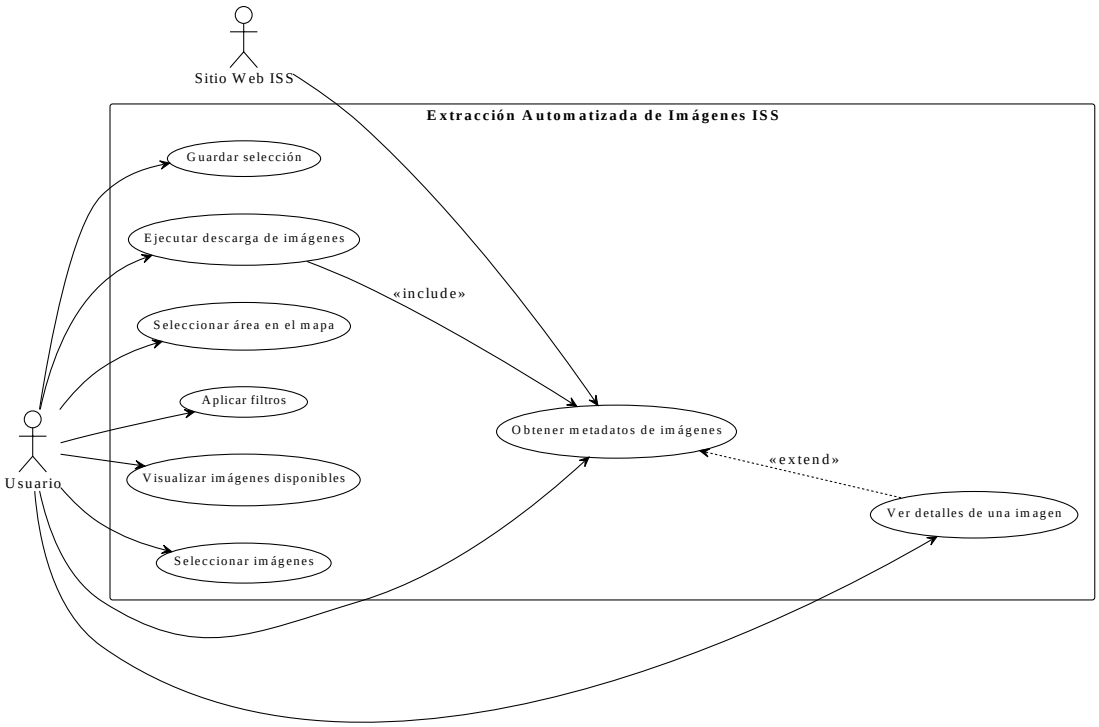

Figura 4.1: Diagrama de casos de uso del Incremento 1. Módulo de extracción web.

Flujos de Actividad Los diagramas de actividad permitieron modelar los procesos internos de forma secuencial, ilustrando cada paso involucrado en la ejecución automatizada del sistema.

La Figura 4.2 detalla el proceso de extracción de enlaces desde el portal de la NASA. El flujo inicia con la configuración de parámetros de búsqueda específicos como cobertura nubosa, sensor activo, inclinación de cámara y rango de fechas. Posteriormente, el sistema ejecuta el scraper conectándose al portal NASA Gateway y aplicando los criterios establecidos para extraer URLs de imágenes

que cumplan con los requisitos definidos.

El proceso contempla validación de estructura de cada enlace extraído antes de agregarlo a la lista temporal. Para evitar sobrecargar el servidor, se implementa un delay controlado de 3-7 segundos entre solicitudes. El camino alternativo principal se presenta en la decisión sobre páginas adicionales: cuando existen más páginas disponibles, el sistema continúa iterativamente el proceso de extracción hasta agotar todos los resultados. Finalmente, se genera un archivo consolidado con todas las URLs extraídas y se guarda en el sistema de archivos para su posterior procesamiento.

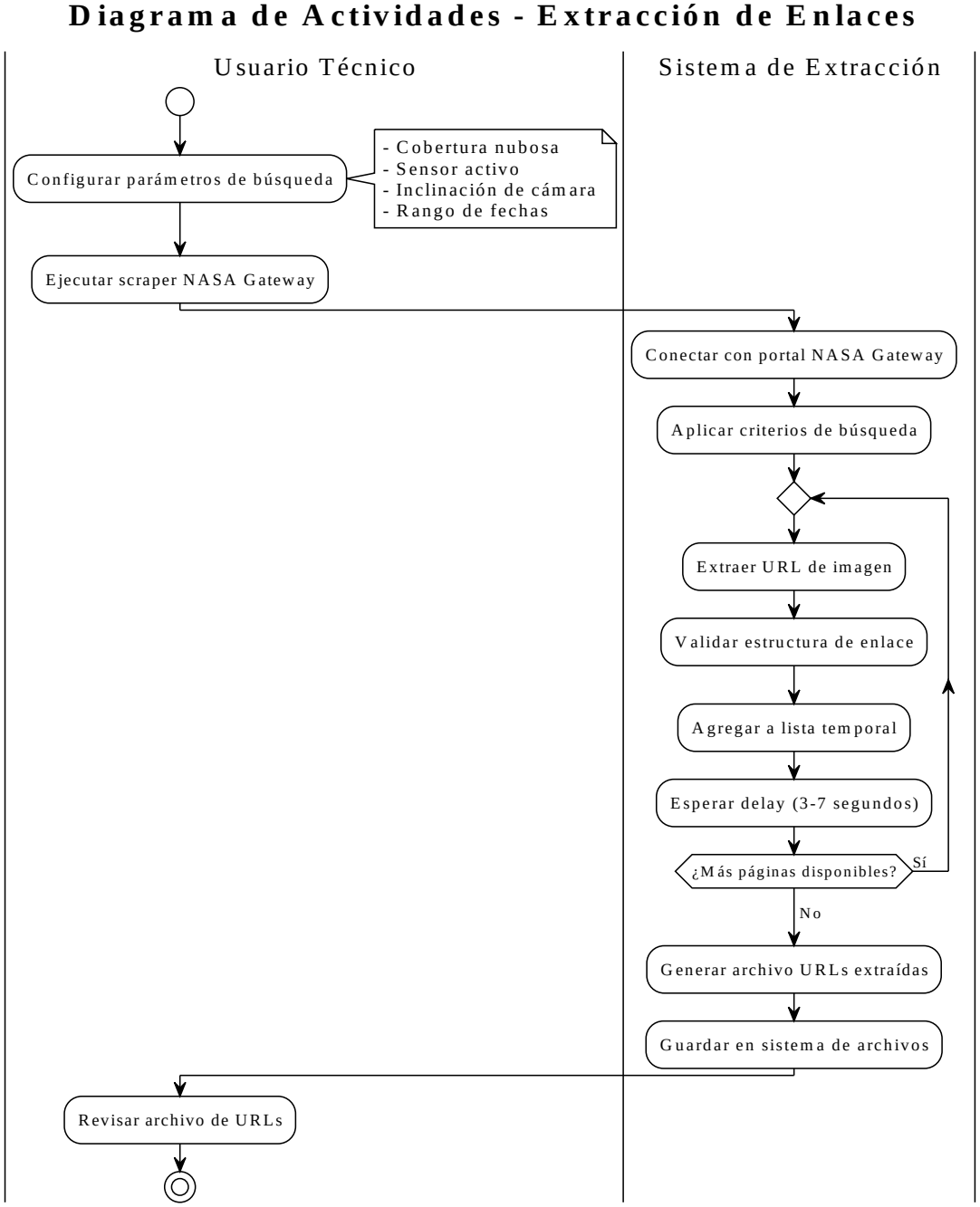

Figura 4.2: Diagrama de actividades del proceso de extracción de enlaces.

La Figura 4.3 representa el flujo de filtrado y procesamiento de metadatos extraídos. El proceso inicia cargando el archivo de URLs previamente extraídas, seguido de un análisis de resolución de imagen y validación de formato, dando preferencia a archivos TIFF. El punto de decisión central evalúa si cada URL cumple con los criterios mínimos establecidos: aquellas que los satisfacen son conservadas, mientras que las que no los cumplen son descartadas automáticamente.

Para las URLs que pasan el filtro inicial, el sistema procede a descargar la página HTML correspondiente y utiliza BeautifulSoup para parsear el contenido. La extracción de metadatos técnicos incluye información crítica como latitud/longitud, altitud, fecha y hora de captura, tipo de cámara, inclinación y cobertura nubosa. Cada conjunto de metadatos es validado para verificar su integridad antes de generar un registro JSON individual. El proceso continúa iterativamente hasta procesar todas las URLs filtradas, culminando con la consolidación de un archivo JSON único que contiene todos los metadatos estructurados.

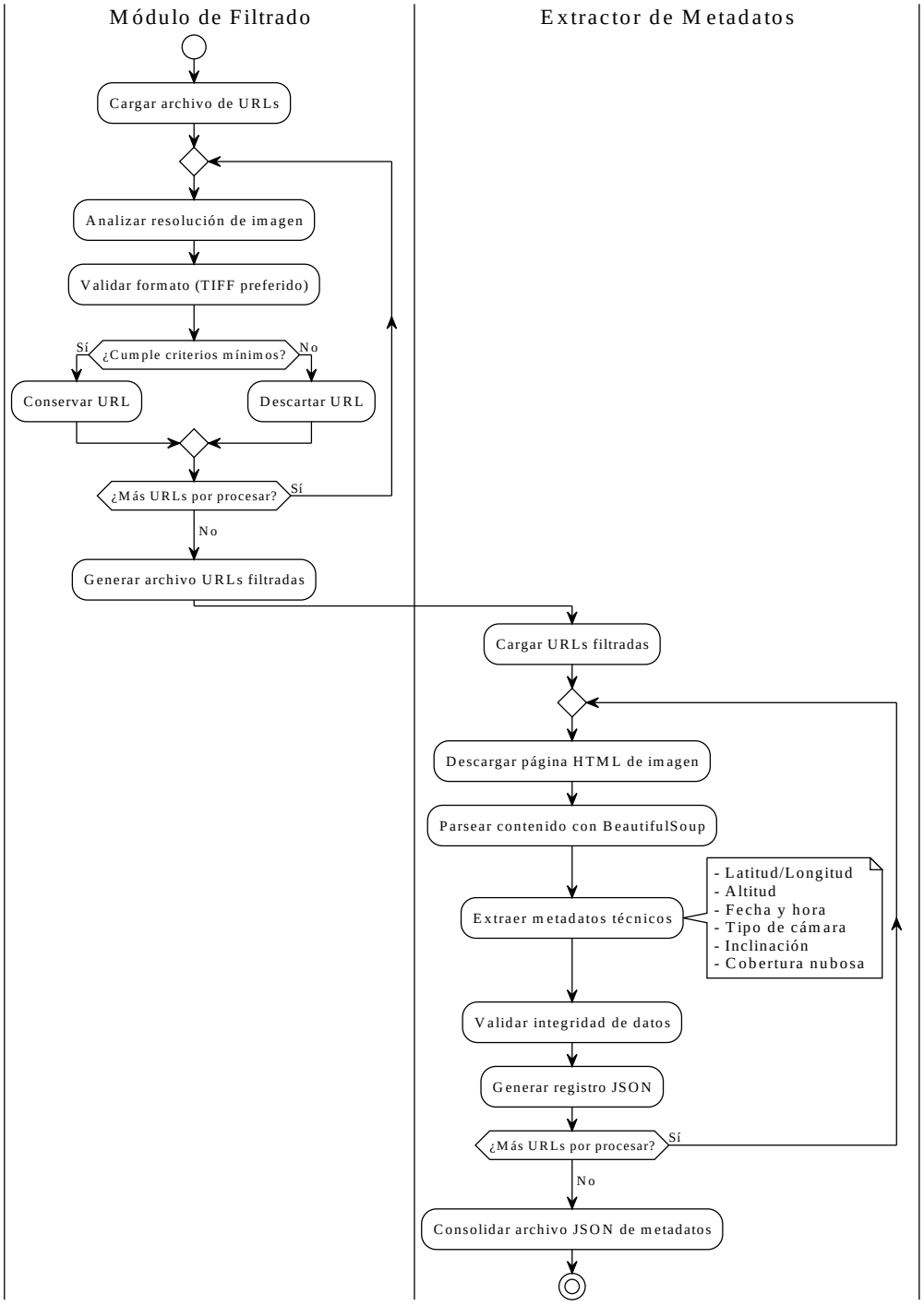

Figura 4.3: Diagrama de actividades del filtrado y procesamiento de metadatos.

La Figura 4.4 presenta el proceso de almacenamiento en base de datos y descarga de imágenes. El flujo inicia conectándose a SQLite e iniciando una transacción para leer los registros de metadatos. Cada registro es validado por unicidad mediante NASA_ID: si el registro ya existe, se omite la inserción; si es

nuevo, se inserta en la tabla Image y se actualizan los índices correspondientes. Una vez procesados todos los registros, se confirma la transacción y se cierra la conexión.

El proceso de descarga utiliza aria2c configurado con ocho conexiones simultáneas para optimizar la velocidad de transferencia. Cada imagen descargada en alta resolución es verificada mediante integridad, y en caso de fallar esta verificación, se ejecuta automáticamente un reintento de descarga. Las imágenes exitosamente descargadas son organizadas en una estructura jerárquica /año/misión/cámara/, complementada con la descarga de metadatos de cámara en formato .txt. El proceso culmina con la generación de un reporte de descarga y la validación final tanto de la estructura de carpetas como de la integridad de la base de datos.

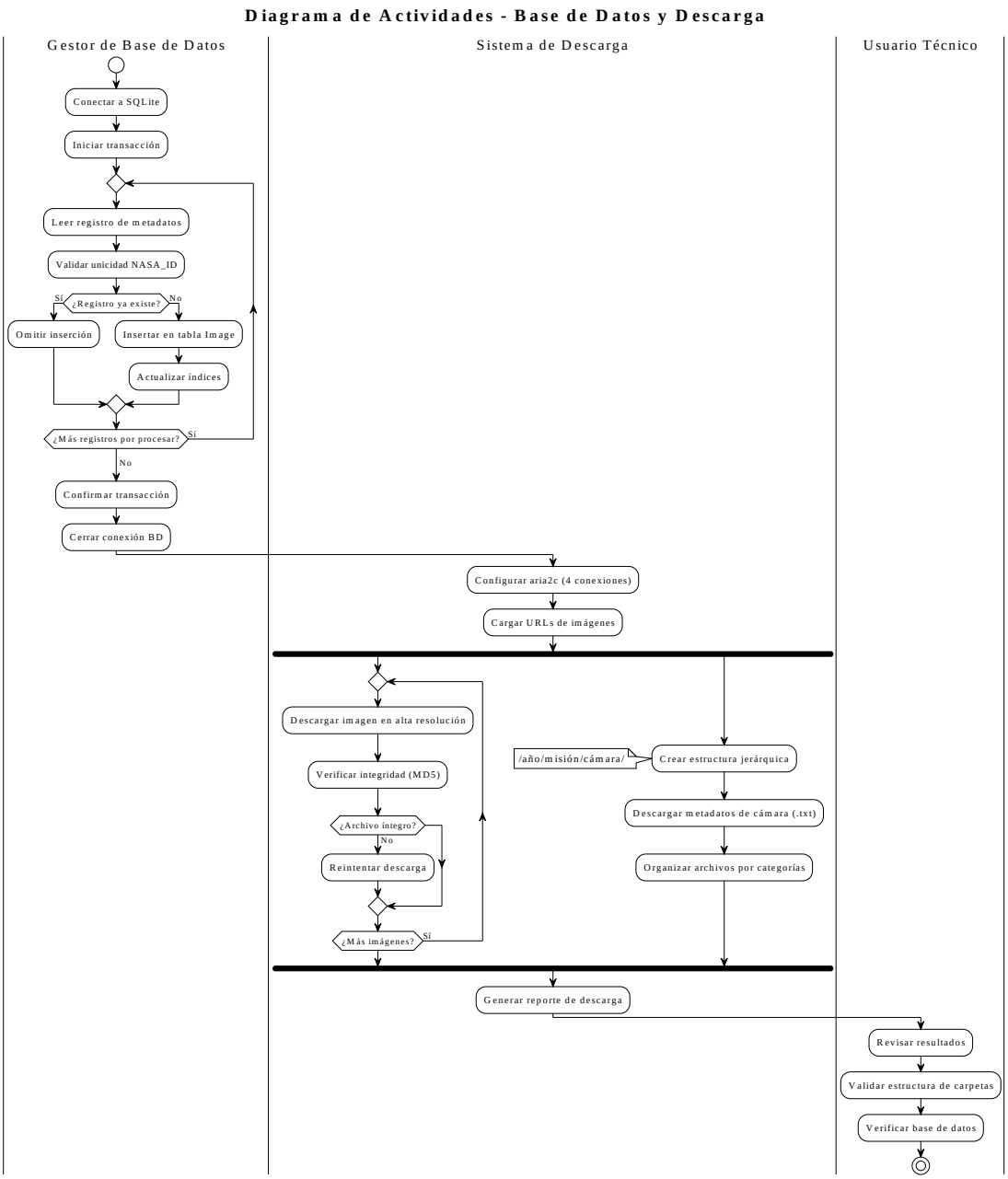

Figura 4.4: Diagrama de actividades del almacenamiento en base de datos y descarga local.

Diseño Orientado a Objetos El diagrama de clases presenta una arquitectura modular que separa claramente las responsabilidades de procesamiento, persistencia y modelado de datos, siguiendo principios de diseño orientado a objetos y patrones arquitectónicos establecidos. Esta separación no solo facilita la escalabilidad del sistema, sino también su testeo y mantenibilidad.

Clase Controladora ImageProcessor La clase `ImageProcessor` actúa como orquestador central del sistema, encapsulando la lógica de negocio para el procesamiento completo de metadatos. Sus métodos privados (`_insert_main_image()`, `_insert_details()`, `_insert_location()`, `_insert_camera_info()`) implementan el patrón Template Method, definiendo un flujo secuencial de inserción que garantiza consistencia transaccional. Esto es clave para evitar registros huérfanos o inconsistentes en la base de datos. Además, los métodos utilitarios (`_parse_date()`, `_parse_time()`, `_to_float()`) permiten normalizar datos provenientes del portal NASA, que pueden estar en formatos variables. El método `_handle_download_and_storage()` coordina la descarga, verificación y organización jerárquica de archivos según estructura temática.

Patrón DAO en MetadataCRUD La clase `MetadataCRUD` implementa el patrón Data Access Object (DAO), proporcionando una abstracción de las operaciones de base de datos. Sus métodos especializados (`create_image()`, `create_image_details()`, `create_map_location()`) encapsulan la lógica SQL, desacoplando la capa de persistencia del resto del sistema. Además, el método `get_paginated_metadatos()` permite consultar grandes volúmenes de datos sin sobrecargar la interfaz, y funciones específicas como `get_image_id_by_nasa_id()` mejoran la eficiencia de búsqueda.

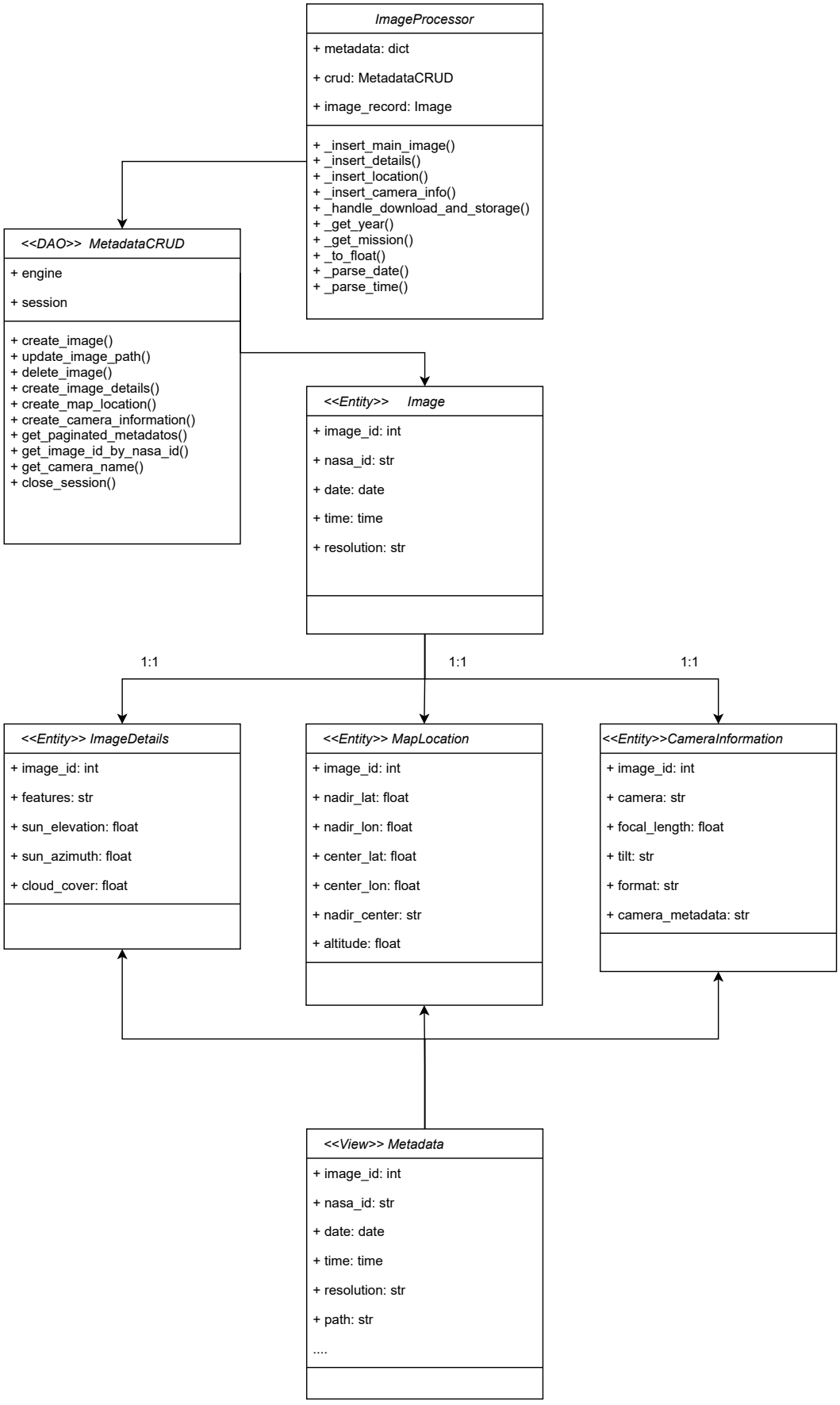

Figura 4.5: Modelo de clases del incremento 1: procesamiento, DAO y entidades del dominio.

Entidades del Dominio Las entidades `Image`, `ImageDetails`, `MapLocation` y `CameraInformation` reflejan una estructura normalizada conforme al modelo ER (ver Figura 4.6):

- **Image**: Actúa como entidad raíz, centralizando identificadores (`nasa_id`) y metadatos generales.
- **ImageDetails**: Registra atributos atmosféricos clave, como `cloud_cover`, relevantes para estudios de calidad lumínica.
- **MapLocation**: Alberga coordenadas geoespaciales, esenciales para análisis espaciales.
- **CameraInformation**: Agrupa datos técnicos que afectan la interpretación científica (e.g., `tilt`, `focal_length`).

La Figura 4.5 incluye además relaciones 1:1 entre las entidades, reflejando la estructura relacional descrita en el modelo ER. Esta correspondencia directa entre clases y tablas garantiza la trazabilidad de datos a nivel lógico y físico.

Vista Integrada de Metadatos La vista `Metadata` actúa como una abstracción lógica que consolida la información distribuida en múltiples tablas, tal como se muestra en el modelo ER (Figura 4.6). Esta vista está optimizada para lectura intensiva y permite a los módulos analíticos consultar de forma eficiente sin necesidad de múltiples operaciones JOIN complejas. Su propósito es reducir la complejidad técnica para quienes analizan contaminación lumínica o condiciones atmosféricas, favoreciendo un acceso ágil y coherente a los datos.

Relaciones y Integridad Las relaciones 1:1 entre entidades aseguran que cada imagen tenga un conjunto único y completo de metadatos. Esta estructura se eligió deliberadamente para reflejar que cada captura satelital posee una sola configuración técnica y geográfica. Se aplican restricciones de clave foránea con `ON DELETE CASCADE`, permitiendo una limpieza automática y segura de registros asociados cuando se elimina una imagen.

Modelo de Datos La Figura 4.6 muestra la estructura entidad-relación, resaltando claves primarias, foráneas y atributos esenciales. La base de datos se implementó con SQLite, elegido por ser una solución embebida, sin servidor, ideal para entornos monousuario y sistemas autónomos. Esto reduce la necesidad de configuraciones complejas, permitiendo portabilidad total del sistema sin depender de servicios externos. Esta implementación cumple el objetivo específico 5, estructurando la base de datos relacional mediante entidades normalizadas (Image, ImageDetails, MapLocation, CameraInformation) que organizan comprehensivamente los metadatos geográficos, atmosféricos y técnicos de cada imagen satelital, garantizando la integridad referencial del sistema.

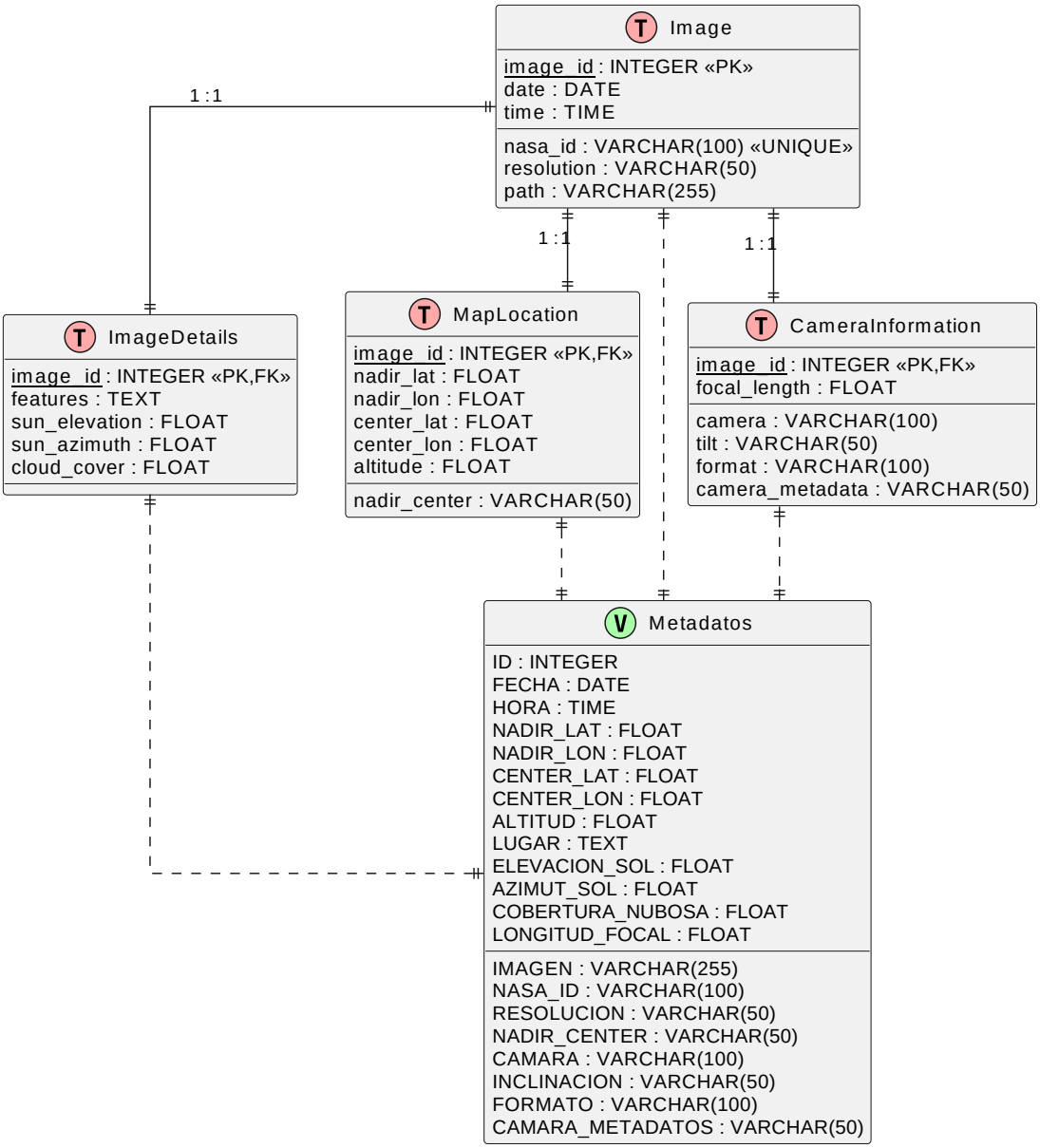

Figura 4.6: Modelo entidad-relación de la base de datos de imágenes.

Interacción entre Componentes: Diagramas de Secuencia Los diagramas de secuencia ilustran la comunicación entre módulos. Por ejemplo:

- La Figura 4.7 describe desde el inicio hasta la búsqueda de imágenes en el mapa. Se incluyen caminos alternativos como búsquedas sin resultados, validación de filtros y cierre anticipado.
- En la Figura 4.8, se prioriza la descarga de imágenes de mayor calidad, y se considera la posibilidad de que algunas URLs no contengan imágenes válidas.

- Las Figuras 4.9 y 4.10 modelan la extracción desde HTML. Se contemplan rutas condicionales como la ausencia de metadatos o la existencia de duplicados.
- Finalmente, las Figuras 4.11–4.13 modelan el procesamiento por lotes. Aquí se destaca el uso de `aria2c` para descargas concurrentes, directorios organizados temáticamente y validaciones de integridad MD5. También se registran logs y se notifica al usuario, incluso en caso de errores parciales.

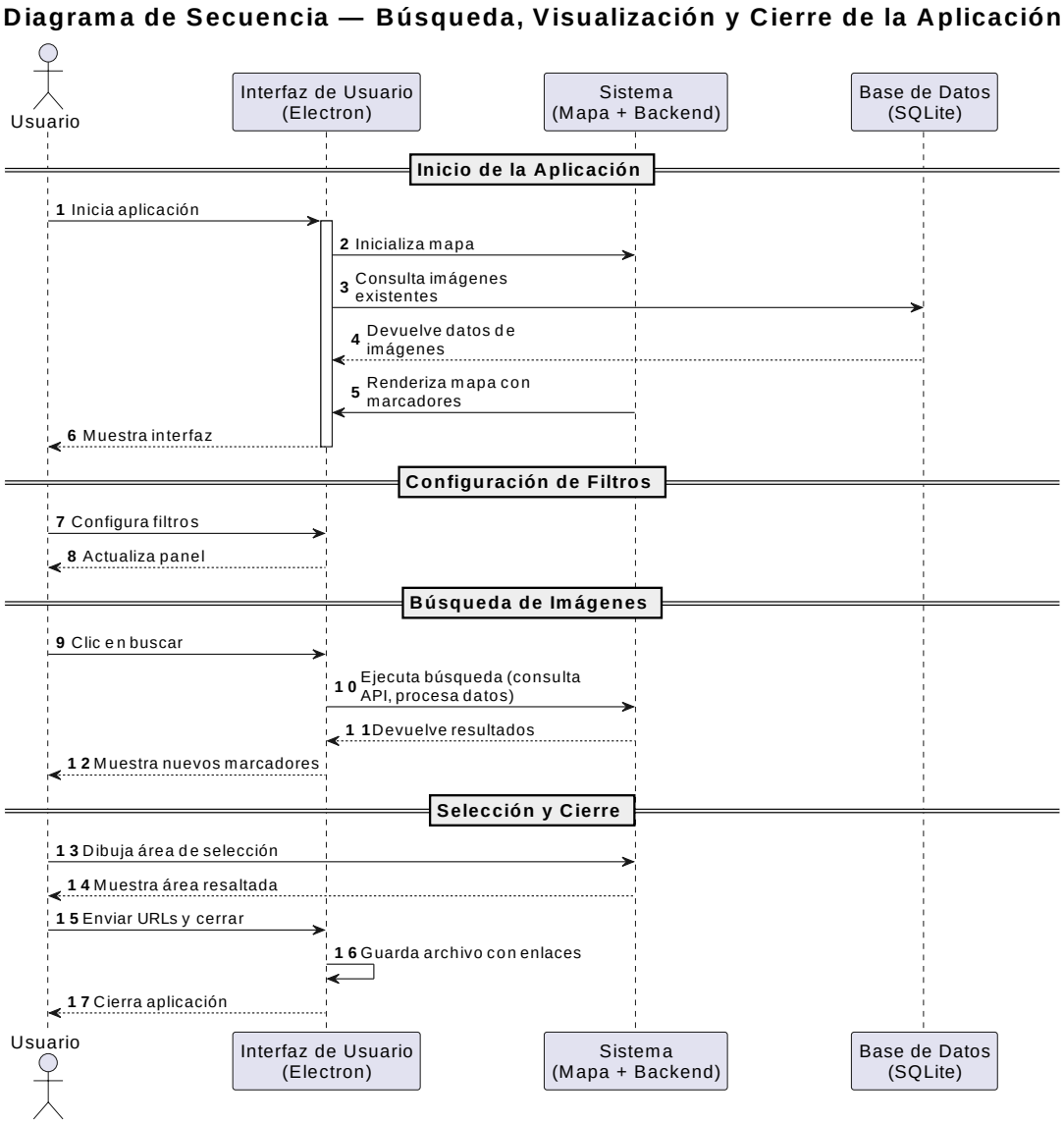

Figura 4.7: Diagrama de secuencia del flujo de búsqueda y filtrado inicial.

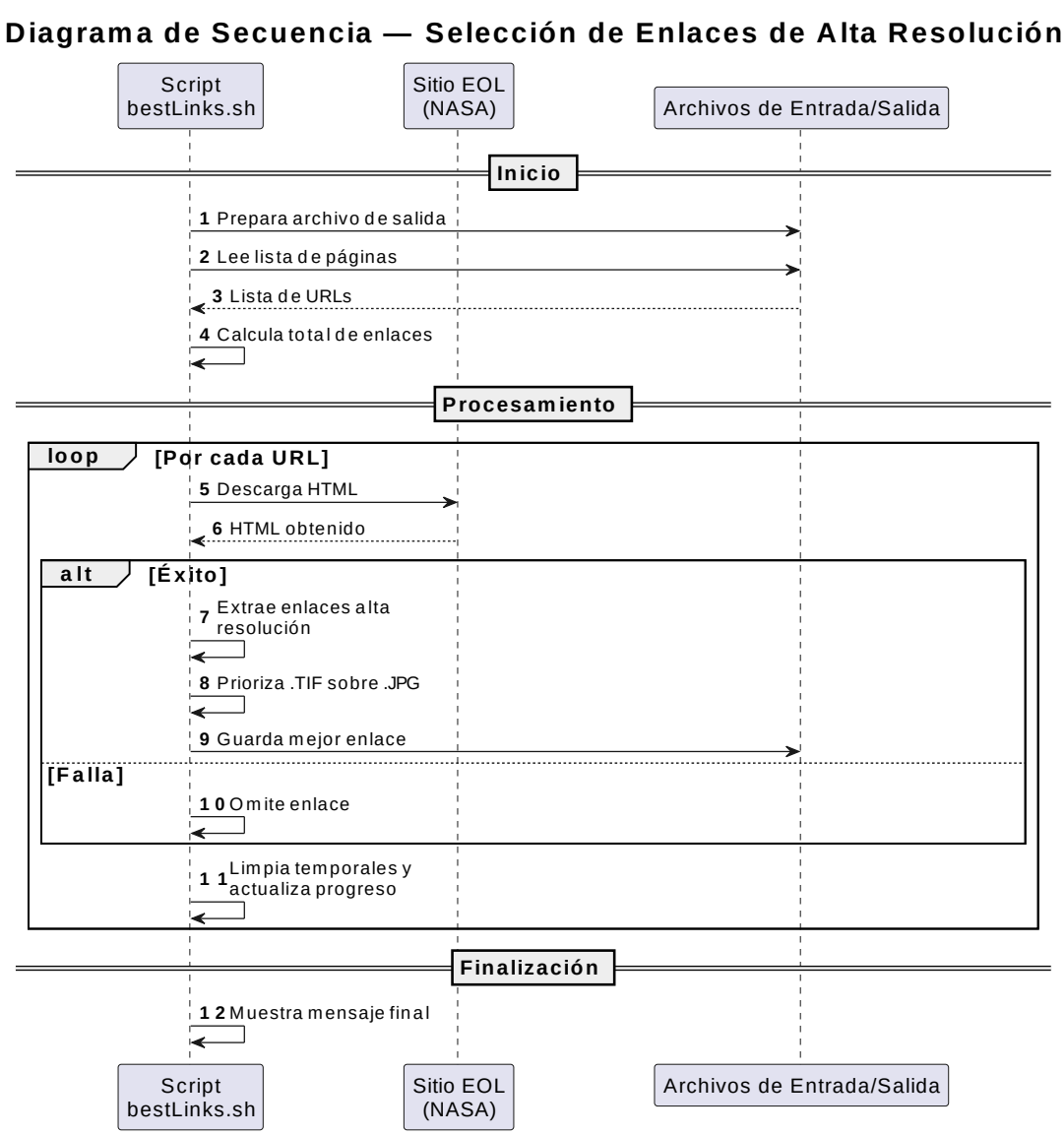

Figura 4.8: Diagrama de secuencia del filtrado por mejor resolución.

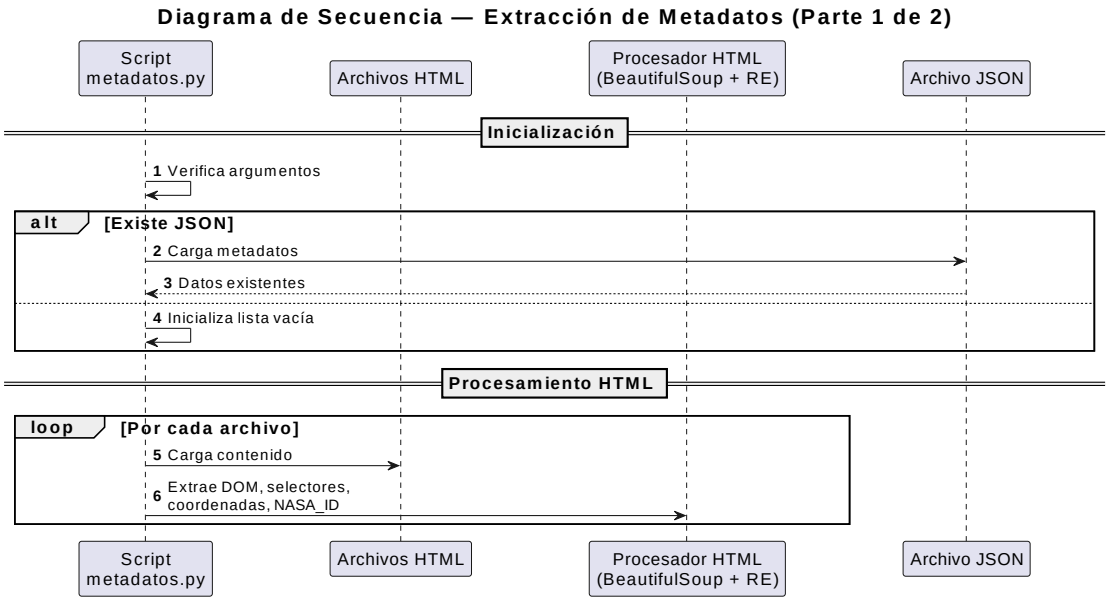

Figura 4.9: Procesamiento de metadatos a partir de archivos HTML (Parte 1 de 2).

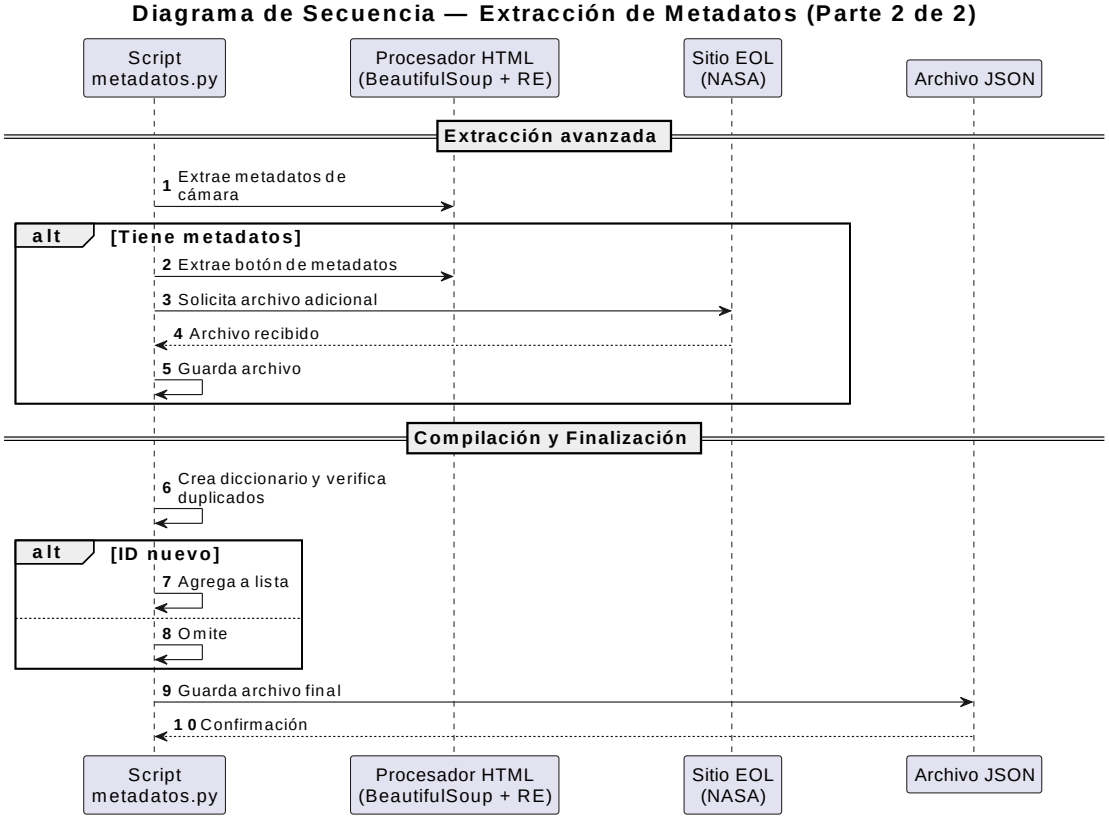

Figura 4.10: Extracción avanzada y guardado de metadatos (Parte 2 de 2).

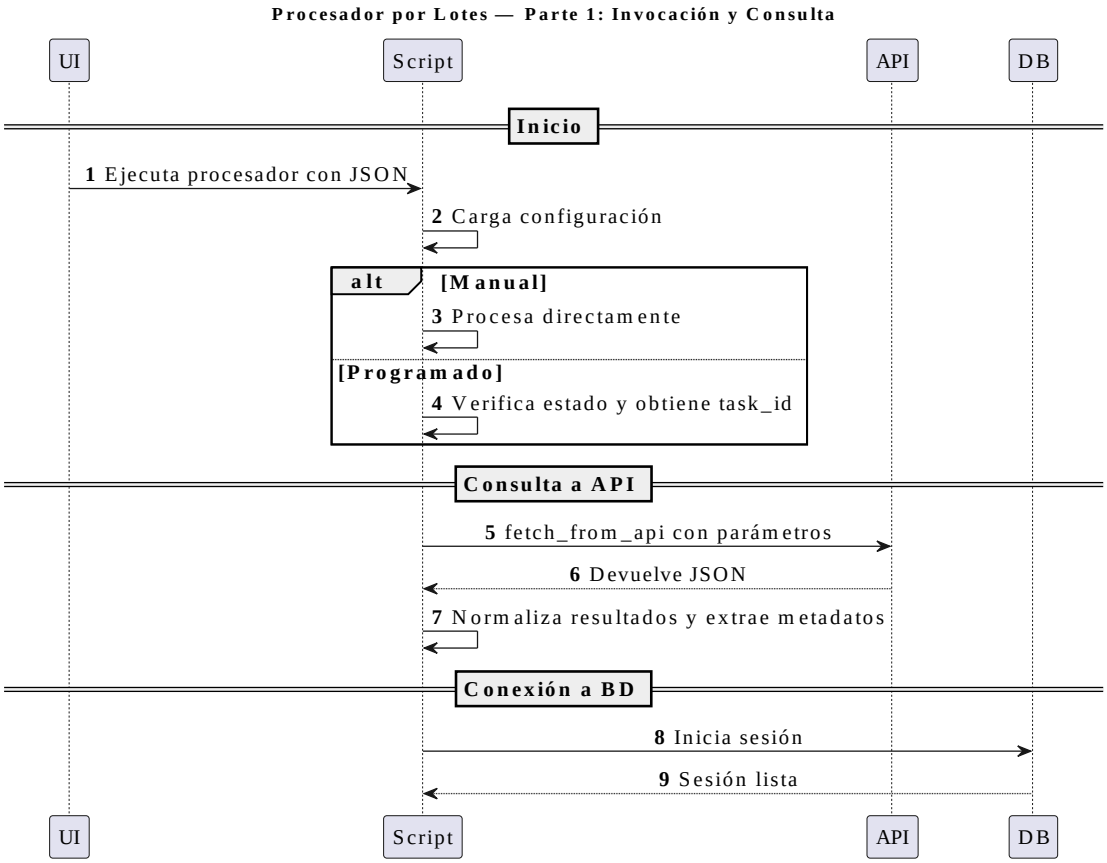

Figura 4.11: Procesador por lotes: Invocación, configuración y consulta (Parte 1 de 3).

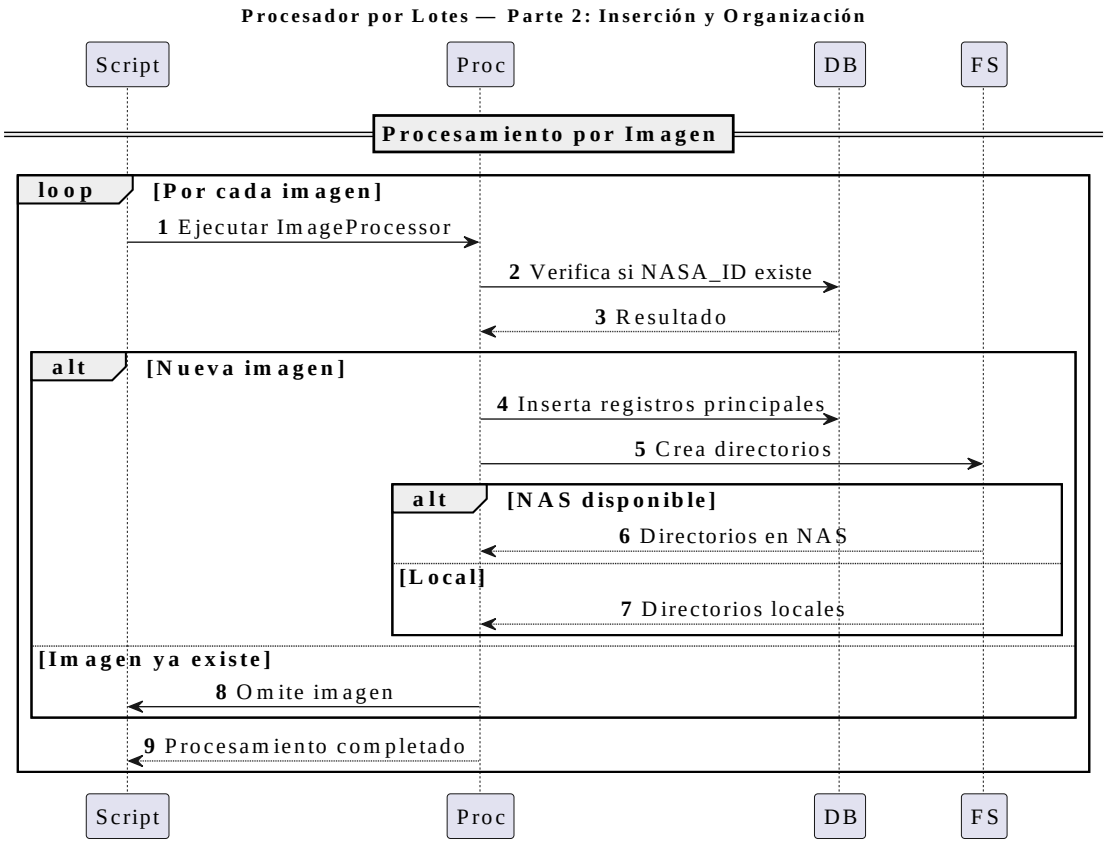

Figura 4.12: Procesador por lotes: Inserción de datos y organización de carpetas (Parte 2 de 3).

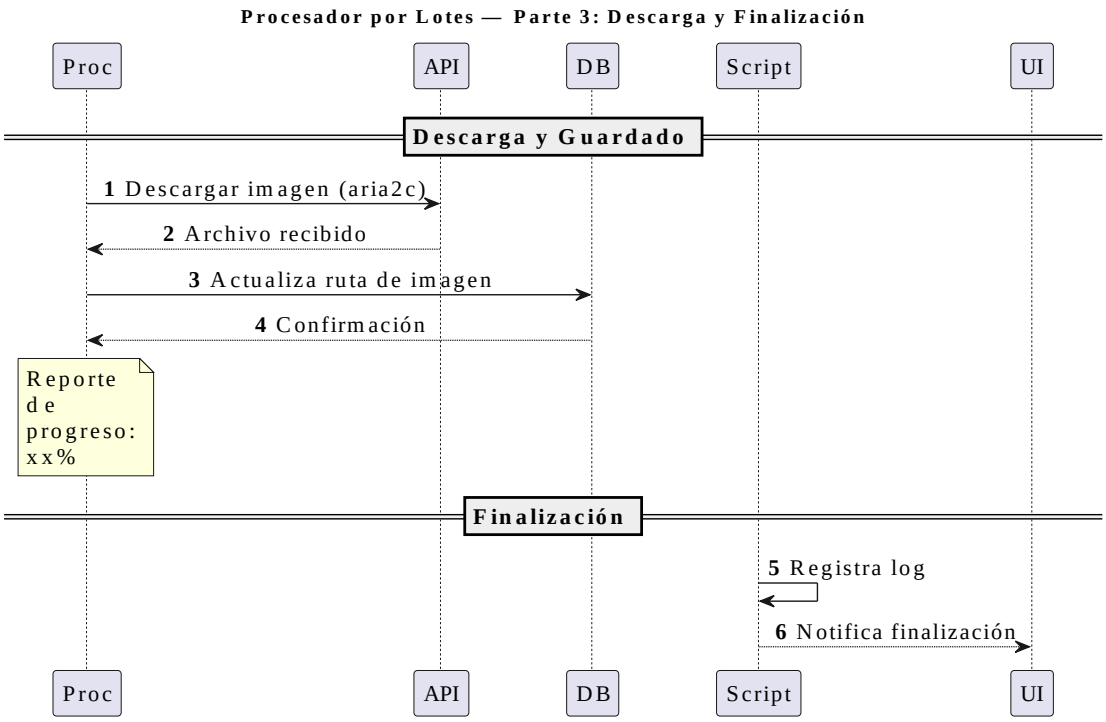

Figura 4.13: Procesador por lotes: Descarga de imágenes y cierre del proceso (Parte 3 de 3).

Resumen de la Fase de Diseño Los modelos desarrollados en esta fase proporcionaron una visión clara y estructurada del sistema, facilitando la transición hacia la implementación. Además, permitieron establecer un lenguaje común entre los diferentes componentes, asegurando coherencia entre el diseño funcional y técnico del incremento.

4.2.4 Fase 3: Implementación

La implementación de este incremento se organizó en módulos independientes que conforman un flujo secuencial: extracción, procesamiento, almacenamiento y organización de imágenes junto con sus metadatos.

El primer paso consistió en el desarrollo de un módulo que extrae los enlaces Python que realiza búsquedas dinámicas en el portal NASA Gateway. Este componente genera URLs de imágenes a partir de criterios como cobertura nubosa, sensor activo o inclinación de la cámara. La ejecución del script produce un archivo con enlaces únicos, como se muestra en la Figura 4.14.

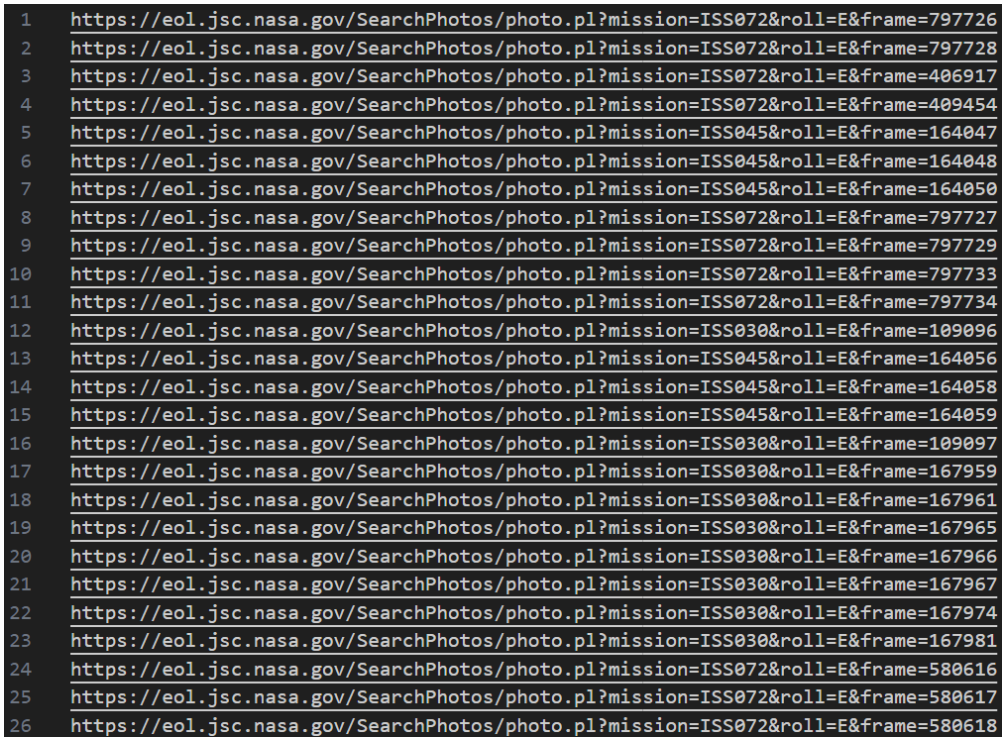

Figura 4.14: Archivo de salida generado por el módulo de extracción de enlaces.

Posteriormente, se implementó un módulo de filtrado que descarta imágenes cuya resolución no cumple con un umbral mínimo predefinido. Este mecanismo materializa el Objetivo Específico 2, garantizando la obtención de imágenes satelitales nocturnas en la mejor resolución disponible, priorizando el formato .tif , utilizando .jpg solo cuando no hay una alternativa superior junto con sus metadatos técnicos. La Figura 4.15 ilustra el resultado de esta etapa.

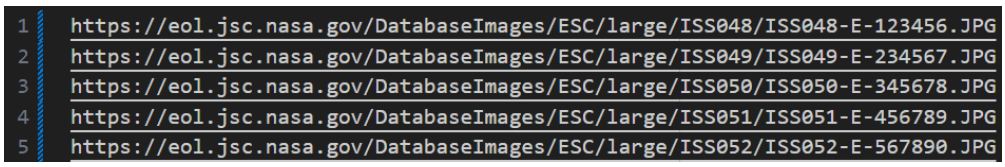

Figura 4.15: Enlaces seleccionados tras aplicar filtros de resolución y tipo de archivo.

Cada URL resultante se emplea como entrada para un extractor de metadatos, también desarrollado en Python. Este módulo analiza las fichas HTML asociadas a cada imagen y recupera atributos técnicos como latitud, longitud, altitud, inclinación, tipo de cámara, y hora local. Los resultados se almacenan en archivos `.json`, como se observa en la Figura 4.16.

```
{
    "NASA_ID": "ISS045-E-164059",
    "FECHA": "2015.12.06",
    "HORA": "08:23:43 GMT",
    "RESOLUCION": "4928 x 3280 pixels",
    "URL": "https://eol.jsc.nasa.gov/DatabaseImages/ESC/large/ISS045/ISS045-E-164059.JPG",
    "NADIR_LAT": 6.1,
    "NADIR_LON": -80.7,
    "CENTER_LAT": null,
    "CENTER_LON": null,
    "NADIR_CENTER": null,
    "ALTITUD": null,
    "LUGAR": "",
    "ELEVACION_SOL": -41.0,
    "AZIMUT_SOL": 115.0,
    "COBERTURA_NUBOSA": 0.0,
    "CAMARA": "NIKON D4 S/N: 2071124",
    "LONGITUD_FOCAL": 400.0,
    "INCLINACION": "",
    "FORMATO": "",
    "CAMARA_METADATA": "scripts/camera_data/ISS045-E-164059.txt.gz"
},
```


Figura 4.16: Ejecución del script de metadatos. Vista previa del archivo JSON generado.

Luego, los metadatos se insertan en una base de datos SQLite mediante el módulo `MetadataCRUD`. Este componente garantiza la unicidad del campo `nasa_id`, valida tipos de datos y asegura la consistencia del registro.

Para completar el flujo, se integró un sistema de descarga automatizada basado en `aria2c`, configurado para usar múltiples conexiones simultáneas. Las imágenes descargadas se organizan localmente en una estructura jerárquica por año, misión y cámara, como se muestra en la Figura 4.17. Además, se almacenan metadatos específicos de cada cámara en una carpeta adicional llamada `camera_data`.

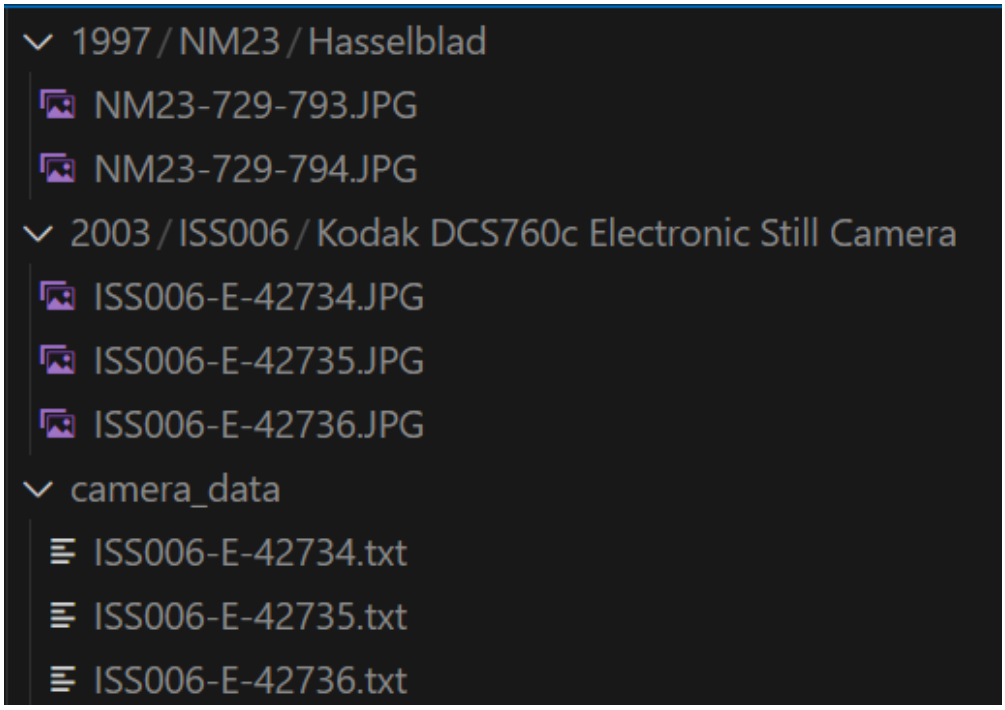

Figura 4.17: Estructura jerárquica local generada por el sistema de organización automática.

Esta estructura jerárquica materializa el Objetivo Específico 6, integrando exitosamente el sistema con el entorno NAS y estableciendo la organización escalable por año, misión y cámara que garantiza la integridad y disponibilidad de los datos satelitales recolectados.

Interfaz técnica del sistema

Para facilitar la interacción con el sistema desarrollado, se construyó una interfaz gráfica utilizando Electron y Node.js. Esta interfaz permite ejecutar los procesos de extracción, descarga y persistencia de forma parametrizada y visual, complementando la interfaz por consola. La interfaz principal del sistema, desde la cual se lanzan los procesos, se muestra en la Figura 4.18.

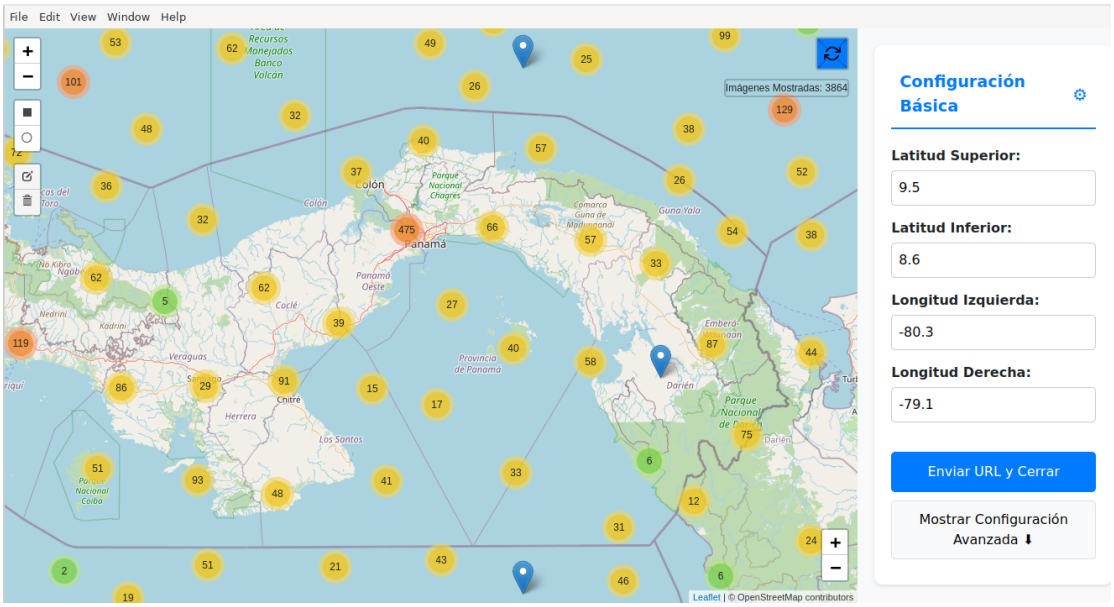

Figura 4.18: Pantalla principal de la interfaz desarrollada con Electron. Se permite lanzar procesos por módulo.

El seguimiento visual del proceso de descarga concurrente está disponible a través de la terminal mostrada en la Figura 4.19.

```
Iniciando aplicación de mapa...
Dibuja un área en el mapa, genera la URL, y cierra la aplicación.
 SECCIÓN
 Procesamiento de Enlaces

Procesando enlaces para alta resolución...
Procesando Enlaces (TIF: 0, JPG: 15): 5/5 : [################################################## ] 100%
Enlaces de alta resolución guardados en: /home/jose/API-NASA/map/scripts/automatica/enlaces_highres.txt
 SECCIÓN
 Procesamiento de URLs

Procesando cada URL en el archivo de enlaces...
Procesando URL: https://eol.jsc.nasa.gov/SearchPhotos/photo.pl?mission=ISS030&roll=E&frame=167959
NASA_ID ISS030-E-167959 ya existe en el JSON. No se duplicará.
 Metadatos guardados en /home/jose/API-NASA/map/scripts/automatica/metadatos.json
Procesando URL: https://eol.jsc.nasa.gov/SearchPhotos/photo.pl?mission=ISS030&roll=E&frame=167966
```

Figura 4.19: Vista del monitoreo de descargas activas con `aria2c`, integración del backend asíncrono.

Esta interfaz actúa como puente entre el usuario técnico y los módulos internos descritos en esta sección, permitiendo gestionar parámetros de ejecución, mostrar estados de avance, y consultar registros procesados.

4.2.5 Fase 4: Pruebas

Cada módulo fue validado de forma individual. Se verificaron los siguientes aspectos:

- Extracción de enlaces y estructura de URLs.
- Filtrado de imágenes por resolución y formato.
- Integridad y consistencia de los metadatos extraídos.
- Correcto almacenamiento en la base de datos.
- Organización jerárquica de archivos locales.

Análisis de Rendimiento

Tabla 4.3: Métricas de rendimiento del sistema de web scraping

Métrica	**Valor**
Tiempo promedio de extracción de metadatos	4.8 s
Tasa de éxito en extracción de metadatos	92 %
Completitud de metadatos geoespaciales	94 %
Volumen procesado	2240 imágenes (9 GB)
Duración total del proceso	2–3 horas

4.2.6 Fase 5: Resultados del Incremento

Logros principales

- Flujo automatizado completo desde búsqueda hasta descarga, cumpliendo el objetivo específico 1 mediante el esquema eficiente de descarga masiva implementado con aria2c y mecanismos de scraping automatizados.
- Recuperación de metadatos técnicos y geoespaciales con alta precisión.

- Base de datos estructurada para análisis posterior.
- Interfaz técnica flexible (CLI + GUI).
- Robustez ante fallos y errores de red.

Limitaciones identificadas

- Acoplamiento a la estructura HTML del portal.
- Metadatos incompletos en imágenes antiguas.
- Variabilidad en formatos de fecha.
- No adaptabilidad dinámica a restricciones de consulta.

Mecanismos de recuperación

- Reintentos automáticos con espera exponencial.
- Registro de errores por URL.
- Transacciones atómicas en base de datos.
- Logging detallado por módulo.

Este incremento estableció una infraestructura sólida que permite la adquisición, procesamiento y organización de imágenes satelitales nocturnas, sirviendo de base para futuras funciones de análisis, visualización y automatización periódica.

4.3 Incremento 2: Explorador Visual de Imágenes y Consulta de Metadatos

4.3.1 Descripción General

Este segundo incremento amplía las funcionalidades del sistema mediante herramientas visuales y técnicas que permiten la validación manual y la

navegación estructurada del conjunto de imágenes descargadas. Mientras que el Incremento 1 abordó la adquisición y organización automática de imágenes satelitales, esta fase se enfocó en facilitar la exploración jerárquica del entorno NAS y el acceso directo a los metadatos almacenados en la base de datos relacional.

Se desarrollaron dos componentes principales con propósitos complementarios:

- **Explorador NAS (GTK+3)**: Aplicación gráfica que permite navegar entre carpetas clasificadas por año, sensor y misión. Ofrece acciones como visualizar, copiar o eliminar imágenes mediante clic derecho.
- **Vista técnica por consola (Textual)**: Interfaz interactiva en terminal que permite consultar, exportar o manipular los metadatos a través de una vista consolidada extraída de la base de datos SQLite.

Ambos módulos fueron diseñados con independencia funcional y se integraron al menú principal del sistema, conservando una arquitectura modular, coherente y de bajo acoplamiento.

4.3.2 Fase 1: Análisis de Requisitos

Se identificaron los objetivos funcionales y técnicos específicos del incremento, con énfasis en la portabilidad, eficiencia en carga, y no interferencia con la lógica de backend existente.

Tabla 4.4: Requerimientos funcionales abordados en el incremento 2

ID	Descripción
RF6	Permitir la navegación jerárquica del entorno NAS con clasificación por año, misión y tipo de sensor.
RF7	Habilitar acciones básicas sobre los archivos: visualizar, descargar y eliminar mediante interfaz gráfica.
RF8	Implementar una interfaz técnica por consola para consultar metadatos almacenados en la base de datos.
RF9	Integrar filtros dinámicos en la vista técnica para facilitar búsquedas por fecha, sensor o misión.

Requerimientos Funcionales

Tabla 4.5: Requerimientos no funcionales abordados en el incremento 2

ID	Descripción
RNF5	Garantizar compatibilidad con entornos Linux, incluyendo distribuciones bajo WSL2.
RNF6	Optimizar tiempos de carga para directorios con más de 1000 imágenes.
RNF7	Diseñar módulos reutilizables e independientes del backend principal.
RNF8	Utilizar bibliotecas estables y libres de dependencias propietarias.

Requerimientos No Funcionales

4.3.3 Fase 2: Diseño del Sistema

El diseño del sistema se centró en la definición estructurada y detallada de los dos módulos principales: el visualizador técnico de metadatos y el explorador gráfico de archivos. Para cada uno, se elaboraron diagramas UML que describen los casos de uso, flujos de actividad, secuencias de interacción y arquitectura de clases. Esta separación modular garantiza una arquitectura escalable y facilita el mantenimiento evolutivo del sistema.

Visualizador Técnico de Metadatos

Casos de Uso La Figura 4.20 muestra los principales casos de uso del visualizador técnico. El usuario puede consultar información de imágenes satelitales, alternar entre fuentes (ISS y NOAA), explorar archivos complementarios, exportar resultados en formato CSV y eliminar registros del sistema. Estas funciones constituyen el núcleo operativo de este módulo técnico.

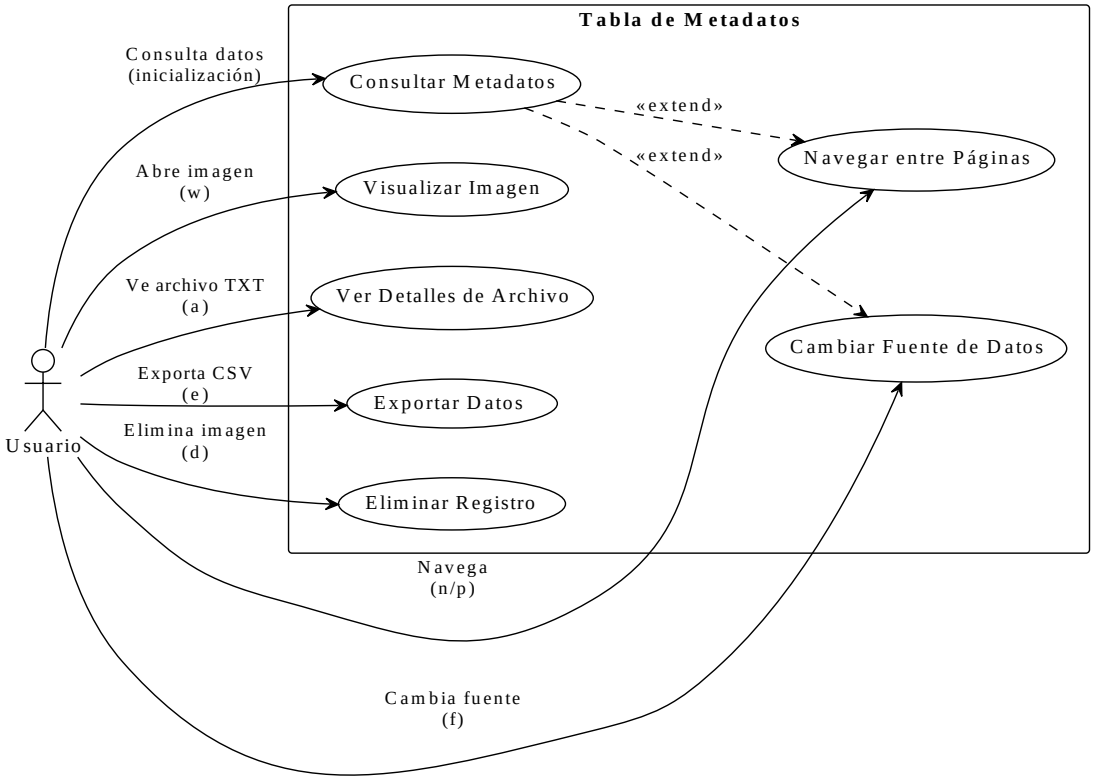

Figura 4.20: Casos de uso del visualizador de metadatos: operaciones de consulta, navegación y gestión de datos satelitales.

Diagramas de Actividad Los diagramas de actividad de este módulo describen en detalle las secuencias operacionales que ejecuta el sistema DataTable para gestión y visualización de metadatos:

- La Figura 4.21 modela la inicialización completa del sistema. El flujo inicia recibiendo parámetros de configuración, seguido de la configuración de la interfaz principal y la creación de la tabla optimizada. Se establecen parámetros iniciales de paginación (offset=0, limit=100) y se define ISS como fuente predeterminada. El proceso culmina agregando las columnas correspondientes, cargando la primera página de datos y presentando la interfaz lista para interacción del usuario.

- La Figura 4.22 representa la navegación entre páginas mediante teclas de control. Para avanzar páginas, el sistema consulta si existen más datos disponibles en la base de datos; si los hay, incrementa el offset y carga la siguiente página, caso contrario notifica que no hay más datos para

mostrar. Para retroceder, verifica que no sea la primera página; si es válido, decrementa el offset y muestra la página anterior, de lo contrario alerta que ya está en la primera página. Este mecanismo asegura navegación controlada en conjuntos de datos extensos.

- La Figura 4.23 detalla la interacción con archivos asociados a cada registro. El sistema primero verifica que la tabla contenga datos y que el usuario haya seleccionado una fila. Dependiendo de la operación solicitada, extrae la ruta de imagen o archivo TXT de la fila seleccionada. Para imágenes, valida la ruta y abre el visor externo con configuración específica de escala y geometría. Para archivos TXT, verifica la existencia del archivo y lo presenta en un visor de texto. Ambos flujos incluyen manejo de errores cuando las rutas no son válidas o los archivos no existen.

- La Figura 4.24 ilustra las operaciones de gestión de datos. La exportación recorre todas las filas de la tabla, convierte los datos a formato CSV y genera un archivo con cabeceras incluidas. La eliminación requiere selección previa de fila, valida que contenga datos suficientes, ejecuta el borrado en base de datos y actualiza la vista para reflejar los cambios. El cambio de fuente alterna entre ISS y NOAA, reinicia la paginación, adapta las cabeceras al formato correspondiente, limpia la tabla actual y recarga los datos con la nueva configuración.

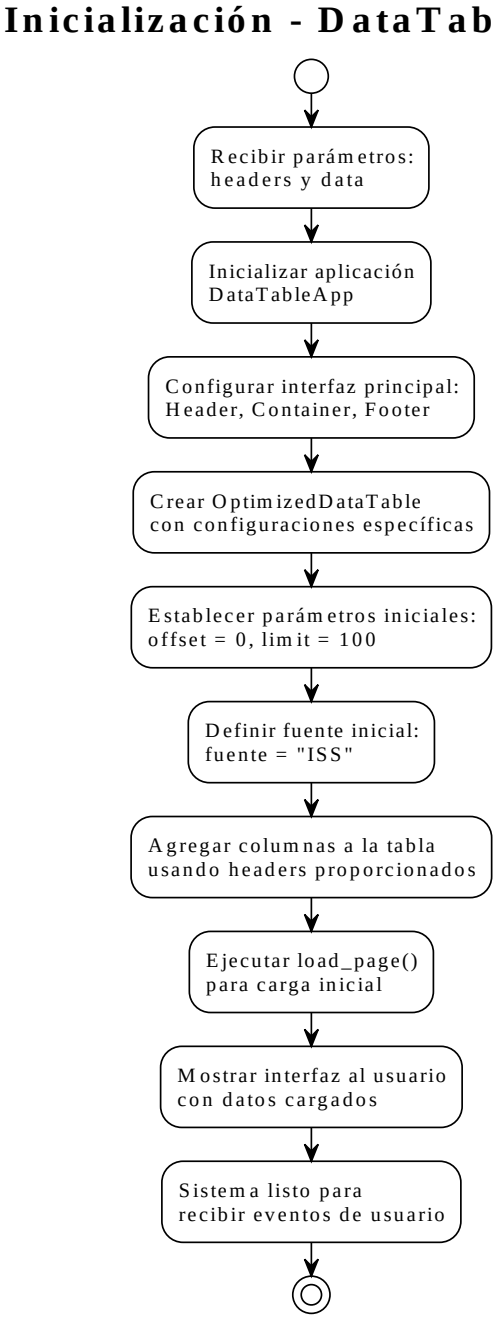

Figura 4.21: Inicialización del visualizador técnico: configuración de entorno y carga de interfaz.

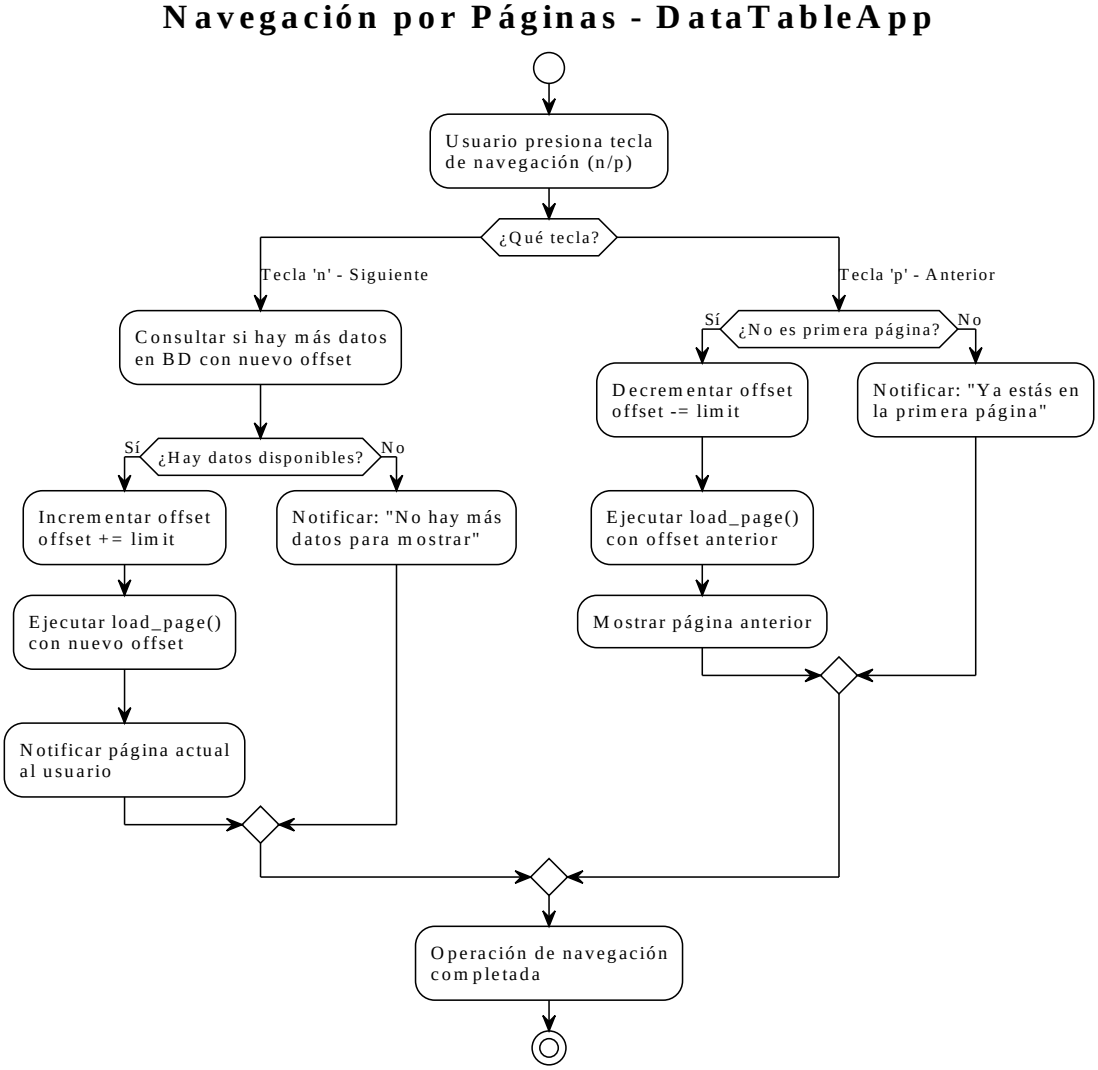

Figura 4.22: Navegación entre páginas de resultados en la tabla de metadatos.

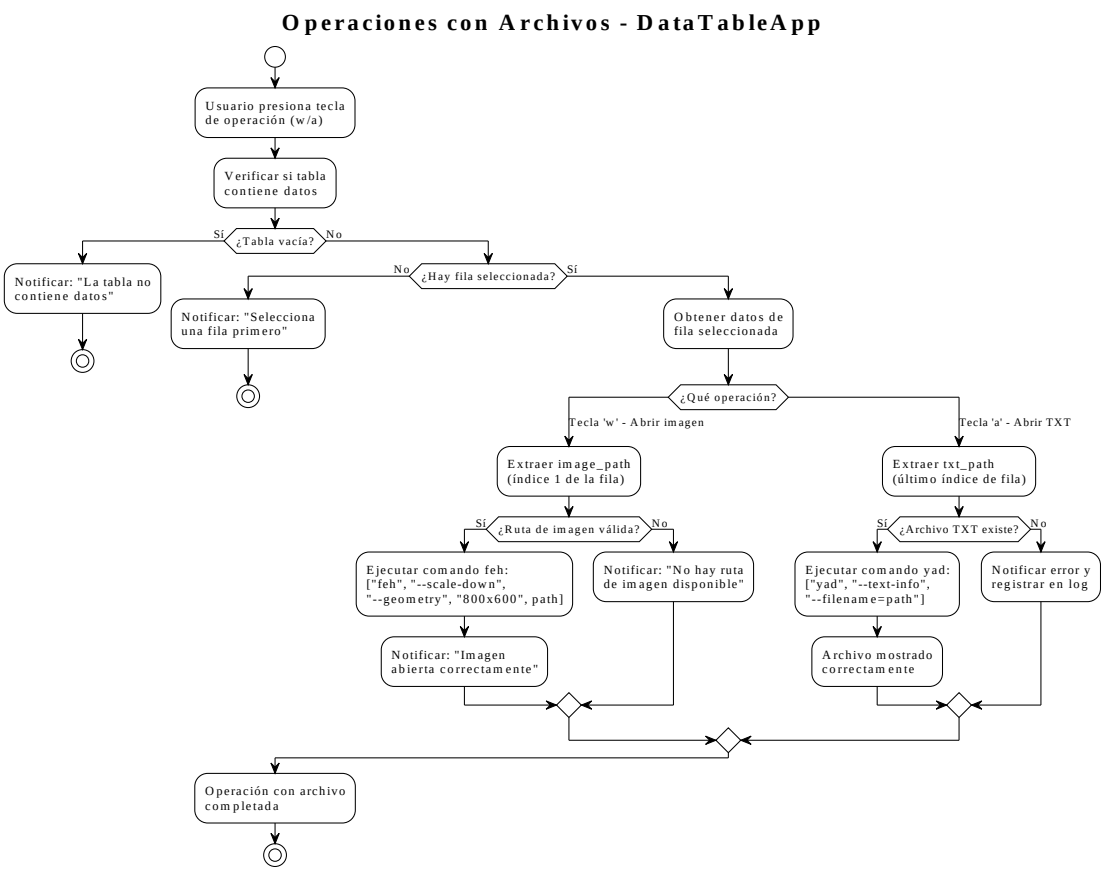

Figura 4.23: Interacción con archivos asociados desde la tabla técnica.

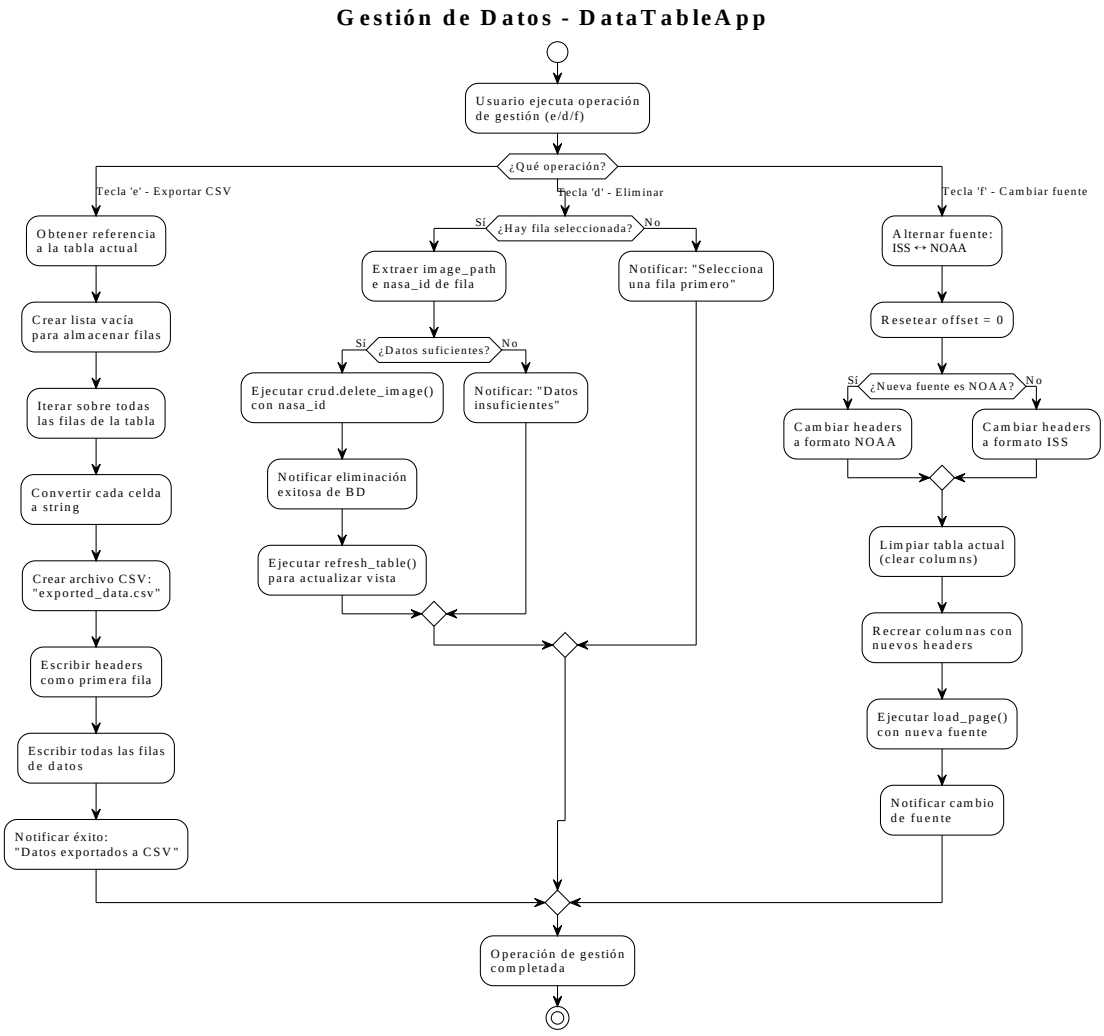

Figura 4.24: Gestión de datos desde la vista técnica: exportación y eliminación.

Diagramas de Secuencia Se diseñaron diagramas de secuencia para representar la interacción temporal entre los elementos del sistema:

- La Figura 4.25 muestra la secuencia de inicialización, donde se configuran las columnas, se consulta la base de datos con paginación, y se cargan los registros iniciales. La operación se dispara al montar la interfaz principal y se confirma al notificar al usuario que los datos han sido cargados.
- En la Figura 4.26, se representa el cambio de fuente de datos entre ISS y NOAA (tecla ’f’). Se actualizan encabezados y datos, ya sea accediendo a la base de datos o cargando desde un archivo JSON, según la fuente seleccionada. Esta lógica condicional permite mantener una sola interfaz con múltiples fuentes heterogéneas.

- La Figura 4.27 ilustra el intento de abrir una imagen desde la tabla (tecla 'w'). Se validan múltiples condiciones: tabla vacía, selección activa, y existencia de ruta válida. En caso de éxito, se lanza un visor externo. Si falla, se notifica el error.

- La Figura 4.28 describe el proceso de exportación de datos a un archivo CSV. Se iteran todas las filas de la tabla, se convierten a texto plano y se guardan con codificación UTF-8. Una notificación final confirma el éxito de la operación.

- En la Figura 4.29, se detalla la eliminación de una imagen seleccionada. El sistema valida que haya una selección activa, extrae el 'nasa id', realiza la operación SQL en la base de datos y actualiza la tabla. Las validaciones previas minimizan errores de ejecución.

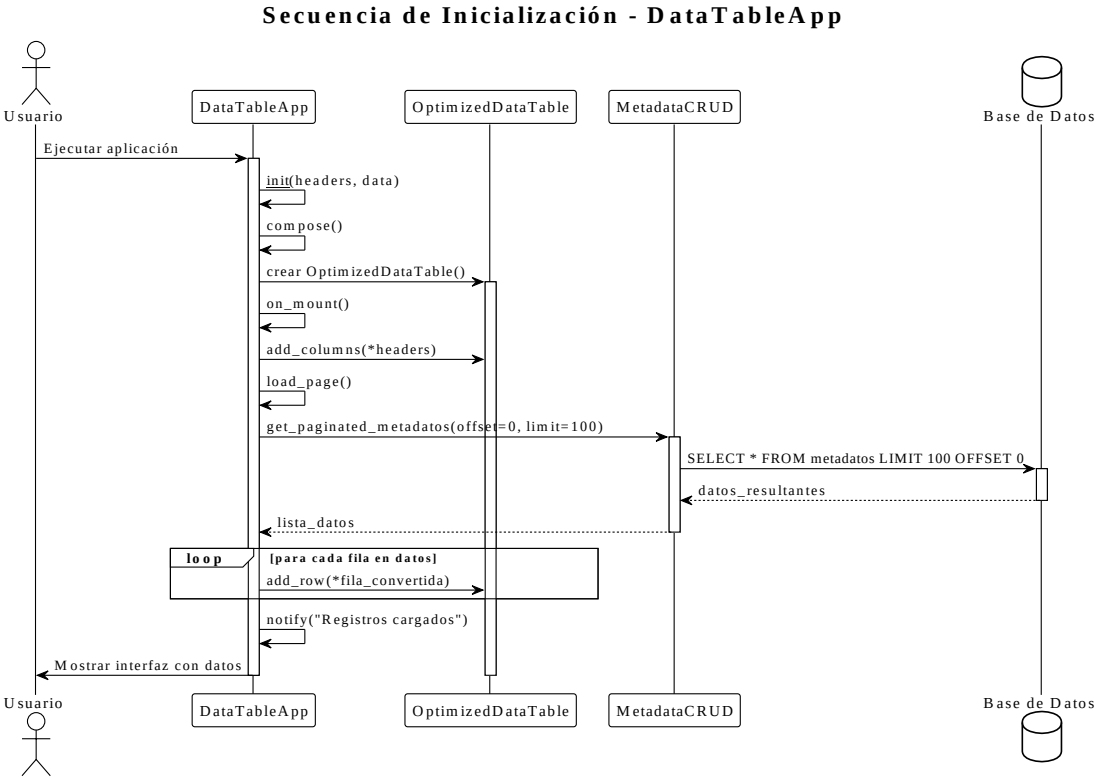

Figura 4.25: Secuencia de inicialización de la tabla de metadatos.

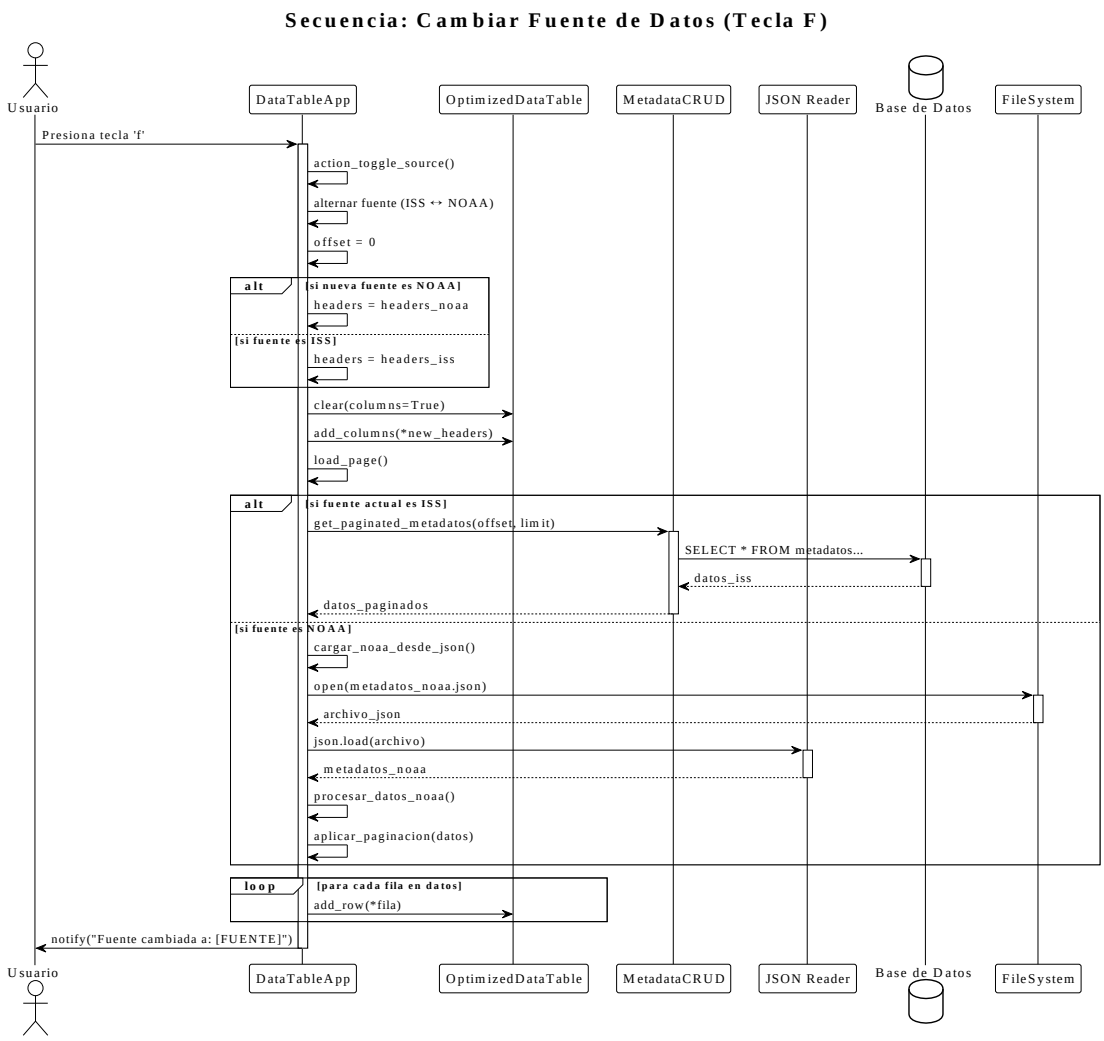

Figura 4.26: Cambio de fuente de datos entre conjuntos ISS y NOAA.

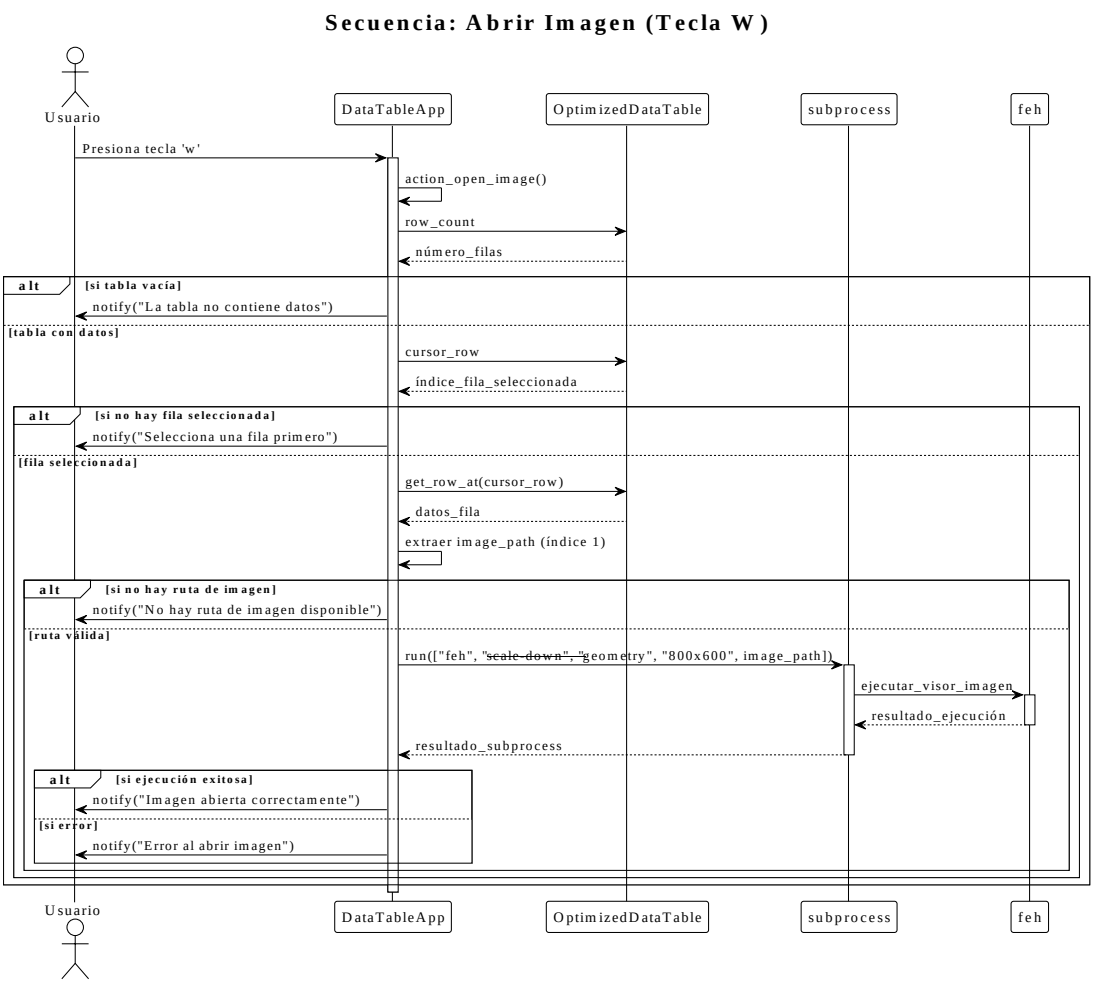

Figura 4.27: Visualización de imágenes desde la tabla de metadatos.

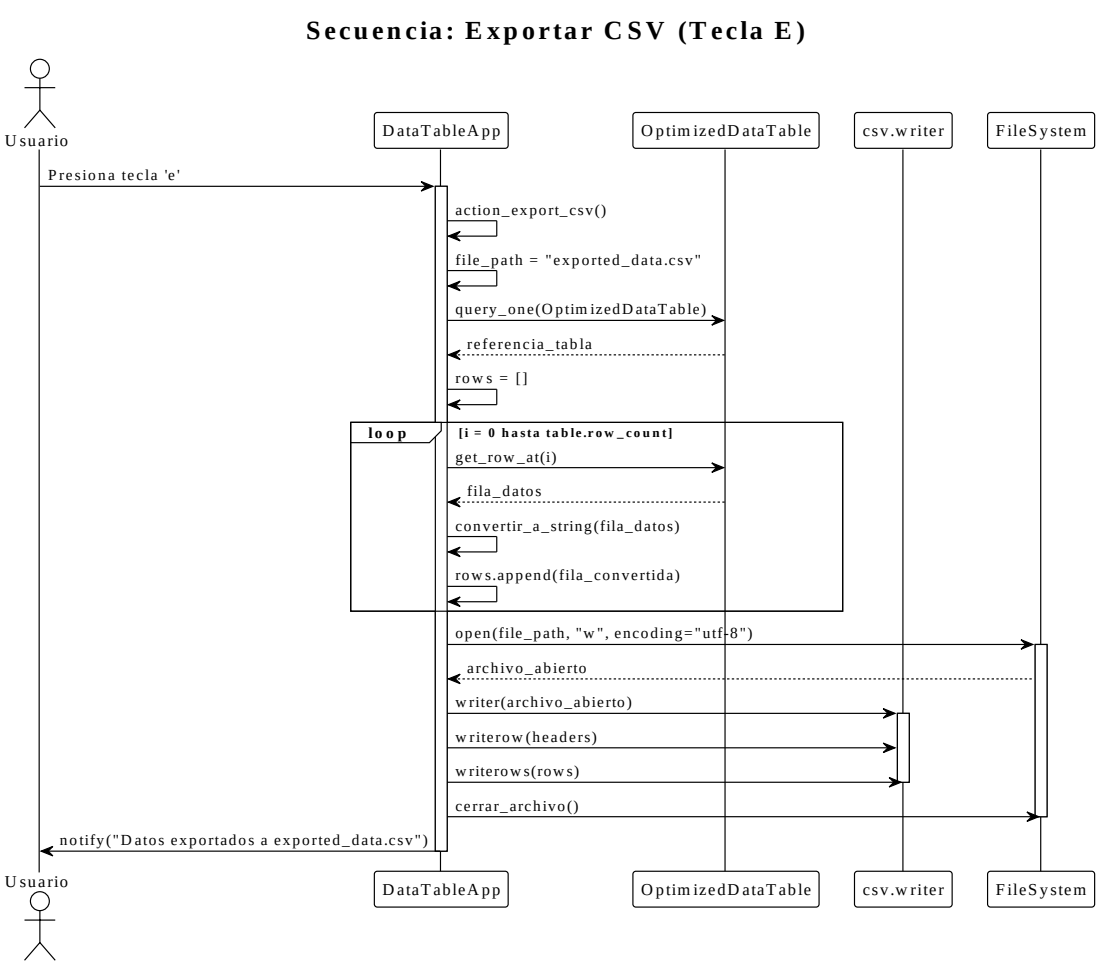

Figura 4.28: Exportación de registros en formato CSV.

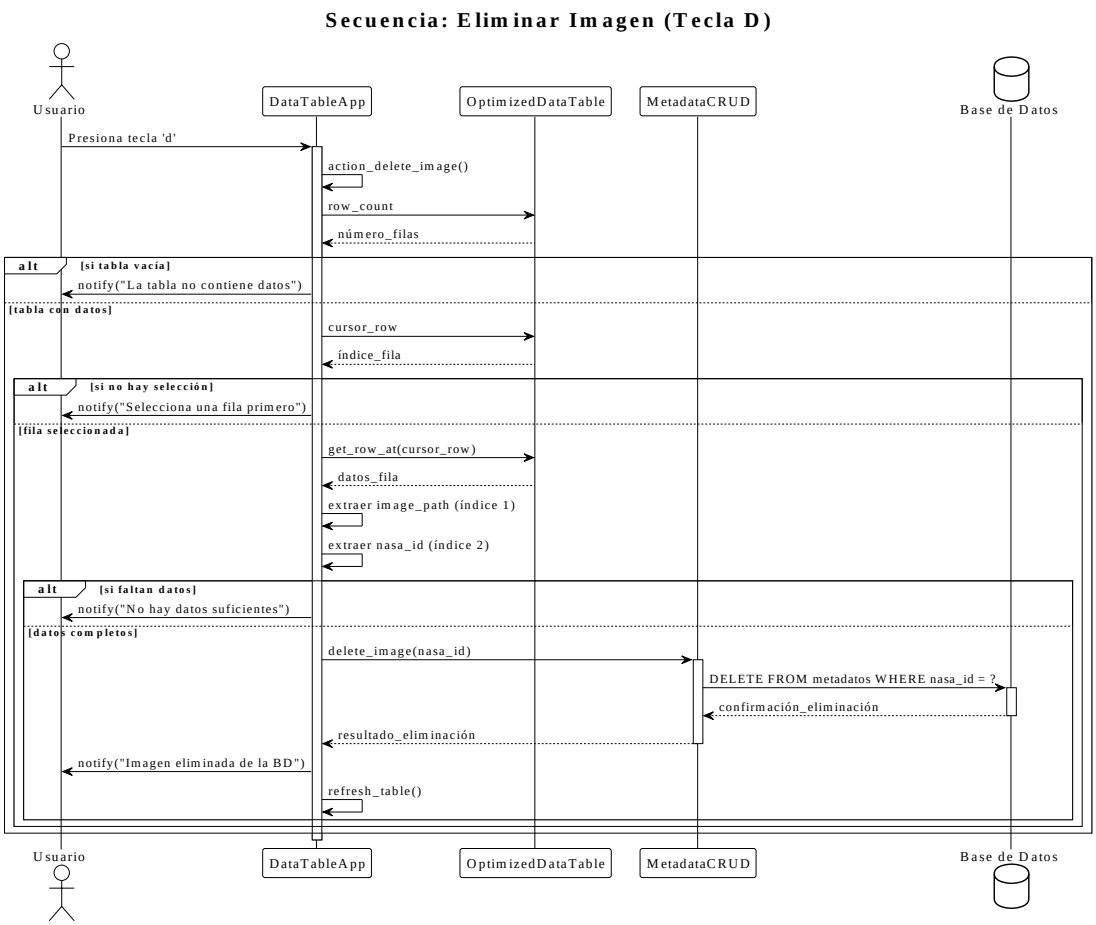

Figura 4.29: Eliminación de registros y actualización de interfaz.

Explorador de Imágenes

Casos de Uso La Figura 4.30 resume las funciones clave del explorador de imágenes. El usuario puede navegar por el sistema de archivos, visualizar imágenes y otros formatos, descargar contenido y eliminar archivos del entorno NAS.

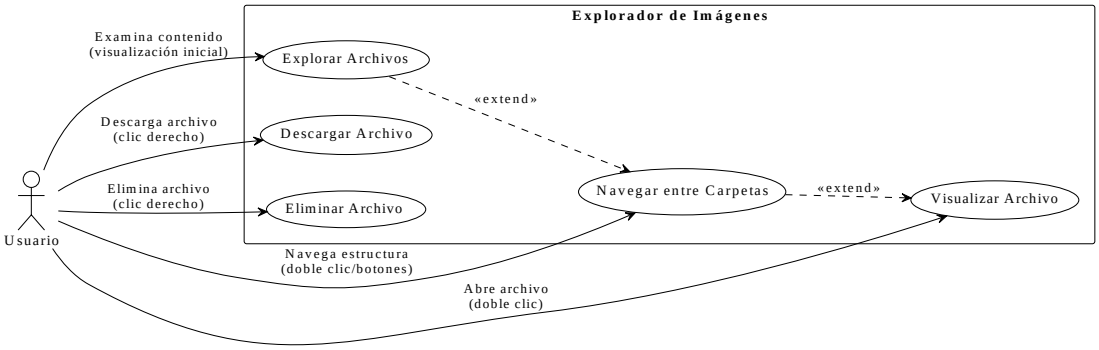

Figura 4.30: Casos de uso del explorador de imágenes: navegación y gestión de archivos.

siguientes:

Diagramas de Actividad Para este componente se elaboraron diagramas que reflejan el comportamiento del sistema ante diferentes eventos, incorporando caminos alternativos y validaciones de entorno que aseguran una experiencia robusta y coherente:

- La Figura 4.35 describe la inicialización del entorno gráfico. Se verifica si el NAS está montado; si no, se recurre a una carpeta local. En ambos casos se establece el directorio base, se carga la estructura en un `TreeView` con íconos MIME y se presenta la ventana principal. Si el directorio no existe, se muestra un error informativo al usuario.

- En la Figura 4.36, se modela el comportamiento al hacer doble clic sobre un elemento. Si se trata de un directorio válido, se navega hacia él. Si es un archivo, se determina su tipo MIME y se lanza la aplicación correspondiente: feh para imágenes .tif y '.jpg', YAD para archivos .txt, y el visor por defecto del sistema para otros tipos. El sistema ignora archivos fuera de los límites definidos.

- La Figura 4.37 detalla el menú contextual que aparece con clic derecho. Ofrece tres opciones: abrir (misma lógica que doble clic), descargar (con selector de destino y copia controlada) y eliminar (con cuadro de confirmación y borrado físico). En cada caso se valida si el usuario confirma o cancela la operación.

- Finalmente, la Figura 4.38 muestra el uso de botones de navegación. Al hacer clic, se verifica que la ruta asociada sea válida y esté dentro del árbol permitido. De ser así, se actualiza el directorio actual, se limpia el árbol y se cargan los nuevos archivos con sus respectivos íconos. En caso contrario, la acción se ignora para evitar errores de navegación.

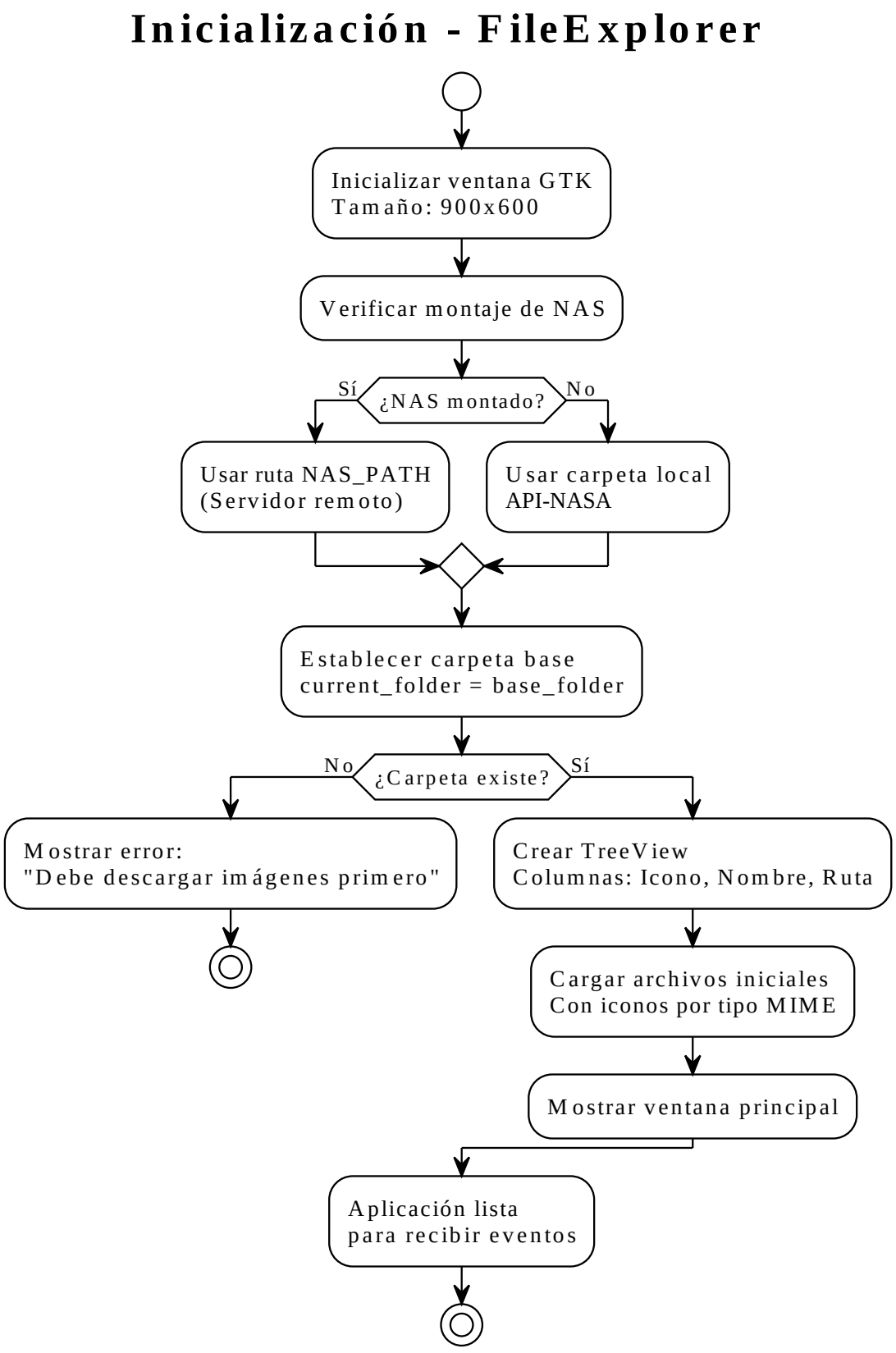

Figura 4.31: Inicialización del FileExplorer: configuración del entorno y carga del árbol de directorios.

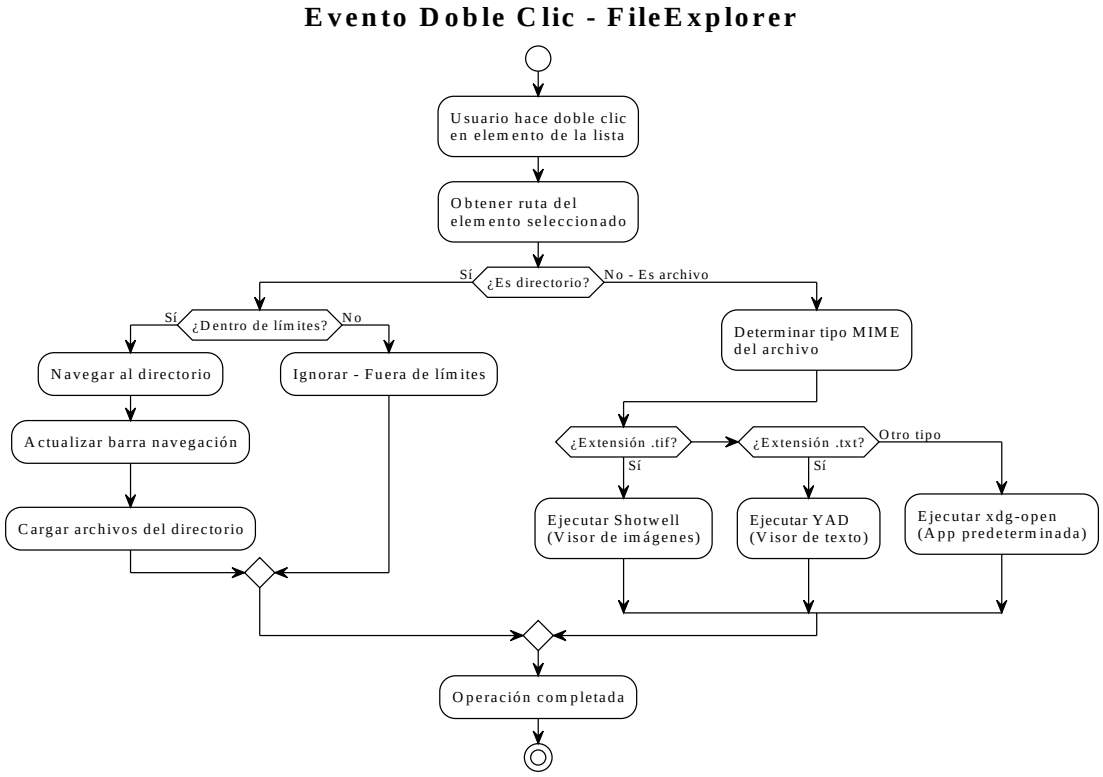

Figura 4.32: Interacción mediante doble clic: apertura de archivos según tipo MIME.

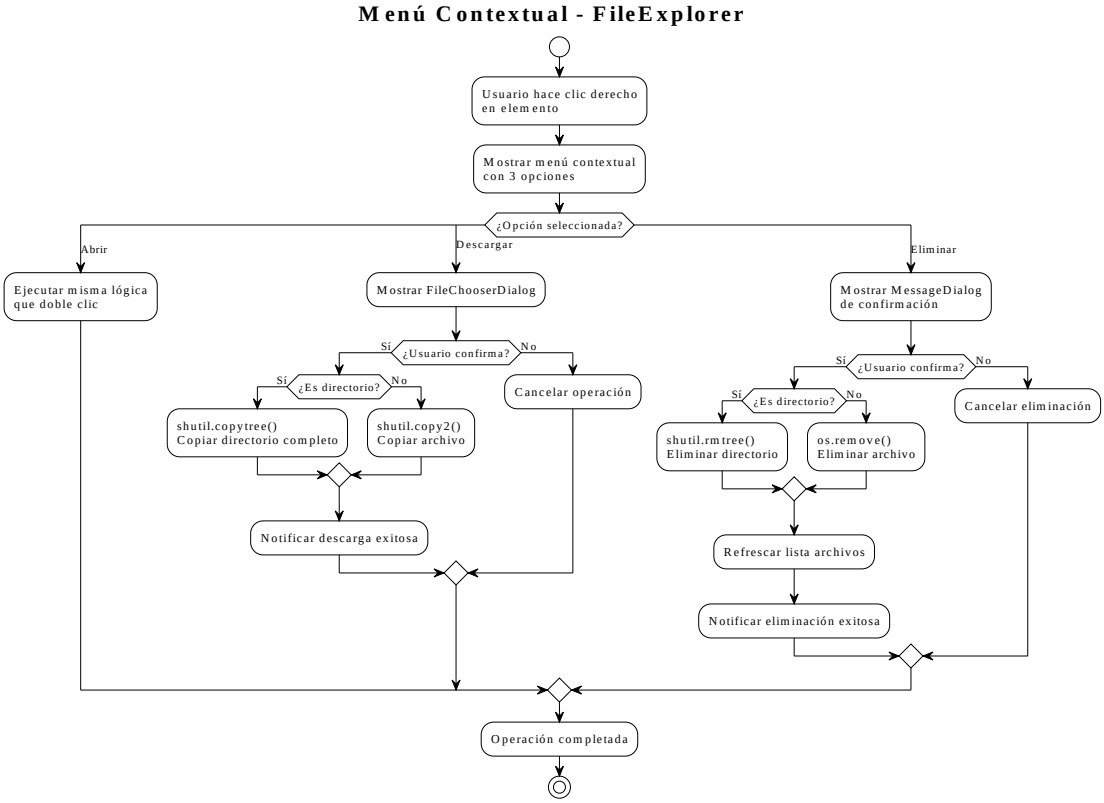

Figura 4.33: Operaciones del menú contextual: apertura, descarga y eliminación.

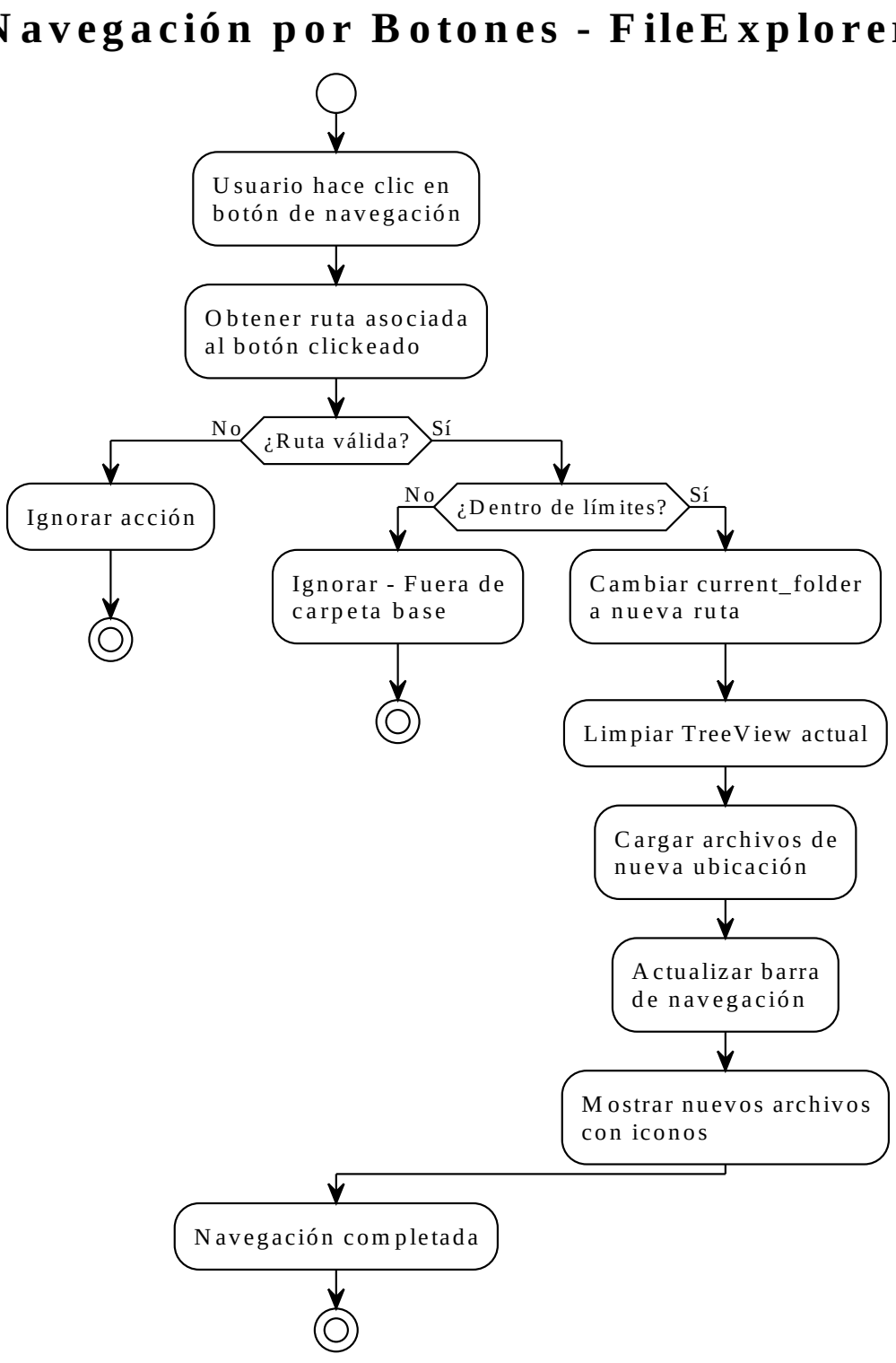

Figura 4.34: Navegación estructurada mediante botones y validación de rutas.

Diagramas de Actividad Para este componente se elaboraron diagramas que reflejan el comportamiento del sistema ante diferentes eventos, incorporando caminos alternativos y validaciones de entorno que aseguran una experiencia

robusta y coherente:

- La Figura 4.35 describe la inicialización del entorno gráfico con ventana GTK de 900x600 píxeles. El sistema primero verifica si el NAS está montado; de ser así, utiliza la ruta NAS_PATH del servidor remoto, caso contrario recurre a la carpeta local .ªPI-NASA". Se establece el directorio base y se verifica su existencia: si no existe, muestra el error "Debe descargar imágenes primero", de lo contrario crea el TreeView con columnas de ícono, nombre y ruta, carga los archivos iniciales con íconos por tipo MIME y presenta la ventana principal lista para recibir eventos.

- La Figura 4.36 modela el comportamiento al hacer doble clic sobre un elemento. Primero obtiene la ruta del elemento seleccionado y determina si es directorio o archivo. Para directorios, verifica que esté dentro de los límites permitidos; si es válido, navega al directorio, actualiza la barra de navegación y carga los archivos, caso contrario ignora la acción. Para archivos, determina el tipo MIME según la extensión: ejecuta Shotwell para archivos .tif, YAD para archivos .txt, y xdg-open para otros tipos de archivo.

- La Figura 4.37 detalla el menú contextual que aparece con clic derecho, ofreciendo tres opciones. Para Abrir, ejecuta la misma lógica que el doble clic. Para "Descargar", muestra FileChooserDialog y si el usuario confirma, utiliza shutil.copytree() para directorios o shutil.copy2() para archivos, notificando descarga exitosa. Para .ºiminar", presenta MessageDialog de confirmación y si se acepta, emplea shutil.rmtree() para directorios u os.remove() para archivos, seguido de refrescar la lista y notificar eliminación exitosa.

- La Figura 4.38 muestra el uso de botones de navegación. Al hacer clic, obtiene la ruta asociada al botón y verifica que sea válida y esté dentro de los límites de la carpeta base. De cumplir ambas condiciones, cambia current_folder a la nueva ruta, limpia el TreeView actual, carga archivos de la nueva ubicación, actualiza la barra de navegación y muestra los nuevos archivos con íconos. En caso contrario, ignora la acción para prevenir navegación fuera de límites.

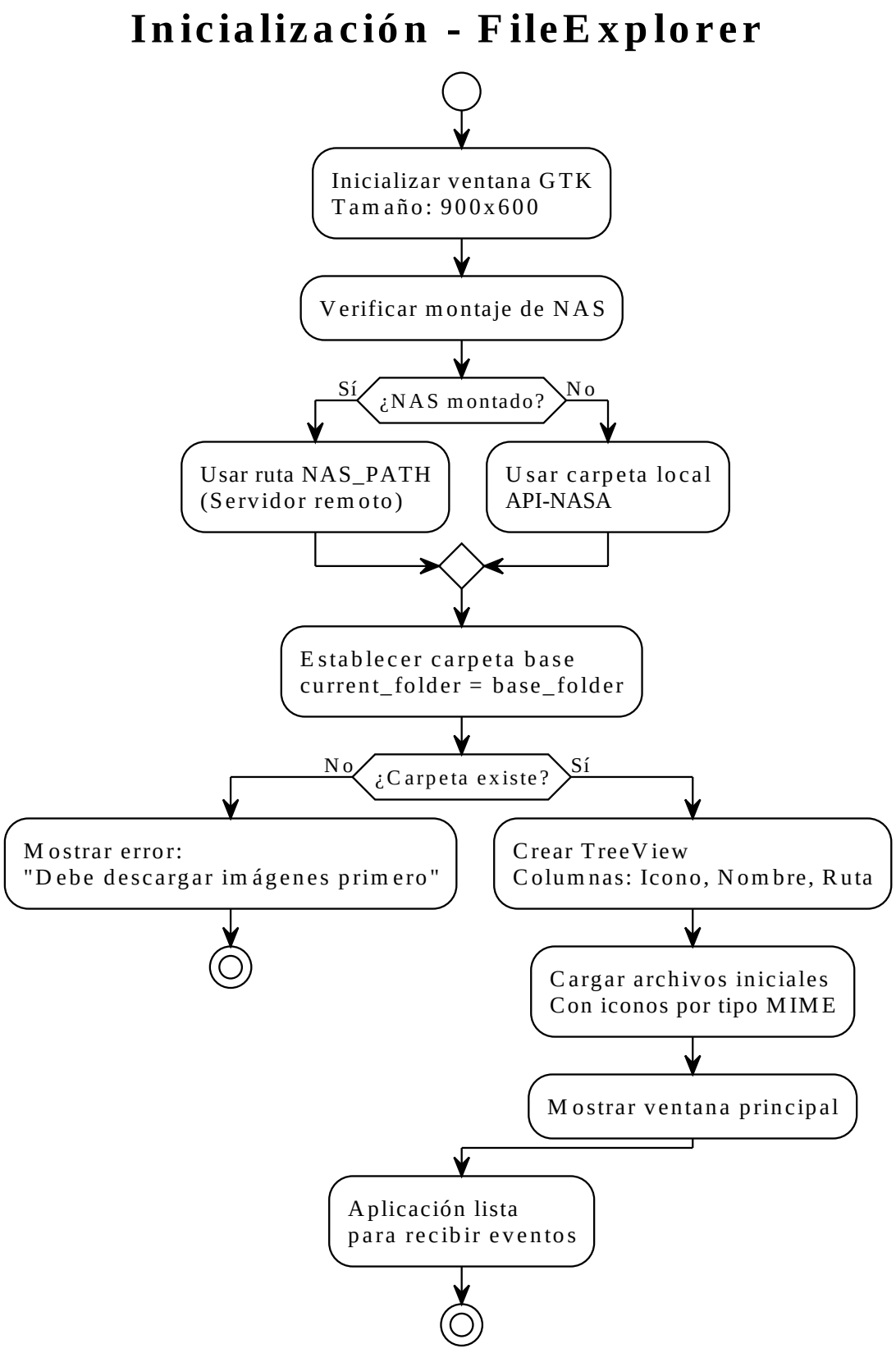

Figura 4.35: Inicialización del FileExplorer: configuración del entorno y carga del árbol de directorios.

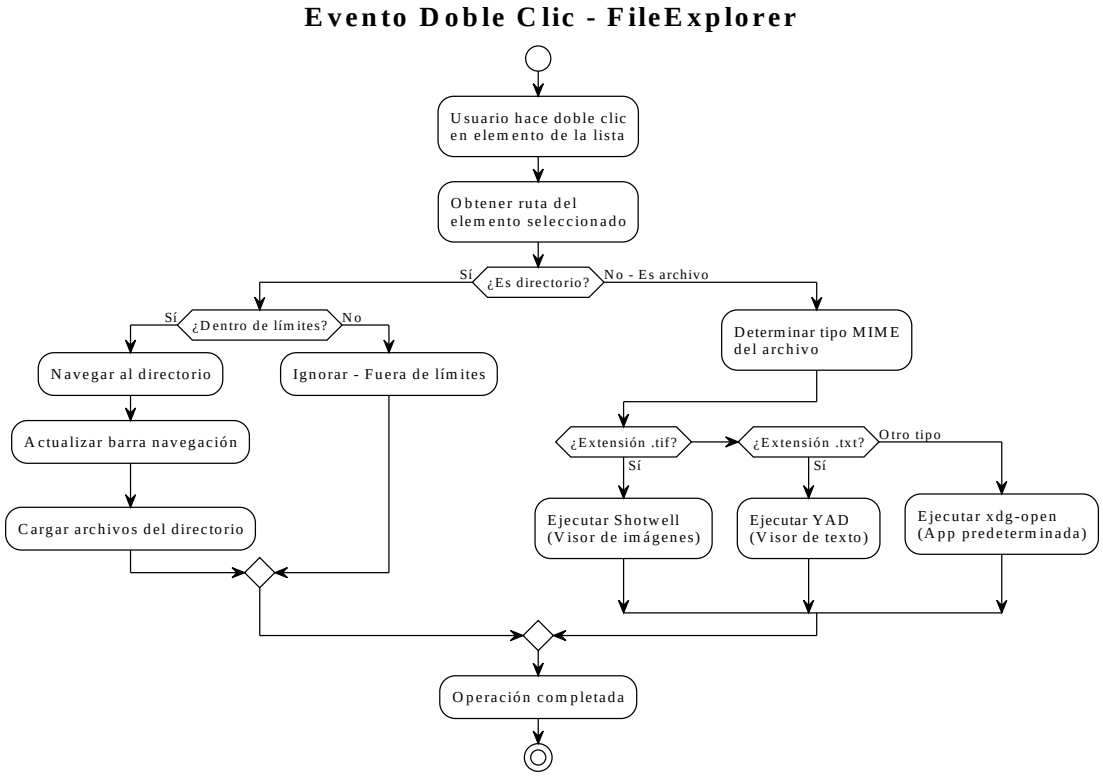

Figura 4.36: Interacción mediante doble clic: apertura de archivos según tipo MIME.

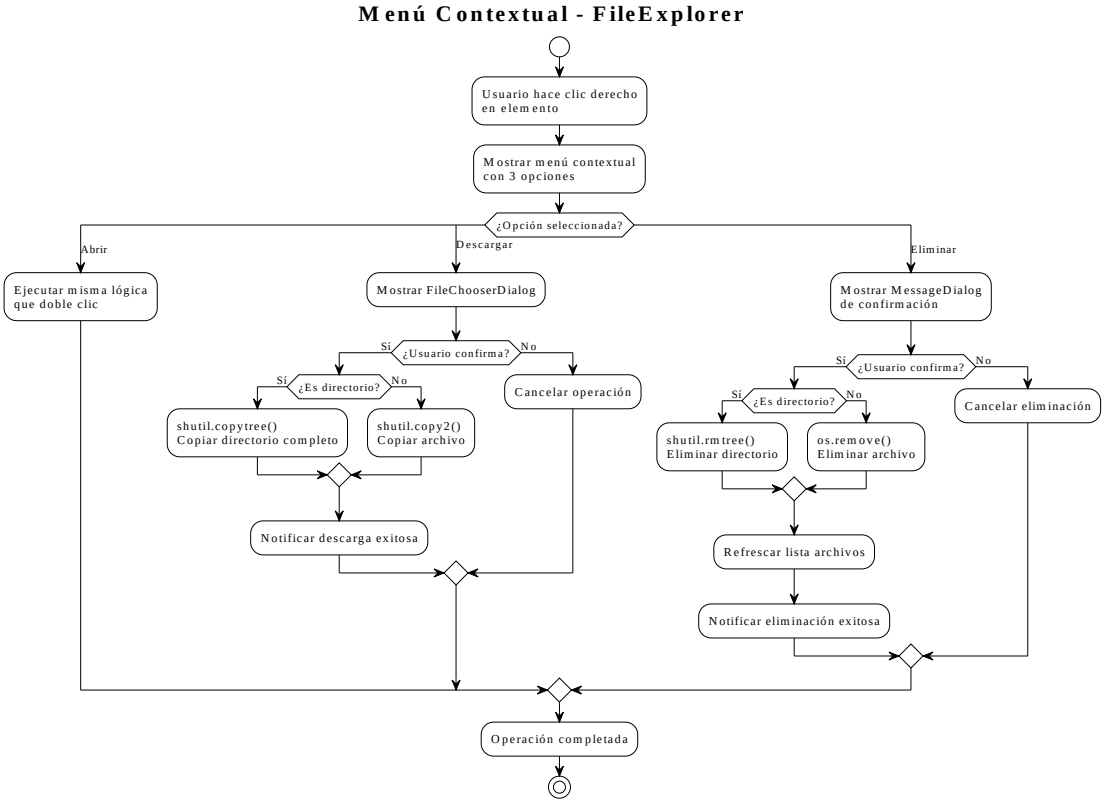

Figura 4.37: Operaciones del menú contextual: apertura, descarga y eliminación.

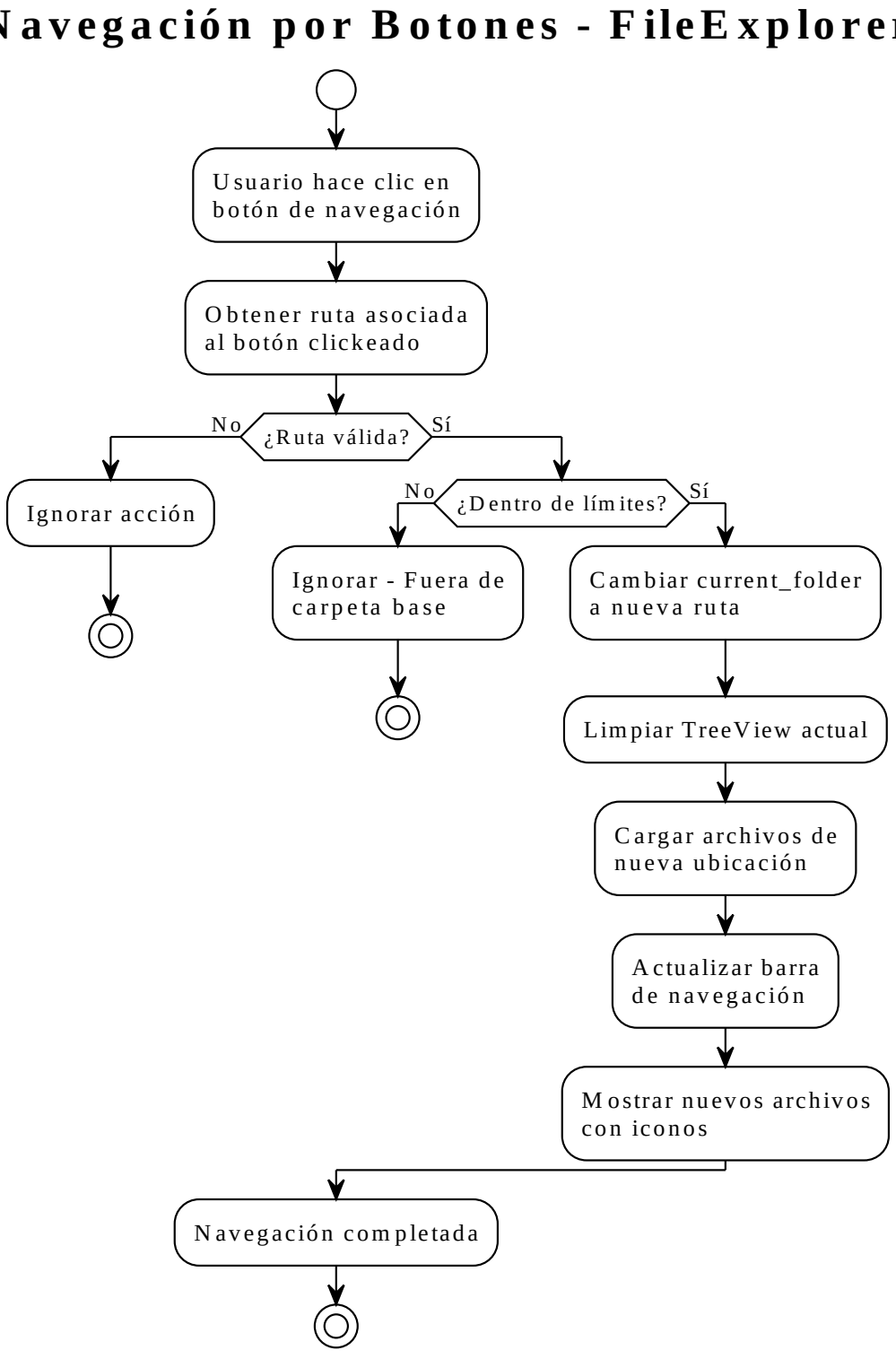

Figura 4.38: Navegación estructurada mediante botones y validación de rutas.

Diagrama de Secuencia La Figura 4.39 representa el flujo completo de interacción entre el usuario, el sistema de archivos (NAS o local) y el entorno GTK del explorador gráfico. El modelo incluye caminos alternativos y decisiones

críticas:

- El flujo inicia con la selección del Explorador Visual, tras lo cual se lanza la aplicación GTK y se valida si el entorno NAS está disponible. En caso negativo, se recurre a un directorio local.
- Posteriormente, el sistema solicita y muestra el contenido del directorio base. Si el usuario navega por carpetas, se repite este ciclo de consulta y renderizado jerárquico.
- Cuando el usuario realiza doble clic sobre una imagen, el sistema invoca la aplicación predeterminada según el tipo MIME. Para archivos ‘.tif‘ se usa Shotwell, y para ‘.txt‘, YAD. También se contempla la apertura con herramientas por defecto si el tipo no es reconocido específicamente.
- En caso de eliminación, se muestra un cuadro de confirmación. Si el usuario acepta, el archivo es borrado del sistema y la vista se actualiza. Si cancela, no ocurre ninguna modificación.
- Finalmente, la descarga de archivos también requiere interacción del usuario. Tras seleccionar “Descargar”, se abre un diálogo de guardado. El archivo es copiado a la ubicación deseada y se notifica el éxito.

Este diagrama no solo muestra el flujo principal, sino que documenta todas las posibles rutas de interacción, garantizando trazabilidad y control en operaciones sensibles sobre archivos reales.

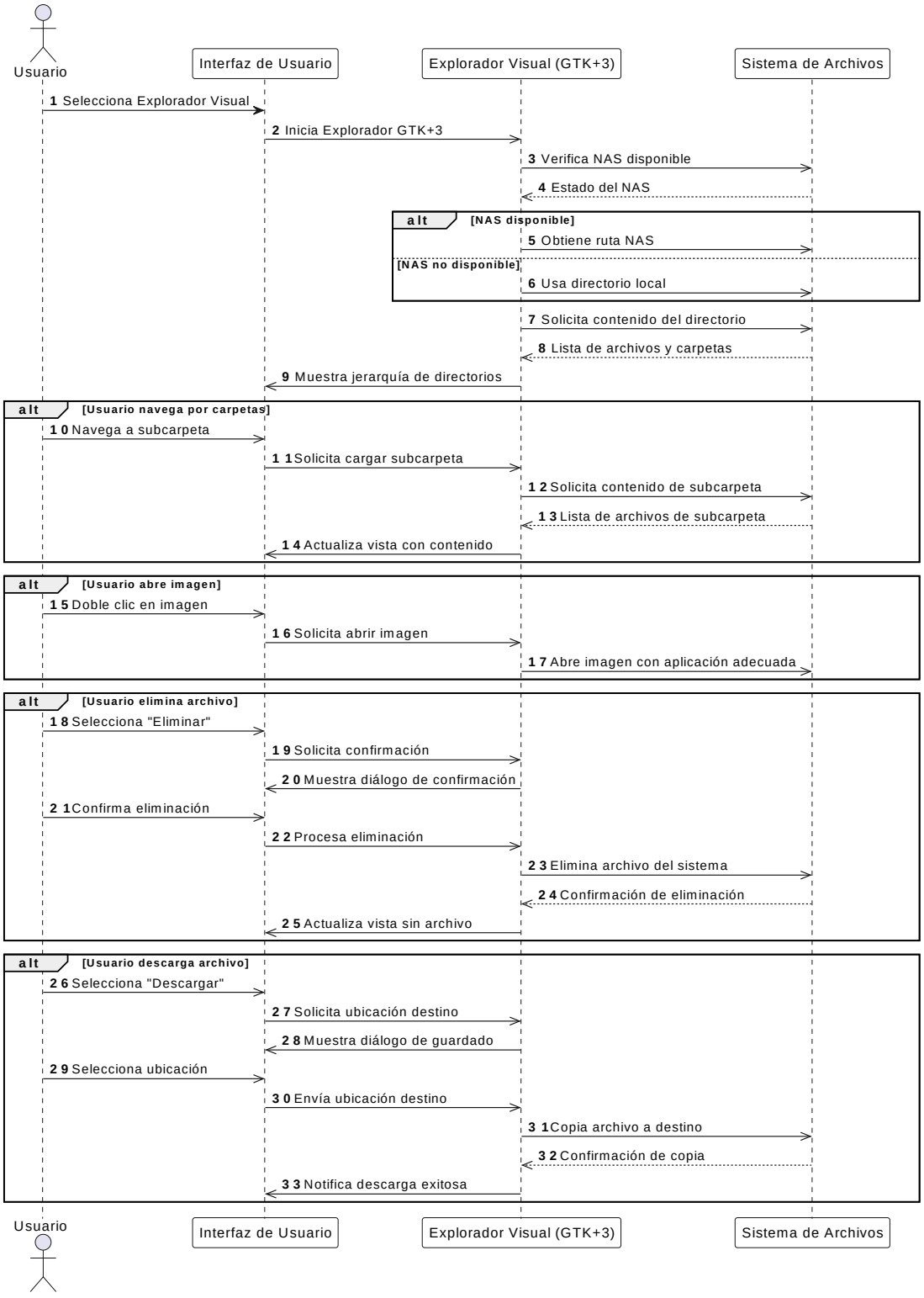

Figura 4.39: Secuencia de uso del explorador de imágenes: navegación, apertura y gestión de archivos.

Modelo de Clases

La arquitectura orientada a objetos del sistema se muestra en la Figura 4.40. Se observa una separación clara entre los módulos `FileExplorer` (gestión de archivos mediante GTK) y `DataTableApp` (visualización técnica mediante Textual), con responsabilidades bien definidas y puntos de integración controlados. Esta modularidad favorece el mantenimiento, la prueba individual de componentes y la reutilización.

- **DataTableApp** es responsable de mostrar y manipular metadatos satelitales. Implementa métodos como `action_open_image()`, `action_export_csv()`, `action_delete_image()` y `refresh_table()`, que encapsulan la lógica de interacción y coordinan acciones sobre la base de datos, garantizando una experiencia fluida y controlada.

- **FileExplorer** gestiona la exploración visual del sistema de archivos. A través de métodos como `load_files()`, `on_double_click()` y `delete_file()`, permite operar sobre el entorno NAS o una carpeta local, con soporte para descarga, eliminación y apertura de archivos según su tipo MIME.

- **Metadata (View)** representa una vista consolidada que expone, en una única estructura, todos los atributos técnicos, espaciales y contextuales de cada imagen. Esta vista se deriva del modelo entidad-relación definido en el Incremento 1 (Figura 4.6), donde se normalizaron las entidades `Image`, `ImageDetails`, `MapLocation` y `CameraInformation`. La vista actúa como una proyección lógica de esas tablas, diseñada para optimizar el acceso y la lectura intensiva de datos por parte del módulo técnico.

Esta división funcional entre navegación gráfica y análisis técnico se traduce en una arquitectura desacoplada y extensible, alineada con principios de diseño limpio y buenas prácticas de ingeniería de software.

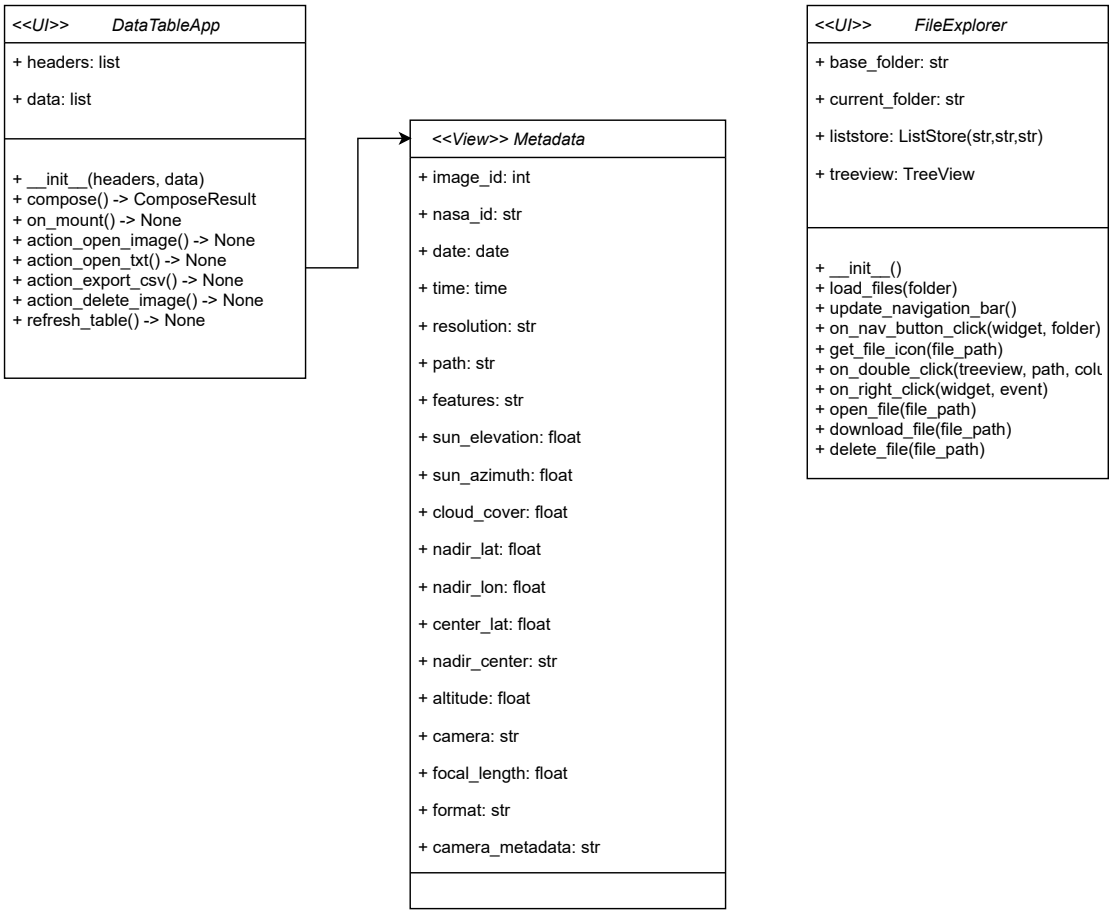

Figura 4.40: Modelo de clases del Incremento 2: separación entre navegación gráfica y vista técnica.

4.3.4 Fase 3: Implementación

La implementación se dividió en dos módulos principales: el explorador gráfico basado en GTK y el visualizador técnico en consola desarrollado con Textual. Ambos fueron integrados al sistema principal sin introducir acoplamientos innecesarios, manteniendo la modularidad definida en la arquitectura.

Explorador GTK El módulo gráfico, implementado en `explorador.py`, ofrece una interfaz jerárquica que permite al usuario explorar imágenes clasificadas por año, sensor y misión. El menú contextual habilita acciones directas sobre archivos: visualización con el visor predeterminado, descarga local mediante diálogo de destino y eliminación segura tanto en el NAS como en la base de datos. La Figura 4.41 muestra la interfaz activa.

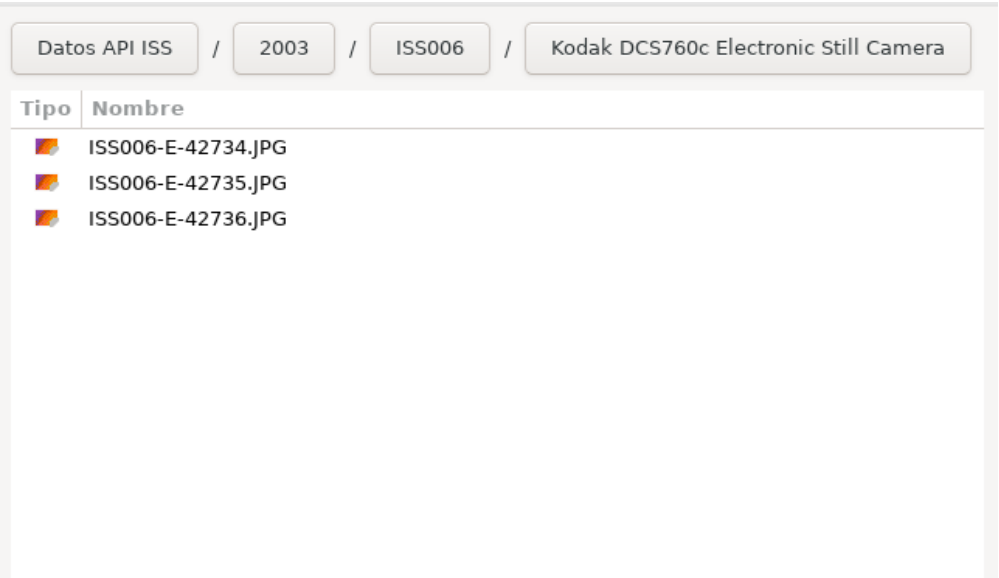

Figura 4.41: Explorador GTK para navegación jerárquica del entorno NAS.

Visualizador Técnico en Consola Desarrollado con la biblioteca `textual`, este componente permite la interacción directa con los metadatos mediante una tabla interactiva que soporta paginación, filtrado dinámico, exportación y operaciones por fila (visualización, eliminación y cambio de fuente de datos). La Figura 4.42 presenta un ejemplo de su ejecución.

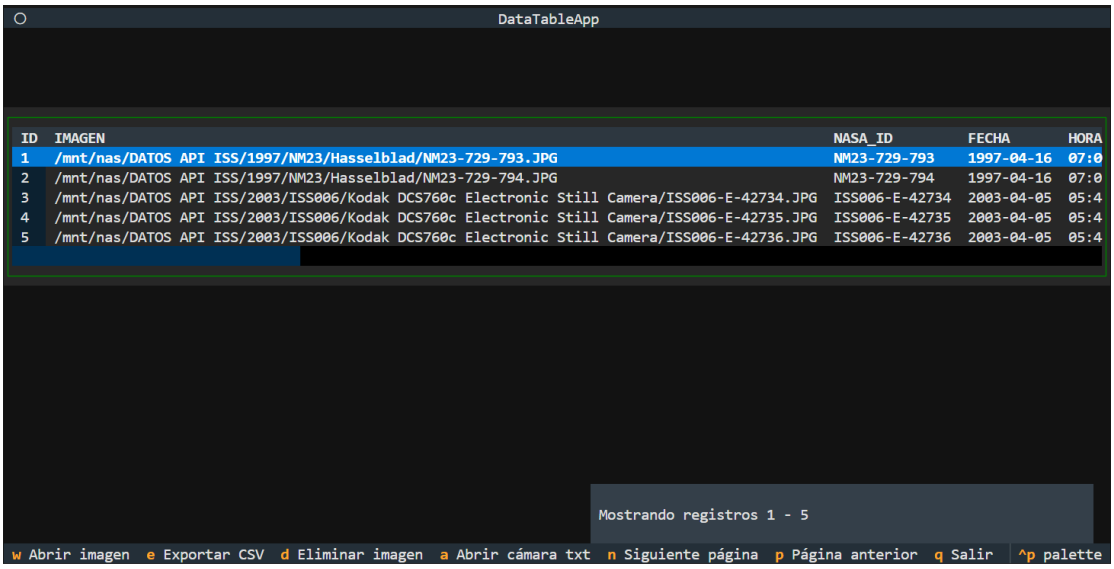

Figura 4.42: Vista técnica en consola: acceso y gestión de metadatos con soporte para múltiples fuentes.

Este módulo opera directamente sobre la vista ‘Metadatos‘ de la base SQLite, manteniendo sincronía con el sistema de archivos físico y permitiendo acciones

eficientes en grandes volúmenes de datos.

4.3.5 Fase 4: Pruebas

Ambos componentes fueron validados en entornos Linux reales, incluyendo distribuciones bajo WSL2, utilizando estructuras de prueba con más de 1000 archivos. Las pruebas se centraron en evaluar la estabilidad, funcionalidad y eficiencia en situaciones de uso intensivo.

Explorador GTK Se evaluó el comportamiento del explorador bajo distintas condiciones de carga:

- Navegación fluida en estructuras anidadas de carpetas.
- Correcta ejecución de acciones desde el menú contextual.
- Detección automática de entorno NAS y fallback al sistema local.
- Manejo de errores ante rutas inexistentes o archivos eliminados.

Visualizador Técnico Las pruebas de la vista técnica incluyeron:

- Carga y navegación estable de registros paginados.
- Visualización directa de archivos relacionados (.tif, .txt).
- Sincronización entre acciones destructivas (eliminación) y la base de datos.
- Exportación de registros a CSV sin pérdida de atributos ni formato.

Análisis de Rendimiento y Eficiencia

Durante las pruebas se recopilaron métricas clave que permiten evaluar el comportamiento de ambos módulos frente a grandes volúmenes de información. La Tabla 4.6 resume los resultados más relevantes.

Tabla 4.6: Métricas de rendimiento del explorador visual y vista técnica

Métrica	**Valor**
Tiempo promedio de carga inicial (Explorador GTK)	1.4 segundos
Tiempo de respuesta en navegación entre carpetas	0.6 segundos
Tiempo de carga para visualización de imágenes	0.8 segundos
Tiempo promedio de consulta a metadatos (vista técnica)	0.9 segundos con carga de 2240 imágenes
Tiempo de exportación CSV (1000 registros)	0.7 segundos

Estas métricas confirman que ambos componentes mantienen un rendimiento óptimo incluso ante estructuras con alta densidad de archivos y consultas complejas.

4.3.6 Fase 5: Resultados del Incremento

Logros Alcanzados

El Incremento 2 logró cumplir satisfactoriamente todos los objetivos planteados. Se destacan los siguientes resultados concretos:

- Exploración eficiente del entorno NAS mediante una interfaz gráfica jerárquica, intuitiva y estable.
- Consulta técnica robusta de metadatos con herramientas interactivas y exportación avanzada.
- Integración limpia y modular con el sistema base, sin generar acoplamientos indeseados.
- Operaciones CRUD extendidas, validadas y sincronizadas con la base de datos y el sistema de archivos.

Limitaciones Técnicas Identificadas

Durante la implementación se identificaron las siguientes limitaciones:

- **Rendimiento en carpetas extensas:** Reducción de fluidez en el explorador GTK al superar los 5000 archivos por carpeta.
- **Compatibilidad de visualización:** Formatos TIFF con compresión especializada requieren visores externos.

Estrategias de Recuperación ante Fallos

Se implementaron mecanismos de robustez ante fallos operativos:

- **Validación de rutas:** Verificación previa de existencia y permisos antes de acceder a cualquier carpeta o archivo.
- **Carga progresiva:** Evita bloqueos en estructuras pesadas permitiendo interacción durante la carga.
- **Manejo de errores de visualización:** Detección de errores en archivos corruptos con fallback automático.
- **Cache de navegación:** Preserva la posición del usuario tras errores o reinicios inesperados.
- **Transacciones atómicas:** Operaciones críticas (como eliminación) se ejecutan bajo transacciones seguras.

4.4 Incremento 3: Consulta Avanzada vía API NASA y Organización de Archivos

4.4.1 Descripción General

Este incremento marcó la transición de un enfoque basado en web scraping hacia una arquitectura más robusta y escalable, fundamentada en el uso directo de la API oficial *NASA Gateway PhotosDatabaseAPI*. El objetivo fue

habilitar consultas programáticas avanzadas, filtrado geográfico y temporal, y la recuperación estructurada de metadatos técnicos de imágenes captadas desde la Estación Espacial Internacional (ISS).
Asimismo, se integraron procesos automáticos para enriquecer los resultados con información adicional obtenida de fuentes complementarias, incluyendo metadatos de cámara y altitud orbital. Se implementaron mecanismos para el control de duplicados mediante verificación del id de la imágen, y un sistema de renombrado estandarizado. Este flujo estableció la base para futuras automatizaciones y tareas periódicas.

4.4.2 Fase 1: Requisitos

Tabla 4.7: Requerimientos funcionales abordados en el incremento 3

ID	Descripción
RF10	Conectarse a la API NASA Gateway para ejecutar consultas filtradas por misión, sensor, hora y región geográfica.
RF11	Renombrar automáticamente los archivos descargados siguiendo una convención estandarizada: `[año]_[misión]_[cámara]_[nasa_id].jpg`.
RF12	Detectar y excluir imágenes duplicadas mediante comparación con la base de datos local (validación por `nasa_id`).

Requerimientos Funcionales

Tabla 4.8: Requerimientos no funcionales abordados en el incremento 3

ID	Descripción
RNF9	Mantener compatibilidad con la estructura jerárquica de almacenamiento definida en incrementos previos.
RNF10	Asegurar nombres de archivo consistentes, legibles y únicos para facilitar búsquedas posteriores.
RNF11	Garantizar la tolerancia a fallos en las llamadas a la API, con reintentos automáticos y logs estructurados.

Requerimientos No Funcionales

4.4.3 Fase 2: Diseño del Sistema

Durante la fase 2 del desarrollo del sistema, se abordó el diseño funcional mediante la elaboración de diagramas de casos de uso, diagramas de actividades y de secuencia correspondientes al tercer incremento del proyecto. Este incremento se centró principalmente en la búsqueda y descarga automatizada de imágenes nocturnas desde la API oficial de la Estación Espacial Internacional (ISS). Cabe destacar que, a nivel arquitectónico, se reutilizó el backend definido en el Incremento 1, específicamente los módulos de procesamiento, descarga y organización de imágenes (ver Figura 4.5) normalización, verificación y persistencia de metadatos (ver Figura 4.6).

Diagramas de Casos de Uso

Los diagramas de casos de uso modelan las funcionalidades principales ofrecidas al usuario en esta etapa del sistema. En la Figura 4.43, se ilustra la interacción del usuario con el sistema para configurar filtros de búsqueda, visualizar resultados sobre un mapa interactivo, activar el modo nocturno y descargar imágenes seleccionadas. El caso de uso se extiende a funciones adicionales como la paginación y la selección de la fuente de coordenadas (frames, nadir, mlcoord), lo que permite personalizar el análisis geoespacial.
Por otro lado, la Figura 4.44 agrupa las operaciones asociadas a la automatización. El usuario puede crear tareas programadas, establecer horarios de ejecución, definir frecuencias e incluso exportar configuraciones. Este diseño amplía la funcionalidad del sistema más allá del uso manual, permitiendo operaciones desatendidas.

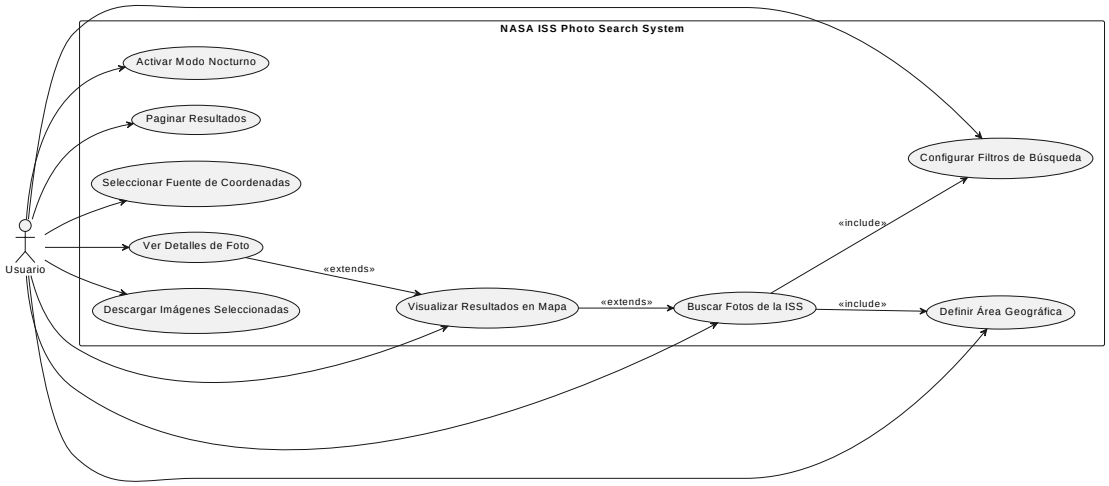

Figura 4.43: Casos de uso del incremento 3: búsqueda de imágenes ISS.

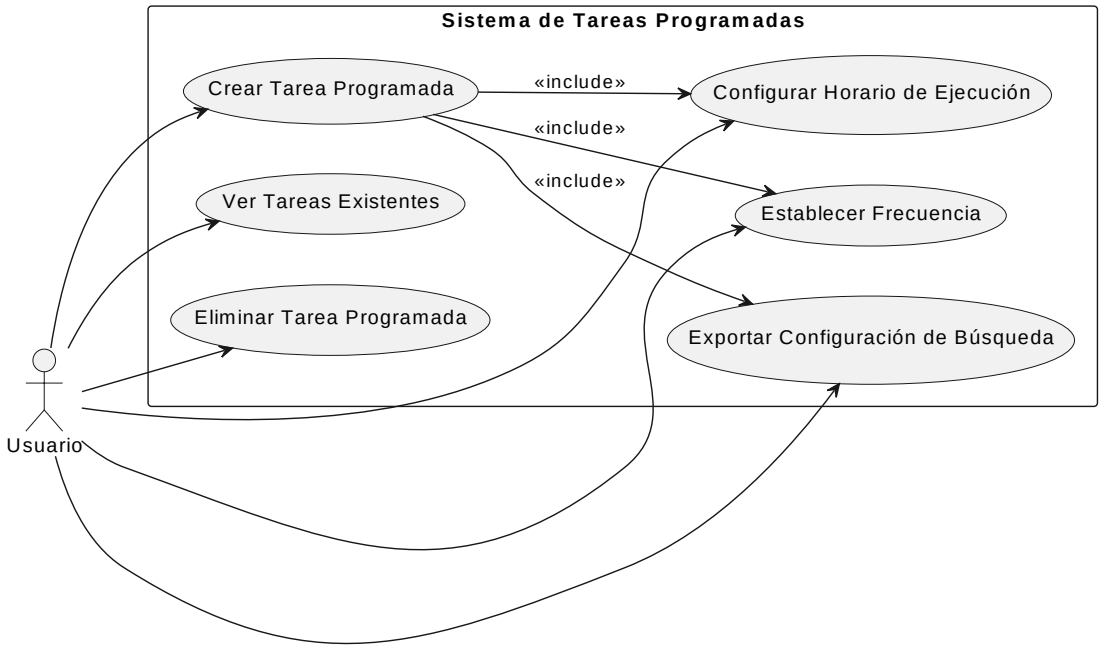

Figura 4.44: Casos de uso del incremento 3: gestión de tareas programadas.

Diagramas de Actividad

Los diagramas de actividad describen los flujos lógicos implementados en el sistema web, contemplando validaciones, procesamientos paralelos y manejo de errores:

- La Figura 4.45 modela la configuración de filtros y consulta a la NASA API. El proceso inicia con la configuración de filtros por parte del usuario,

seguida de validación del sistema que muestra errores si los filtros no son válidos. Una vez validados, se aplican campos por defecto y se limpia el mapa anterior. El sistema procesa tres fuentes en paralelo: "frames", "nadirz "mlcoord". Para cada fuente que soporte filtros de tiempo nocturno (ptime), se agregan automáticamente si el modo nocturno está activado. Se construye la consulta y parámetros de la API, se ejecuta la llamada y se procesan los datos de respuesta normalizando campos y generando URLs de preview. El sistema verifica duplicados en base de datos: si ya existe, ignora el resultado; si es nuevo, lo agrega a los resultados. Finalmente combina todos los resultados, los muestra en el panel y crea marcadores en el mapa aplicando clustering.

- La Figura 4.46 representa el proceso de descarga inmediata de imágenes. El flujo inicia cuando el usuario hace clic en "Descargar Ahora", verificando primero que existan resultados disponibles; si no los hay, muestra mensaje de error. Una vez confirmados los resultados, inicializa el contador de tiempo y prepara el progreso de descarga. En la fase de preparación de metadatos, procesa los primeros 5 resultados extrayendo información básica, obteniendo datos NADIR/Altitud y descargando metadatos de cámara. Combina todos los metadatos y guarda el archivo JSON correspondiente. Posteriormente inicia la barra de progreso e invoca el proceso Python de descarga, recibiendo actualizaciones continuas del progreso e incrementando el contador. Una vez finalizada la descarga, detiene el contador de tiempo y muestra mensaje de éxito.

- La Figura 4.47 detalla la programación de tareas automatizadas mediante el Programador de Tareas de Windows. El proceso inicia cuando el usuario abre el modal de configuración de cron, especifica hora de inicio, selecciona frecuencia y, si la misma lo requiere, también especifica el intervalo numérico. Tras confirmar la creación, se valida la entrada mostrando errores si es necesario. Si hay resultados para exportar, procede con la preparación de tarea generando un ID único, construyendo la query de búsqueda y los parámetros de retorno, y creando el objeto de

tarea. En la fase de persistencia, lee el archivo tasks.json existente, agrega la nueva tarea y escribe el archivo actualizado. Para la programación del sistema, construye el comando WSL y configura los argumentos del scheduler según el tipo de frecuencia: Una vez para ejecución única, *Diaria* para repetición diaria, *Semanal* para cada semana, o *HORA/MINUTO* para repetición por intervalo. Finalmente ejecuta schtasks.exe y notifica el resultado: mensaje de éxito si la tarea se creó exitosamente, o mensaje de error en caso contrario.

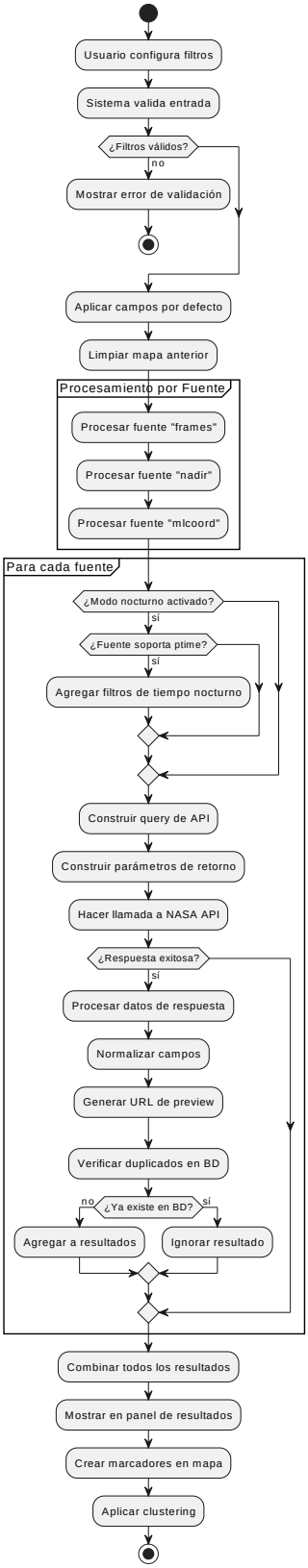

Figura 4.45: Diagrama de actividad: configuración y envío de filtros a la API.

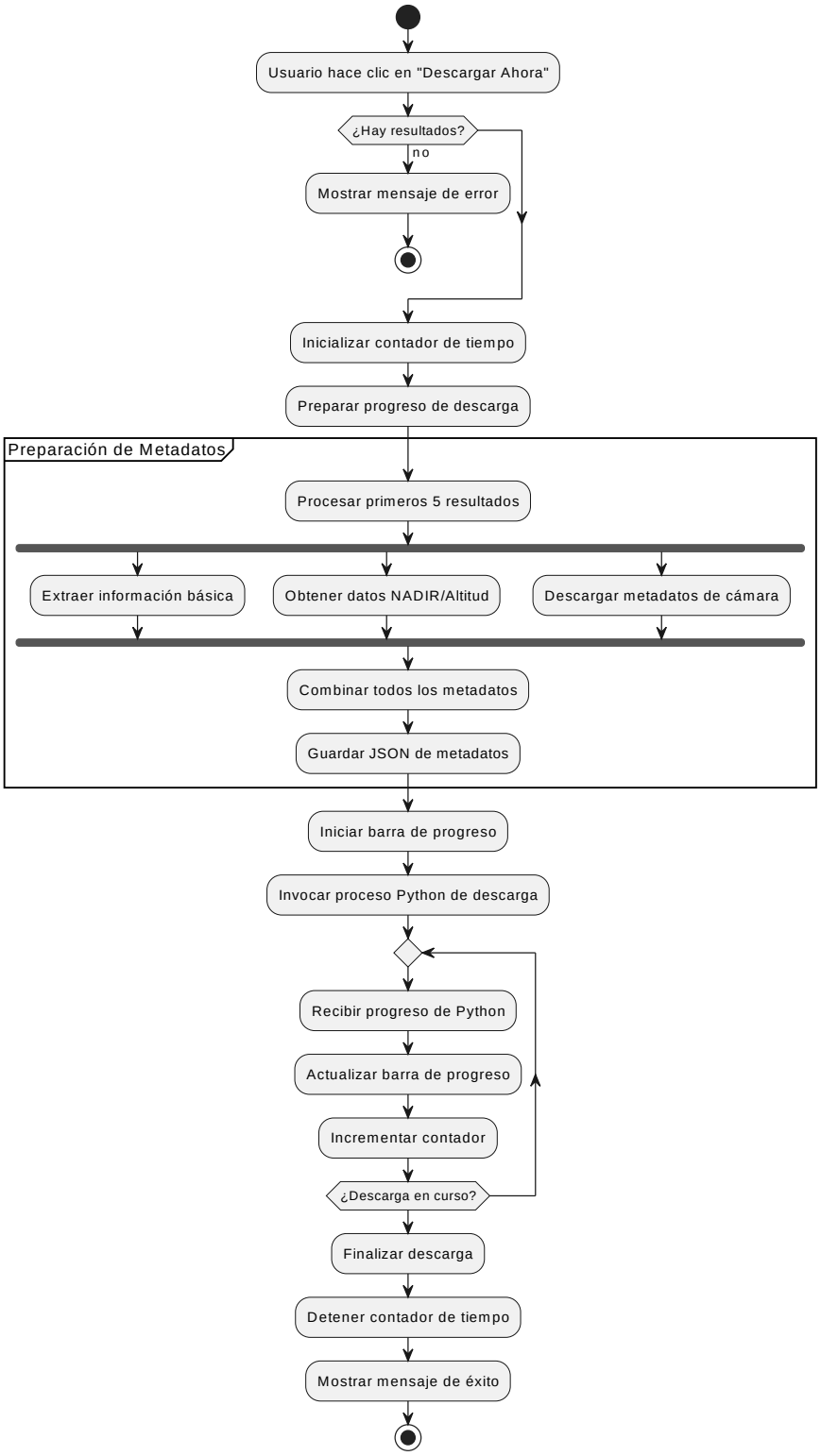

Figura 4.46: Diagrama de actividad: descarga de imágenes y procesamiento de metadatos.

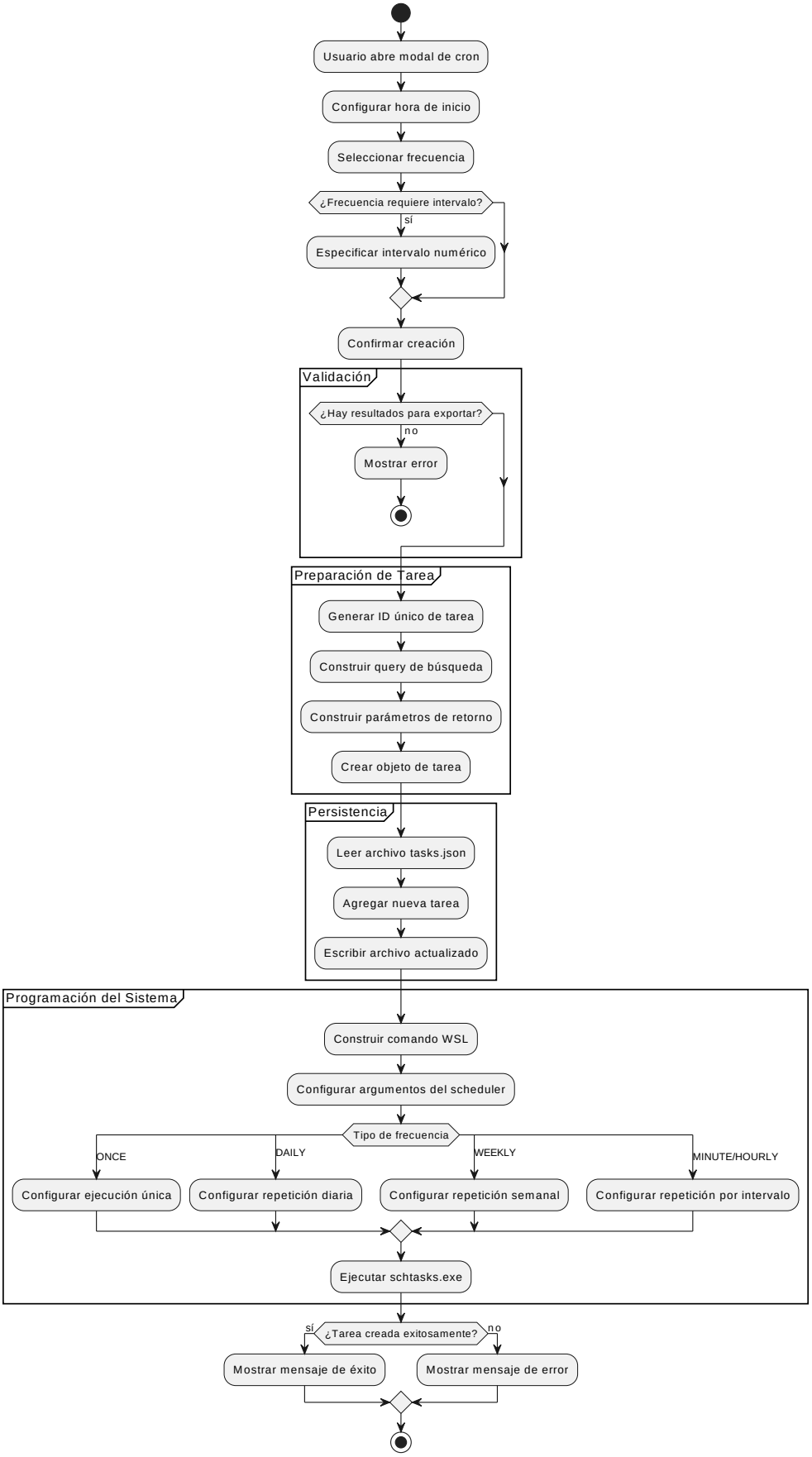

Figura 4.47: Diagrama de actividad: programación de tareas automáticas.

Diagramas de Secuencia

Los diagramas de secuencia muestran la interacción detallada entre los componentes del sistema, incorporando tanto rutas principales como caminos alternativos con validaciones y manejo de errores:

- La Figura 4.48 representa el flujo completo de búsqueda con filtros definidos. El proceso inicia cuando el usuario ejecuta fetchData() en la Vue App, la cual valida los filtros de entrada y aplica campos por defecto mediante aplicarCamposDefault(). Posteriormente limpia el mapa actual con clearMap() y procede a iterar sobre las tres fuentes disponibles: frames, nadir y mlcoord. Para cada fuente, el queryBuilder construye la consulta específica con buildQuery(filtros, fuente, boundingBox) generando el queryString correspondiente, y construye los parámetros de retorno con buildReturn(returnFields, fuente). La Vue App ejecuta fetch(apiUrl) hacia la NASA API, recibiendo los datos en crudo (rawData[]). Para cada foto en los resultados, normaliza los campos, obtiene la URL de preview mediante getImageUrl(photo), y verifica la existencia en base de datos consultando el NASA_ID. Si la imagen no existe previamente, la agrega a los resultados; si ya existe, ignora el resultado para evitar duplicados. Finalmente, addMarkersToMap(map, results) coloca los marcadores y clusterGroup en el mapa, y muestra los resultados al usuario.

- La Figura 4.49 modela el proceso completo de configuración y ejecución de tareas programadas. El flujo inicia cuando el usuario ejecuta abrirModalCron(), mostrando el modal de configuración en la Vue App. Una vez configurada la tarea, el usuario confirma con confirmarCrearCron(config), enviando los parámetros a taskManager. Este componente utiliza queryBuilder para construir la consulta con buildQuery(filtros) obteniendo el queryString, y construye los campos de retorno con buildReturn(returnFields) generando el returnString. El taskManager crea un objeto nuevaTarea con toda la configuración, lee el archivo tasks.json existente del Sistema de Archivos para obtener las tareas previas, agrega la nueva tarea a la lista y escribe el tasks.json

actualizado, confirmando el éxito de la operación. Posteriormente, crea la tarea en Windows Task Scheduler, recibe la respuesta del sistema y envía el mensaje de éxito o error correspondiente a la Vue App para mostrar el resultado al usuario.

- La Figura 4.50 muestra la extracción detallada de metadatos desde el sitio web de la NASA mediante web scraping. El proceso inicia cuando la Vue App solicita obtenerCamaraMetadata(nasaId) al componente NASA Web. Este construye la URL de la foto específica y utiliza axios.get(photoUrl) para obtener el HTML de la página completa. Una vez recibido el HTML, utiliza cheerio.load(html) para crear el objeto que permite manipular el DOM. Busca específicamente el botón de metadata en la página y, si el elemento botón existe, extrae la URL del atributo onclick, construye la URL completa de metadatos y ejecuta axios.get(metadataUrl) para descargar el contenido de metadatos. Posteriormente verifica en el Sistema de Archivos si el archivo ya existe localmente; si no existe o el contenido es diferente al actual, utiliza writeFileSync(metadata) para sobrescribir el archivo localmente, confirma el éxito de la operación y retorna el path del archivo a la Vue App.

Reutilización de Componentes del Incremento 1 Es importante destacar que para las fases de procesamiento, descarga, guardado en base de datos y organización de archivos, este incremento reutiliza completamente la arquitectura desarrollada en el incremento 1 (ver Sección 4.2.3). Específicamente, se aprovechan las clases ImageProcessor, MetadataCRUD y las entidades del dominio (Image, ImageDetails, MapLocation, CameraInformation) descritas en las Figuras 4.5 y 4.3, garantizando consistencia en el almacenamiento y evitando duplicación de código entre incrementos.

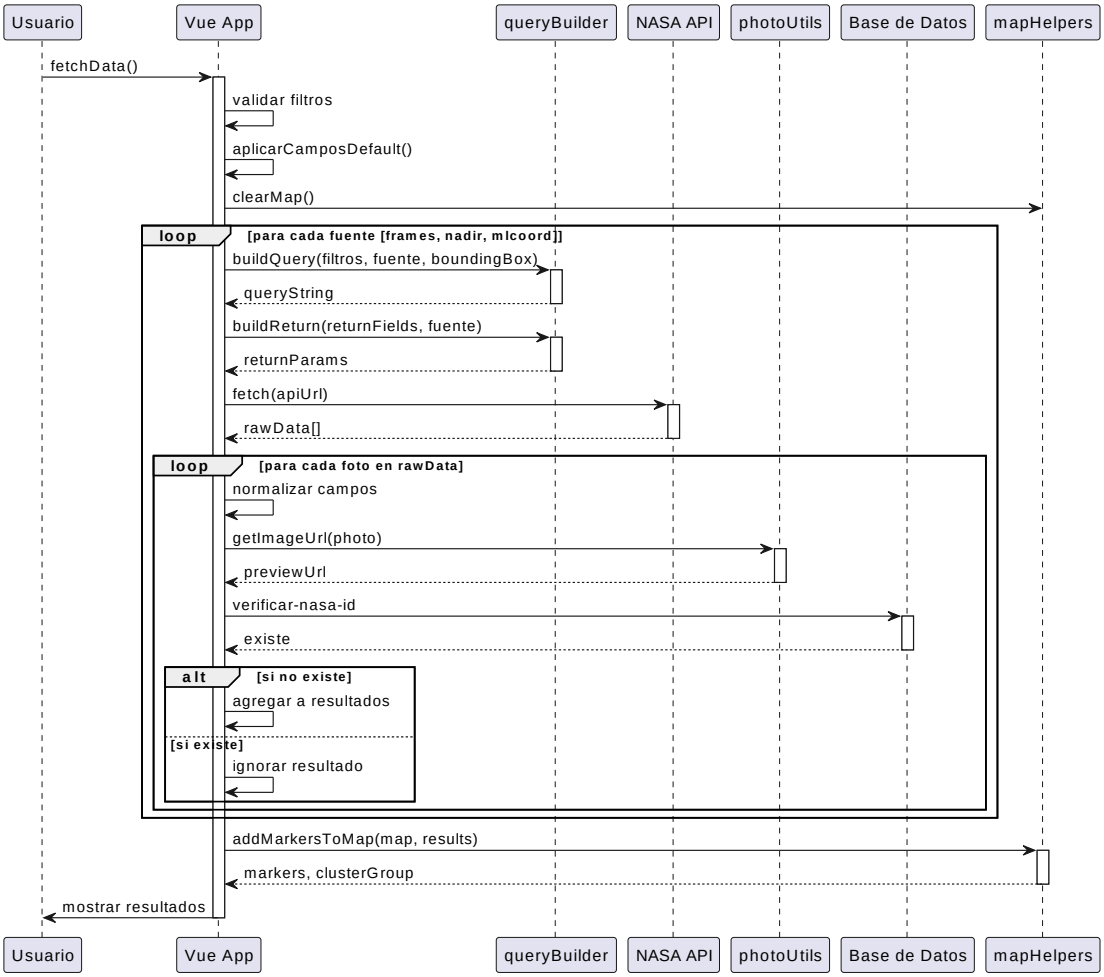

Figura 4.48: Diagrama de secuencia: búsqueda de imágenes con filtros definidos.

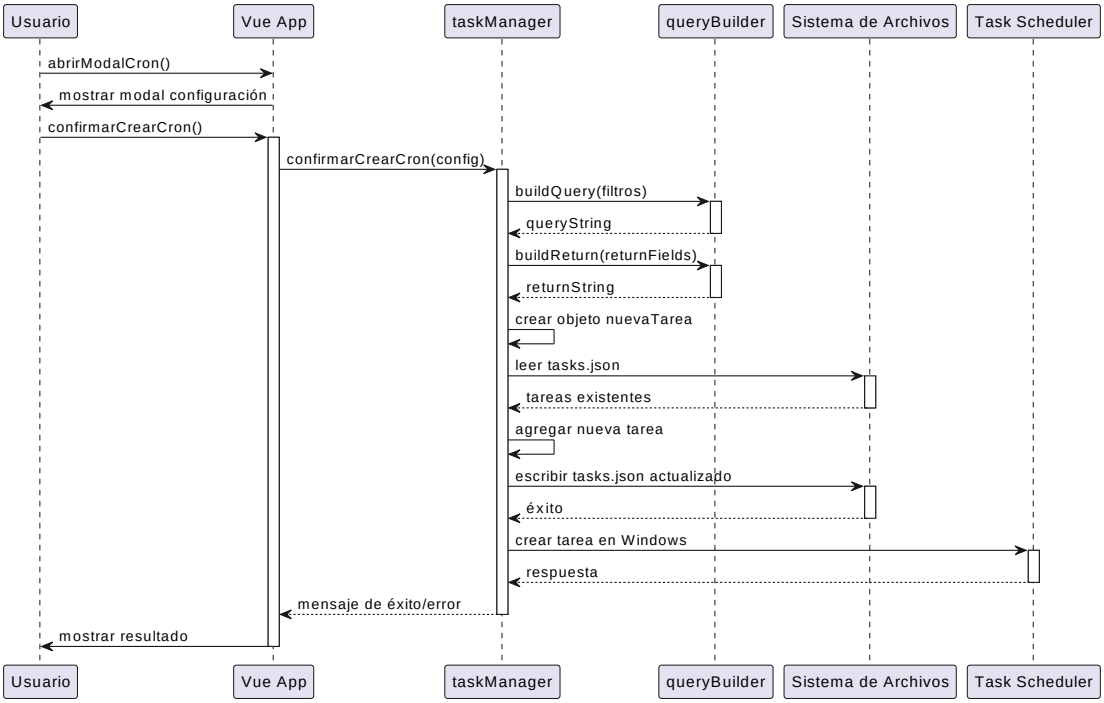

Figura 4.49: Diagrama de secuencia: configuración y ejecución de tareas programadas.

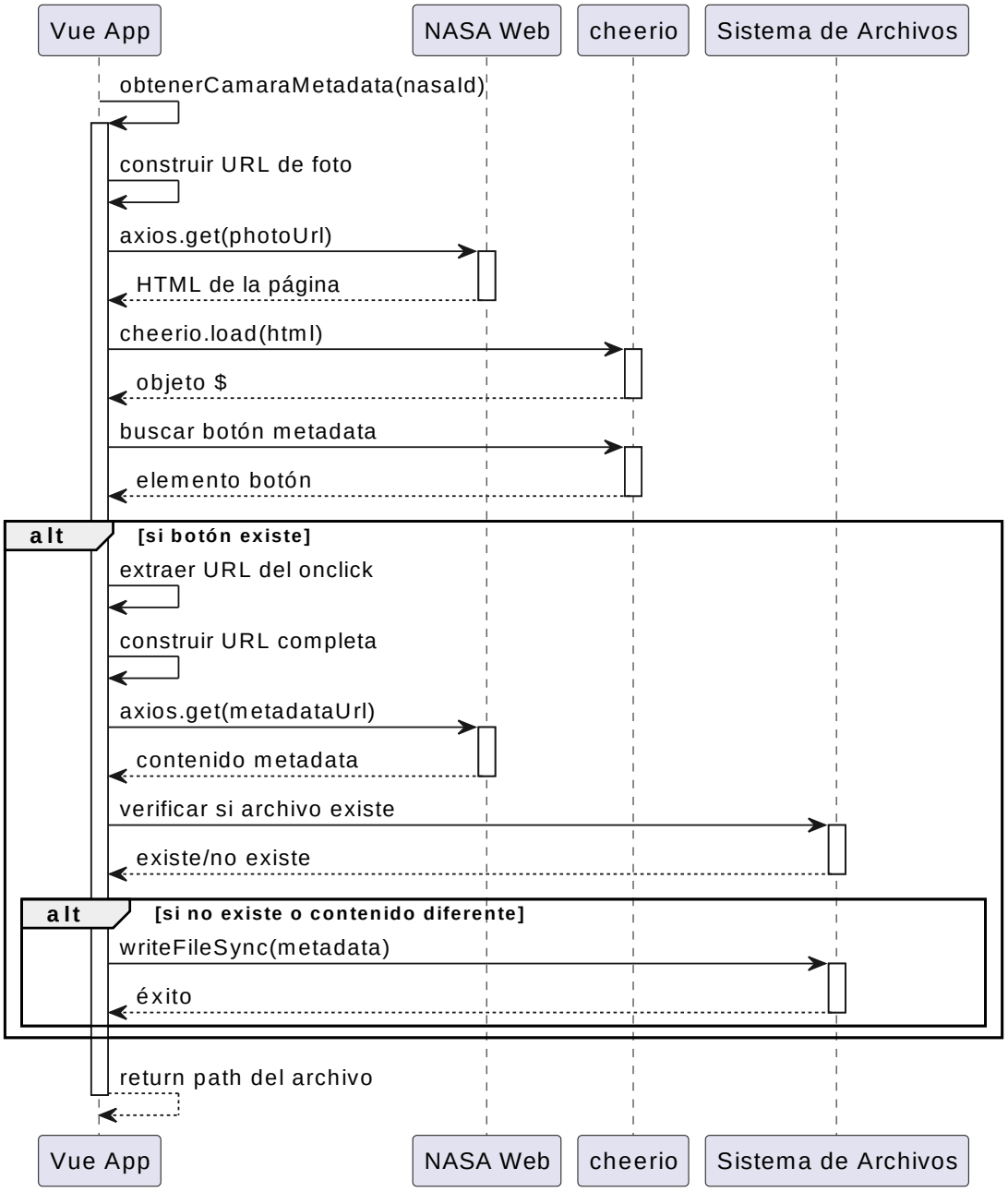

Figura 4.50: Diagrama de secuencia: extracción de metadatos desde NASA Web.

4.4.4 Fase 3: Implementación

Este incremento introdujo una nueva interfaz avanzada para la búsqueda, descarga y programación de imágenes satelitales nocturnas a través de la API oficial `NASA PhotosDatabaseAPI`. La implementación se compone de tres capas funcionales principales: la interfaz de usuario web, el backend de integración

API y el procesamiento de imágenes.

Interfaz de usuario La interfaz gráfica se construyó con Vue.js sobre un entorno HTML personalizado (`template.html`) y se carga dinámicamente desde `periodica.html`. Permite al usuario:

- Aplicar filtros compuestos por tabla, campo, operador y valor.
- Seleccionar campos de retorno por tabla de forma granular.
- Dibujar una región geográfica en un mapa Leaflet para definir un Bounding Box.
- Elegir entre tres fuentes de coordenadas: `frames`, `nadir` y `mlcoord`.
- Activar el modo nocturno, que restringe la búsqueda a horas específicas.

Los resultados se presentan en tarjetas con vista previa, metadatos esenciales y coordenadas geográficas, permitiendo seleccionar imágenes para acciones posteriores.

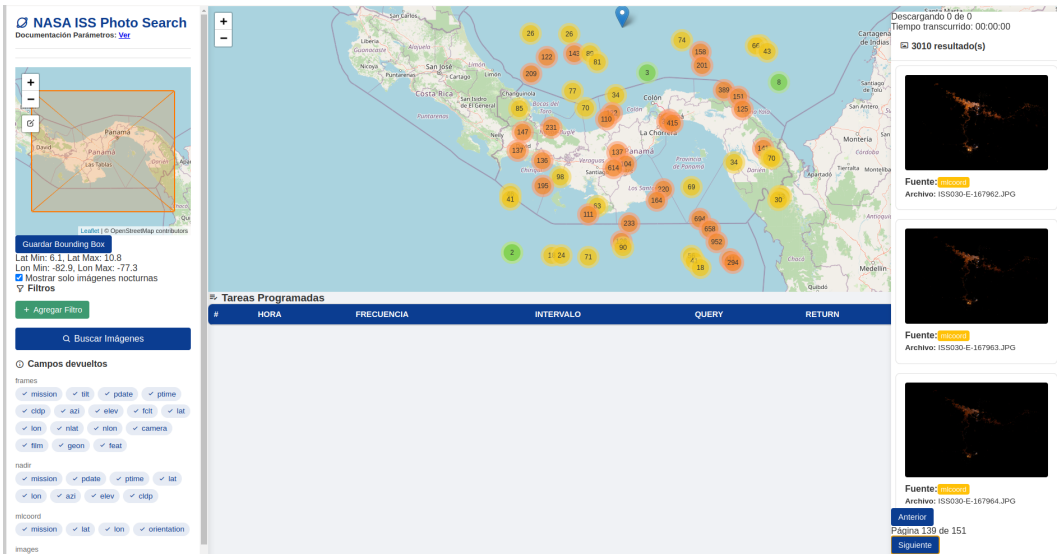

Figura 4.51: Interfaz web para búsqueda avanzada, visualización y filtrado sobre la API de NASA.

Enriquecimiento y verificación Una vez obtenidos los resultados desde la API, el sistema:

- Enriqueció los registros con metadatos adicionales obtenidos vía scraping desde la ficha individual de cada imagen.
- Verificó en tiempo real si cada imagen ya existía en la base de datos local.
- Aplicó reglas de validación por resolución (solo `large` o `highres`).
- Renombró los archivos de forma estándar con el patrón: `[fecha]_[sensor]_[altitud]_[nasa_id].jpg`.

Automatización y descarga directa Se integró una capa de automatización que permite programar la ejecución periódica de consultas usando cron o acciones manuales desde la interfaz. Las descargas pueden ejecutarse:

- Directamente desde la interfaz (`descargarAhora`) con barra de progreso y temporizador.
- Como tarea programada mediante JSON, procesada por `run_batch_processor.py`, que descarga metadatos, crea registros en base de datos y almacena las imágenes con `aria2c`.

4.4.5 Fase 4: Pruebas

Se realizaron pruebas funcionales exhaustivas sobre los siguientes componentes clave:

- **Consultas dinámicas:** Se verificó la correcta construcción de URLs a partir de filtros compuestos y la recuperación de resultados desde múltiples fuentes de coordenadas.
- **Validación previa:** Se comprobó que las imágenes duplicadas eran omitidas automáticamente al detectar su `NASA_ID` en la base de datos.
- **Renombrado y metadatos:** Se inspeccionaron manualmente los archivos JSON generados, confirmando la inclusión de altitud, hora local, cámara y formato.

- **Tareas programadas:** Se ejecutaron pruebas controladas con el script `run_batch_processor.py` validando su ejecución con archivos de metadatos y generación de logs detallados.
- **Descarga directa:** Se monitoreó la ejecución asincrónica desde la interfaz con barra de progreso, asegurando que los archivos se guardaban según la estructura jerárquica esperada.

Análisis de Rendimiento

Tabla 4.9: Métricas de rendimiento del sistema de descarga vía API

Métrica	**Valor**
Tiempo total del proceso	40 - 45 minutos
Tasa de éxito en descarga	100 %
Volumen procesado	3018 imágenes (25 GB)
Peso promedio por imagen	3.6 MB
Velocidad promedio de descarga	1.5 MB/s
Formato predominante	2700 JPG (89.5 %) 318 tif (10.5 %)
Rango de tamaño de archivos	0.4 MB – 28.6 MB

Nota: Aunque el rendimiento de descarga vía API fue superior al de web scraping en términos de velocidad y volumen, se observó que los metadatos obtenidos no son tan completos. En particular, algunos campos geoespaciales y técnicos no están disponibles directamente a través de la API.

4.4.6 Fase 5: Resultados del Incremento

Los resultados del tercer incremento consolidan una capa avanzada de automatización, filtrado y control de calidad en el sistema. Entre los logros más relevantes se encuentran:

- **Automatización programada:** Se logró la ejecución periódica de tareas de descarga mediante archivos JSON y scripts en segundo plano, cumpliendo directamente el objetivo específico 3 de configurar el sistema para realizar descargas automáticas sin necesidad de intervención manual.

- **Enriquecimiento inteligente:** Cada imagen obtenida desde la API fue complementada con información adicional extraída del portal, sin intervención manual.
- **Control de duplicados:** Se evitó la redundancia de datos mediante comparación en tiempo real contra la base local.
- **Procesamiento completo:** Se integraron múltiples scripts en un flujo coherente que ejecuta descarga, extracción, renombrado, inserción en base de datos y almacenamiento físico.

Limitación observada: La API no proporciona todos los metadatos disponibles a través del scraping del portal, lo que limita ciertos análisis geoespaciales avanzados.

4.5 Incremento 4: Descarga Automatizada desde Google Earth Engine

4.5.1 Descripción General

Este cuarto incremento constituye una expansión estratégica del sistema, enfocado en la incorporación de fuentes científicas calibradas de observación nocturna mediante la integración con Google Earth Engine (GEE). El objetivo principal fue establecer un mecanismo eficiente para la adquisición automatizada de imágenes satelitales nocturnas provenientes de plataformas especializadas como VIIRS (Visible Infrared Imaging Radiometer Suite) y DMSP-OLS (Defense Meteorological Satellite Program – Operational Linescan System).
El entorno técnico continúa basándose en la configuración híbrida Windows 11 con WSL2 (Ubuntu 24.04), aprovechando los servicios de autenticación OAuth de Google Cloud Platform y la potencia de procesamiento geoespacial de Earth Engine. Este incremento complementa los desarrollos previos al añadir fuentes de datos calibradas científicamente, aumentando significativamente el valor analítico del sistema para aplicaciones de investigación ambiental y urbanística.

4.5.2 Fase 1: Análisis de Requisitos

La fase inicial de análisis permitió delimitar el alcance funcional del incremento y establecer las especificaciones técnicas necesarias para garantizar una integración efectiva con las plataformas de observación terrestre seleccionadas. Se identificaron los requisitos prioritarios para este incremento y se definieron criterios específicos para su validación.

Requisitos Funcionales Los requerimientos funcionales priorizados para este incremento se centraron en establecer mecanismos de conexión, extracción y procesamiento específicos para datos científicos de observación nocturna. La Tabla 4.10 detalla estos requisitos fundamentales.

Tabla 4.10: Requerimientos funcionales abordados en el incremento 4

ID	Descripción
RF13	Conectarse a Google Earth Engine para descargar imágenes VIIRS y DMSP-OLS para la cuenca del Canal de Panamá, implementando procesos de autenticación y consulta mediante API.
RF14	Exportar imágenes con parámetros definidos: resolución espacial, área geográfica delimitada y periodo temporal seleccionable, garantizando la consistencia de formatos.
RF15	Incorporar las imágenes descargadas al flujo existente de almacenamiento y catalogación, manteniendo la coherencia con los modelos de datos previamente establecidos.
RF16	Desarrollar un sistema de tareas programables para la actualización periódica del repositorio con nuevas imágenes disponibles en las colecciones científicas.
RF17	Implementar visualización específica para imágenes calibradas NOAA, con controles de filtrado por año, sensor y tipo de procesamiento.

Requisitos No Funcionales Complementando los requerimientos funcionales, se identificaron características técnicas esenciales para garantizar la calidad, seguridad y mantenibilidad del sistema, tal como se muestra en la Tabla 4.11.

Tabla 4.11: Requerimientos no funcionales abordados en el incremento 4

ID	Descripción
RNF12	Garantizar autenticación y permisos adecuados en la cuenta de Google Cloud, implementando mecanismos de renovación automática de credenciales y gestión segura de tokens.
RNF13	Evitar descargas duplicadas mediante control de fechas y sensores ya almacenados, implementando verificaciones de integridad en múltiples niveles del sistema.
RNF14	Mantener trazabilidad de la fuente, sensor, y resolución en los nombres de archivo y metadatos, asegurando la consistencia con estándares internacionales de datos geoespaciales.
RNF15	Optimizar el rendimiento de las tareas de exportación para minimizar el consumo de recursos computacionales y cuotas de API de Google Earth Engine.
RNF16	Asegurar la compatibilidad de los formatos exportados (GeoTIFF) con los sistemas de visualización y análisis desarrollados en incrementos previos.

La identificación temprana de estos requisitos permitió establecer una base sólida para las fases subsecuentes del incremento, orientando las decisiones de diseño e implementación hacia objetivos claramente definidos.

4.5.3 Fase 2: Diseño del Sistema

En el marco del cuarto incremento del sistema, se consolidaron funcionalidades clave asociadas a la visualización, exportación, carga local y automatización de tareas relacionadas con las imágenes nocturnas provistas por el sistema NOAA. Para modelar estas capacidades se elaboraron diagramas UML que permitieron describir el comportamiento, interacción y estructura de los componentes nuevos.

Diagramas de Casos de Uso

Los diagramas de casos de uso modelan las interacciones principales del usuario con el sistema:

- La Figura 4.52 representa funcionalidades vinculadas a la visualización de luces nocturnas: selección de fecha, manipulación de capas, regeneración de mosaicos del mapa y exportación de nuevas imágenes.
- La Figura 4.53 cubre el subsistema de tareas automáticas, permitiendo la configuración de frecuencias, verificación de cambios, y gestión completa de tareas programadas.
- La Figura 4.54 muestra las acciones disponibles dentro del visor geoespacial local, como acceder a imágenes ya descargadas, explorar metadatos y controlar el zoom o la capa base.

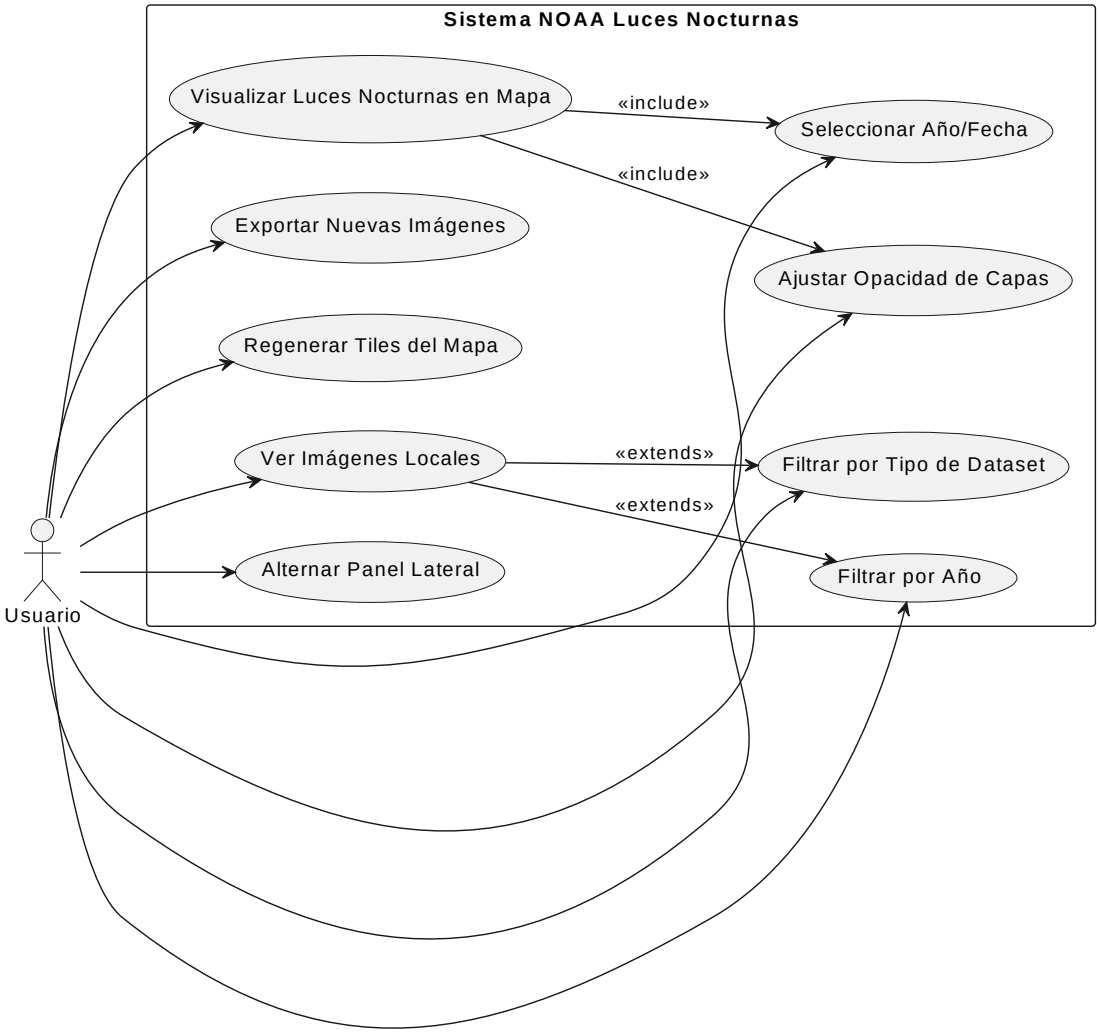

Figura 4.52: Casos de uso: visualización y exportación de imágenes NOAA.

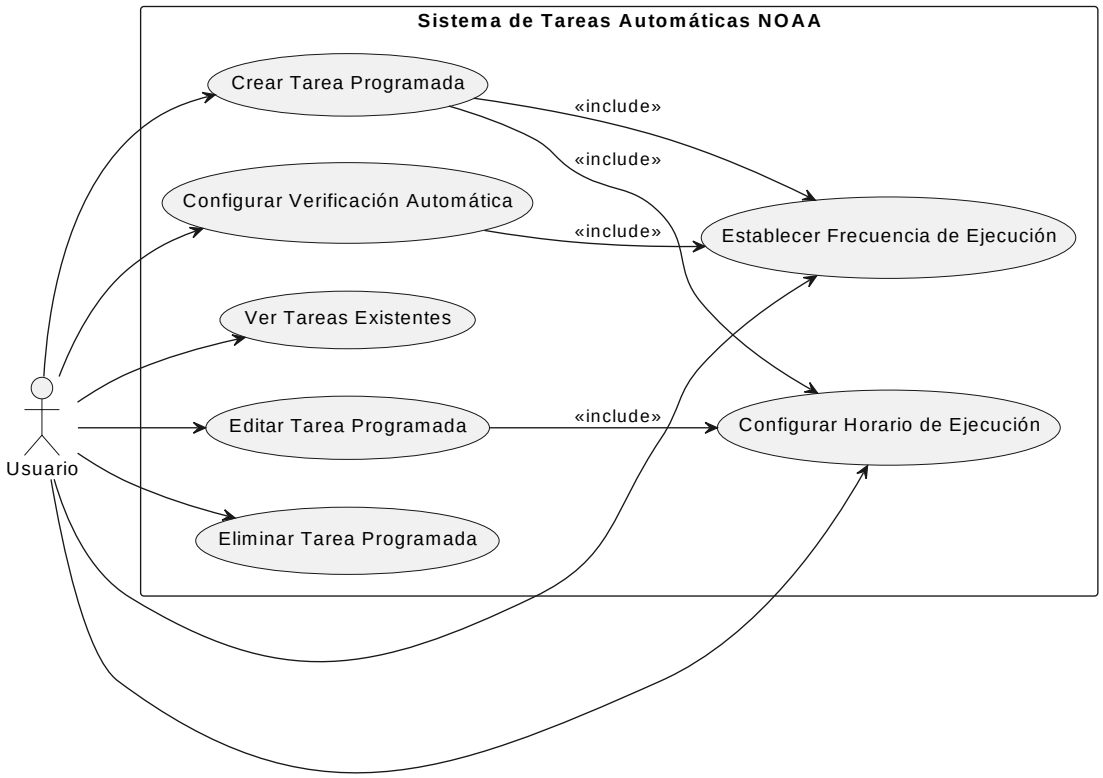

Figura 4.53: Casos de uso: sistema de tareas automáticas NOAA.

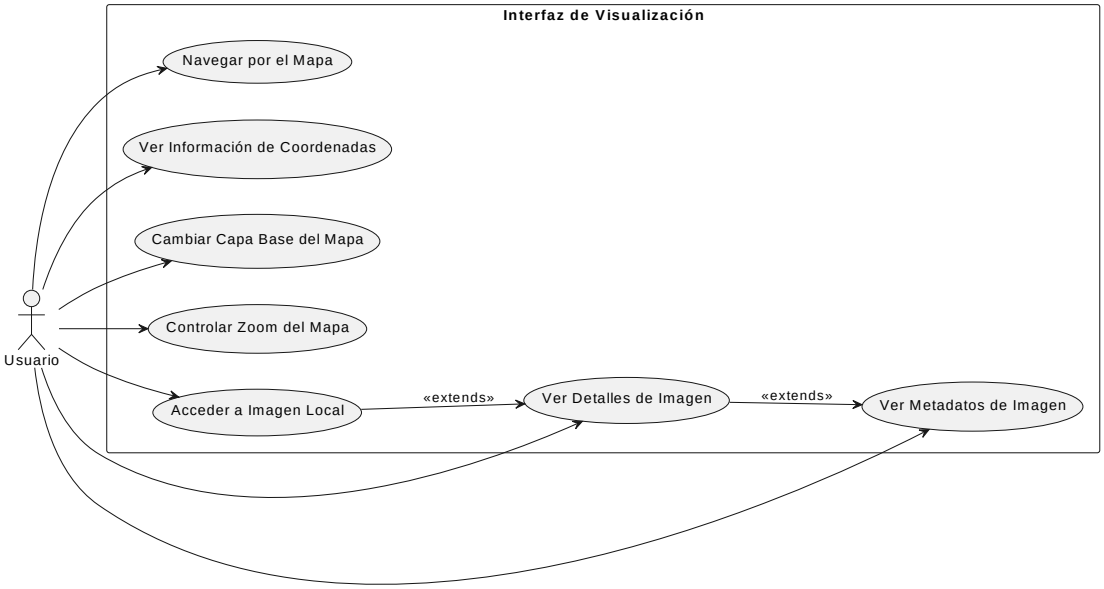

Figura 4.54: Casos de uso: exploración de imágenes y metadatos locales.

Diagramas de Actividad

Los diagramas de actividad describen flujos clave en este incremento, contemplando validaciones, procesamientos en paralelo y manejo robusto de errores:

- La Figura 4.55 representa el proceso completo de exportación de imágenes NOAA desde Earth Engine. El flujo inicia cuando el usuario hace clic en “Exportar Imágenes”, mostrando la barra de progreso e inicializando Earth Engine. En la fase de búsqueda de imágenes nuevas, carga metadatos existentes y obtiene las colecciones DMSP y VIIRS. Procesa cada colección filtrando por la región de Panamá y ordenando por fecha descendente, identifica imágenes no exportadas y cuenta el total disponible. Durante el lanzamiento de exportaciones, selecciona cada imagen no exportada, crea tareas de exportación a Drive, las inicia en Earth Engine y actualiza el progreso hasta completar todas las imágenes. En la fase de monitoreo y descarga, espera la completación de cada tarea: si se completa exitosamente, descarga la imagen desde Drive, la mueve a carpeta local, actualiza metadatos y tiles JSON, y actualiza el progreso real; si falla, registra el error correspondiente. El proceso continúa hasta completar todas las tareas pendientes, finalizando con la barra de progreso y mensaje de éxito.

- La Figura 4.56 modela la carga de imágenes locales desde archivo de metadatos. El proceso inicia cuando el usuario hace clic en “Cargar imágenes locales”, mostrando indicador de carga y leyendo el archivo metadatos_noaa.json. Si el archivo no existe, muestra error de archivo no encontrado; si existe, procede a procesar los metadatos. En la construcción de lista, extrae el ID de cada imagen, determina el dataset correspondiente (DMSP o VIIRS), construye la ruta local del archivo y crea objetos de imagen con metadatos completos. Una vez procesados todos los metadatos, guarda la lista en caché, aplica los filtros actuales configurados y muestra las imágenes en el panel lateral.

- La Figura 4.57 detalla la gestión completa de tareas automatizadas. El

flujo inicia cuando el usuario abre el modal de tareas, cargando las tareas existentes desde JSON. El sistema presenta tres opciones principales: crear nueva tarea o eliminar tarea. Para crear nueva tarea, configura hora de ejecución, selecciona frecuencia, establece intervalo si aplica y valida la configuración; si es válida, elimina tarea previa si existe, construye comando WSL/bash, crea la tarea en Windows Scheduler, guarda en archivo JSON y muestra mensaje de éxito; si no es válida, muestra error de validación. Para editar tarea existente, rellena el formulario con datos actuales, permite modificación y sigue el mismo flujo de validación y creación. Para eliminar tarea, la elimina de Windows Scheduler, actualiza el archivo JSON y muestra confirmación. En todos los casos, finaliza cerrando el modal.

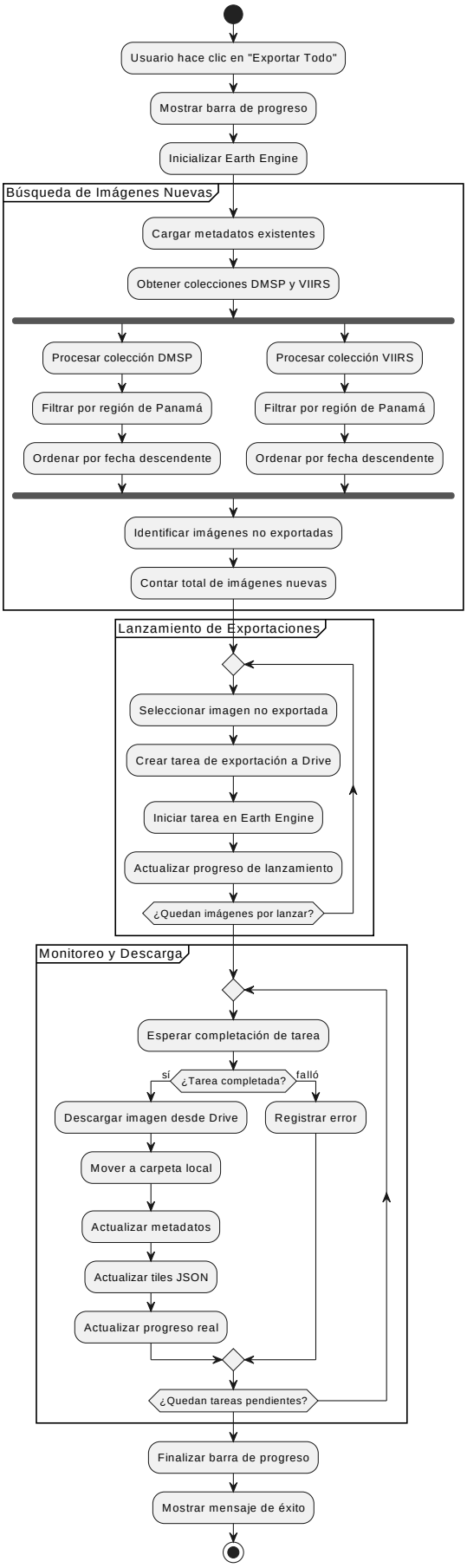

Figura 4.55: Diagrama de actividad: proceso de exportación de imágenes NOAA.

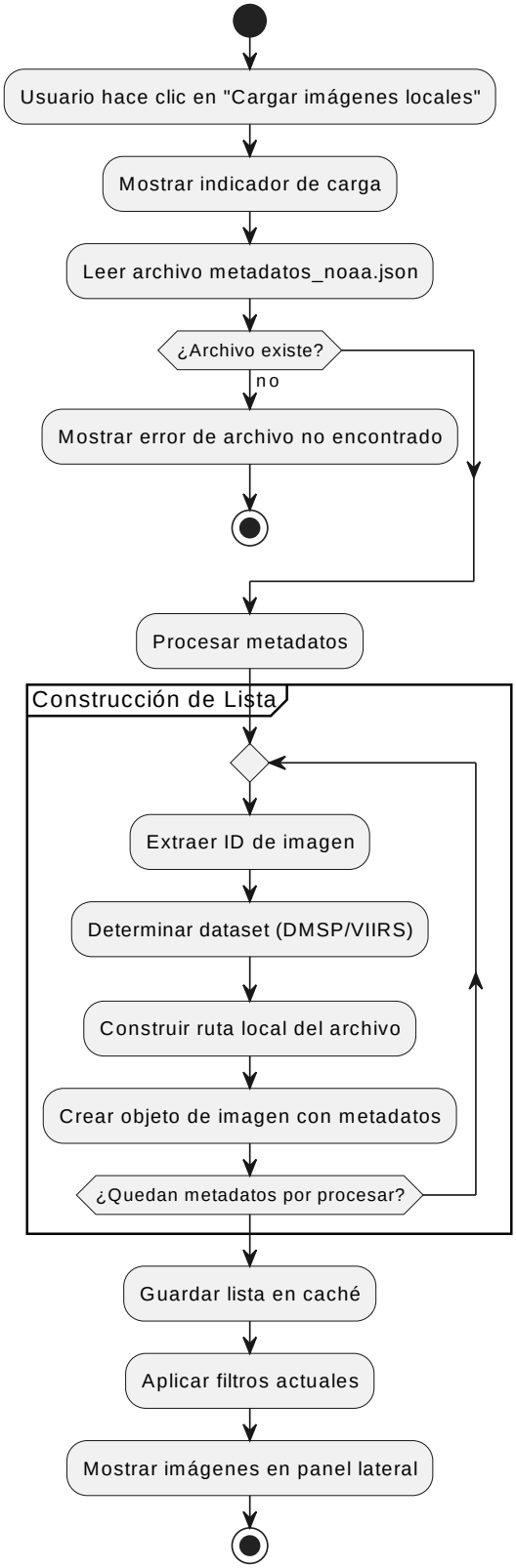

Figura 4.56: Diagrama de actividad: carga y filtrado de imágenes locales.

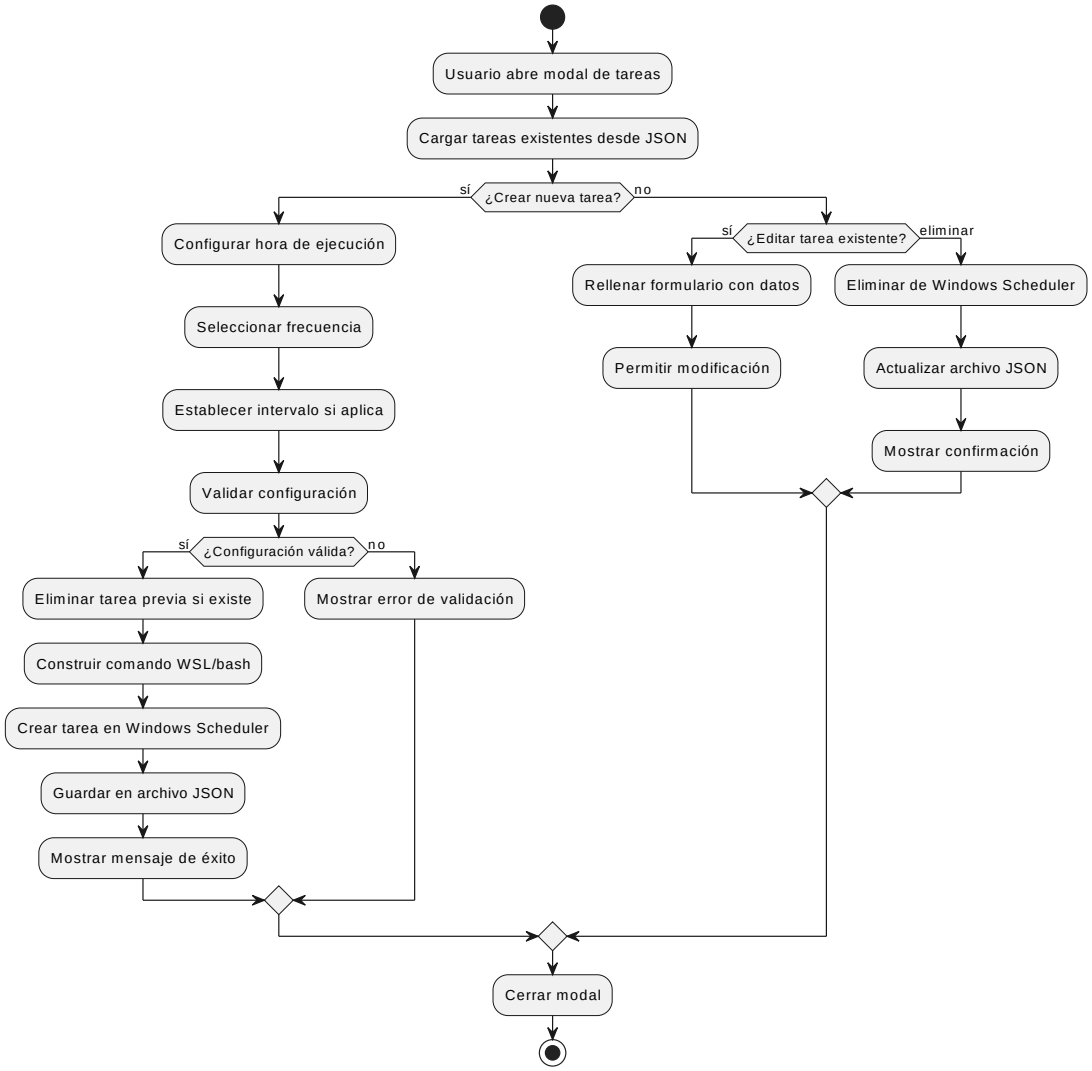

Figura 4.57: Diagrama de actividad: configuración y gestión de tareas automatizadas.

Diagramas de Secuencia

Los diagramas de secuencia muestran la interacción detallada entre componentes del sistema NOAA, evidenciando comunicación entre procesos y manejo asíncrono de operaciones:

- La Figura 4.58 modela la exportación masiva de imágenes desde Earth Engine. El proceso inicia cuando el usuario solicita exportar imágenes, mostrando el estado de *Exportando...* con barra de progreso. El sistema ejecuta un script Python que accede a las colecciones de datos NOAA (DMSP y VIIRS) en Earth Engine. Para cada imagen nueva, crea una tarea de exportación hacia Google Drive y la inicia automáticamente. El

sistema monitorea continuamente el progreso de todas las tareas: cuando una tarea se completa, utiliza rclone para descargar la imagen resultante desde Drive, la mueve a la carpeta local del NAS y actualiza los metadatos correspondientes. Una vez finalizadas todas las descargas, muestra el mensaje de *Completado*.

- La Figura 4.59 muestra la creación de tareas automáticas en el sistema operativo. El flujo inicia cuando el usuario abre el modal de configuración de tareas y define los parámetros deseados (frecuencia, hora, etc.). Al confirmar la configuración, el sistema valida los datos del formulario y construye el comando necesario para programar la tarea. Utiliza el Programador de Tareas de Windows para registrar la nueva tarea en el sistema operativo. Si la operación es exitosa, muestra mensaje de confirmación y cierra el modal; en caso de error, presenta el mensaje de error correspondiente.

- La Figura 4.60 representa la regeneración de los mosaicos del mapa. El proceso inicia cuando el usuario solicita regenerar los tiles, mostrando el estado *Regenerando tiles...* . El sistema ejecuta un script Python que inicializa el procesador NOAA y carga los metadatos existentes de las imágenes. Para cada imagen, solicita a Earth Engine que genere una nueva URL de mosaico con token actualizado, actualiza el archivo de configuración de tiles y guarda los cambios. Una vez completado el proceso, muestra *Tiles regenerados* y recarga automáticamente la interfaz para mostrar los mosaicos actualizados.

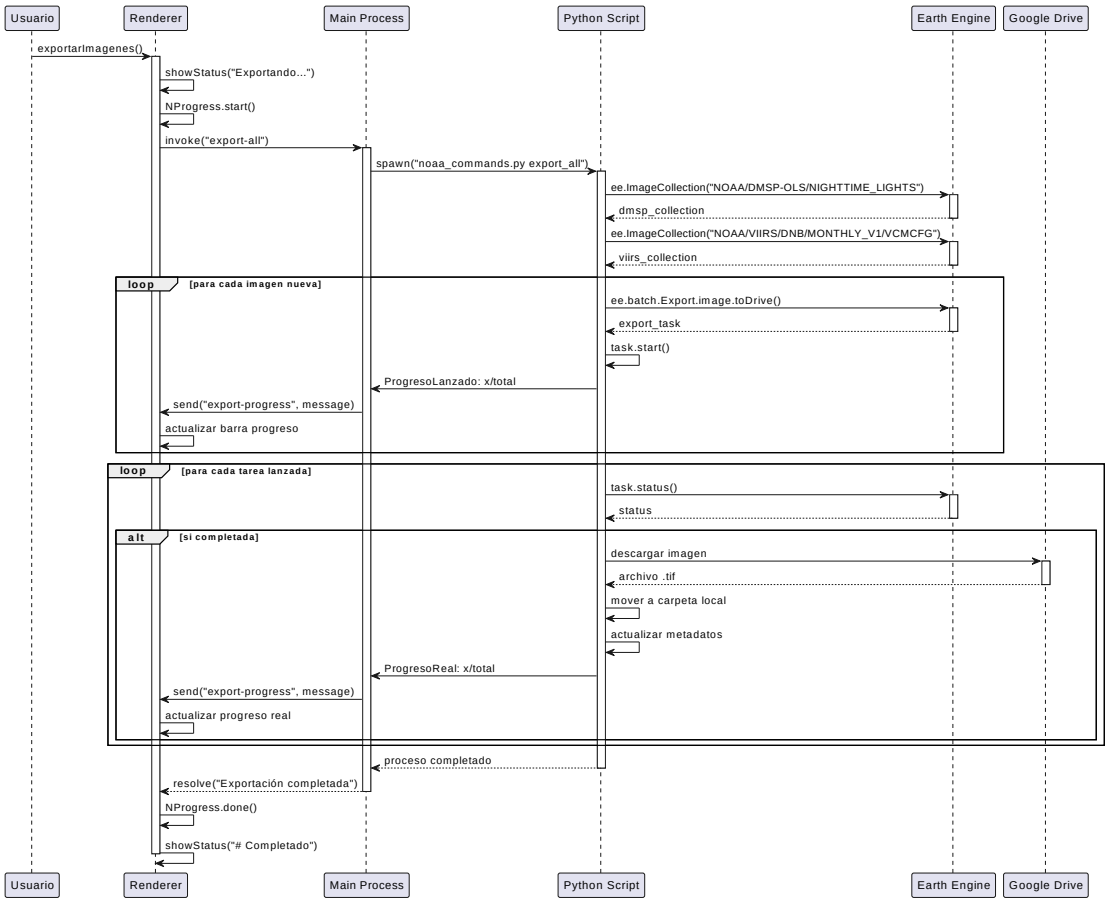

Figura 4.58: Diagrama de secuencia: exportación de imágenes desde Earth Engine.

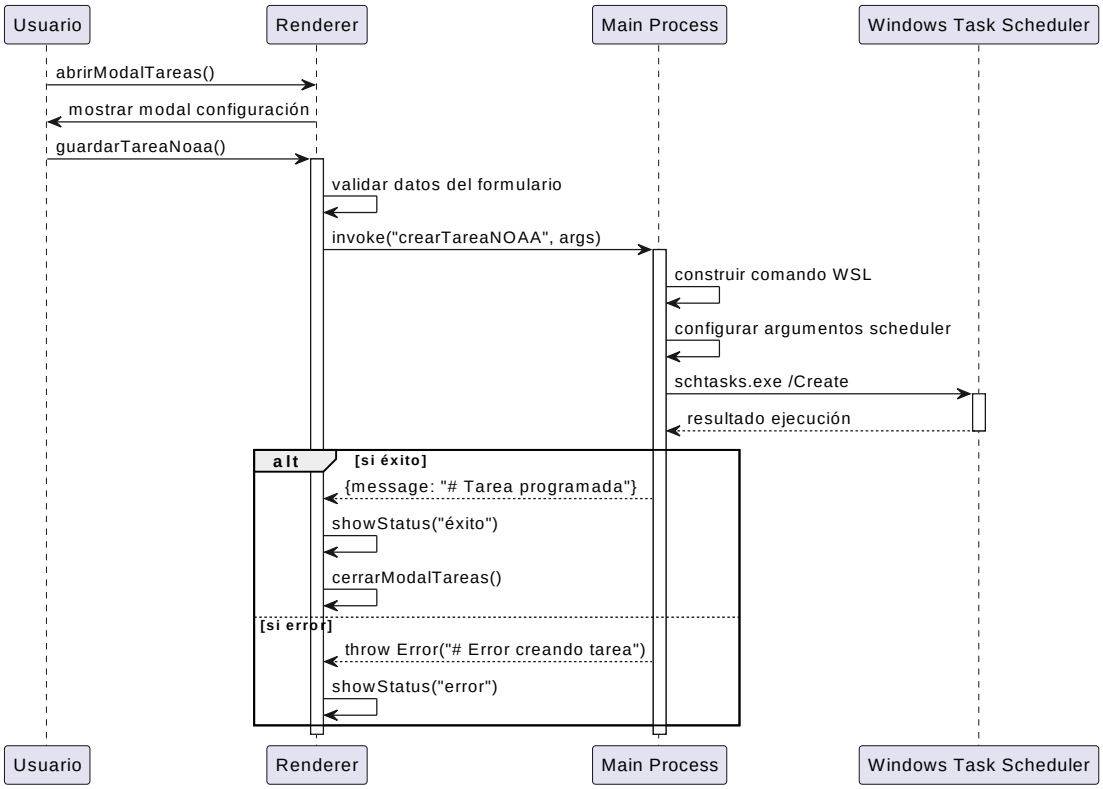

Figura 4.59: Diagrama de secuencia: programación de tareas automáticas.

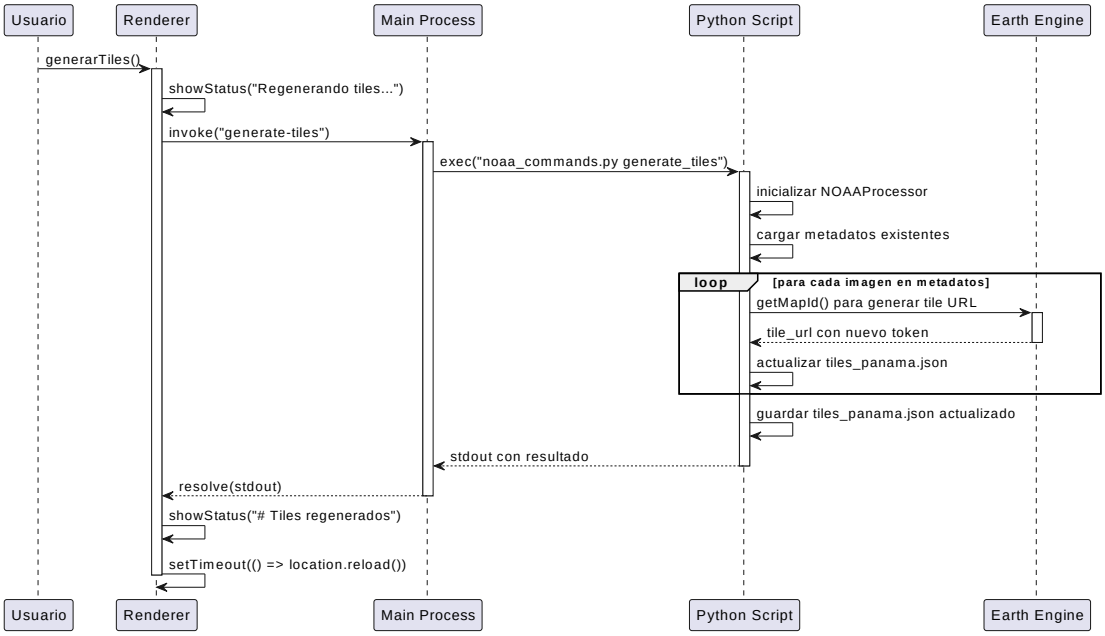

Figura 4.60: Diagrama de secuencia: regeneración de mosaicos del mapa.

Diseño Orientado a Objetos La arquitectura de este incremento se basó en un modelo orientado a objetos con separación clara de responsabilidades. La Figura 4.61 muestra las clases principales:

Diseño Orientado a Objetos La arquitectura de este incremento se basó en un modelo orientado a objetos con separación clara de responsabilidades. La Figura 4.61 muestra la clase principal NOAAProcessor que encapsula toda la lógica de integración con Google Earth Engine. Esta clase incluye configuración específica para la región de Panamá mediante ee.Geometry.Rectangle([-83.1, 7.0, -77.2, 9.6]), gestión de carpetas (drive_folder, nas_folder, local_folder), y métodos especializados como export_all_images() para exportación masiva, wait_move() para monitoreo de tareas, y get_metadata() para extracción de información. Este enfoque modular permite escalar o adaptar cada componente por separado sin romper la funcionalidad global, facilitando la integración con diferentes fuentes de datos geoespaciales.

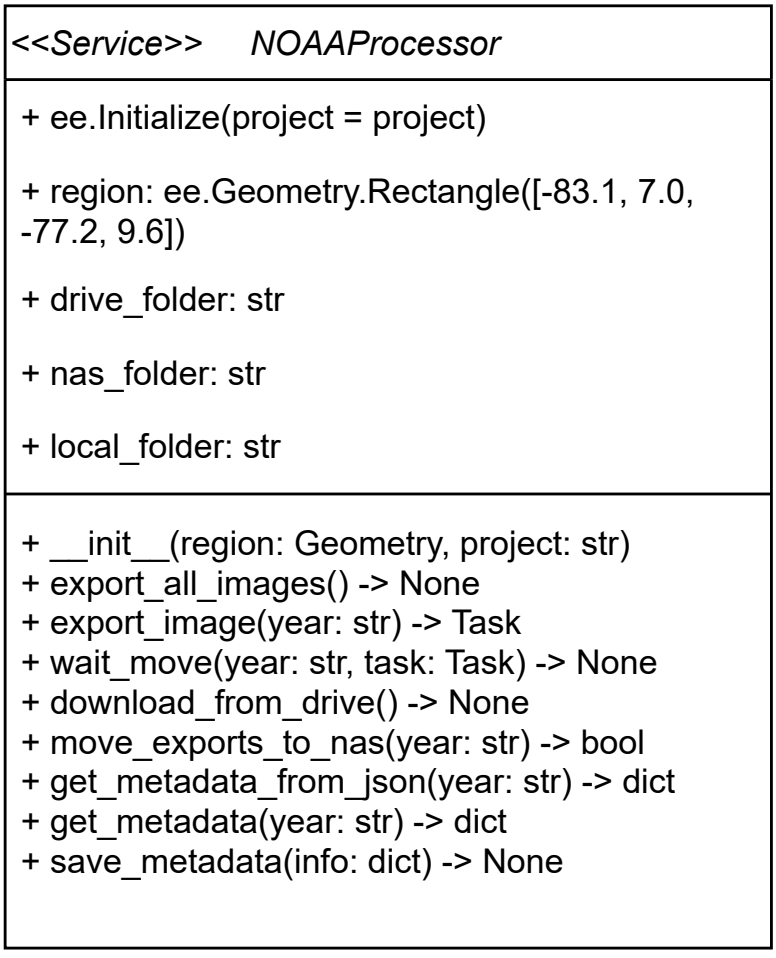

Figura 4.61: Diagrama de clases del incremento 4: módulos de procesamiento NOAA.

Estos modelos desarrollados durante la fase de diseño proporcionaron una visión estructurada del sistema, facilitando la transición hacia la implementación

y estableciendo un lenguaje común para todos los componentes del incremento.

4.5.4 Fase 3: Implementación

La implementación de este incremento se organizó en tres componentes principales: el backend de procesamiento, el sistema de tareas programadas y la interfaz de visualización. Cada componente fue desarrollado siguiendo los principios de modularidad, reutilización y robustez establecidos en incrementos anteriores.

Backend de procesamiento NOAA El núcleo funcional fue implementado en el módulo `noaa_processor.py`, mediante la clase `NOAAProcessor`. Este componente central permite:

- Establecer conexión autenticada con Google Earth Engine mediante credenciales OAuth.
- Recuperar y consultar las colecciones científicas `NOAA/DMSP-OLS/NIGHTTIME_LIGHTS` y `NOAA/VIIRS/DNB/MONTHLY_V1/VCMCFG`.
- Aplicar filtros geoespaciales para delimitar la región de interés (cuenca del Canal de Panamá).
- Implementar filtros temporales para seleccionar imágenes por año, mes o rango de fechas.
- Identificar de forma eficiente las imágenes no descargadas previamente mediante comparación con registros locales.
- Configurar y lanzar tareas de exportación asíncronas hacia Google Drive con parámetros optimizados.
- Monitorear el estado de las tareas en ejecución y gestionar reintentos automáticos cuando sea necesario.

Sistema de gestión de tareas El módulo `noaa_commands.py` se implementó como una interfaz de línea de comandos que permite ejecutar distintas operaciones de forma desatendida o programada. Las principales funcionalidades implementadas incluyen:

- `export_all`: descubre y exporta todas las imágenes nuevas disponibles en un rango temporal.
- `get_metadata`: consulta y actualiza los metadatos disponibles por año y sensor.
- `download_pending`: descarga desde Google Drive las imágenes ya exportadas por tareas asíncronas.
- `process_downloaded`: procesa las imágenes descargadas, generando tiles y actualizando el catálogo.
- `run_task`: ejecuta tareas específicas definidas en archivos JSON de configuración.
- `schedule_task`: programa la ejecución periódica de tareas mediante cron.

Cada imagen exportada atraviesa un flujo de procesamiento completo que incluye:

1. Descarga desde Google Drive mediante `rclone` con verificación de integridad.
2. Conversión al formato estándar y aplicación de mejoras visuales cuando sea necesario.
3. Traslado a la estructura jerárquica dentro del entorno NAS según año y sensor.
4. Registro de metadatos técnicos y geoespaciales en un archivo estructurado JSON.
5. Generación de tiles y archivos auxiliares para visualización optimizada en el frontend.

6. Actualización del índice maestro con referencias a las nuevas imágenes incorporadas.

Interfaz web para control y visualización Se desarrolló una interfaz especializada en `noaa.html` utilizando Leaflet, Bootstrap y componentes personalizados. Esta interfaz permite:

- Visualizar imágenes NOAA (tanto VIIRS como DMSP) sobre un mapa interactivo con controles de zoom y desplazamiento.
- Aplicar filtros dinámicos por año, tipo de sensor y método de procesamiento.
- Ajustar la opacidad y visibilidad de cada capa mediante controles deslizantes.
- Consultar los metadatos completos de cada imagen mediante paneles informativos.
- Ejecutar procesos de exportación completos con un solo clic desde la interfaz.
- Configurar y programar tareas automáticas (frecuencia, hora, parámetros) desde un panel modal.
- Monitorizar el estado de las tareas en ejecución mediante indicadores visuales.

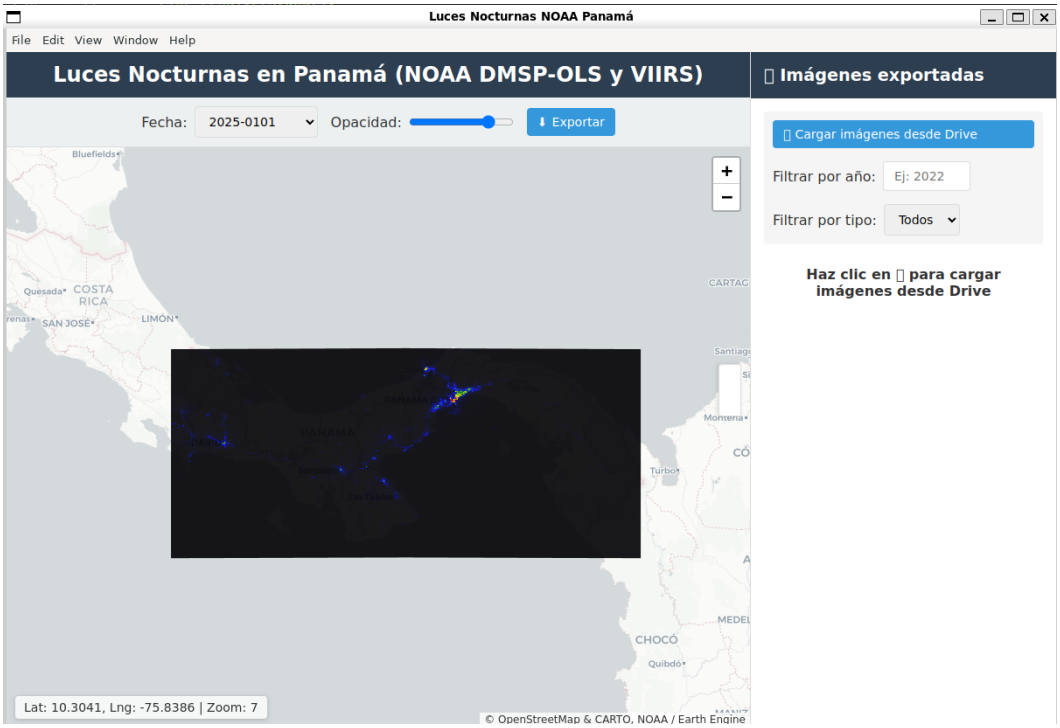

Figura 4.62: Interfaz NOAA para exploración, exportación y tareas automatizadas.

La comunicación entre el frontend y el backend se implementó mediante el mecanismo `ipcRenderer` de Electron, permitiendo iniciar procesos asincrónicos, leer archivos de configuración y emitir notificaciones en tiempo real desde el núcleo del sistema hacia la interfaz gráfica.

Figura 4.63: Modal de configuración de tareas programadas para actualización automática.

Esta implementación integral garantiza una experiencia fluida para el usuario técnico, combinando la potencia analítica de Earth Engine con la facilidad de uso de una interfaz moderna y responsive.

4.5.5 Fase 4: Pruebas

La verificación del incremento se enfocó en validar el correcto funcionamiento de cada componente individual, así como la integración completa del flujo de trabajo desde la consulta en Earth Engine hasta la visualización final en la interfaz. Se diseñaron pruebas específicas para cada aspecto crítico del sistema.

Pruebas Funcionales Se realizaron pruebas funcionales exhaustivas para verificar la operatividad de los diferentes módulos:

- **Autenticación:** Se verificó la correcta autenticación con Google Earth Engine, incluyendo la renovación automática de tokens y la gestión de errores de sesión.
- **Consulta de colecciones:** Se comprobó la capacidad del sistema para acceder a las colecciones VIIRS y DMSP-OLS, aplicando diversos filtros y verificando la consistencia de los resultados.
- **Exportación controlada:** Se validó el proceso completo de exportación de imágenes, desde la creación de la tarea en GEE hasta su finalización, con monitoreo de progreso y gestión adecuada de finalizaciones.
- **Filtrado efectivo:** Se confirmó que el sistema solo descargara imágenes no presentes en la base de datos local, evitando duplicados mediante la comparación con el archivo `metadatos_noaa.json`.
- **Integridad de descarga:** Las imágenes descargadas fueron verificadas en cuanto a su formato, estructura de datos interna, cobertura espacial y resolución esperada según los parámetros de exportación.
- **Visualización geoespacial:** Se comprobó la correcta generación y carga de tiles sobre el mapa interactivo, verificando su respuesta a controles de

opacidad, cambios de año y regeneración manual de mosaicos.

Pruebas de Integración Además de las pruebas de componentes individuales, se realizaron pruebas de integración para verificar el correcto funcionamiento del sistema como un todo:

- **Flujo completo:** Se validó el proceso end-to-end desde la consulta en Earth Engine hasta la visualización en la interfaz, pasando por todos los pasos intermedios.
- **Tareas programadas:** Se ejecutaron tareas programadas tanto desde archivos JSON como mediante la interfaz gráfica, verificando su correcta ejecución en los momentos establecidos.
- **Coexistencia con otros módulos:** Se verificó que el nuevo incremento no interfiriera con las funcionalidades existentes de incrementos anteriores.
- **Compatibilidad de almacenamiento:** El sistema se validó tanto en entornos locales como en montajes sobre NAS, confirmando la consistencia del flujo de datos y la disponibilidad de archivos en ambos contextos.

Métricas de Rendimiento Durante la fase de pruebas se monitorizaron diversos parámetros para evaluar el desempeño del sistema, recopilando métricas relevantes que se muestran en la Tabla 4.12.

Tabla 4.12: Métricas de rendimiento del sistema de descarga desde Google Earth Engine

Métrica	**Valor**
Tiempo promedio de exportación en GEE (por lote de 6 imágenes)	3.4 minutos
Tiempo de descarga desde Google Drive (promedio)	25 segundos por imagen
Velocidad de transferencia promedio	4.2 MB/s
Tamaño promedio de imágenes VIIRS	850 KB
Tamaño promedio de imágenes DMSP-OLS	409 KB
Volumen total de datos procesados (muestra inicial)	84.3 MB (191 imágenes)
Tasa de éxito en exportación automática	100 %
Tiempo promedio de generación de tiles para visualización	12 segundos por imagen
Tiempo total del proceso	8.4 minutos

Estas métricas demuestran un rendimiento satisfactorio del sistema, con tiempos de procesamiento adecuados para operaciones por lotes y una alta tasa de éxito en las tareas automatizadas.

4.5.6 Fase 5: Resultados del Incremento

Este incremento permitió ampliar significativamente el alcance del sistema mediante la integración de fuentes científicas calibradas con cobertura global y datos estandarizados. Los principales resultados obtenidos fueron:

- **Integración completa con Earth Engine:** Se estableció una conexión robusta con la plataforma GEE, desarrollando herramientas de automatización para el manejo de grandes volúmenes de datos provenientes de sensores como DMSP-OLS y VIIRS, cumpliendo así el Objetivo Específico 4.

- **Corpus de imágenes calibradas:** Se integraron con éxito las colecciones VIIRS y DMSP-OLS, con un total de 127 imágenes nuevas descargadas (65 VIIRS y 62 DMSP) que abarcan el periodo 2012-2024 para la cuenca del Canal de Panamá.
- **Organización estructurada:** Se registraron y organizaron los archivos exportados siguiendo la estructura jerárquica establecida en el entorno NAS, manteniendo consistencia con los incrementos previos.
- **Metadatos enriquecidos:** Se generaron y actualizaron automáticamente los metadatos técnicos y geoespaciales, incluyendo información específica de calibración, sensor y procesamiento para cada imagen.
- **Visualización especializada:** Se desarrolló una interfaz gráfica moderna para la exploración, consulta y descarga de imágenes NOAA, con controles adaptados a este tipo específico de datos.
- **Automatización completa:** Se implementó un sistema de tareas programables que permite la actualización periódica del repositorio sin intervención manual, garantizando acceso a las imágenes más recientes.
- **Documentación técnica:** Se generó documentación detallada sobre los procesos implementados, facilitando el mantenimiento y la expansión futura del sistema.

Limitaciones Técnicas Identificadas

Durante el desarrollo e implementación del Incremento 4, se identificaron diversas limitaciones técnicas que impactan el funcionamiento del sistema:

- **Cuotas de API:** Google Earth Engine impone límites en la cantidad de tareas simultáneas y el volumen de datos exportables diariamente, lo que puede retrasar procesos masivos de descarga.
- **Resolución limitada:** Las colecciones NOAA disponibles en Earth Engine tienen una resolución espacial máxima de 15 arcsegundos (aproximadamente 500m en el ecuador), inferior a las imágenes astronáuticas del incremento 1.

- **Dependencia de conectividad:** El proceso completo depende de una conexión estable a los servicios de Google, lo que puede generar vulnerabilidades ante problemas de red.
- **Latencia en exportaciones:** Las tareas de exportación en Earth Engine operan de forma asíncrona y pueden quedar en cola durante periodos variables según la carga del sistema.

Estrategias de Recuperación ante Fallos

Para garantizar la robustez del sistema frente a las limitaciones identificadas, se implementaron los siguientes mecanismos de recuperación:

- **Sistema de reintentos adaptativos:** Las tareas fallidas se reprograman automáticamente con intervalos exponenciales, hasta un máximo configurable de intentos.
- **Verificación multinivel:** Se realizan comprobaciones de integridad en diferentes etapas del proceso (exportación, descarga, procesamiento) para detectar fallos tempranamente.
- **Estado persistente:** El sistema mantiene un registro detallado del estado de cada tarea, permitiendo reanudar operaciones interrumpidas sin duplicar esfuerzos.
- **Notificaciones automáticas:** Se implementaron alertas que informan al administrador sobre errores críticos que requieren intervención manual.
- **Logging extensivo:** Se mantiene un registro detallado de todas las operaciones, facilitando el diagnóstico y resolución de problemas.

Estos mecanismos han demostrado ser efectivos durante las pruebas del sistema, garantizando una alta disponibilidad y fiabilidad incluso en condiciones sub-óptimas.

4.6 Integración, Automatización y Despliegue del Sistema

4.6.1 Síntesis Funcional de los Módulos

Tras completar la implementación de los cuatro incrementos funcionales, el sistema automatizado para la adquisición y gestión de imágenes satelitales nocturnas ha alcanzado un estado operativo completo y robusto. Cada módulo desarrollado desempeña un rol específico y sinérgico dentro del flujo general del sistema:

- **Módulo ISS - NASA Gateway**: Especializado en la extracción y enriquecimiento de imágenes RGB mediante técnicas de web scraping y consultas a la API oficial de la NASA, garantizando la obtención de fotografías de alta calidad con sus metadatos correspondientes.
- **Módulo GEE (Google Earth Engine)**: Responsable de la descarga sistemática de imágenes calibradas radiométricamente desde las colecciones científicas VIIRS y DMSP-OLS, aplicando los filtros geoespaciales y temporales definidos por el usuario.
- **Explorador Visual**: Interfaz gráfica intuitiva que facilita la navegación jerárquica y la manipulación eficiente de archivos en el entorno NAS, permitiendo visualizar, filtrar y organizar el catálogo completo de imágenes.
- **Gestión de Metadatos**: Sistema de registro en base de datos SQLite y herramienta de consulta técnica vía consola, que mantiene actualizada toda la información contextual y científica asociada a cada imagen.

Estos componentes han sido integrados en un sistema modular, escalable y completamente automatizado, optimizado para el monitoreo continuo y sistemático de la contaminación lumínica a través de imágenes satelitales nocturnas.

Adicionalmente, se definió un segundo caso de uso centrado en la interacción del usuario con el menú del sistema. Este modelo, presentado en la Figura 4.64,

facilita la comprensión de las operaciones disponibles en la interfaz técnica, así como su ejecución parametrizada desde consola.

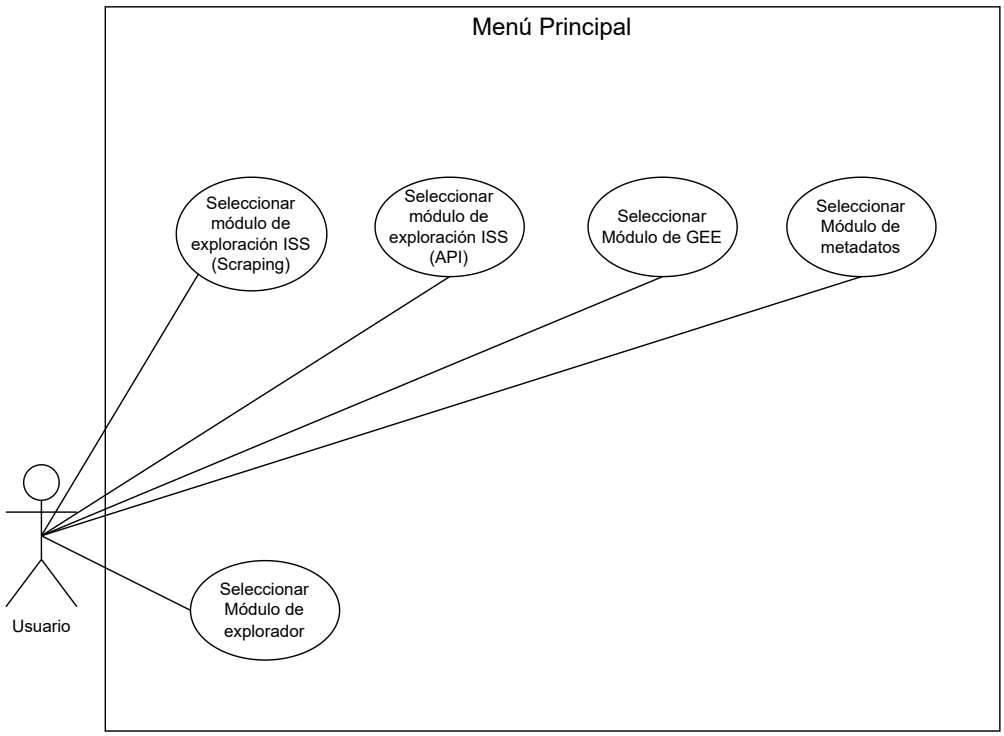

Figura 4.64: Diagrama de casos de uso del Menú del Sistema. Interacción con el menú técnico.

Para sintetizar cuantitativamente la evolución en la eficiencia de los distintos métodos de descarga desarrollados a lo largo del proyecto, se presenta la Tabla 4.13. Esta tabla resume los tiempos aproximados requeridos por cada método, agrupados por fuente satelital e incremento funcional. La comparación revela una mejora significativa: se pasó de procesos manuales que podían tomar más de cinco horas (en el caso de la ISS) a sistemas automatizados que completan las descargas en menos de 10 minutos (en el caso de NOAA con `rclone` y descargas paralelas).

Además, los resultados mostraron una reducción sostenida en la ocurrencia de errores: mientras que los métodos manuales presentaron tasas de falla cercanas al 50 %, los sistemas automatizados lograron tasas inferiores al 1 %, con algunos módulos alcanzando un 100 % de éxito en las descargas. También se observó un aumento progresivo en la inclusión de metadatos técnicos, pasando de coberturas parciales (alrededor del 60–80 %) a extracciones completas en los métodos finales vía API.

Tabla 4.13: Comparativa de tiempos de descarga por método e incremento de software

Fuente	**Método**	**Tiempo aprox. de descarga**
ISS	Manual	Prolongado (alto volumen)
ISS	Web scraping, sin paralelismo	5+ horas
ISS	Web scraping con descargas paralelas	2–3 horas
ISS	API sin metadatos, con descargas paralelas	20 minutos
ISS	API con metadatos, web scraping y descargas paralelas	45–50 minutos
NOAA	Manual	Prolongado (alto volumen)
NOAA	GEE API sin descargas paralelas	50 minutos
NOAA	GEE API con descargas paralelas y `rclone`	8.4 minutos

4.6.2 Despliegue en Infraestructura Mixta

El sistema ha sido desplegado exitosamente en un entorno híbrido Windows-Linux mediante la tecnología WSL2 (Windows Subsystem for Linux 2), lo que ha permitido aprovechar simultáneamente las capacidades de automatización nativas de Windows junto con la potencia y flexibilidad de las herramientas de línea de comandos de Linux. La arquitectura final de despliegue se ilustra en detalle en la Figura 4.65.

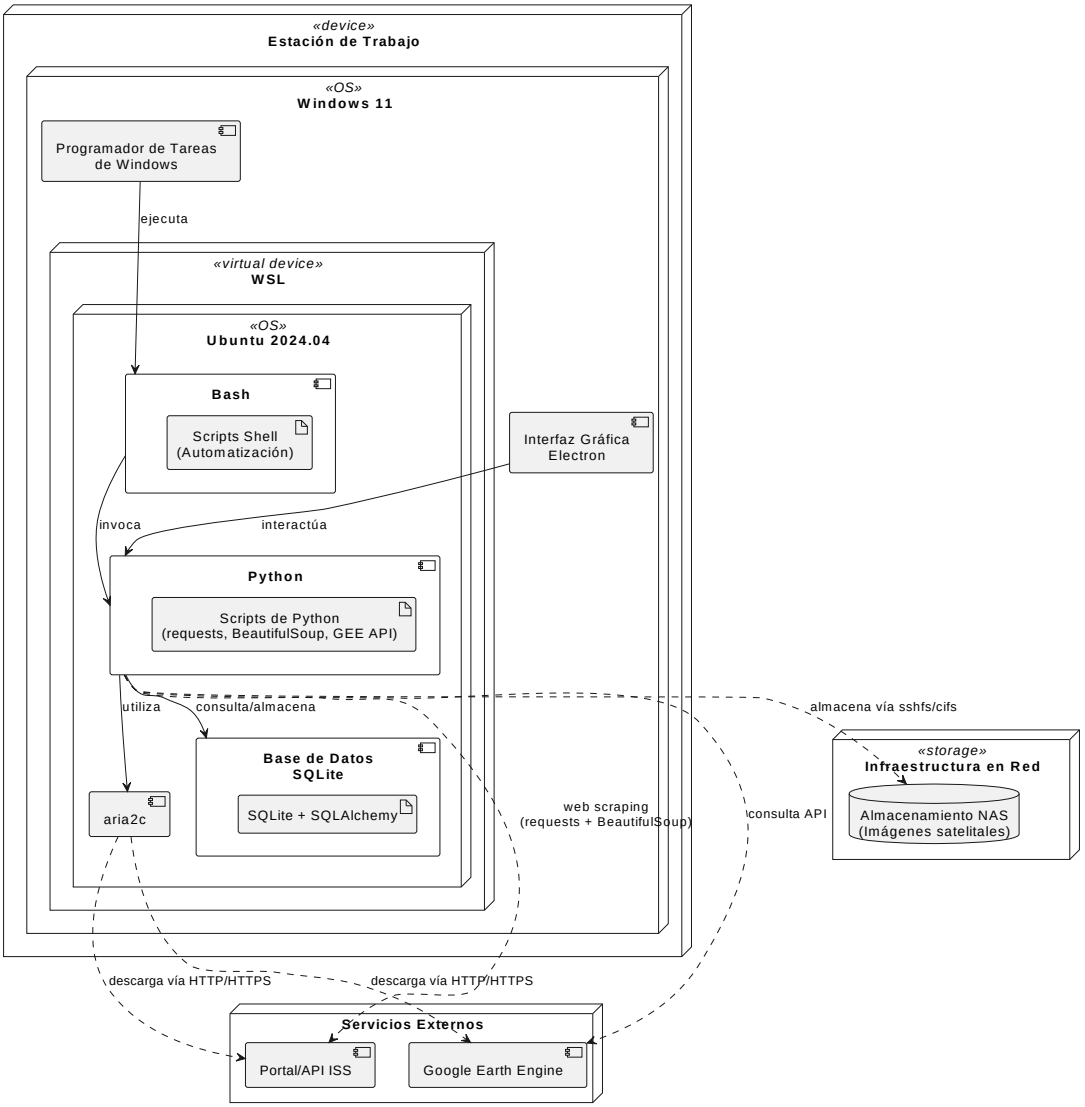

Figura 4.65: Diagrama de despliegue del sistema en entorno mixto (WSL2 + NAS).

Los elementos fundamentales de la infraestructura de despliegue incluyen:

- Estación de trabajo principal equipada con Windows 11 y subsistema WSL2 ejecutando Ubuntu 24.04 LTS, que proporciona el entorno de ejecución principal para los componentes del sistema.
- Scripts de automatización desarrollados en Python y Shell Bash, alojados y ejecutados desde el subsistema Linux, aprovechando las bibliotecas nativas para el procesamiento de datos geoespaciales.
- Programador de tareas de Windows configurado meticulosamente para ejecutar los scripts automáticamente según tarea programada por el

usuario, minimizando la carga de red y optimizando el rendimiento.

- Sistema de almacenamiento centralizado implementado en un servidor NAS, montado en la estación de trabajo mediante `sshfs` de forma remota y con `cifs` de forma local, que garantiza la persistencia y disponibilidad de los datos.

- Repositorio de código fuente alojado en GitHub, con control de versiones distribuido y autenticación segura mediante claves SSH, facilitando el mantenimiento continuo.

4.6.3 Automatización Periódica y Generación de Logs

Para garantizar la operación desatendida y continua del sistema, se han implementado tareas periódicas que ejecutan los módulos de adquisición y organización de datos sin requerir intervención manual. Estas tareas programadas activan scripts que realizan las siguientes operaciones:

- Consultan sistemáticamente las APIs y fuentes de datos configuradas para identificar y descargar nuevas imágenes disponibles, evitando duplicaciones innecesarias.

- Almacenan los archivos en el servidor NAS siguiendo una estructura jerárquica predefinida, optimizando tanto el espacio de almacenamiento como la velocidad de acceso.

- Documentan exhaustivamente toda actividad del sistema en archivos de log con marcas de tiempo precisas, facilitando la auditoría y resolución de problemas potenciales.

Este enfoque de automatización ha eliminado la necesidad de supervisión constante, permitiendo que el sistema opere de manera autónoma durante periodos prolongados mientras mantiene un registro detallado de todas sus operaciones.

4.6.4 Control de Versiones y Seguridad del Sistema

Todo el código fuente del sistema se gestiona mediante el sistema de control de versiones Git y se encuentra alojado en un repositorio privado en la plataforma GitHub. Para garantizar la seguridad e integridad del desarrollo, se han implementado las siguientes prácticas:

- Utilización de autenticación SSH con claves de alta entropía para todas las operaciones con el repositorio remoto, eliminando la dependencia de contraseñas.
- Estructura organizada de ramas por módulo funcional que permite el desarrollo paralelo y la integración controlada de nuevas características:
  - `main`: rama estable que contiene exclusivamente código probado y funcional.
  - `scraping-nasa`, `api-nasa` ,`gee-module`, `gui-utils`: ramas dedicadas al desarrollo de componentes específicos.
- Configuración rigurosa del archivo `.gitignore` para excluir automáticamente archivos binarios, datos sensibles, credenciales y claves de API del control de versiones.
- Implementación de revisiones de código obligatorias antes de la integración en la rama principal, asegurando la calidad y seguridad del código.

Estas medidas garantizan tanto la trazabilidad completa del desarrollo como la protección de datos sensibles y credenciales de acceso utilizadas por el sistema.

4.6.5 Estado Operativo Actual

El sistema se encuentra actualmente en un estado completamente operativo, cumpliendo el Objetivo Específico 7 mediante su despliegue exitoso en una estación de trabajo local equipada con WSL2, utilizando conexiones remotas seguras vía SSH para su gestión y operación integrada con el entorno NAS.

Demostrando las siguientes capacidades:

- Funcionamiento autónomo con actualización diaria de su base de imágenes, incorporando nuevos datos de manera consistente y fiable sin intervención manual.
- Gestión eficiente de un catálogo que supera las 3000 imágenes provenientes de diversas misiones satelitales, incluyendo ISS, VIIRS y DMSP-OLS, todas ellas correctamente categorizadas y accesibles.
- Organización óptima de los datos en una estructura jerárquica por sensor, año y tipo, facilitando tanto el acceso programático como la navegación manual de los archivos.
- Mantenimiento de un sistema exhaustivo de logs y metadatos que permite la auditoría completa de operaciones y proporciona la base para análisis posteriores de los datos recopilados.
- Resiliencia ante fallos de red o interrupciones temporales, con capacidad de recuperación automática y continuación de operaciones sin pérdida de datos.

Conclusiones

El sistema automatizado desarrollado cumplió exitosamente con el objetivo general de facilitar la descarga, organización y almacenamiento de imágenes satelitales nocturnas, enfocadas en la cuenca hidrográfica del Canal de Panamá. Mediante la integración de diversas tecnologías como APIs, scripts en Python, Google Earth Engine, herramientas de automatización y almacenamiento en un NAS, se estableció un flujo de trabajo eficiente y sostenible. Asimismo, los objetivos específicos se alcanzaron de manera satisfactoria. Entre ellos destacan la descarga masiva de imágenes a través de aria2c, la estructuración de metadatos en una base de datos relacional, y la automatización completa del proceso, eliminando la necesidad de intervención manual. Cada uno de los cinco incrementos implementados fortaleció la funcionalidad del sistema, aportando a su escalabilidad y versatilidad.

La incorporación de datos provenientes de diferentes fuentes satelitales, como la ISS, VIIRS y DMSP-OLS, permitió construir una base de datos multitemporal y multiespectral, altamente útil para el análisis espacial de la contaminación lumínica. La resolución visual ofrecida por la ISS, combinada con la cobertura histórica de VIIRS y DMSP-OLS, brindó un enfoque complementario y robusto que enriqueció los resultados obtenidos.

Además, la arquitectura modular del sistema, el uso de tecnologías de código abierto y la orientación a tareas programadas consolidan su valor como una herramienta técnica confiable. Su diseño permite adaptarlo a otros contextos geográficos y a distintos sensores satelitales, lo que facilita su aplicación en estudios ambientales y geoespaciales diversos.

Finalmente, si bien se enfrentaron algunas limitaciones como la ausencia de georreferenciación automática en muchas imágenes provenientes de la ISS y las diferencias de formato entre plataforma, se ha demostrado que es viable automatizar de forma efectiva la adquisición de datos para investigaciones ambientales avanzadas.

Recomendaciones

Con base en el desarrollo e implementación del sistema de descarga automática de imágenes satelitales nocturnas, se considera necesario evaluar la migración hacia una plataforma web institucional que permita el acceso remoto por parte de investigadores, técnicos y entidades interesadas en la consulta y descarga de datos, eliminando así la dependencia de operaciones locales y ampliando significativamente el alcance del proyecto.

Paralelamente, resulta fundamental explorar la integración de nuevas fuentes satelitales, especialmente productos especializados como Black Marble de NASA o imágenes de alta resolución provenientes de satélites comerciales, siempre que sus licencias, resolución temporal y disponibilidad se alineen con los objetivos específicos del proyecto. Esta diversificación de fuentes enriquecería considerablemente la base de datos disponible para análisis de contaminación lumínica.

La sostenibilidad del sistema requiere el desarrollo de documentación técnica integral y procedimientos de despliegue reproducibles que incluyan instrucciones detalladas de instalación, configuración, mantenimiento y actualización. Esta documentación garantizará la continuidad operativa del sistema y facilitará su gestión por parte de futuros técnicos o desarrolladores que se incorporen al proyecto.

Desde el punto de vista técnico, se recomienda implementar herramientas de monitoreo y respaldo automático que permitan auditar continuamente el estado de las descargas, optimizar el uso del almacenamiento NAS y verificar la integridad de los archivos almacenados. Los respaldos programados aumentarían significativamente la robustez del sistema ante posibles fallos inesperados o pérdidas de información.

La arquitectura modular desarrollada debe preservarse y expandirse estratégicamente para facilitar la escalabilidad del sistema, permitiendo que futuras funcionalidades, nuevos sensores o regiones geográficas adicionales puedan integrarse sin requerir reestructuraciones completas de la plataforma existente.

Finalmente, se sugiere evaluar la interoperabilidad del sistema con otras

plataformas de análisis científico, servicios de visualización existentes y entornos de investigación colaborativa, lo que podría promover su reutilización en diversos contextos científicos y fomentar el desarrollo de investigaciones interdisciplinarias en el campo de la contaminación lumínica y estudios ambientales relacionados.

Referencias Bibliográficas

[1] P. Richard and P. Jeyaraj, “Advances in agricultural biotechnology,” in *Proceedings of the International Conference on Agricultural and Biological Sciences*. ResearchGate, 2022. [Online]. Available: https://www.researchgate.net/publication/366095148_Advances_in_Agricultural_Biotechnology

[2] L. Gomes *et al.*, “An overview of platforms for big earth observation data management and analysis,” *Remote Sensing*, vol. 12, no. 8, 2020.

[3] S. Bará and F. Falchi, “Artificial light at night: a global disruptor of the night-time environment,” *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 378, no. 1892, p. 20220352, 2023. [Online]. Available: https://doi.org/10.1098/rstb.2022.0352

[4] Real Academia Española, “Contaminación lumínica,” s.f., consultado el 4 de marzo de 2025. [Online]. Available: https://dpej.rae.es/lema/contaminaci%C3%B3n-lum%C3%ADnica

[5] R. R. Coronado-Orrillo, J. E. Matienzo-Mendoza, M. Guanilo-Delgado, and M. E. Zevallos-Loyaga, “La contaminación lumínica en las áreas urbanas del perú. revisión sistemática,” *Cienciamatria. Revista Interdisciplinaria de Humanidades, Educación, Ciencia y Tecnología*, vol. 10, no. 19, pp. 347–361, 2024.

[6] A. Stoken and K. Fisher, “Find my astronaut photo: Automated localization and georectification of astronaut photography,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2023, pp. 6196–6205.

[7] A. Stoken, P. Ilhardt, M. Lambert, and K. Fisher, “(street) lights will guide you: Georeferencing nighttime astronaut photography of earth,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2024, pp. 492–501.

[8] Y. Chen, B. Zhao, F. Zhang, and W. Huang, “Cloud-based storage and computing for remote sensing big data: A technical review,” *International Journal of Digital Earth*, vol. 15, no. 1, pp. 1–20, 2022.

[9] E. Dritsas and M. Trigka, “Remote sensing and geospatial analysis in the big data era: A survey,” *Remote Sensing*, vol. 17, no. 3, p. 550, 2025.

[10] W. Zhang, M. Liu, and L. Chen, “Design of scalable web gis microservice framework for geospatial data access,” in *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. X-5/W1-2023, 2023, pp. 45–52.

[11] A. Brenning and S. Henn, “Web scraping: a promising tool for geographic data acquisition,” *Preprint*, 2023. [Online]. Available: https://www.researchgate.net/publication/371175762

[12] J. M. González García and J. A. Pérez Pérez, “Modelo incremental en el desarrollo de software,” *Revista Cubana de Ingeniería*, vol. 12, no. 5, p. e1046, 2021, iSSN 2218-3620. [Online]. Available: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2218-36202021000500175

Anexos

```
import requests

# Definicion de parametros
base_url = (
    "https://eol.jsc.nasa.gov/SearchPhotos/"
    "PhotosDatabaseAPI/PhotosDatabaseAPI.pl"
)

query = (
    "frames|lat|ge|7.0|frames|lat|le|9.6|"
    "frames|lon|ge|-83.1|frames|lon|le|-77.2|"
    "frames|elev|lt|0"
)

# Campos que se desean recuperar
return_fields = "images|directory|images|filename"

# Clave de API (si aplica)
api_key = "your_key"

# Parametros de la solicitud
params = {
    "query": query,
    "return": return_fields,
    "key": api_key
}

# Solicitud GET a la API
response = requests.get(base_url, params=params)

# Conversion de respuesta JSON (formato no formateado)
data = response.json()

# Extraccion del primer resultado
if data:
    image = data[0]
    directory = image["images.directory"]
```

```
    filename = image["images.filename"]

    # Construccion de la URL de la imagen
    image_url =
    f"https://eol.jsc.nasa.gov/DatabaseImages/{directory}/{filename}"

    # Descarga de la imagen
    img_data = requests.get(image_url).content
    with open(filename, "wb") as f:
        f.write(img_data)
else:
    print("No se encontraron resultados.")
```

Código 1: Consulta a la API de NASA Gateway con filtros espaciales y condición nocturna (Python)

```
var roi = ee.Geometry.Rectangle([-83.1, 7.0, -77.2, 9.6]); // Region de Panama

var viirs = ee.ImageCollection('NOAA/VIIRS/DNB/MONTHLY_V1/VCMSLCFG')
              .filterDate('2023-01-01', '2023-01-31')
              .filterBounds(roi)
              .median()
              .clip(roi);

Map.centerObject(roi, 8);
Map.addLayer(viirs, {min: 0.0, max: 0.3}, 'VIIRS enero 2023');

// Exportar a Google Drive
Export.image.toDrive({
  image: viirs,
  description: 'VIIRS_Panama_Enero2023',
  region: roi,
  scale: 500,
  maxPixels: 1e13
});
```

Código 2: Ejemplo de script en Google Earth Engine para el sensor VIIRS (JavaScript)