UNIVERSIDAD SIGLO 21
La educación evoluciona

Trabajo Práctico 3

Materia: SEMINARIO DE PRÁCTICA DE INFORMATICA – INF275

Carrera: Licenciatura en informática.

Titular Experto: Pablo Virgolini

Titular Disciplinar: Hugo Frias

Alumno: Sebastian Sanfilippo

Año: 2025

Fecha de entrega:05/10/2005

ÍNDICE

TABLA DE CONTENIDOS

SIZPOPE — Sistema de Identificación de Zonas Potenciales de Pesca (PFZ) del Mar Argentino

Introducción

SIZPOPE (Sistema de Identificación de Zonas Potenciales de Pesca) es una plataforma informática orientada a servicios que integra datos satelitales de libre acceso para proveer información operativa en tiempo casi real al sector pesquero y a organismos científicos y de control del Mar Argentino. A partir de productos Nivel 2 de Temperatura Superficial del Mar (SST) y Clorofila-a (Chl-a) —provenientes de misiones como Sentinel-3, NOAA-20/21, S-NPP, Terra y Aqua— el sistema ejecuta procesos de fusión, normalización y control de calidad que permiten estimar y visualizar Zonas Potenciales de Pesca (PFZ) con trazabilidad de metadatos.

El enfoque se centra en identificar frentes térmicos y áreas de alta productividad primaria como indicadores de caladeros, combinando gradientes de temperatura y concentración de clorofila en ventanas temporales configurables y, cuando corresponde, parametrizadas por especie objetivo. Los resultados se publican como capas geoespaciales (WMTS/WMS) consumibles en un visor web que ofrece gestión de Áreas de Interés (AOI), línea de tiempo, consulta por píxel y suscripciones de alertas cuando el índice PFZ supera umbrales definidos dentro de las AOI.

La propuesta busca reducir tiempos de búsqueda y costos de operación, mejorar la seguridad y favorecer prácticas de pesca sostenibles a través de una "fuente única de verdad" abierta y verificable. La arquitectura por capas (datos, negocio y presentación) y la adopción de estándares abiertos garantizan escalabilidad, interoperabilidad y facilidad de evolución hacia nuevas plataformas, sensores y funcionalidades sin comprometer la coherencia del diseño ni la reproducibilidad de los resultados.

Justificación

1. **Optimización operativa**: focaliza esfuerzos donde coexisten condiciones oceanográficas favorables (rango térmico por especie + alta productividad primaria) reduciendo millas navegadas y consumo de combustible.

2. **Aumento de capturas y selectividad**: dirigir la prospección a PFZ incrementa la probabilidad de éxito y promueve selectividad por especie.
3. **Sostenibilidad**: disminuir esfuerzo en zonas menos productivas y apoyar la gestión basada en evidencia.
4. **Acceso abierto**: aprovecha datos satelitales de libre disponibilidad, democratizando el acceso a información de alto valor.
5. **Gestión del riesgo y seguridad**: integra pronósticos/estado del mar (capacidad de extensión) y reduce exposición innecesaria.

Definiciones del proyecto y del sistema

Alcance (MVP del prototipo)

El proyecto abarca el análisis, diseño, desarrollo de un prototipo funcional implementado según requerimiento con Java & MySQL, y despliegue inicial de un sistema de generación de mapas. Se incluirá la integración de fuentes de datos satelitales como los NOAA Class y/o JPSS de la NASA, la creación de una base de datos de usuarios y ubicaciones, y el desarrollo de una interfaz de visualización en un mapa georreferenciado. El prototipo incluirá la funcionalidad de registro de usuarios y la visualización de productos PFZ.

Objetivos del proyecto

Automatizar ingestión L2 (SST/Chl-a):

Implementar la decarga y registro automatico escenas L2 de SST y Chl-a de al menos 4 plataformas (p. ej., Sentinel-3, NOAA-20/21, S-NPP) con una frecuencia mínima diaria, logrando ≥95% de ejecuciones exitosas durante un piloto de 30 días.

Calcular índice PFZ por celda con latencia controlada:

Desplegar el servicio de cálculo (PFZModel) para generar un run diario con latencia extremo-a-extremo ≤90 minutos desde la disponibilidad de las escenas, alcanzando ≥90% de cobertura espacial útil (en días sin nubosidad extrema) durante el piloto.

Publicar mapas PFZ/SST/Chl-a como tiles OGC:

Exponer capas WMTS/XYZ para PFZ, SST y Chl-a con tiempo de respuesta p95 ≤500 ms desde caché de tiles y p95 ≤1200 ms para cache miss, medido sobre 7 días de operación continua antes del primer hito.

Visor web operativo con AOI y consulta por píxel:

Entregar un visor web que permita crear/editar/eliminar AOI, visualizar timeline y consultar valores por coordenada (PFZ, SST, Chl-a), alcanzando una puntuación SUS ≥75 en pruebas de usabilidad con ≥10 usuarios objetivo antes del segundo hito.

Alertas por especie/umbral en AOI:

Habilitar suscripciones configurables (especie, umbral, AOI) y notificaciones por email/push cuando el índice PFZ supere el umbral en la AOI, con tasa de entrega ≥98% y tiempo de disparo ≤15 min tras el run diario, durante 4 semanas consecutivas.

Exportación interoperable:

Permitir la descarga de resultados en GeoTIFF (raster) y GeoJSON (isócronas/contornos), validando que 100% de los archivos cumplan con CRS EPSG:4326 y metadatos mínimos (fuente, fecha, versión de modelo) para el tercer hito.

Seguridad y cumplimiento básico:

Incorporar autenticación y autorización (RBAC), cifrado TLS y auditoría de eventos críticos, con 0 findings críticos en un pentest de caja gris previo al pase a pre-producción.

Disponibilidad y observabilidad:

Lograr ≥99,5% de disponibilidad mensual del backend y del servicio de mapas, instrumentando métricas, logs correlacionados y alertas (SLOs) en un tablero único antes del cuarto hito.

Escalabilidad controlada:

Demostrar que el sistema soporta ≥3× el volumen base de escenas y ≥20 usuarios concurrentes en el visor, manteniendo p95 ≤1,5 s por tile dinámico, verificado en un test de carga previo al cierre del proyecto.

Definición del sistema SIZPOPE

Descripción general.

El SIZPOPE será un sistema orientado a servicios que, mediante el procesamiento de datos satelitales de acceso público, identificará potenciales zonas de pesca. La detección de estas zonas es algo que no recae en el software, sino que lo realizan los procesadores y son parte de los productos estándar de los procesadores de imágenes satelitales. Una vez finalizada la adquisición de los datos estos quedan disponibles en diferentes fuentes públicas y se presentarán en un mapa georreferenciado al que podrán

acceder los usuarios registrados. Los usuarios podrán definir áreas de interés para recibir alertas personalizadas vía el medio de notificación seleccionado.

Funcionamiento (alto nivel).

1. **Ingreso de datos**: descarga programada de L2 SST y L2 Chlor-A de múltiples misiones.

2. **Preprocesamiento**: correcciones y normalización (máscara de nubes, QC por banderas de calidad, reproyección, mosaico y compositado).

3. **Fusión/Modelado PFZ**: cálculo de **índice PFZ** por celda (0–1) combinando: (a) cercanía al **rango térmico** por especie; (b) **percentil** local de Chlor-A; (c) **gradientes térmicos** (frentes); (d) continuidad espacial en ventana móvil.

4. **Persistencia**: metadatos en **MySQL** (escenas/productos, AOIs, usuarios, versiones de modelo). Ráster PFZ y capas de entrada como COG/GeoTIFF con tiling para mapas.

5. **Publicación**: API REST en Java y servidor de mapas (p.ej., GeoServer) para WMS/WMTS/XYZ tiles.

6. **Alertas**: coincidencia AOI × PFZ>umbral → notificación (email/web/push).

Suposiciones y dependencias. Disponibilidad estable de catálogos de datos; conectividad robusta; capacidad de almacenamiento; librerías geoespaciales (GDAL/PROJ); servidores de mapas; autenticación/autorizarción (OIDC/OAuth2).

Elicitación

La elicitación de requisitos para SIZPOPE se planteó como un proceso iterativo y controlado, orientado a capturar necesidades reales del dominio pesquero y a traducirlas en comportamientos verificables del sistema. El objetivo fue entender cómo, cuándo y para qué se utilizan las variables oceanográficas (SST y Chl-a) en la identificación de caladeros, qué decisiones operativas dependen de esa información y cuáles son las restricciones técnicas, regulatorias y operativas que condicionan su uso en campo. Para ello se definió un alcance inicial centrado en el Mar Argentino y en usuarios con perfiles diferenciados (capitanes y oficiales de puente, analistas oceanográficos, planificadores de flota, operadores del centro de monitoreo y administradores de plataforma), con el propósito explícito de cubrir tanto necesidades tácticas (día a día) como estratégicas (planeamiento y evaluación histórica).

Métodos de Elicitación

La estrategia metodológica combinó entrevistas semiestructuradas y observación de prácticas operativas con análisis documental y prototipado evolutivo. Las entrevistas se utilizaron para explorar flujos críticos

(definición de AOI, lectura de mapas PFZ/SST/Chl-a, consulta por píxel, exportaciones y alertas), descubrir reglas de negocio (umbrales por especie, ventanas temporales, criterios de calidad de escena, políticas de acceso) y relevar restricciones de entorno (conectividad variable a bordo, necesidades de cache y trabajo desconectado, formatos de intercambio). La observación permitió contrastar discursos con uso real de herramientas cartográficas, identificar cuellos de botella (latencia de tiles, paletas poco legibles, CRS inconsistentes) y registrar artefactos existentes (planillas, reportes de marea, avisos a la navegación). El análisis documental abarcó lineamientos técnicos de productos L2, convenciones de metadatos, estándares OGC y prácticas de trazabilidad. Con los hallazgos preliminares se construyeron prototipos de baja y media fidelidad del visor y formularios (AOI, suscripciones, exportación), que se emplearon en walkthroughs para validar lenguaje, flujos y tolerancias de error. Cuando fue pertinente, se aplicaron cuestionarios breves para medir percepción de utilidad y facilidad de uso, y se recogieron métricas objetivas (tiempos de tarea, tasa de éxito, errores por sesión) como insumo para los objetivos de usabilidad.

El proceso produjo un conjunto de artefactos interrelacionados: mapa de stakeholders y responsabilidades, catálogos de objetivos operativos y métricas de éxito, historias de usuario y escenarios de uso, modelos de dominio preliminares, glosario controlado (términos oceanográficos y geoespaciales), y una primera matriz de trazabilidad que vincula objetivos ↔ requisitos funcionales/no funcionales ↔ casos de uso ↔ criterios de aceptación. La priorización se realizó con MoSCoW, distinguiendo capacidades indispensables para la operación (p. ej., ingestión L2, cálculo PFZ, visor con AOI e inspección por píxel, publicación WMTS y alertas) de funcionalidades deseables pero diferibles (exportaciones avanzadas, CDNs regionales, aprendizaje de preferencias por usuario). Paralelamente, se documentaron supuestos (disponibilidad periódica de escenas y metadatos confiables, adopción de EPSG:4326 como CRS base, políticas de acceso por rol) y restricciones (latencia objetivo en near real time, límites de ancho de banda a bordo, cumplimiento de seguridad y auditoría).

Para asegurar verificabilidad, cada requisito se formuló con criterios de aceptación claros (datos de entrada, comportamiento observable, salida esperada y tolerancias), apoyándose en ejemplos concretos: intersección PFZ>umbral dentro de un AOI para disparo de alerta en menos de 15 minutos; tiempos de respuesta p95 para tiles desde caché y con cache miss; validaciones topológicas de AOI y consistencia de CRS en exportaciones. Asimismo, se identificaron riesgos y se definieron mitigaciones tempranas: cobertura útil reducida por nubosidad (composición temporal y multisensor), latencias por picos de demanda (precálculo de tiles, edge cache), variabilidad de conectividad (modo degradado y reintentos), y sesgos por parametrización de especie (versionado de modelos y rollbacks). Finalmente, se estableció un plan de iteraciones de elicitación con puntos de control al cierre de cada hito, de forma que los aprendizajes de uso real retroalimenten los requisitos, actualicen la matriz de trazabilidad y mantengan alineados objetivos, casos de uso y roadmap técnico.

Conocimiento del negocio

Contexto: El sector pesquero del Mar Argentino opera con altísima variabilidad espacio-temporal: masas de agua, frentes térmicos, viento y disponibilidad de nutrientes condicionan la presencia de cardúmenes.

Las decisiones clave (¿dónde operar?, ¿cuándo mover la flota?, ¿qué esfuerzo asignar?) se toman con ventanas de tiempo cortas y bajo incertidumbre. SIZPOPE se posiciona como un sistema de apoyo operativo que convierte datos satelitales abiertos (SST y Chl-a L2) en inteligencia geoespacial accionable (PFZ) para planificar, ejecutar y evaluar la actividad.

Flujo actual (simplificado). Prospección por experiencia histórica, reportes informales y pocas fuentes oceanográficas integradas; decisiones con demoras y alta incertidumbre.

Dolores detectados. - Dispersión de fuentes de datos y barreras técnicas SIG. - Falta de producto integrado **SST+Chlor-A** listo para el puente de mando. - Costos crecientes por navegación "a ciegas". - Baja trazabilidad para justificar zonas sugeridas.

KPIs y criterios de aceptación.

- Cobertra útil diaria (%): área con datos válidos sobre área total de interés.
- Latencia extremo-a-extremo (ingesta→PFZ→tiles): objetivo ≤ 90 min.
- p95 de respuesta de tiles: ≤ 500 ms (hit), ≤ 1200 ms (miss).
- Hit-rate operativo: proporción de salidas a zonas con PFZ alto que resultan en captura efectiva.
- Tasa de entrega de alertas: ≥ 98% en ≤ 15 min tras PFZRun.
- Disponibilidad del servicio: ≥ 99,5% mensual (backend + mapas).
- SUS (usabilidad) del visor: ≥ 75 con usuarios representativos.

Propuesta de solución de arquitectura

Arquitectura propuesta

La solución se basará en una arquitectura orientada a servicios compuesta por tres capas principales:

1. Capa de datos
2. Capa de negocios
3. Capa de presentación

Este diseño modular permitirá desarrollar el sistema de manera eficiente, ya que la capa de negocio expondrá una API REST que puede ser consumida por cualquier tipo de cliente (una aplicación de escritorio, una interfaz web o una aplicación móvil). Esto garantiza la flexibilidad y escalabilidad del proyecto, al cumplir con los requisitos de utilizar Java y MySQL para el backend .

Capa de datos

- Persistencia operacional: SIZPOPE utiliza MySQL 8 comoBD relacional para gestionar usuarios, roles, permisos, parámetros por especie, catálogos de productos satelitales y auditoría. Las geometrías de AOI (polígonos y radios) se almacenan con tipos GEOMETRY y SRID adecuado.
- Datos geoespaciales/ráster: Los productos satelitales (SST, Clorofila-a y el índice PFZ resultante) se guardan como COG/GeoTIFF/NetCDF en almacenamiento de objetos o filesystem, y se publican como tiles a través de un servidor de mapas (p. ej., GeoServer). Para acelerar búsquedas por ubicación/tiempo (AOI × fecha, cobertura, calidad), puede complementarse con un índice geoespacial (p. ej., Elasticsearch con geo_shape para AOI y metadatos de escenas).

Capa de negocio

El núcleo del sistema está implementado en Java (Spring Boot). Se encarga de:

- Ingestar catálogos de SST/Chl-a (multisensor) y aplicar controles de calidad.
- Preprocesar (reproyección, máscaras de nubes, mosaicos, compositados) y modelar el puntaje PFZ por celda según parámetros por especie (rango térmico, percentiles de clorofila, frentes térmicos).
- Cruzar resultados con Áreas de Interés (AOI) del usuario y emitir alertas cuando el índice supere umbrales configurados.
- Orquestar la publicación de capas/títulos y exponerlas mediante API (REST/GraphQL), además de registrar trazabilidad y métricas operativas.

Capa de presentación

Se ofrece una aplicación web que consume la API para mostrar mapas interactivos con línea de tiempo, filtros por especie/fecha y herramientas de inspección de píxel. La visualización geográfica se implementa con OpenLayers o Leaflet, integrando capas PFZ/SST/Chl-a y leyendas dinámicas. Desde la UI se pueden definir AOI, descargar GeoTIFF/GeoJSON/PDF y configurar alertas.

Ventajas de la solución

- Automatización de la prospección: reduce trabajo manual; el sistema ingesta, procesa y publica PFZ de forma programada.
- Casi en tiempo real: acorta el ciclo “dato satelital → decisión”, mejorando la oportunidad de salida hacia zonas favorables.
- Decisiones coherentes: una fuente única de verdad (PFZ + metadatos + calidad) para flotas, analistas y autoridades.

- Accesible y usable: interfaz web responsiva; no requiere software SIG especializado en el punto de uso.
- Escalable y extensible: arquitectura modular que admite nuevas variables (corrientes, batimetría, viento) y nuevos sensores sin rediseños profundos.
- Trazabilidad y control: registro de versiones del modelo, parámetros por especie y cobertura/nubes para justificar recomendaciones.

Etapa de análisis

Funcionalidades y Requerimientos No Funcionales

En esta sección se establece la definición precisa del producto software, delimitando su alcance y sus principales características conforme a los objetivos del proyecto SIZPOPE. El análisis se articula en dos dimensiones clave: los Requerimientos Funcionales, que detallan el qué del sistema (las acciones y servicios que SIZPOPE debe ejecutar, como la gestión de eventos y la definición de áreas de interés), y los Requerimientos No Funcionales, que definen el cómo (las restricciones de calidad, rendimiento, seguridad y usabilidad) que aseguran la fiabilidad y el éxito operativo del sistema.

Requerimientos funcionales (RF).

ID	Descripción
RF-01	El sistema debe permitir registrar usuario, creando una cuenta con email y contraseña.
RF-02	El sistema debe permitir iniciar sesión autenticando credenciales
RF-03	El sistema debe permitir crear área de interés (polígono/radio como AOI)
RF-04	El sistema debe permitir editar un área de interés (AOI)
RF-05	El sistema debe permitir eliminar un área de interés (AOI)
RF-06	El sistema debe permitir descargar y registrar productos L2 de SST.
RF-07	El sistema debe permitir descargar y registrar productos L2 de Chl-a.
RF-08	El sistema debe calcular índice PFZ por celda.
RF-09	El sistema debe generar y exponer tiles/WMTS de PFZ.
RF-10	El sistema debe mostrar mapa PFZ en el visor web.
RF-11	El sistema debe mostrar mapa SST en el visor web.
RF-12	El sistema debe mostrar mapa Chl-a en el visor web.
RF-13	El sistema debe mostrar metadatos de la celda (SST/Chl-a/PFZ) en el visor web.
RF-14	El sistema debe notificar al usuario cuando PFZ supera umbral en su AOI.
RF-15	El sistema debe permitir configurar suscripción de alertas según especies/variables.
RF-16	El sistema debe permitir mostrar historial PFZ
RF-17	El sistema debe permitir habilitar una nueva fuente satelital nueva (plataforma/sensor).
RF-18	El sistema debe permitir crear usuario (admin)

RF-19	El sistema debe permitir editar usuario (admin)
RF-20	El sistema debe permitir deshabilitar usuario (admin)
RF-21	El sistema debe permitir Descargar PFZ en GeoTIFF/GeoJSON.

Requerimientos no funcionales (RNF).

ID	Descripción
RNF-01	El sistema debe procesar y mostrar los datos de los focos de calor con una latencia mínima para que la información sea "near real time".
RNF-02	El sistema debe proteger la información de los usuarios y garantizar la integridad de los datos. Se deben implementar mecanismos de autenticación y autorización.
RNF-03	El sistema debe ser capaz de manejar un aumento en el volumen de datos satelitales y en la cantidad de usuarios.
RNF-04	El sistema debe estar disponible 24/7, ya que los incendios pueden ocurrir en cualquier momento.
RNF-05	La interfaz de usuario debe ser intuitiva y fácil de usar, incluso para personal no técnico.
RNF-06	El código fuente del sistema debe ser modular y estar bien documentado para facilitar futuras actualizaciones, correcciones de errores y la adición de nuevas funcionalidades. Esto es crucial para un proyecto de código abierto.

Aplicación del Proceso Unificado de Desarrollo (PUD)

Incepción (2–3 semanas) - Alcance MVP, stakeholders, riesgos, costeo, visión de arquitectura. - Casos de uso claves definidos al 20–30% (nivel objetivo). Prototipo UI de mapas.

Elaboración (4–5 semanas) - Arquitectura base ejecutable: ingestión mínima (1 sensor SST + 1 Chlor-A), pipeline y tiles PFZ. - Mitigación de riesgos técnicos (proyecciones, máscara de nubes, performance de tiles). - Casos de uso al 80% y modelo de datos estabilizado.

Construcción (6–8 semanas) - Implementación completa del MVP: AOIs, alertas, descargas, panel de calidad, hardening. - Pruebas unitarias/integración, pruebas con usuarios piloto (capitanes/analistas SIG).

Transición (2 semanas) - Despliegue en entorno productivo; capacitación; manual de usuario; métricas de adopción y calidad.

Riesgos principales y mitigación - **Cobertura por nubes** → compositados multidiarios, fusión multisensor, gap-filling. - **Latencia de datos** → colas por producto; publicación incremental. - **Desalineación especie-parámetros** → calibración iterativa con CPUE/historicos; tablero de validación

Inicio del análisis: Casos de uso

ID	CU-01
Nombre	Registrar usuario
Propósito	Permitir a un usuario crear una nueva cuenta para acceder a funcionalidades autenticadas.
Actores	Usuario (capitán/analista) no registrado, Sistema de Autenticación
Precondiciones	• No existe una cuenta activa con el email provisto. • Disponibilidad del servicio de correo y base de datos.
Disparador	Solicitud de alta desde pantalla de registro.
Flujo básico	1. El usuario accede a la página de registro del sistema. 2. El sistema presenta un formulario de registro. 3. El usuario ingresa su nombre, apellido, dirección de correo electrónico, y una contraseña según politica de seguridad. 4. El sistema valida que los campos obligatorios estén completos y que la dirección de correo electrónico no esté ya registrada. 5. Si los datos son válidos, el sistema encripta la contraseña y crea un nuevo registro de usuario en la base de datos. 6. El sistema envía un correo electrónico de confirmación de registro al usuario.
Flujos alternativos	[A1 - Email ya registrado] 1. El sistema rechaza el alta. 2. Se sugiere recuperar contraseña. [A2 - Política de contraseña incumplida] 1. Se indica el requisito incumplido y se solicita corrección. [A3 - Falla del correo de verificación] 1. Se permite reintentar envío de correo o verificar más tarde.
Postcondiciones	• Cuenta creada, rol por defecto asignado, correo de verificación enviado. • El Usuario puede autenticarse.
Excepciones	• Corte de servicio de DB. • Corte de servicio de correo. • Error de validación no controlado
RF/RNF asociados	RF-01, RNF-02

ID	CU-02
Nombre	Iniciar sesión
Propósito	Permitir a el usuario autenticar credenciales y habilitar una sesión segura.
Actores	Usuario registrado, Sistema de Autenticación
Precondiciones	• Usuario registrado y verificado. • Servicio Auth y DB disponibles.
Disparador	Envío de credenciales en login.
Flujo básico	1. El usuario accede a la página principal del sistema. 2. El sistema presenta un formulario de autenticación. 3. El usuario ingresa su usuario y contraseña. 4. El sistema valida que los campos obligatorios estén completos y coincidan con los registrados. 5. El sistema informa al actor que ingreso correctamente y lo lleva a la pagina inicial.
Flujos alternativos	[A1 - Credenciales inválidas] 1. Se informa el error sin revelar si el email existe. 2. Se permite reintentar con rate limiting. [A2 - Usuario deshabilitado] 1. Se rechaza acceso y se sugiere contacto con el administrador.
Postcondiciones	• El Usuario se logueo al sistema tiene una sesión activa • Se emite token y registro en auditoría.
Excepciones	• Servicio de autenticación caído. • Rate limit por intentos, bloqueo de cuenta.
RF/RNF asociados	RF-02, RNF-02

ID	CU-03
Nombre	Crear AOI (Área de interés)
Propósito	Permitir a el usuario definir un área de interés como polígono o radio para suscripciones y análisis.
Actores	Usuario autenticado
Precondiciones	• Usuario registrado y verificado con sesión activa. • Servicio visor de mapas disponible.
Disparador	Selección de 'Nueva AOI' en el visor.

Flujo básico	1. El usuario navega a la sección de AOI. 2. El sistema presenta un mapa interactivo con herramientas de dibujo. 3. El actor utiliza las herramientas para dibujar un polígono sobre una ubicación. 4. El actor asigna un nombre y especie/parametrización opcional. 5. El actor confirma la creación del área de interés. 6. El sistema valida geometría (SRID 4326, no autointersectante). 7. El sistema guarda las coordenadas geográficas del área y el nombre asociado en la base de datos MySQL, vinculándolas a la cuenta del usuario. 8. El sistema muestra un mensaje de confirmación.
Flujos alternativos	[A1 - Geometría inválida] 1. Se informa error y se solicita corrección (autointersección, tamaño, SRID). [A2 - Límite de AOIs alcanzado] 1. Se rechaza la creación y se sugiere eliminar alguna existente.
Postcondiciones	• El usuario tiene un AOI creada y asociada a su cuenta. • El AOI queda disponible para intersección PFZ
Excepciones	• Falla de persistencia. • Timeout del servidor de mapas.
RF/RNF asociados	RF-03, RNF-5

ID	CU-04
Nombre	Editar área de interés (AOI)
Propósito	Permitir a el Usuario actualizar la geometría o atributos de un AOI existente del usuario.
Actores	Usuario autenticado
Precondiciones	• Usuario registrado y verificado con sesión activa. • Servicio visor de mapas disponible. • AOI existente del usuario.
Disparador	Selección de 'Editar AOI'.
Flujo básico	1. El Usuario navega a la sección de AOI y Selecciona 'Editar'. 2. Se selecciona el AOI en el mapa. 3. Se carga geometría/atributos. 4. Se modifican vértices/atributos. 5. Se valida y persiste con versión. 6. Se confirma actualización del AOI.

Flujos alternativos	[A1 - Geometría inválida] 1. Se informa error y se solicita corrección (autointersección, tamaño, SRID). [A2 - Validación falla] 1. Se rechazan cambios; se muestran errores específicos.
Postcondiciones	El usuario genera AOI actualizado.
Excepciones	• Falla de persistencia. • Timeout del servidor de mapas.
RF/RNF asociados	RF-04

ID	CU-05
Nombre	Eliminar área de interés (AOI)
Propósito	Permitir a el Usuario Eliminar un AOI del para que no se use en alertas ni análisis posteriores.
Actores	Usuario autenticado
Precondiciones	• Usuario registrado y verificado con sesión activa. • Servicio visor de mapas disponible. • AOI existente del usuario.
Disparador	Selección de 'Eliminar AOI'.
Flujo básico	1. El Usuario navega a la sección de AOI y Selecciona 'Eliminar'. 2. Se selecciona el AOI en el mapa. 3. El sistema solicita confirmación explícita. 4. El sistema borra o marca inactivo. 5. Se confirma eliminación de AOI.
Flujos alternativos	[A1 - Geometría inválida] 1. El sistema informa error y se solicita corrección (autointersección, tamaño, SRID). [A2 - Validación falla] 1. El sistema rechaza cambios; se muestran errores específicos.
Postcondiciones	• El AOI eliminado deja de considerarse en alertas/consultas. • El usuario libera un cupo de AOI
Excepciones	• Falla de persistencia. • Timeout del servidor de mapas.
RF/RNF asociados	RF-05

ID	CU-06
Nombre	Ingestar producto SST (L2)

Propósito	Descargar, validar y registrar escenas L2 de SST para su posterior procesamiento.
Actores	Usuario autenticado, Servicio planificador, Servicio de ingestión
Precondiciones	• Usuario registrado y verificado con sesión activa. • La fuente de datos satelital debe estar accesible. • Hay espacio de almacenamiento disponible.
Disparador	Job programado o ejecución manual de ingesta de datos
Flujo básico	1. El Usuario mediante el planificador dispara la tarea. 2. Se consulta catálogo por ventana temporal seleccionado el productos correspondiente. 3. Se descargan archivos L2 de SST y se verifica su integridad. 4. El sistema extrae metadatos clave para la identificacion. 5. El sistema registra escenas en el catálogo MySQL.
Flujos alternativos	[A1 - Falla de red/endpoint] 1. El sistema realiza reintentos con backoff; se muestra alerta operativa. [A2 - Archivo corrupto] 1. El sistema descarta y se intenta nueva descarga.
Postcondiciones	• El sistema registra escenas SST con metadatos. • El sistema realiza un QC inicial.
Excepciones	• Quota excedida. • almacenamiento insuficiente..
RF/RNF asociados	RF-06, RNF-01, RNF-03

ID	CU-07
Nombre	Ingestar producto Chl-a (L2)
Propósito	Descargar, validar y registrar escenas L2 de Chl-a para su posterior procesamiento.
Actores	Usuario autenticado, Servicio planificador, Servicio de ingestión
Precondiciones	• Usuario registrado y verificado con sesión activa. • La fuente de datos satelital debe estar accesible. • Hay espacio de almacenamiento disponible.
Disparador	Job programado o ejecución manual de ingesta de datos
Flujo básico	1. El Usuario mediante el planificador dispara la tarea. 2. Se consulta catálogo por ventana temporal seleccionado el productos correspondiente. 3. Se descargan archivos L2 de Chl-a y se verifica su integridad. 4. El sistema extrae metadatos clave para la identificacion. 5. El sistema registra escenas en el catálogo MySQL.

Flujos alternativos	[A1 — Falla de red/endpoint] 1. Reintentos con backoff; alerta operativa. [A2 — Archivo corrupto] 1. Se descarta y se intenta nueva descarga.
Postcondiciones	• El sistema registra escenas Chl-a con metadatos. • El sistema realiza un QC inicial.
Excepciones	• Quota excedida. • almacenamiento insuficiente..
RF/RNF asociados	RF-07, RNF-01, RNF-03

ID	CU-08
Nombre	Calcular índice PFZ
Propósito	Generar el índice PFZ por celda a partir de SST/gradiente térmico y Chl-a parametrizados por especie.
Actores	Usuario autenticado, Servicio PFZModel
Precondiciones	• Usuario registrado y verificado con sesión activa. • Porductos SST y Chl-a en grilla común. • Parámetros por especie vigentes.
Disparador	Job programado o ejecución manual de ingesta de datos
Flujo básico	1. El usuario navega hasta la seccion planificador y dispara la tarea de calcular PZF. 2. El sistema construye compuestos y máscaras de calidad. 3. El sistema calcula gradiente térmico y se normaliza Chl-a. 4. El sistema aplica la parametrización por especie selecionada. 5. El sistema computa score PFZ [0..1] y se versiona. 6. El sistema genera ráster PFZ y tiles preliminares.
Flujos alternativos	[A1 — Cobertura insuficiente (nubes)] 1. Se marca calidad baja o se combina multisensor. [A2 — Parámetros inconsistentes] 1. Se detiene el run; se solicita corrección.
Postcondiciones	• PFZRun generado; ráster PFZ y metadatos. • El sistema realiza un QC inicial. • Se muestra el mapa resueltante.
Excepciones	• Error en CRS. • Memoria insuficiente para mosaicos.
RF/RNF asociados	RF-08, RNF-01, RNF-03

ID	CU-09
Nombre	Publicar tiles/WMTS PFZ

Propósito	Publicar el resultado PFZ como tiles WMTS/XYZ para consumo por el visor.
Actores	Usuario autenticado, Servicio Mapas
Precondiciones	• Usuario registrado y verificado con sesión activa. • El PFZRun debe válido y el ráster estar accesible. • Servidor de mapas debe estar operativo
Disparador	Evento de finalización de cálculo PFZ.
Flujo básico	1. El sistema carga el producto generado por el calulo de PFZ. 2. El sistema configurara estilos, aplicara las leyendas, generara metadatos y los aplica sobre el producto a visualizar. 3. El sistema carga tiles críticos asociados al Área de interes AOI. 4. El sistema expone WMTS/XYZ en la ventana de visualización.
Flujos alternativos	[A1 — Falla de publicación] 1. Se realiza el Rollback y el reintento; se alerta al operador. [A2 — Parámetros inconsistentes] 1. Se detiene el run; se solicita corrección.
Postcondiciones	• PFZRun generado; ráster PFZ y metadatos. • El sistema realiza un QC inicial. • Se muestra el mapa resueltante.
Excepciones	• Error en CRS. • Memoria insuficiente para mosaicos.
RF/RNF asociados	RF-09, RNF-04

ID	CU-10
Nombre	Visualizar mapa PFZ
Propósito	Permitir al usuario visualizar la capa PFZ en el visor web.
Actores	Usuario autenticado, Servicio Mapas, Servicio de Visualización
Precondiciones	• Usuario registrado y verificado con sesión activa. • La capa PFZ debe estar publicada ser accesible accesible. • Servidor de mapas debe estar operativo
Disparador	Evento de finalización de cálculo PFZ.
Flujo básico	1. El sistema carga el producto generado por el calulo de PFZ. 2. El sistema configurara estilos, aplicara las leyendas, generara metadatos y los aplica sobre el producto a visualizar. 3. El sistema carga tiles críticos asociados al Área de interes AOI. 4. El sistema expone WMTS/XYZ en la ventana de visualización.
Flujos alternativos	[A1 — Falla de publicación] 1. Se realiza el Rollback y el reintento; se alerta al operador.

	[A2 — Parámetros inconsistentes] 1. Se detiene el run; se solicita corrección.
Postcondiciones	• Se realizaron los controles de calidad de la imagen • Se vizualiza PFZ con leyendas
Excepciones	• Error en CRS. • Memoria insuficiente para mosaicos.
RF/RNF asociados	RF-10, RNF-05

ID	CU-11
Nombre	Visualizar mapa SST
Propósito	Permitir al usuario visualizar la capa de SST para contexto térmico en visor web.
Actores	Usuario autenticado, Servicio Mapas, Servicio de Visualización
Precondiciones	• Usuario registrado y verificado con sesión activa. • La capa SST debe estar publicada ser accesible accesible. • Servidor de mapas debe estar operativo
Disparador	El usuario selecciona capa SST en la interfaz de visualización.
Flujo básico	1. El usuario navega hasta la seccion visualizacion y selecciona la capa SST a visualizar 2. El sistema consulta la base de datos de productos y recupera la información más reciente. 3. El sistema superpone el AOI con el producto SST en un mapa georreferenciado en la interfaz de usuario. 4. El usuario puede hacer clic en un punto para ver detalles, como la hora de detección, la fuente de datos y valores T. 5. El sistema actualiza el mapa periódicamente para mostrar nuevos eventos.
Flujos alternativos	[A1 — Falla de publicación] 1. Se realiza el Rollback y el reintento; se alerta al operador. [A2 — Parámetros inconsistentes] 1. Se detiene el run; se solicita corrección.
Postcondiciones	• Se muestra mapa SST • Se notifica al usuario.
Excepciones	• Error en CRS. • Fallo en servidor de mapas.
RF/RNF asociados	RF-11, RNF-05

ID	CU-12
Nombre	Visualizar mapa Chl-A
Propósito	Permitir al usuario Visualizar la capa de Clorofila-a para contexto de productividad en visor web.
Actores	Usuario autenticado, Servicio Mapas, Servicio de Visualización
Precondiciones	• Usuario registrado y verificado con sesión activa. • La capa Chl-A debe estar publicada ser accesible. • Servidor de mapas debe estar operativo
Disparador	El usuario selecciona capa Chl-A en la interfaz de visualización.
Flujo básico	1. El usuario navega hasta la seccion visualizacion y selecciona la capa SST a visualizar 2. El sistema consulta la base de datos de productos y recupera la información más reciente. 3. El sistema superpone el AOI con el producto Chl-A en un mapa georreferenciado en la interfaz de usuario. 4. El usuario puede hacer clic en un punto para ver detalles, como la hora de detección, la fuente de datos y valores de concentración de Chl-A. 5. El sistema actualiza el mapa periódicamente para mostrar nuevos eventos.
Flujos alternativos	[A1 — Falla de publicación] 1. Se realiza el Rollback y el reintento; se alerta al operador. [A2 — Parámetros inconsistentes] 1. Se detiene el run; se solicita corrección.
Postcondiciones	• Se muestra mapa Chl-A • Se notifica al usuario.
Excepciones	• Error en CRS. • Fallo en servidor de mapas.
RF/RNF asociados	RF-12, RNF-05

ID	CU-13
Nombre	Consultar Celda
Propósito	Permitir al usuario consultar valores (PFZ, SST, Chl-a) y metadatos para una coordenada.
Actores	Usuario autenticado, Servicio Mapas, Servicio de Visualización

Precondiciones	• Usuario registrado y verificado con sesión activa. • La capas deben estar publicadas ser accesibles. • Servidor de mapas debe estar operativo.
Disparador	El usuario realiza un Click sobre una celda especifica del mapa.
Flujo básico	1. Sobre un mapa que se esta visualizando el usuario hace clic sobre una celda especifica 2. El sistema reconoce el click y traduce la consulta. 2. Se envía la consulta con el query para identify (lat,lon,fecha,capa). 3. El sistema devuelve valores/flags correspondientes con las identificacion.. 4. El sistema muestra un popup con datos y metadatos.
Flujos alternativos	[A1 — Punto fuera de cobertura] 1. Se informa que no hay datos para esa coordenada/fecha. [A2 — Parámetros inconsistentes] 1. Los valores de la celda no son consistentes o fuera de escala.
Postcondiciones	• Popup con valores/metadatos; coordenadas capturadas. • Se notifica al usuario.
Excepciones	• Desfase de proyección; (Bowtie effect) • Error en modulo identify.
RF/RNF asociados	RF-13, RNF-05

ID	CU-14
Nombre	Configurar suscripción de alertas
Propósito	Permitir al usuario Definir especie, umbral y AOI para recibir notificaciones cuando PFZ supere umbral.
Actores	Usuario autenticado, Servicio Mapas, Servicio de Visualización
Precondiciones	• Usuario registrado y verificado con sesión activa. • El usuario debe tener al menos un AOI configurado.
Disparador	Apertura del módulo 'Alertas'.
Flujo básico	1. El usuario navega a la sección de preferencias de notificaciones. 2. El sistema presenta opciones de suscripción (por notificaciones para eventos confirmados por especie, umbral, AOI y frecuencia). 3. El usuario selecciona las opciones que le interesan. 4. El usuario guarda los cambios. 5. El sistema actualiza las preferencias del usuario en la base de datos MySQL.

Flujos alternativos	[A1 — Parámetros inválidos] 1. Informar error y solicitar corrección. [A2 — Usuario no guarda cambios] 1. El sistema no guarda los cambio en la confiugración de notificaciones
Postcondiciones	• Popup con valores/metadatos; coordenadas capturadas. • Se notifica al usuario.
Excepciones	• Desfase de proyección; (Bowtie effect) • Error en modulo identify.
RF/RNF asociados	RF-14, RNF-05

ID	CU-15
Nombre	Recibir alerta PFZ
Propósito	Notificar al usuario cuando PFZ supera el umbral dentro de sus AOI.
Actores	Sistema de noficifaciones, Usuario
Precondiciones	• El usuario debe tener suscripciones activas con AOI definida. • El sistema debe haber termiando un RunPFZ reciente. • Debe haber un canal de notificación disponible.
Disparador	PFZ ≥ umbral dentro de AOI del usuario.
Flujo básico	1. El sistema debe intersectar el producto generado PFZ>umbral con AOI por usuario. 2. El sistema generara un evento de alerta con la informacion de AOI, especie, scorey timestamp. 3. El sistema debera enviar la notificación correnpondiente con el metodo seleccionado (email/push/web). 4. El sistema registrara envio/entrega/lectura.
Flujos alternativos	[A1 — Canal no disponible] 1. El sistema reintenta y/o hace el fallback al metodo alternativo (si esta definido).
Postcondiciones	• La elerta es emitida y registrada por el sistema • El usuario es informado.
Excepciones	• Congestión del servicio de correo/push. • Error en modulo notify.
RF/RNF asociados	RF-15, RNF-02

ID	CU-16
Nombre	Mostrar historial PFZ
Propósito	Explorar series temporales/mapas PFZ históricos por fecha/área.
Actores	Usuario autenticado.

Precondiciones	• Usuario registrado y verificado con sesión activa. • El servicio historico debe estar disponible
Disparador	El usuario accede al módulo Histórico.
Flujo básico	1. El usuario navega la interfaz y selecciona Abrir 'Histórico'. 2. El usuario define el rango temporal y el AOI de interes. 3. El sistema filtra los datos en la base y presentando mapas/series y estadísticas. 4. El sistema ofrece exportar selección a GeoTIFF/GeoJSON (opcional).
Flujos alternativos	[A1 — Rango vacío] 1. Indicar que no hay datos para los filtros aplicados.
Postcondiciones	• Se vizualiza el listado en pantalla filtrados por criterios seleccionados. • El usuario es informado.
Excepciones	• Consulta fuera de rango valido.
RF/RNF asociados	RF-16, RNF-05

ID	CU-17
Nombre	Habilitar nueva fuente satelital
Propósito	Registrar una nueva plataforma/sensor y su configuración de ingestión.
Actores	Administrador
Precondiciones	• Rol administrador configurado y asignado. • El endpoint de la nueva fuenta de datos debe estar accesible. • Espacio disponible para nuevas descargas. • Credenciales configuradas para el endpoint.
Disparador	El administrador realiza el alta de producto (plataforma/sensor/variable).
Flujo básico	1. El administrador navega en la interfza hasta el menu 'Fuentes' y selecciona 'Nueva'. 2. El administrador completa en formulario con los datos de plataforma, sensor, variables y endpoint. 3. El sistema realiza la prueba de conectividad y valida el esquema. 4. El administrador guarda los cambios y habilita la nueva fuente. 5. El sistema programa la primer descarga.
Flujos alternativos	[A1 — Prueba de conectividad falla] 1. Rechazar alta; pedir corregir configuración.
Postcondiciones	• Nueva fuente de datos creada y habilitada. • Primer health check exitoso.

Excepciones	• Credenciales inválidas. • Formatos incompatibles con SST y Chlo-A
RF/RNF asociados	RF-17, RNF-03

ID	CU-18
Nombre	Administrar usuarios
Propósito	Permitir al administrador gestionar cuentas de usuarios
Actores	Administrador
Precondiciones	• Usuario con Rol Administrador con sesión activa.
Disparador	El administrador inicia la creación de nuevo usuario
Flujo básico	1. El administrador navega al panel de gestión de usuarios. 2. El sistema muestra una lista de todos los usuarios registrados. 3. El administrador selecciona un usuario para ver su perfil. 4. El administrador puede modificar la información del perfil del usuario (nombre, correo electrónico, etc.). 5. El administrador puede cambiar el estado de la cuenta (habilitar, deshabilitar). 6. El administrador puede eliminar la cuenta de un usuario. 7. El sistema confirma los cambios realizados.
Flujos alternativos	[A1 — Email ya registrado] 1. El sistema muestra un mensaje indicando el error y cancela la creación. [A2 — No hay usuarios registrados] 1. El sistema muestra un mensaje indicando que no hay usuarios para gestionar.
Postcondiciones	• El perfil del usuario es actualizado, desactivado o eliminado del sistema.
Excepciones	-
RF/RNF asociados	RF-18, RF-19, RF-20, RNF-02, RNF-03

ID	CU-19
Nombre	Exportar PFZ
Propósito	Permitir exportar productos PFZ en formatos GeoTIFF/GeoJSON para análisis externo.
Actores	Usuario Registrado, Analista
Precondiciones	• Usuario registrado y verificado con sesión activa. • Las capas deben estar publicadas y ser accesibles. • EL servidor de mapas debe estar operativo.
Disparador	El suaurio selecciona 'Descargar' en el visor.

Flujo básico	1. El usaurio en la interfaz de de visualizacón selecciona 'Exportar'. 2. El usuario selecciona área, rango temporal y formato. 3. El sistema valida los límites de tamaño y ventana temporal. 4. El sistema genera archivo y proveer enlace de descarga en la misma interfaz.
Flujos alternativos	[A1 — Límite excedido] 1. Sugerir reducir área/rango o usar exportación por lotes
Postcondiciones	• Se genera el archivo solicitado con CRS EPSG:4326 y metadatos mínimos.
Excepciones	• Fallo en conversión del formato a exportar • El almacenamiento no cuenta con espacio disponible
RF/RNF asociados	RF-21, RNF-05

Matriz de trazabilidad de requerimientos

ID Requisito	Req. Funcional/No Funcional	Caso de Uso	Clases de Dominio	Artefacto de implementación (Prototipo)
RF-01	Registrar usuario	CU-01	User, AuthService	Módulo de Registro (Formulario y lógica Java)
RF-02	Iniciar sesión	CU-02	User, AuthService	Módulo de Autenticación y Perfiles
RF-03	Crear AOI	CU-03	User, AOI	Módulo de Mapa Interactivo
RF-04	Editar AOI	CU-04	User, AOI	Módulo de Mapa Interactivo
RF-05	Eliminar AOI	CU-05	User, AOI	Módulo de Mapa Interactivo
RF-06	Ingestar L2 SST	CU-06	Product, Scene, Catalog	Módulo de Consumo de API Externa (Clase Java)
RF-07	Ingestar L2 Chl-a	CU-07	Product, Scene, Catalog	Módulo de Consumo de API Externa (Clase Java)

RF-08	Calcular índice PFZ	CU-08	PFZRun, PFZModel, Scene	Módulo de Mapa Interactivo
RF-09	Publicar tiles PFZ	CU-09	PFZRun, PFZTile, MapService	Módulo de Mapa Interactivo
RF-10	Mostrar mapa PFZ	CU-10	MapService, PFZRun	Módulo de Mapa Interactivo
RF-11	Mostrar mapa SST	CU-11	MapService, Scene	Módulo de Mapa Interactivo
RF-12	Mostrar mapa Chl-a	CU-12	MapService, Scene	Módulo de Mapa Interactivo
RF-13	Mostrar metadatos de celda	CU-13	MapService, PFZRun, Scene	Módulo de Mapa Interactivo
RF-14	Notificar alerta PFZ en AOI	CU-15	Alert, AOI, PFZRun, NotificationService	Módulo de Notificación (Lógica Java)
RF-15	Configurar suscripciones de alertas	CU-14	Subscription, AOI, User	Perfil de Usuario y Opciones
RF-16	Mostrar historial PFZ	CU-16	PFZRun	Módulo de Historial y Filtros
RF-17	Habilitar nueva fuente satelital	CU-17	Product, Catalog	Módulo de Configuración del Administrador
RF-18	Crear usuario (admin)	CU-18	User, AuthService	Módulo de Configuración del Administrador
RF-19	Editar usuario (admin)	CU-18	User, AuthService	Módulo de Configuración del Administrador
RF-20	Deshabilitar usuario (admin)	CU-18	User, AuthService	Módulo de Configuración del Administrador
RF-21	Descargar PFZ	CU-19	PFZRun, ExportService	Modulo de Descargas
RNF-01	Rendimiento (near real-time)	CU-06, CU-07, CU-08, CU-09, CU-10	IngestionJob, PreprocessJob, PFZModel, MapService	Optimización de consultas a la base de datos
RNF-02	Seguridad (authN/authZ, integridad)	CU-01, CU-02, CU-18,	Usuario, Administrador	Módulo de Autenticación y Perfiles

RNF-03	Escalabilidad (datos/usuarios/sensores)	CU-06, CU-07, CU-09, CU-17	Todas las Clases	Arquitectura del sistema (Java, MySQL)
RNF-04	Disponibilidad 24×7	CU-09, CU-15	Todas las Clases	Estrategia de despliegue y monitoreo del servidor
RNF-05	Usabilidad (UI intuitiva)	CU-03, CU-10, CU-11, CU-12, CU-13, CU-14, CU-16, CU-19	N/A (UX/UI)	Diseño de interfaz de usuario
RNF-06	Mantenibilidad/Modularidad	CU-17 (transversal)	Todas las Clases	Estándares de programación y documentación

Diagrama de casos de uso.

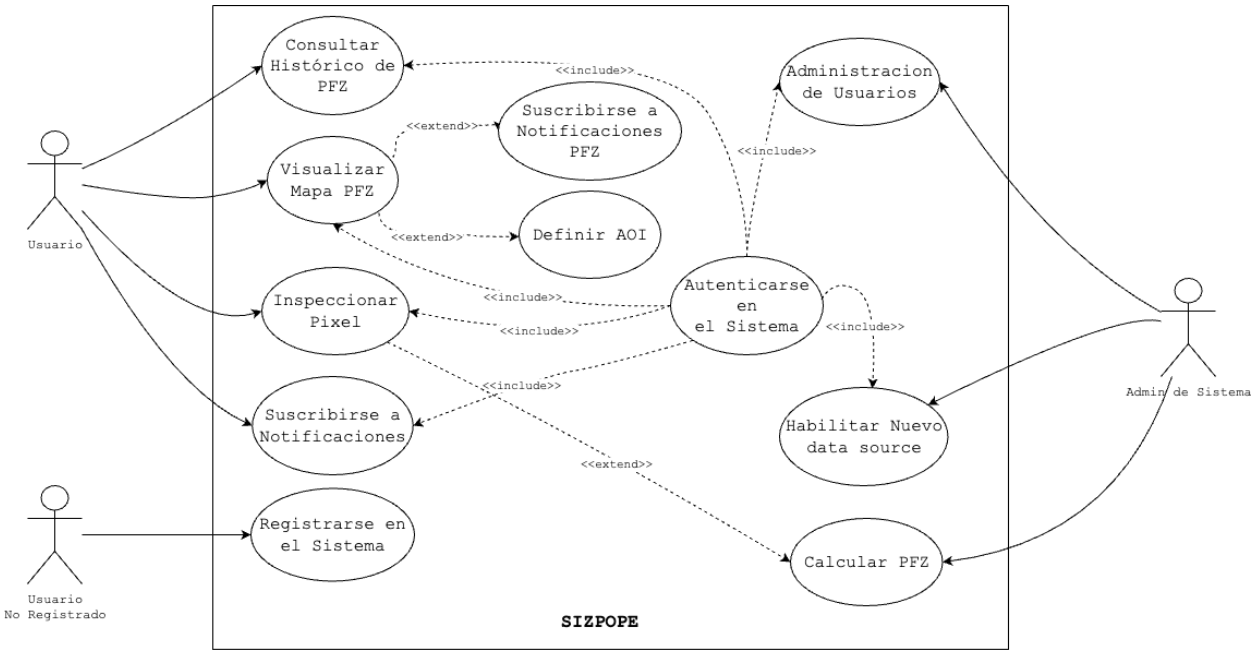
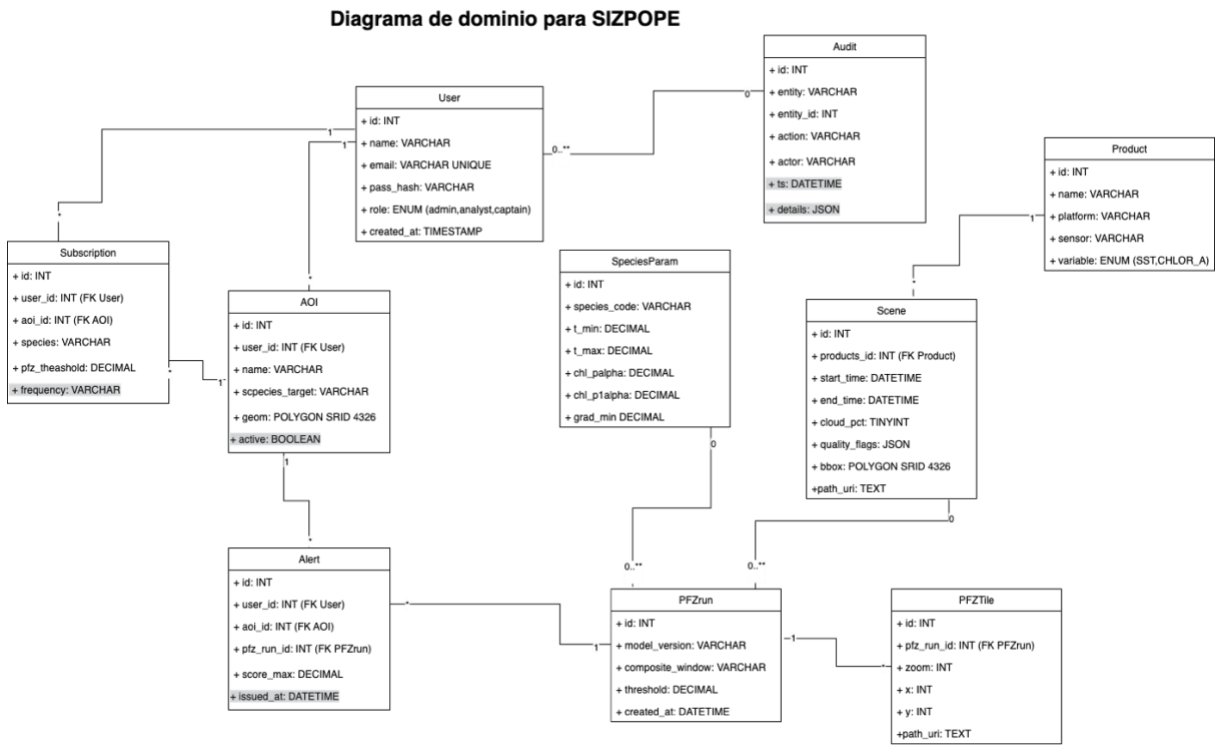
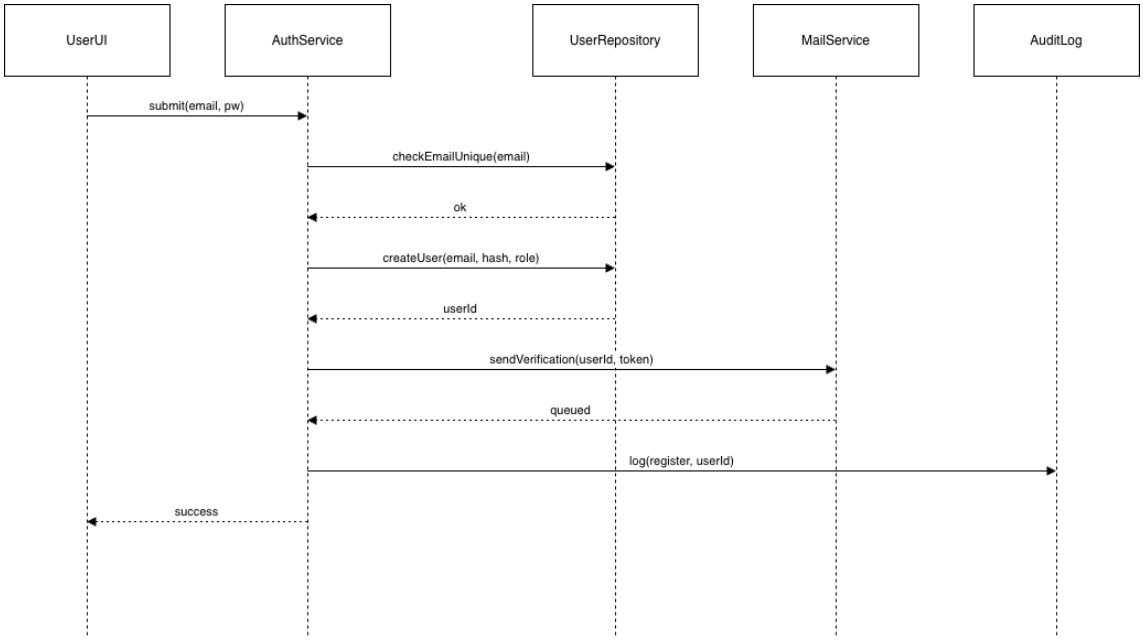


Diagrama de dominio

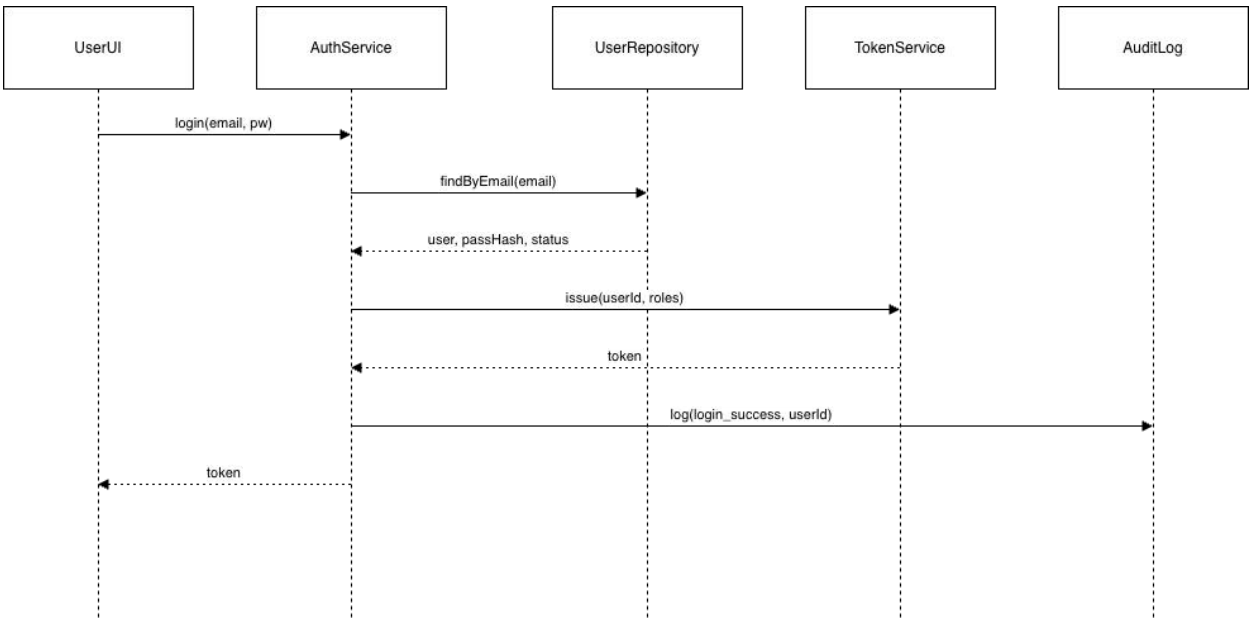


Diagramas de secuencia

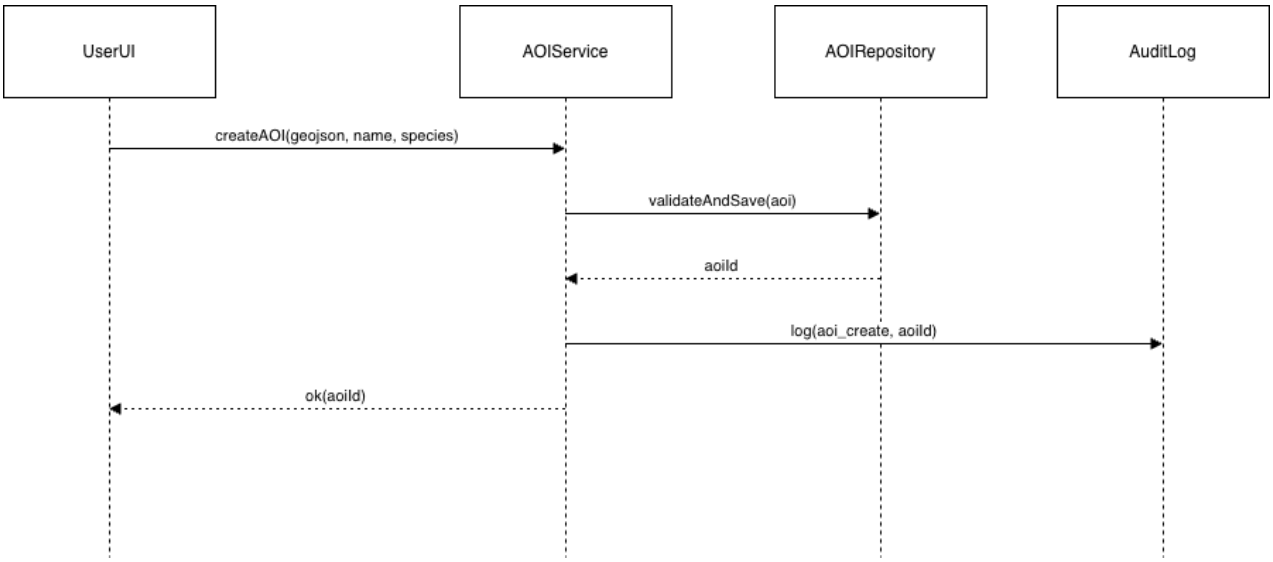
CU-01 Registrar Usuario



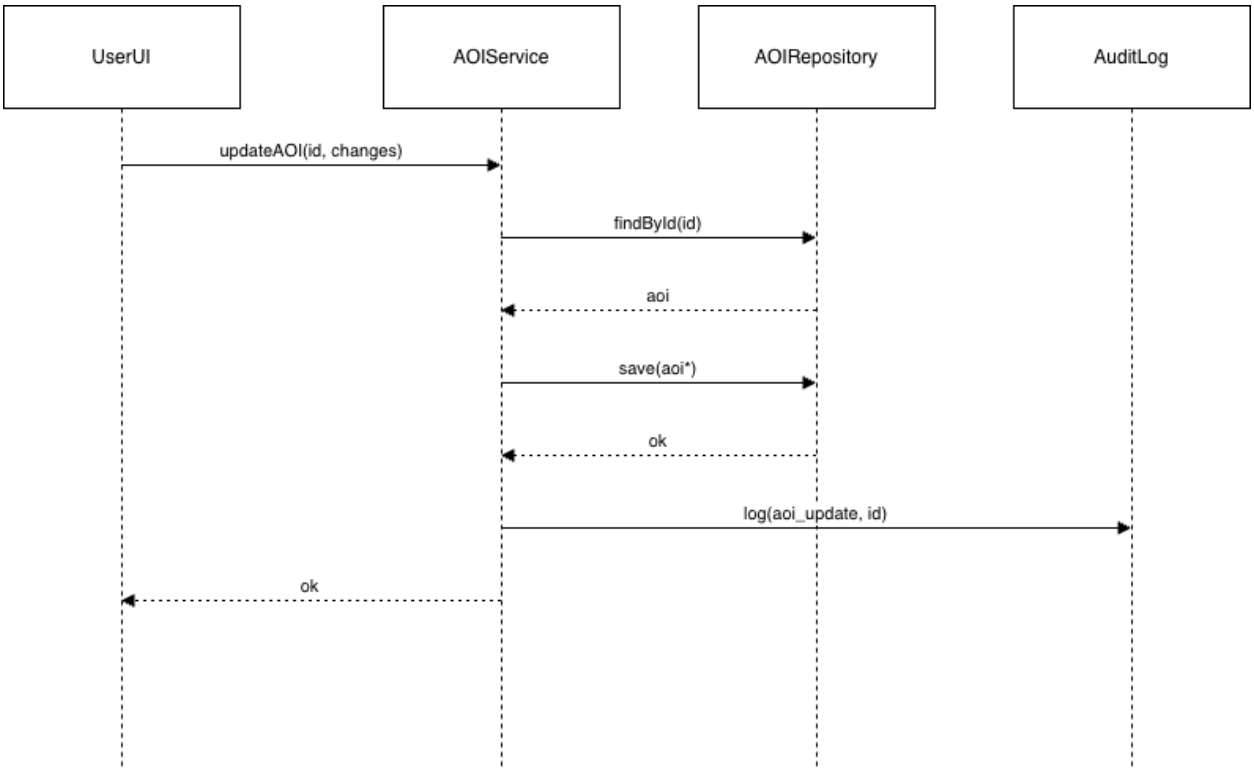
CU-02 Iniciar sesión



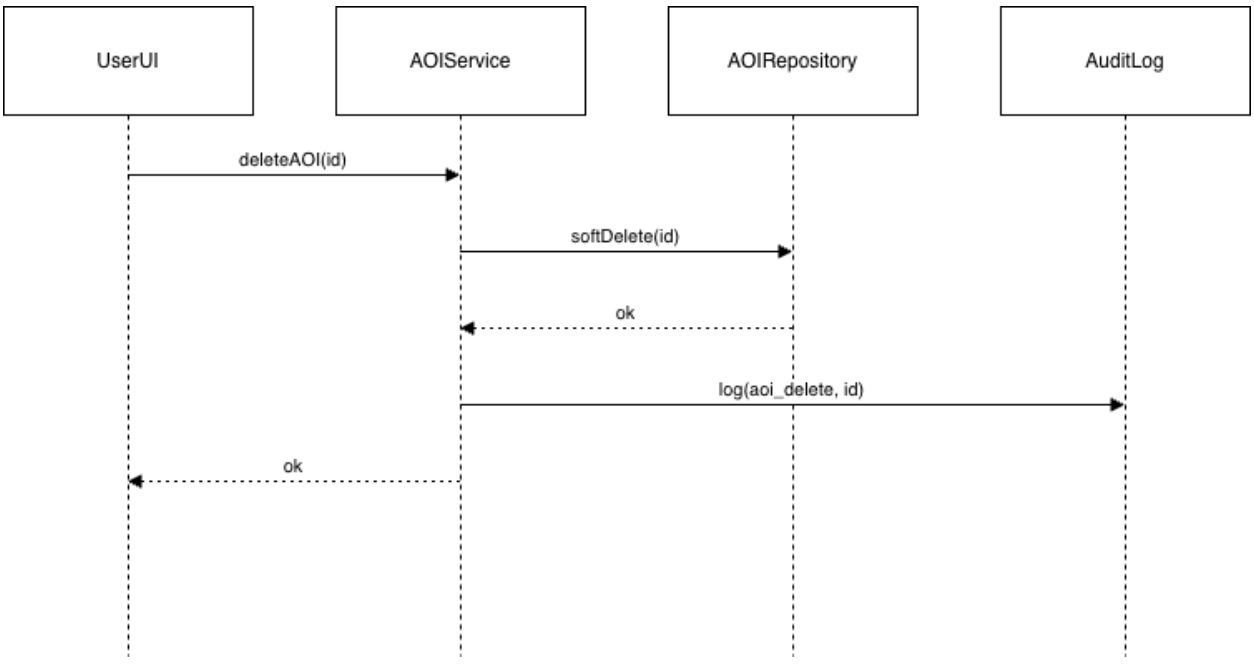
CU-03 Crear AOI (Área de interés)



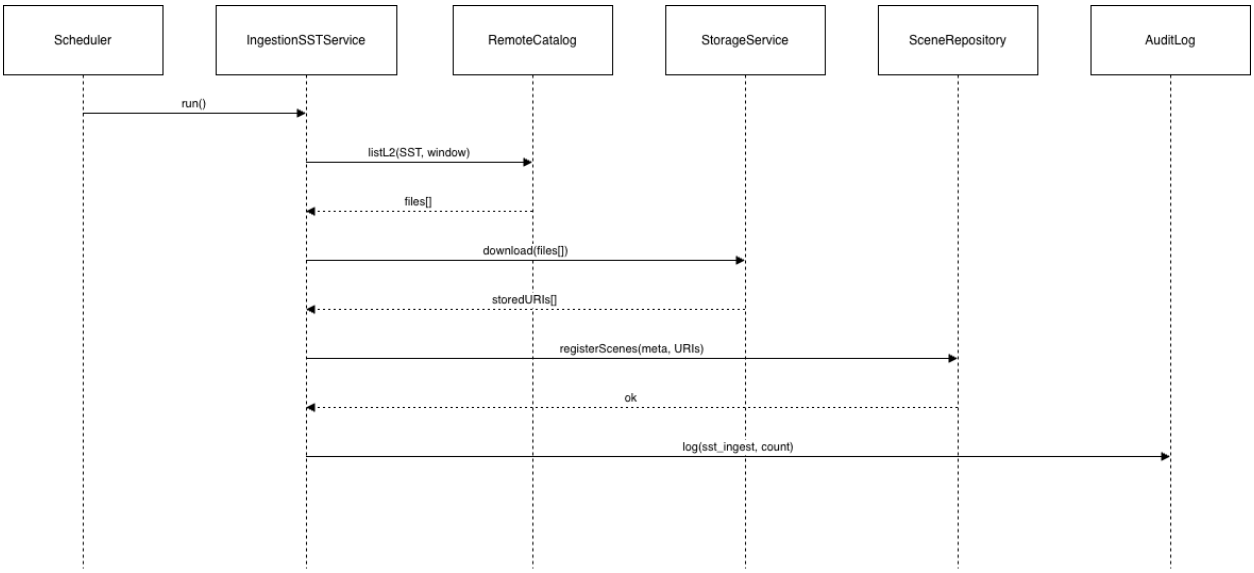
CU-04 Editar área de interés (AOI)



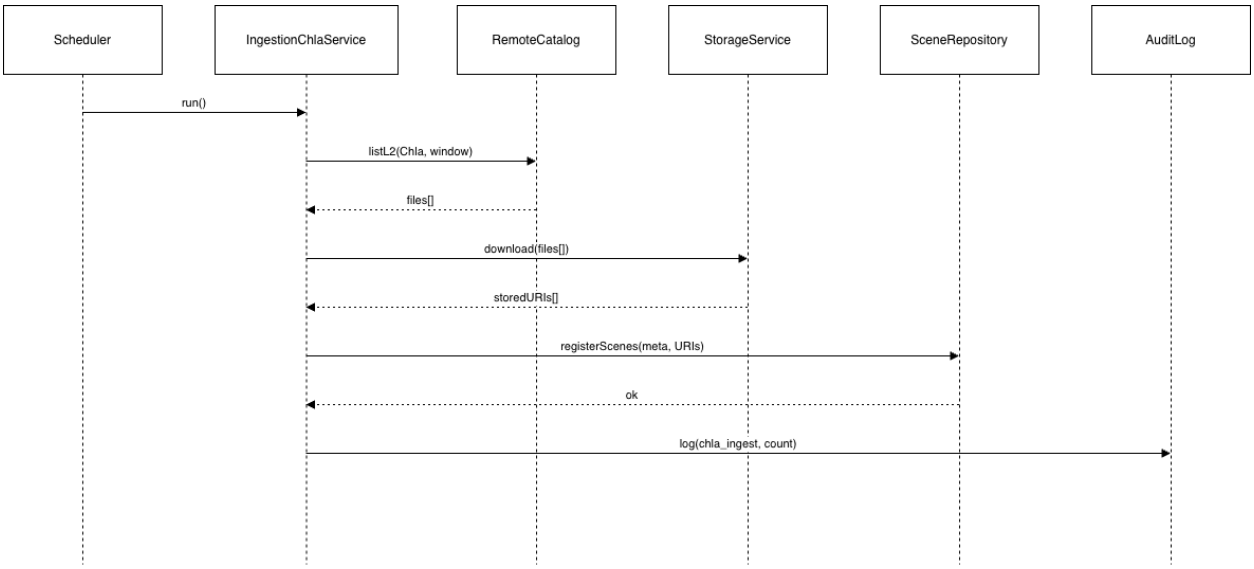
CU-05 Eliminar área de interés (AOI)



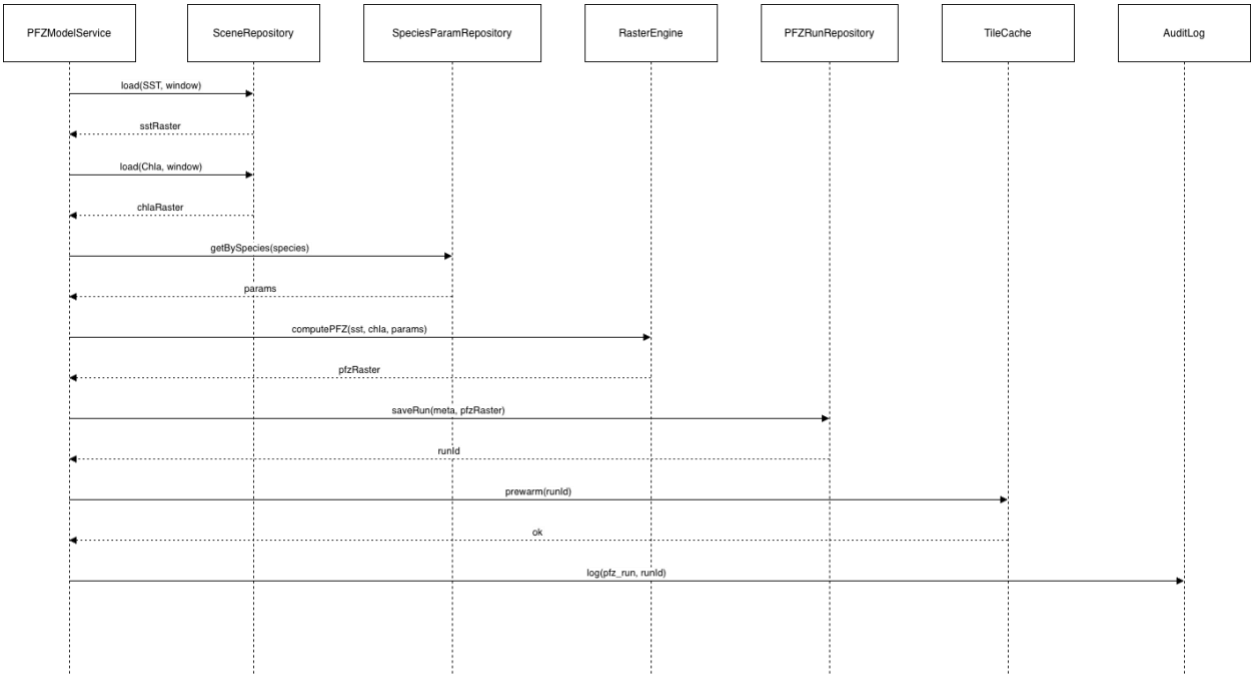
CU-06 Ingestar producto SST (L2)



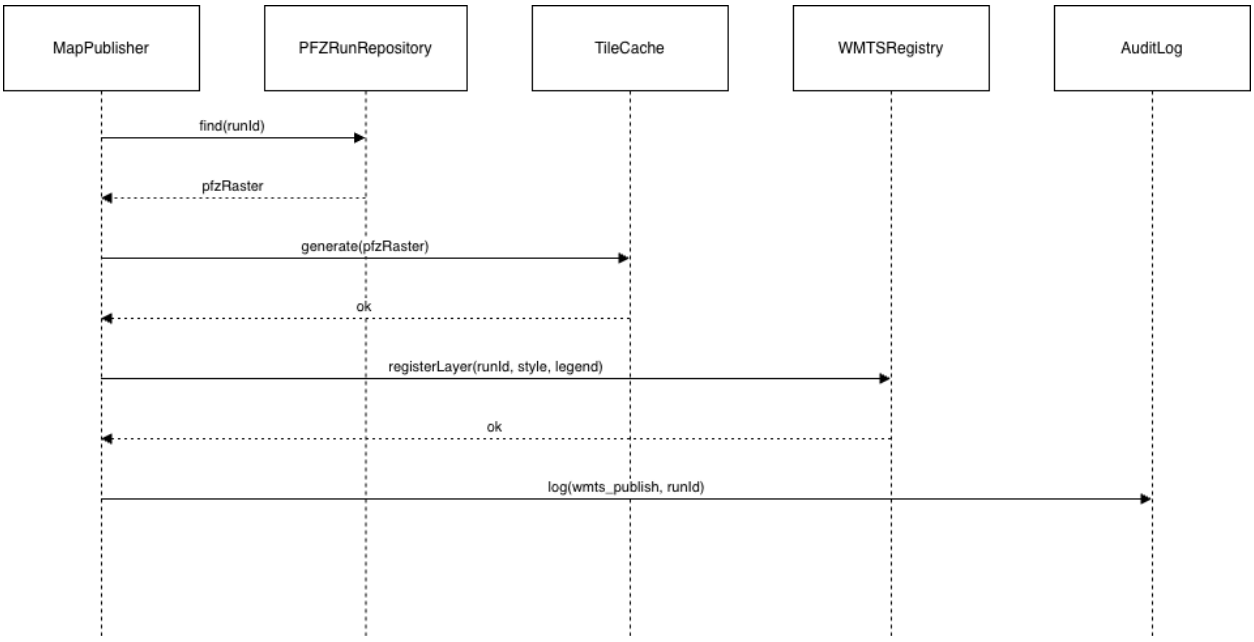
CU-07 Ingestar producto Chl-a (L2)



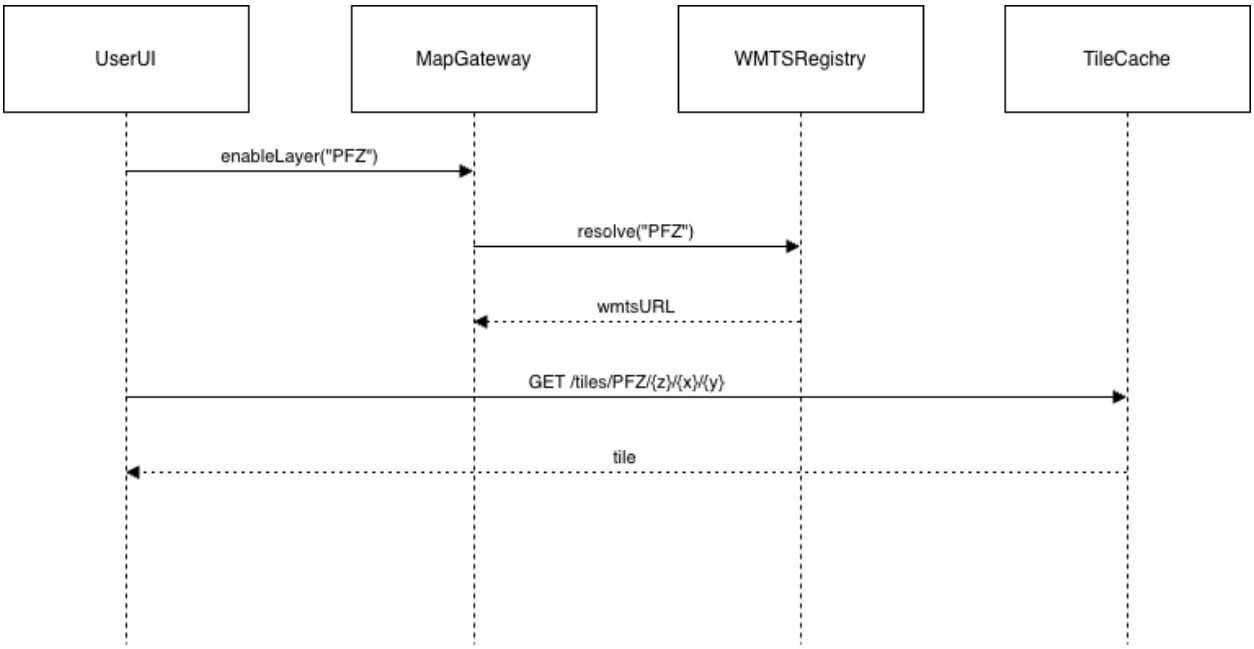
CU-08 Calcular índice PFZ



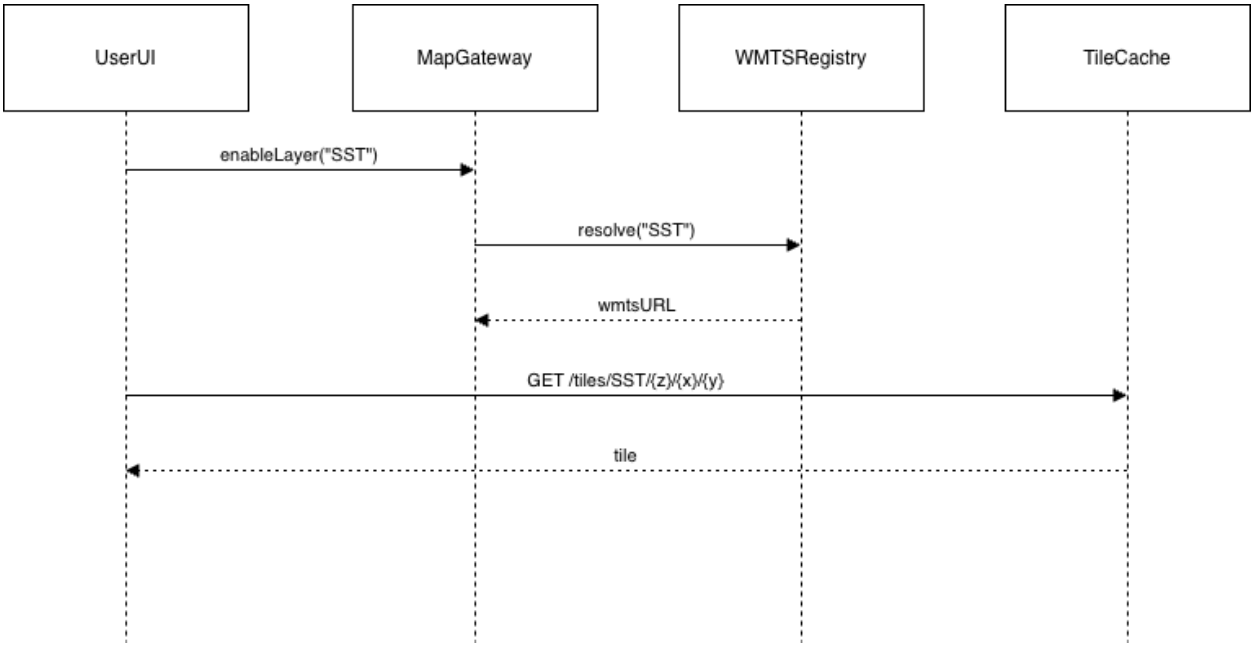
CU-09 Publicar tiles/WMTS PFZ



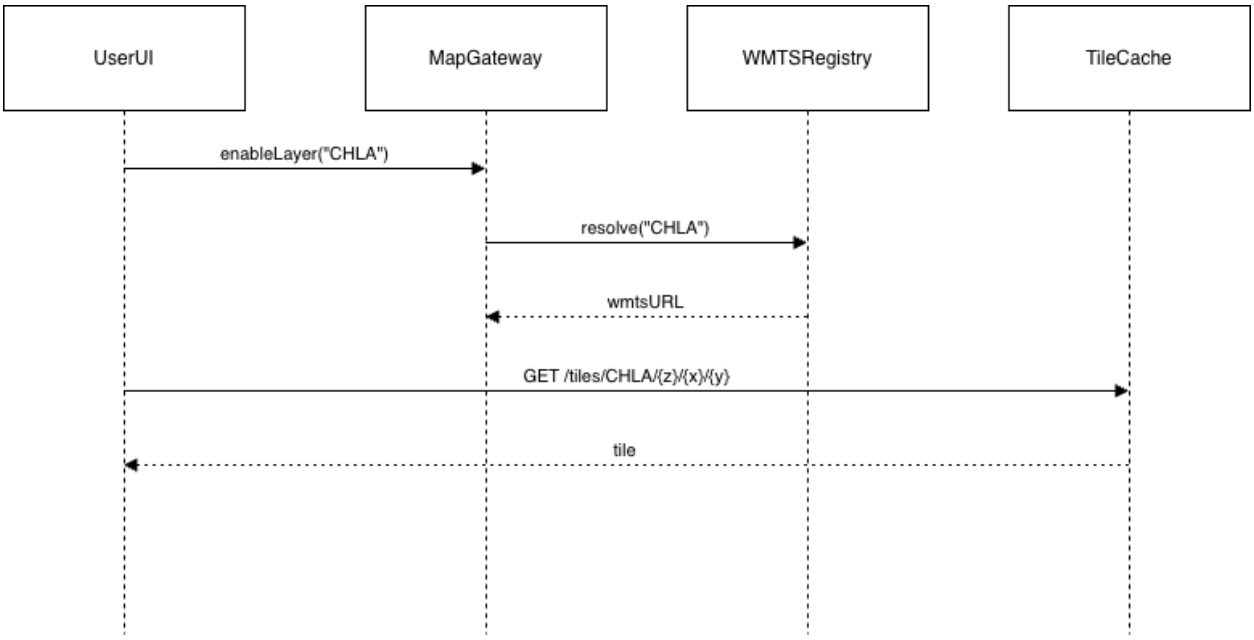
CU-10 Visualizar mapa PFZ



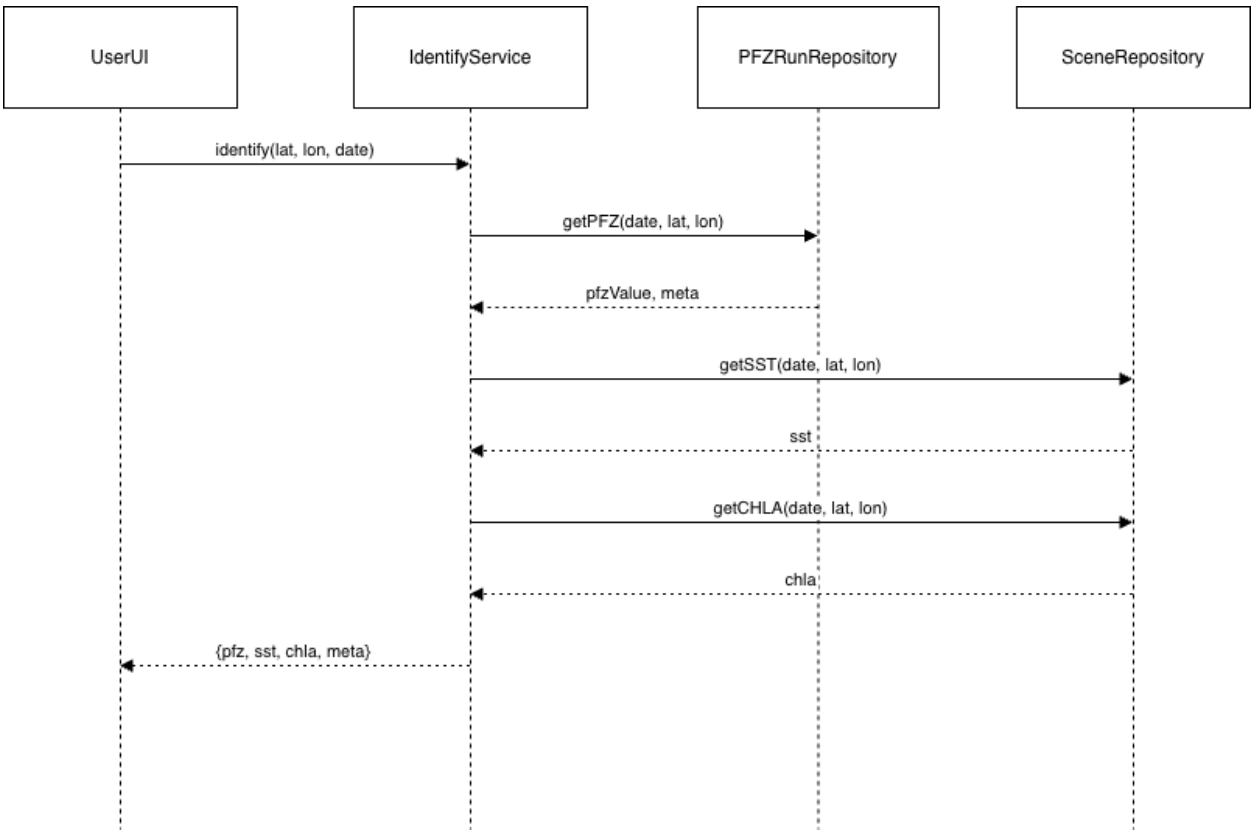
CU-11 Visualizar mapa SST



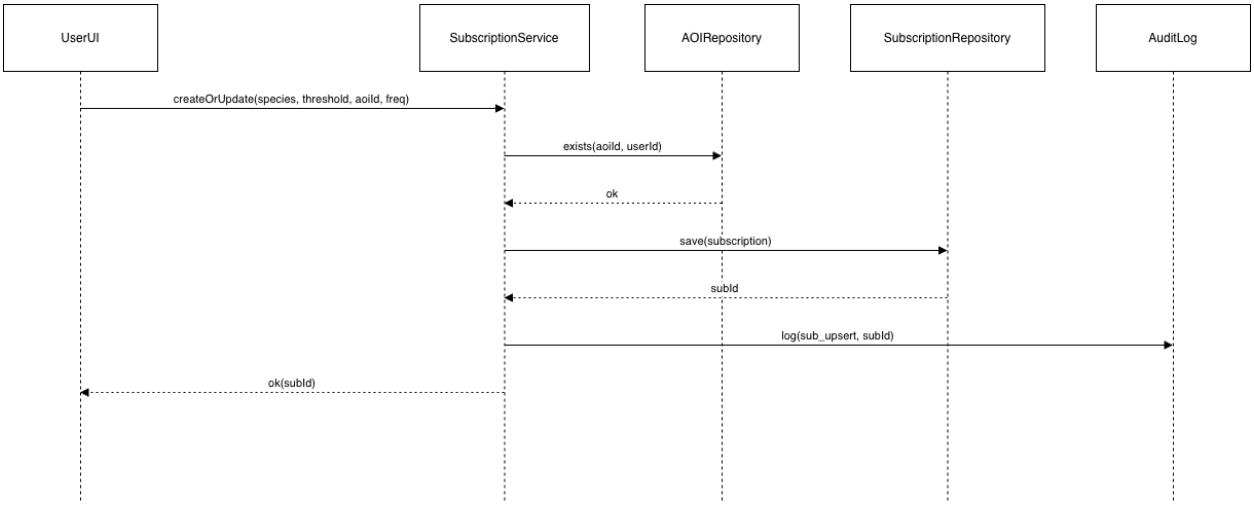
CU-12 Visualizar mapa Chl-A



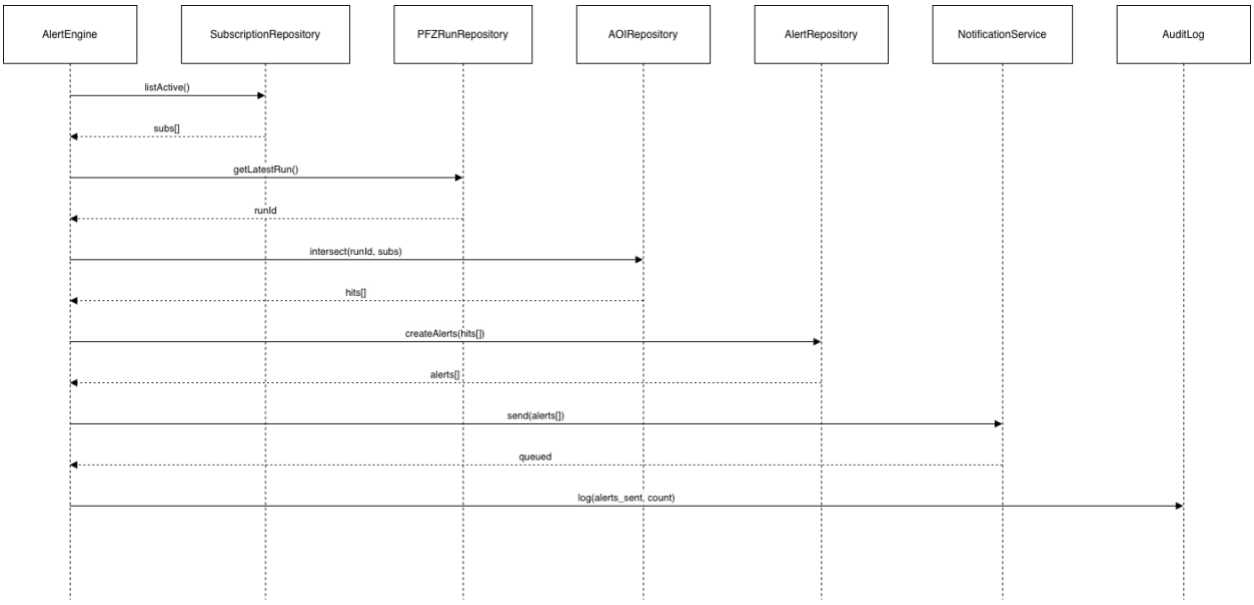
CU-13 Consultar Celda



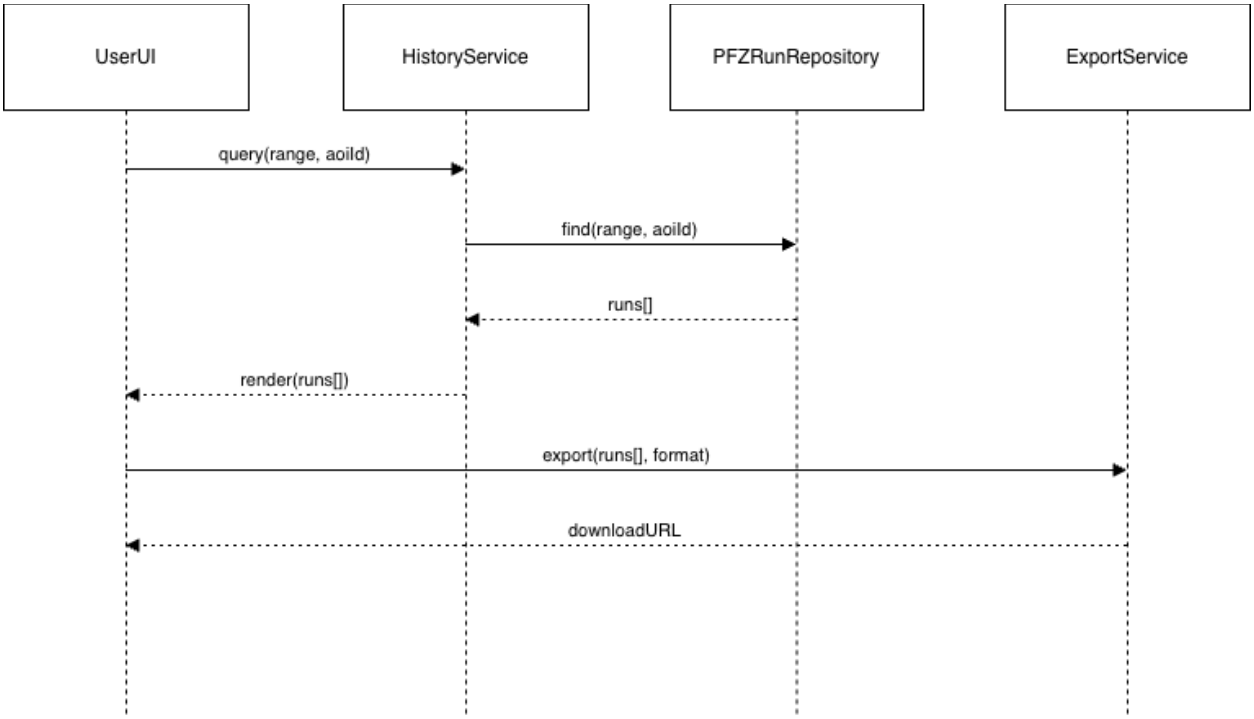
CU-14 Configurar suscripción de alertas



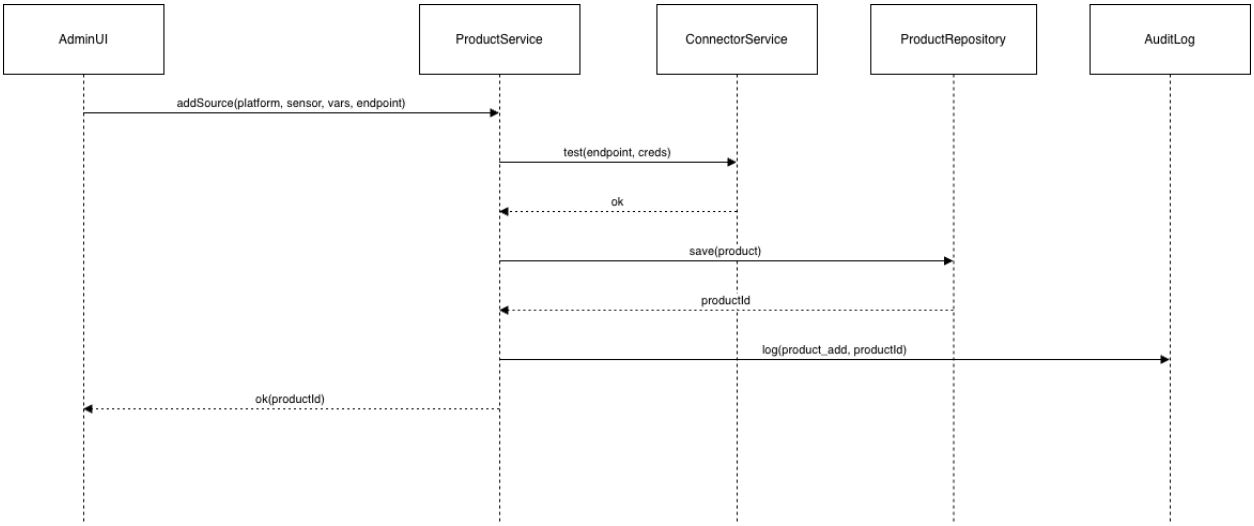
CU-15 Recibir alerta PFZ



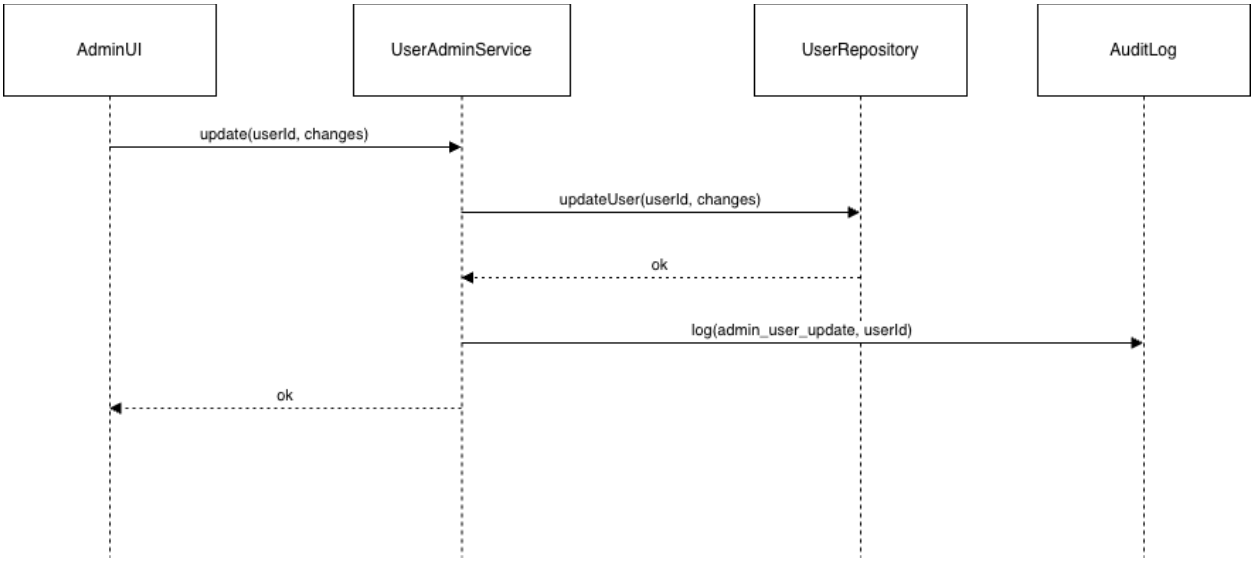
CU-16 Mostrar historial PFZ



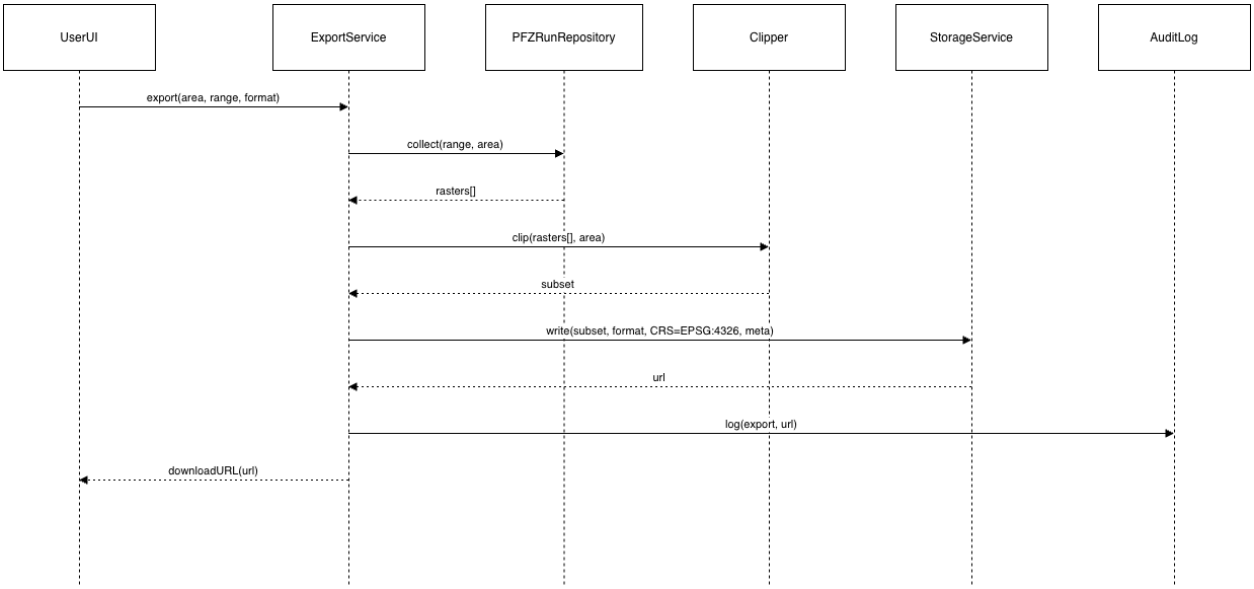
CU-17 Habilitar nueva fuente satelital



CU-18 Administrar usuarios



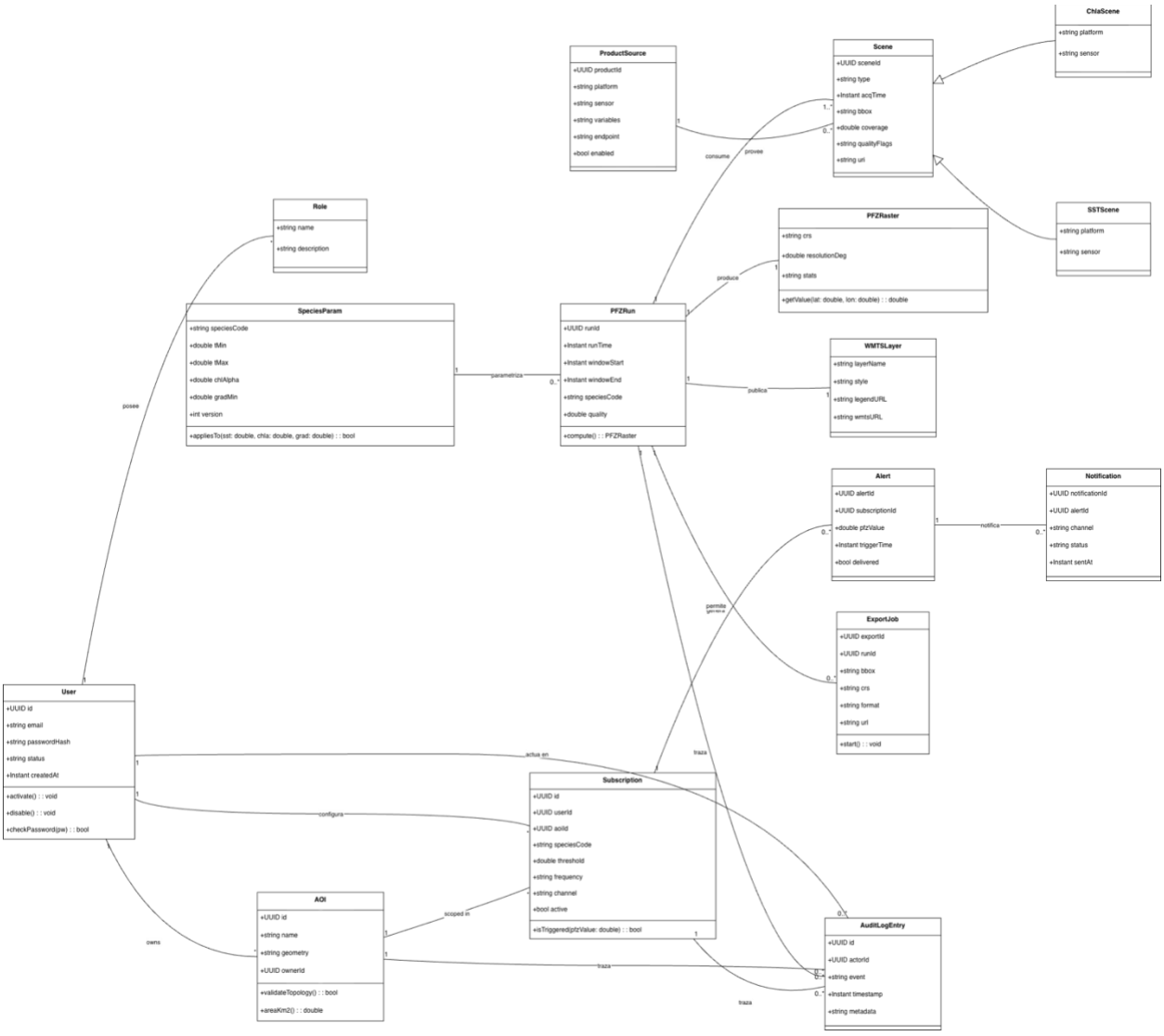
CU-19 Exportar PFZ



Etapa de diseño.

El siguiente diagrama es el establecimiento de la arquitectura general del software, las estructuras de datos y las interfaces que se traducirán en código. El Diagrama de Clases de Diseño detalla la organización interna del sistema, definiendo las clases, sus atributos, sus operaciones y las relaciones entre ellas.

Siguiendo las mejores prácticas del diseño orientado a objetos y el Proceso Unificado de Desarrollo (PUD), se garantiza la separación de responsabilidades y una alta cohesión. Esta estructura intenta ser consistente con el comportamiento dinámico definido previamente en los Diagramas de Secuencia, sirviendo como guía directa para la codificación del sistema SIZPOPE.



Etapa de Implementación

Modelo de pruebas

El objetivo general de la prueba es garantizar que el Sistema SIZPOPE cumpla integralmente con todos los Requerimientos Funcionales y No Funcionales definidos, asegurando la calidad y la fiabilidad de la información en un sistema de alertas de misión crítica.

Esto implica verificar tres aspectos fundamentales:

- **Integridad del Flujo de Datos:** Asegurar el correcto end-to-end (de extremo a extremo) del ciclo de vida de un evento: desde la captura de datos geográficos (WKT) en la Capa de Presentación, su persistencia en la Capa de Datos (MySQL), la aplicación de la lógica de negocio (Validación), hasta su correcta visualización en el componente GIS del Dashboard principal.
- **Seguridad y Acceso por Roles (RBAC)**: Confirmar que el Control de Acceso Basado en Roles (RBAC) funcione de manera estricta, impidiendo que usuarios no autorizados (Ciudadanos) accedan a funciones de alto privilegio (Administración, Validación).
- **Usabilidad y Rendimiento:** Validar que la interfaz de usuario (Java Swing) sea intuitiva para todos los roles y que el sistema mantenga una respuesta ágil en la consulta y filtrado de eventos desde la base de datos (rendimiento de la capa de datos)

CU-01 — Registrar usuario

Pruebas Unitarias

ID	TC-U-01-01
Nombre	Registro válido (camino feliz)
Objetivo	Crear usuario con credenciales válidas y emitir correo de verificación.
Precondiciones	No existe usuario con el email. Servicios operativos y conectividad a cola de correo.
Datos de entrada	email=usuario@example.com ; password=Otra!Clave2025
Componentes involucrados	CUT: AuthService — **Dobles/colaboradores:** UserRepository (mock), PasswordHasher (fake/spy), TokenService (stub), MailService (mock/spy), AuditLog (mock), Clock (stub), IdGenerator (stub), TransactionManager (mock)
Pasos	1. Invocar AuthService.register(email, password) 2. Verificar UserRepository.checkEmailUnique(emailNormalizado) 3. Verificar PasswordHasher.hash(password) 4. Verificar UserRepository.createUser({...passwordHash...}) 5. Verificar TokenService.generateVerification(userId) 6. Verificar MailService.sendVerification(userId, token) 7. Verificar AuditLog.log('register', userId, metadata) 8. Confirmar TransactionManager.commit()

Resultado esperado	Usuario creado en estado **PENDING**; token de verificación emitido; correo encolado; auditoría registrada.
Postcondiciones	Usuario persistido; evento de auditoría con timestamp de Clock; transacción confirmada.
Prioridad	Alta
Tipo	Unitario
Caso de Uso	CU-01
Requisitos relacionados	RF-01, RNF-01, RNF-02, RNF-05, RNF-06
Criterios de aceptación	Retorna userId no nulo; passwordHash ≠ password; **1** correo emitido; commit sin rollback; logs sin errores.

ID	TC-U-01-02
Nombre	Email duplicado
Objetivo	Rechazar el alta cuando el email ya existe.
Precondiciones	Existe usuario con usuario@example.com.
Datos de entrada	email=usuario@example.com ; password=Otra!Clave2025
Componentes involucrados	CUT: AuthService — **Dobles/colaboradores:** UserRepository (mock), AuditLog (mock), TransactionManager (mock)
Pasos	1. Configurar UserRepository.checkEmailUnique(email) → False 2. Invocar AuthService.register(email, password) 3. Capturar y verificar la excepción/control de error de dominio
Resultado esperado	Error Conflict('EMAIL_TAKEN'); no se persiste usuario ni se envía correo.
Postcondiciones	Estado inalterado; auditoría de intento fallido (motivo EMAIL_TAKEN).
Prioridad	Alta
Tipo	Unitario (Seguridad)
Caso de Uso	CU-01
Requisitos relacionados	RF-01, RNF-02, RNF-06
Criterios de aceptación	createUser y sendVerification **no** llamados; respuesta y mensaje consistentes; sin efectos colaterales.

ID	TC-U-01-03
Nombre	Política de contraseña (fuerza mínima)
Objetivo	Rechazar contraseñas débiles antes de persistir.
Precondiciones	Política activa: ≥10 caracteres, 1 mayúscula, 1 minúscula, 1 dígito, 1 símbolo.
Datos de entrada	email=usuario@example.com ; password=Otra!Clave2025

Componentes involucrados	CUT: AuthService — **Dobles/colaboradores:** *(ninguno, si la validación ocurre previo a repo/hasher)*
Pasos	1. Invocar AuthService.register(email, 'weak') 2. Capturar y verificar el error de validación 3. Verificar ausencia de interacción con UserRepository y PasswordHasher
Resultado esperado	Error de validación PASSWORD_WEAK.
Postcondiciones	Sin efectos colaterales; no se crea usuario; (opcional) auditoría de intento inválido.
Prioridad	Alta
Tipo	Unitario (Validación)
Caso de Uso	CU-01
Requisitos relacionados	RF-01, RNF-02, RNF-06
Criterios de aceptación	UserRepository y PasswordHasher **no** invocados; mensaje claro; logs sin warnings críticos.

Pruebas de Integración

ID	TC-I-01-01
Nombre	Registro end-to-end con correo
Objetivo	Validar el pipeline completo desde API hasta persistencia y notificación por correo.
Precondiciones	API Gateway y AuthService desplegados; DB MySQL accesible; cola/SMTP mockeado y operativo; reloj estable; logs habilitados.
Datos de entrada	HTTP POST /auth/register con { email: "usuario@example.com", password: "Str0ng!Pass2025" }
Componentes involucrados	API Gateway (REST), **AuthService**, **UserRepository** (DB MySQL), **PasswordHasher**, **TokenService**, **MailService** (mock/cola), **AuditLog**, **TransactionManager**
Pasos	1. Realizar POST /auth/register con email y password 2. Verificar en DB que se creó el usuario con status=PENDING y passwordHash válido 3. Verificar en cola/SMTP mock un único mensaje de verificación con token parseable 4. Validar que TokenService generó token con TTL correcto5. Consultar AuditLog y corroborar evento register con userId y requestId correlacionado
Resultado esperado	Respuesta **201 Created**; usuario persistido; correo de verificación encolado; auditoría registrada.
Postcondiciones	Usuario en PENDING; correo pendiente de entrega; trazabilidad completa en logs.

Prioridad	Alta
Tipo	Integración
Caso de Uso	CU-01
Requisitos relacionados	RF-01, RNF-01, RNF-02, RNF-04, RNF-06
Criterios de aceptación	Hash no reversible; un único correo emitido; latencias dentro de umbral; logs sin errores; consistencia de IDs entre capas.

ID	TC-I-01-02
Nombre	Rollback si falla SMTP (REQUIRE_MAIL)
Objetivo	Asegurar consistencia transaccional cuando el envío de correo falla bajo política estricta.
Precondiciones	MailService configurado para fallar (timeout/500); TransactionManager en modo REQUIRE_MAIL; DB limpia.
Datos de entrada	HTTP POST /auth/register con { email: "failmail@example.com", password: "Str0ng!Pass2025" }
Componentes involucrados	API Gateway (REST), **AuthService**, **UserRepository**, **PasswordHasher**, **TokenService**, **MailService** (falla forzada), **AuditLog**, **TransactionManager**
Pasos	1. Realizar POST /auth/register con email válido 2. Forzar fallo en MailService.sendVerification (timeout/500) 3. Verificar respuesta HTTP de error controlado (503/500 con código de negocio) 4. Verificar en DB que no quedó el usuario persistido (rollback efectivo) 5. Verificar en AuditLog evento de incidente con causa y requestId
Resultado esperado	Transacción revertida; usuario **no** creado; error controlado devuelto al cliente; auditoría del incidente.
Postcondiciones	Sin usuarios “zombie”; métricas de fallo incrementadas; alerta operativa si corresponde.
Prioridad	Alta
Tipo	Integración
Caso de Uso	CU-01
Requisitos relacionados	RF-01, RNF-04, RNF-06
Criterios de aceptación	DB sin registro de usuario; se evidencia rollback; mensaje de error consistente; ningún correo en cola.

ID	TC-I-01-03
Nombre	Normalización de email (API + DB)
Objetivo	Verificar que el email se normaliza (trim/lowecase) en todo el flujo antes de validar y persistir.

Precondiciones	No existe cuenta para usuario@example.com; validadores activos; auditoría habilitada.
Datos de entrada	HTTP POST /auth/register con { email: " Usuario@Example.com ", password: "Str0ng!Pass2025" }
Componentes involucrados	API Gateway (REST), **AuthService**, **UserRepository**, **PasswordHasher**, **TokenService**, **MailService** (mock/cola), **AuditLog**
Pasos	1. Realizar POST /auth/register con email con espacios y mayúsculas 2. Verificar que checkEmailUnique recibe usuario@example.com (normalizado) 3. Verificar en DB el usuario almacenado con email en minúsculas y sin espacios 4. Verificar auditoría con email normalizado en metadata (cuando aplique) 5. Verificar un único correo en cola asociado al userId creado
Resultado esperado	Alta exitosa con email canónico; validaciones realizadas sobre el valor normalizado; correo emitido.
Postcondiciones	Datos canónicos persistidos; auditoría consistente con el valor normalizado.
Prioridad	Media
Tipo	Integración
Caso de Uso	CU-01
Requisitos relacionados	RF-01, RNF-06
Criterios de aceptación	checkEmailUnique y persistencia usan el mismo email normalizado; no hay duplicados; un único correo.

Pruebas de aceptación

ID	TC-A-01-01
Nombre	Alta desde UI (flujo visible)
Objetivo	Validar la experiencia de usuario y la confirmación de alta incluyendo el correo de verificación.
Precondiciones	UI web operativa; navegador soportado; API y AuthService activos; DB accesible; MailService (SMTP/cola) disponible; auditoría habilitada.
Datos de entrada	Email válido y contraseña fuerte (p. ej. usuario@example.com, Str0ng!Pass2025).
Componentes involucrados	Front-end Web (Registro), **API Gateway (REST)**, **AuthService**, **UserRepository (MySQL)**, **PasswordHasher**, **TokenService**, **MailService (SMTP/cola)**, **AuditLog**

Pasos	1. Abrir la pantalla de registro en la UI 2. Completar email y contraseña con valores válidos 3. Enviar el formulario de registro 4. Verificar que la UI muestra mensaje de éxito/confirmación del alta 5. Verificar la recepción del correo de verificación en la casilla del usuario
Resultado esperado	La UI confirma el alta y el usuario recibe un correo de verificación con enlace/token válido.
Postcondiciones	Usuario creado en estado **PENDING**; correo de verificación disponible; registro de auditoría presente.
Prioridad	Alta
Tipo	Aceptación
Caso de Uso	CU-01
Requisitos relacionados	RF-01, RNF-01, RNF-05
Criterios de aceptación	Mensajes claros y accesibles; tiempo de respuesta adecuado; un único correo emitido; sin errores visibles en UI/console.

ID	TC-A-01-02
Nombre	Mensaje de error claro (email duplicado)
Objetivo	Verificar que el usuario comprende el error y las acciones sugeridas cuando el email ya existe.
Precondiciones	Existe una cuenta registrada con usuario@example.com; UI y API operativas.
Datos de entrada	Email duplicado (usuario@example.com) y contraseña válida.
Componentes involucrados	Front-end Web (Registro), **API Gateway (REST)**, **AuthService**, **UserRepository**, **AuditLog**
Pasos	1. Abrir la pantalla de registro en la UI 2. Completar email con uno ya registrado y contraseña válida 3. Enviar el formulario de registro 4. Verificar que la UI muestra un mensaje de error entendible y no técnico 5. Verificar que la UI sugiere opciones (recuperar contraseña/usar otro email)
Resultado esperado	La UI informa “email ya registrado” y orienta a la acción correcta; no se crea cuenta nueva.
Postcondiciones	Sin cambios en DB; evento de auditoría del intento fallido.
Prioridad	Media
Tipo	Aceptación
Caso de Uso	CU-01
Requisitos relacionados	RF-01, RNF-05

Criterios de aceptación	No se exponen detalles internos; foco vuelve al campo email; accesible para lector de pantalla.

ID	TC-A-01-03
Nombre	Accesibilidad y validaciones en cliente
Objetivo	Comprobar que la UI bloquea entradas inválidas y provee ayudas accesibles por campo.
Precondiciones	Validaciones en cliente activas; navegador soportado; UI accesible (atributos ARIA/roles).
Datos de entrada	Email con formato inválido y contraseña que no cumple la política mínima.
Componentes involucrados	Front-end Web (Registro)
Pasos	1. Abrir la pantalla de registro en la UI 2. Completar el email con formato inválido y contraseña débil 3. Intentar enviar el formulario 4. Verificar que la UI no permite el envío y presenta mensajes de ayuda por campo (accesibles) 5. Corregir los campos y verificar que la UI habilita el envío
Resultado esperado	La UI impide el envío con datos inválidos y muestra mensajes de ayuda accesibles; al corregir, el envío se habilita.
Postcondiciones	Sin llamadas al backend mientras existan errores de validación; experiencia consistente.
Prioridad	Media
Tipo	Aceptación
Caso de Uso	CU-01
Requisitos relacionados	RF-01, RNF-05
Criterios de aceptación	Mensajes por campo legibles por tecnología asistiva; no hay tráfico a API con datos inválidos; comportamiento consistente en navegadores soportados.

CU-02 Iniciar sesión

Pruebas Unitarias

ID	TC-U-02-01
Nombre	Login válido (camino feliz)
Objetivo	Autenticar credenciales válidas y emitir token JWT.
Precondiciones	Existe usuario activo con email y passwordHash consistente; políticas de contraseña y bloqueo activas.

Datos de entrada	email=usuario@example.com ; password=Str0ng!Pass2025
Componentes involucrados	CUT: AuthService — **Dobles/colaboradores:** UserRepository (mock), PasswordHasher (fake/spy), TokenService (stub), AuditLog (mock), Clock (stub)
Pasos	1. Invocar AuthService.login(email, password) 2. Verificar UserRepository.findByEmail(emailNormalizado) 3. Verificar PasswordHasher.verify(password, passHash) 4. Verificar TokenService.issue(userId, roles) y reclamos estándar (iss, sub, iat, exp) 5. Verificar AuditLog.log('login_success', userId, metadata)
Resultado esperado	Token emitido; login exitoso.
Postcondiciones	Auditoría de éxito registrada; contador de fallos (si existiera) en cero.
Prioridad	Alta
Tipo	Unitario
Caso de Uso	CU-02
Requisitos relacionados	RF-02, RNF-02, RNF-01, RNF-06
Criterios de aceptación	TokenService.issue llamado una vez; verify positivo; claims con expiración válida.

ID	TC-U-02-02
Nombre	Contraseña incorrecta
Objetivo	Rechazar autenticación cuando la verificación del hash falla.
Precondiciones	Existe usuario activo; hash almacenado válido; política de conteo de fallos habilitada.
Datos de entrada	email=usuario@example.com ; password=Wrong#2025
Componentes involucrados	CUT: AuthService — **Dobles/colaboradores:** UserRepository (mock), PasswordHasher (fake/spy), TokenService (stub), AuditLog (mock)
Pasos	1. Invocar AuthService.login(email, password) 2. Verificar UserRepository.findByEmail(emailNormalizado) 3. Forzar PasswordHasher.verify(...) → False 4. Verificar que no se llama a TokenService.issue(...) 5. Verificar AuditLog.log('login_failed', userId/email, cause='BAD_CREDENTIALS')
Resultado esperado	Error UNAUTHORIZED sin token.
Postcondiciones	Incremento de contador de fallos (si aplica); auditoría de intento fallido.
Prioridad	Alta
Tipo	Unitario (Seguridad)
Caso de Uso	CU-02
Requisitos relacionados	RF-02, RNF-02
Criterios de aceptación	No emisión de token; mensaje genérico (sin filtrar si email existe).

ID	TC-U-02-03
Nombre	Usuario bloqueado/inactivo
Objetivo	Impedir autenticación cuando el estado del usuario no es activo.
Precondiciones	Usuario con status=LOCKED o DISABLED; políticas RBAC vigentes.
Datos de entrada	email=locked@example.com ; password=Cualquier!2025
Componentes involucrados	CUT: AuthService — **Dobles/colaboradores:** UserRepository (mock), PasswordHasher (fake/spy), TokenService (stub), AuditLog (mock)
Pasos	1. Invocar AuthService.login(email, password) 2. Verificar UserRepository.findByEmail(emailNormalizado) → user.status=LOCKED 3. Verificar que no se llama a PasswordHasher.verify ni a TokenService.issue 4. Verificar AuditLog.log('login_blocked', userId/email, cause='LOCKED')
Resultado esperado	Error FORBIDDEN sin token.
Postcondiciones	Auditoría de bloqueo; sin cambios en credenciales.
Prioridad	Media
Tipo	Unitario (Seguridad)
Caso de Uso	CU-02
Requisitos relacionados	RF-02, RNF-02
Criterios de aceptación	No exposición de motivo sensible; no hay emisión de token ni verificación de hash.

Pruebas de Integración

ID	TC-I-02-01
Nombre	Login API→DB→JWT (feliz)
Objetivo	Validar flujo completo: API, consulta en DB, verificación hash y emisión de token.
Precondiciones	API Gateway y AuthService desplegados; DB con usuario activo; reloj y auditoría habilitados.
Datos de entrada	POST /auth/login { email: "usuario@example.com", password: "Str0ng!Pass2025" }
Componentes involucrados	API Gateway (REST), **AuthService**, **UserRepository** (MySQL), **PasswordHasher**, **TokenService**, **AuditLog**
Pasos	1. Realizar POST /auth/login con credenciales válidas 2. Verificar 200 OK con access_token y expires_in 3. Decodificar JWT y validar claims (iss, sub, iat, exp, roles, jti) 4. Verificar en AuditLog la entrada login_success con correlación a la request
Resultado esperado	Token válido con expiración prevista; auditoría correcta.

Postcondiciones	Sesión del cliente activa; traza en logs.
Prioridad	Alta
Tipo	Integración
Caso de Uso	CU-02
Requisitos relacionados	RF-02, RNF-01, RNF-02, RNF-06
Criterios de aceptación	Latencia bajo umbral; reloj consistente para iat/exp; firma JWT válida.

ID	TC-I-02-02
Nombre	Throttling/Rate limit tras fallos consecutivos
Objetivo	Verificar backoff/limit tras N fallos seguidos para misma IP/usuario.
Precondiciones	Política de rate limit activa (p. ej., 5 intentos/5 min); contador por sujeto e IP operativo.
Datos de entrada	6 intentos con password incorrecto.
Componentes involucrados	API Gateway (RateLimiter), **AuthService**, **AuditLog**, **UserRepository**, **PasswordHasher**
Pasos	1. Ejecutar 5 POST /auth/login con password incorrecto 2. Ejecutar el 6º intento dentro de la ventana temporal 3. Verificar respuesta 429 Too Many Requests con cabeceras de retry (si aplica) 4. Verificar auditoría de rate limit y métricas de seguridad
Resultado esperado	Bloqueo temporal del endpoint/usuario según política.
Postcondiciones	Ventana y contador actualizados; logs de seguridad disponibles.
Prioridad	Alta
Tipo	Integración (Seguridad)
Caso de Uso	CU-02
Requisitos relacionados	RF-02, RNF-02, RNF-04
Criterios de aceptación	No fuga de información (mismo mensaje para email válido/inválido); headers coherentes.

ID	TC-I-02-03
Nombre	Normalización de email en login
Objetivo	Garantizar que el email se normaliza antes de consultar DB y verificar credenciales.
Precondiciones	Usuario registrado como usuario@example.com.
Datos de entrada	POST /auth/login { email: " Usuario@Example.com ", password: "Str0ng!Pass2025" }
Componentes involucrados	API Gateway, **AuthService**, **UserRepository**, **PasswordHasher**, **TokenService**, **AuditLog**

Pasos	1. Realizar POST /auth/login con email con espacios y mayúsculas 2. Verificar que la consulta a DB usa usuario@example.com (normalizado) 3. Verificar 200 OK con token válido4. Verificar auditoría con email normalizado (cuando aplique)
Resultado esperado	Autenticación exitosa con email canónico.
Postcondiciones	Trazabilidad consistente con valor normalizado.
Prioridad	Media
Tipo	Integración
Caso de Uso	CU-02
Requisitos relacionados	RF-02, RNF-06
Criterios de aceptación	Sin duplicación de usuarios por mayúsculas/espacios; token emitido.

Pruebas de aceptación

ID	TC-A-02-02
Nombre	Mensaje de error genérico (credenciales inválidas)
Objetivo	Evitar revelar si el email existe y guiar al usuario al remediar el error.
Precondiciones	UI y API operativas.
Datos de entrada	usuario@example.com ; Wrong#2025
Componentes involucrados	Front-end Web (Login), **API Gateway (REST)**, **AuthService**, **UserRepository**, **PasswordHasher**, **AuditLog**
Pasos	1. Abrir pantalla de login en la UI 2. Completar email válido y contraseña incorrecta 3. Enviar formulario 4. Verificar mensaje no técnico: “Credenciales inválidas” 5. Verificar enfoque en campo clave y sugerencias (reintentar/recuperar contraseña)
Resultado esperado	La UI informa error sin filtrar existencia de cuenta; no hay sesión.
Postcondiciones	Auditoría de intento fallido; sin token en cliente.
Prioridad	Media
Tipo	Aceptación
Caso de Uso	CU-02
Requisitos relacionados	RF-02, RNF-05
Criterios de aceptación	No revela si email existe; tiempo de respuesta adecuado; accesible.

ID	TC-A-02-03
Nombre	Cuenta bloqueada/inactiva (UX)

Objetivo	Mostrar un mensaje claro y acciones sugeridas cuando la cuenta no está disponible.
Precondiciones	Usuario con status=LOCKED o DISABLED.
Datos de entrada	locked@example.com ; Cualquier!2025
Componentes involucrados	Front-end Web (Login), **API Gateway (REST)**, **AuthService**, **UserRepository**, **AuditLog**
Pasos	1. Abrir pantalla de login en la UI 2. Completar email bloqueado e ingresar contraseña válida 3. Enviar formulario 4. Verificar mensaje claro: “Cuenta bloqueada” o “Cuenta deshabilitada” (sin detalles técnicos) 5. Verificar que la UI muestra opciones de asistencia (contacto/soporte)
Resultado esperado	No se inicia sesión; UI informa estado de la cuenta y sugiere acciones.
Postcondiciones	Auditoría de intento bloqueado; sin cambios en credenciales.
Prioridad	Media
Tipo	Aceptación
Caso de Uso	CU-02
Requisitos relacionados	RF-02, RNF-05
Criterios de aceptación	Mensaje consistente; no expone razones internas; experiencia accesible.

CU-03 Crear AOI (Área de interés)

Pruebas Unitarias

ID	TC-U-03-01
Nombre	Crear AOI válido (camino feliz)
Objetivo	Persistir un AOI válido y retornar su aoiId.
Precondiciones	Usuario autenticado con permisos para crear AOI.
Datos de entrada	geojson polígono válido (cerrado, sin autointersección), name="Plataforma Sur", species=["merluza"].
Componentes involucrados	CUT: AOIService — **Dobles/colaboradores:** AOIRepository (mock), AuditLog (mock), GeometryValidator (stub), OwnershipPolicy (stub), IdGenerator (stub)
Pasos	1. Invocar AOIService.createAOI(geojson, name, species) 2. Verificar GeometryValidator.validate(geojson) → válido 3. Verificar OwnershipPolicy.check(currentUser) → autorizado 4. Verificar AOIRepository.validateAndSave(aoi) 5. Verificar AuditLog.log('aoi_create', aoiId, metadata)

Resultado esperado	Retorna aoiId; AOI válido preparado para uso.
Postcondiciones	AOI persistido y asociado al ownerId del usuario.
Prioridad	Alta
Tipo	Unitario
Caso de Uso	CU-03
Requisitos relacionados	RF-03, RNF-02, RNF-06
Criterios de aceptación	aoiId no nulo; validaciones llamadas; auditoría registrada.

ID	TC-U-03-02
Nombre	GeoJSON inválido
Objetivo	Rechazar AOI con geometría inválida.
Precondiciones	Usuario autenticado.
Datos de entrada	geojson con polígono no cerrado o self-intersection.
Componentes involucrados	CUT: AOIService — **Dobles/colaboradores:** GeometryValidator (stub), AOIRepository (mock), AuditLog (mock)
Pasos	1. Invocar AOIService.createAOI(geojsonInválido, name, species) 2. Forzar GeometryValidator.validate(...) → error GEOMETRY_INVALID 3. Verificar que no se llama a AOIRepository.validateAndSave 4. Verificar AuditLog.log('aoi_create_failed', reason='GEOMETRY_INVALID')
Resultado esperado	Error de validación; AOI no persistido.
Postcondiciones	Sin cambios en almacenamiento.
Prioridad	Alta
Tipo	Unitario (Validación)
Caso de Uso	CU-03
Requisitos relacionados	RF-03, RNF-06
Criterios de aceptación	Mensaje claro; sin efectos colaterales; auditoría del intento fallido.

ID	TC-U-03-03
Nombre	Especie no soportada
Objetivo	Rechazar AOI cuando incluye especie inexistente en catálogo de parámetros.
Precondiciones	Catálogo de especies en SpeciesParamRepository.
Datos de entrada	species=["kraken"] (no existente), geojson válido, name="Zona X".
Componentes involucrados	CUT: AOIService — **Dobles/colaboradores:** SpeciesParamRepository (stub), AOIRepository (mock), AuditLog (mock)

Pasos	1. Invocar AOIService.createAOI(geojson, name, speciesNoSoportada) 2. Verificar SpeciesParamRepository.getBySpecies('kraken') → vacío 3. Verificar que no se llama a AOIRepository.validateAndSave 4. Verificar AuditLog.log('aoi_create_failed', reason='SPECIES_UNSUPPORTED')
Resultado esperado	Error SPECIES_UNSUPPORTED.
Postcondiciones	Sin persistencia de AOI.
Prioridad	Media
Tipo	Unitario (Reglas de negocio)
Caso de Uso	CU-03
Requisitos relacionados	RF-03, RNF-06
Criterios de aceptación	Mensaje claro; no hay efectos en repositorio.

Pruebas de Integración

ID	TC-I-03-01
Nombre	Alta de AOI API→Servicio→DB
Objetivo	Validar el flujo completo desde el endpoint hasta la persistencia y auditoría.
Precondiciones	API y AOIService desplegados; DB accesible; auditoría habilitada.
Datos de entrada	POST /aoi { name: "Plataforma Sur", species: ["merluza"], geojson: <polígono válido> }
Componentes involucrados	API Gateway (REST), **AOIService**, **GeometryValidator**, **AOIRepository** (MySQL), **AuditLog**
Pasos	1. Realizar POST /aoi con payload válido 2. Verificar 201 Created con aoiId 3. Validar en DB el AOI con ownerId del usuario y geometría almacenada en CRS esperado 4. Verificar registro en AuditLog aoi_create con aoiId
Resultado esperado	AOI persistido y visible para el dueño.
Postcondiciones	AOI disponible para suscripciones y consultas.
Prioridad	Alta
Tipo	Integración
Caso de Uso	CU-03
Requisitos relacionados	RF-03, RNF-01, RNF-02, RNF-06
Criterios de aceptación	Respuesta bajo umbral; consistencia de CRS/precisión; trazabilidad completa.

ID	TC-I-03-02
Nombre	Rechazo por geometría inválida (API)
Objetivo	Confirmar que la API devuelve error de validación coherente ante geometría inválida.
Precondiciones	Validadores activos; auditoría habilitada.
Datos de entrada	POST /aoi con geojson inválido (self-intersection).
Componentes involucrados	API Gateway (REST), **AOIService**, **GeometryValidator**, **AuditLog**
Pasos	1. Realizar POST /aoi con geojson inválido 2. Verificar respuesta 400 Bad Request con código GEOMETRY_INVALIaD 3. Verificar que no hay inserción en DB para ese AOI 4. Verificar auditoría aoi_create_failed con causa
Resultado esperado	Error 400 con motivo claro; sin persistencia.
Postcondiciones	Estado inalterado; logs de validación.
Prioridad	Alta
Tipo	Integración
Caso de Uso	CU-03
Requisitos relacionados	RF-03, RNF-06
Criterios de aceptación	Mensaje consistente; no se crean registros parciales.

ID	TC-I-03-03
Nombre	Normalización y metadatos
Objetivo	Validar que name y species se normalizan y que se guardan metadatos mínimos.
Precondiciones	Catálogo de especies; auditoría habilitada.
Datos de entrada	POST /aoi { name: " plataforma SUR ", species: ["Merluza"], geojson: <válido> }
Componentes involucrados	API Gateway, **AOIService**, **SpeciesParamRepository**, **AOIRepository**, **AuditLog**
Pasos	1. Realizar POST /aoi con valores no canónicos 2. Verificar species “Merluza” → “merluza” (canon) 3. Verificar name → “Plataforma Sur” (trim/título) 4. Verificar en DB metadatos de creación (ownerId, createdAt, CRS) 5. Verificar auditoría con valores canónicos
Resultado esperado	Datos normalizados persistidos; auditoría coherente.
Postcondiciones	AOI listo para búsquedas y filtros consistentes.
Prioridad	Media
Tipo	Integración

Caso de Uso	CU-03
Requisitos relacionados	RF-03, RNF-06
Criterios de aceptación	Búsquedas por nombre/especie encuentran el AOI con valores normalizados.

Pruebas de aceptación

ID	TC-A-03-01
Nombre	Crear AOI desde UI (mapa)
Objetivo	Validar UX para dibujar polígono, cargar nombre/especies y crear AOI.
Precondiciones	UI con visor; herramientas de dibujo habilitadas; usuario logueado.
Datos de entrada	Polígono dibujado en el mapa; name="Plataforma Sur"; species=["merluza"].
Componentes involucrados	Front-end Web (Visor AOI), **API Gateway (REST)**, **AOIService**, **AOIRepository**, **AuditLog**
Pasos	1. Abrir visor y seleccionar “Crear AOI” 2. Dibujar polígono y confirmar geometría 3. Completar nombre y especies 4. Enviar formulario de creación5 5. Verificar confirmación visual y listado de AOIs con el nuevo ítem
Resultado esperado	AOI visible en la capa del usuario con nombre y especies seleccionadas.
Postcondiciones	AOI disponible para configurar suscripciones.
Prioridad	Alta
Tipo	Aceptación
Caso de Uso	CU-03
Requisitos relacionados	RF-03, RNF-05
Criterios de aceptación	Mensajes claros; controles accesibles; sin errores en consola.

ID	TC-A-03-02
Nombre	Feedback de error (geometría inválida)
Objetivo	Mostrar errores comprensibles cuando la geometría no es válida.
Precondiciones	Validadores activos en cliente y servidor.
Datos de entrada	Polígono autointersectado.
Componentes involucrados	Front-end Web (Visor AOI), **API Gateway**, **AOIService**, **AuditLog**

Pasos	1. Dibujar un polígono autointersectado 2. Enviar el formulario 3. Verificar que la UI muestra mensaje “Geometría inválida” y destaca el área 4. Corregir polígono y reintentar la creación 5. Verificar creación exitosa con la geometría corregida
Resultado esperado	UI bloquea envío inválido y guía la corrección; luego permite crear.
Postcondiciones	AOI solo se crea con geometría válida.
Prioridad	Media
Tipo	Aceptación
Caso de Uso	CU-03
Requisitos relacionados	RF-03, RNF-05
Criterios de aceptación	Mensaje no técnico y accesible; sin duplicar solicitudes.

ID	TC-A-03-03
Nombre	Restricción por permisos (RBAC)
Objetivo	Impedir crear AOI si el rol carece de privilegios.
Precondiciones	Usuario con rol sin permiso AOI_CREATE.
Datos de entrada	Polígono válido; name="AOI Limitado".
Componentes involucrados	Front-end Web (Visor AOI), **API Gateway**, **AOIService**, **OwnershipPolicy/RBAC**, **AuditLog**
Pasos	1. Abrir “Crear AOI” con usuario sin permisos 2. Completar datos válidos 3. Enviar formulario 4. Verificar que la UI informa “Permiso insuficiente” 5. Verificar que no aparece el AOI en el listado
Resultado esperado	Operación denegada con mensaje claro; sin persistencia.
Postcondiciones	Auditoría de denegación de acceso.
Prioridad	Media
Tipo	Aceptación
Caso de Uso	CU-03
Requisitos relacionados	RF-03, RNF-02, RNF-05
Criterios de aceptación	UX consistente; no filtra detalles internos de seguridad.

CU-04 Editar área de interés (AOI)

Pruebas Unitarias

ID	TC-U-04-01
Nombre	Edición válida (camino feliz)
Objetivo	Actualizar un AOI existente con cambios válidos.
Precondiciones	aoiId existente y del ownerId actual; cambios válidos.
Datos de entrada	id=aoi-123, changes={ name: "Plataforma Sur V2", species: ["merluza","caballa"] }.
Componentes involucrados	CUT: AOIService — **Dobles/colaboradores:** AOIRepository (mock), GeometryValidator (stub, si hay cambios de geometría), OwnershipPolicy (stub), AuditLog (mock)
Pasos	1. Invocar AOIService.updateAOI(id, changes) 2. Verificar AOIRepository.findById(id) → aoi existente 3. Verificar OwnershipPolicy.check(aoi.ownerId, currentUser) → autorizado 4. Verificar validación de campos (name/species) y geometría si corresponde 5. Verificar AOIRepository.save(aoi*) con cambios aplicados 6. Verificar AuditLog.log('aoi_update', id, metadata)
Resultado esperado	Retorna ok; AOI actualizado.
Postcondiciones	AOI persistido con nuevos valores; auditoría registrada.
Prioridad	Alta
Tipo	Unitario
Caso de Uso	CU-04
Requisitos relacionados	RF-04, RNF-02, RNF-06
Criterios de aceptación	Cambios visibles en repositorio; sin modificar campos no incluidos en changes.

ID	TC-U-04-02
Nombre	AOI inexistente
Objetivo	Rechazar edición cuando el aoiId no existe.
Precondiciones	aoiId no presente en DB.
Datos de entrada	id=aoi-XYZ, changes={ name: "Nuevo nombre" }.
Componentes involucrados	CUT: AOIService — **Dobles/colaboradores:** AOIRepository (mock), AuditLog (mock)

Pasos	1. Invocar AOIService.updateAOI(idInexistente, changes) 2. Verificar AOIRepository.findById(idInexistente) → null 3. Verificar que no se llama a AOIRepository.save(aoi*) 4. Verificar AuditLog.log('aoi_update_failed', idInexistente, reason='NOT_FOUND')
Resultado esperado	Error NOT_FOUND.
Postcondiciones	Sin cambios en DB.
Prioridad	Alta
Tipo	Unitario (Errores)
Caso de Uso	CU-04
Requisitos relacionados	RF-04, RNF-06
Criterios de aceptación	Mensaje claro y consistente; ningún efecto lateral.

ID	TC-U-04-03
Nombre	Falta de permisos (owner)
Objetivo	Bloquear la edición si el usuario no es dueño ni tiene rol con privilegio.
Precondiciones	AOI existe pero ownerId != currentUser.
Datos de entrada	id=aoi-123, changes={ species: ["merluza"] }.
Componentes involucrados	CUT: AOIService — **Dobles/colaboradores:** AOIRepository (mock), OwnershipPolicy (stub), AuditLog (mock)
Pasos	1. Invocar AOIService.updateAOI(id, changes) 2. Verificar AOIRepository.findById(id) → aoi con otro owner 3. Verificar OwnershipPolicy.check(aoi.ownerId, currentUser) → no autorizado 4. Verificar que no se llama a AOIRepository.save(aoi*) 5. Verificar AuditLog.log('aoi_update_denied', id, reason='FORBIDDEN')
Resultado esperado	Error FORBIDDEN.
Postcondiciones	AOI sin cambios; auditoría de denegación.
Prioridad	Media
Tipo	Unitario (Seguridad)
Caso de Uso	CU-04
Requisitos relacionados	RF-04, RNF-02
Criterios de aceptación	No se exponen detalles internos de la política; sin efectos en repositorio.

Pruebas de Integración

ID	TC-I-04-01
Nombre	PUT /aoi/{id} (feliz)

Objetivo	Validar el flujo de edición desde API hasta DB y auditoría.
Precondiciones	AOI existente del usuario; API y servicio activos.
Datos de entrada	PUT /aoi/aoi-123 { name: "Plataforma Sur V2", species: ["merluza","caballa"] }.
Componentes involucrados	API Gateway (REST), **AOIService**, **AOIRepository** (MySQL), **OwnershipPolicy**, **AuditLog**
Pasos	1. Realizar PUT /aoi/{id} con payload válido 2. Verificar 200 OK con ok=true 3. Verificar en DB que el AOI refleja los cambios 4. Verificar AuditLog aoi_update con id y requestId
Resultado esperado	AOI actualizado; respuesta 200.
Postcondiciones	Auditoría consistente con los cambios.
Prioridad	Alta
Tipo	Integración
Caso de Uso	CU-04
Requisitos relacionados	RF-04, RNF-01, RNF-06
Criterios de aceptación	Latencia bajo umbral; datos consistentes en lectura posterior.

ID	TC-I-04-02
Nombre	Rechazo por geometría inválida en edición
Objetivo	Confirmar respuesta de error cuando la nueva geometría es inválida.
Precondiciones	AOI existente; validadores activos.
Datos de entrada	PUT /aoi/aoi-123 { geometry: <self-intersection> }.
Componentes involucrados	API Gateway, **AOIService**, **GeometryValidator**, **AOIRepository**, **AuditLog**
Pasos	1. Realizar PUT /aoi/{id} con geometría inválida 2. Verificar 400 Bad Request GEOMETRY_INVALID 3. Verificar que el registro en DB no cambió 4. Verificar AuditLog aoi_update_failed con causa
Resultado esperado	Error 400; AOI sin cambios.
Postcondiciones	Estado intacto; traza de validación.
Prioridad	Alta
Tipo	Integración
Caso de Uso	CU-04
Requisitos relacionados	RF-04, RNF-06
Criterios de aceptación	Mensaje claro y consistente; sin registros parciales.

ID	TC-I-04-03
Nombre	Control de permisos (RBAC/Owner)
Objetivo	Asegurar que solo el dueño o roles autorizados pueden editar.
Precondiciones	AOI con ownerId distinto al usuario actual; RBAC activo.
Datos de entrada	PUT /aoi/aoi-123 { name: "Cambio no permitido" }.
Componentes involucrados	API Gateway, **AOIService**, **OwnershipPolicy**, **AOIRepository**, **AuditLog**
Pasos	1. Realizar PUT /aoi/{id} con usuario sin permisos 2. Verificar 403 Forbidden 3. Verificar que el AOI en DB no refleja cambios 4. Verificar AuditLog aoi_update_denied
Resultado esperado	Denegación por permisos; sin persistencia.
Postcondiciones	Auditoría de denegación.
Prioridad	Media
Tipo	Integración (Seguridad)
Caso de Uso	CU-04
Requisitos relacionados	RF-04, RNF-02
Criterios de aceptación	No se filtra información sensible; consistencia de estado.

Pruebas de aceptación

ID	TC-A-04-01
Nombre	Editar AOI desde UI (atributos)
Objetivo	Validar que el usuario puede editar nombre/especies desde el panel.
Precondiciones	Usuario logueado; AOI visible en listado del usuario.
Datos de entrada	name="Plataforma Sur V2", species=["merluza","caballa"].
Componentes involucrados	Front-end Web (Visor/Listado AOI), **API Gateway**, **AOIService**, **AOIRepository**, **AuditLog**
Pasos	1. Abrir el listado de AOIs en la UI 2. Seleccionar AOI y abrir edición 3. Modificar nombre/especies 4. Guardar cambios 5. Verificar actualización visual y confirmación
Resultado esperado	AOI actualizado en UI y backend.
Postcondiciones	Cambios reflejados en consultas/visor.
Prioridad	Alta
Tipo	Aceptación
Caso de Uso	CU-04
Requisitos relacionados	RF-04, RNF-05
Criterios de aceptación	Mensajes claros; controles accesibles; sin errores de consola.

ID	TC-A-04-02
Nombre	Editar geometría desde UI (validación y guardado)
Objetivo	Permitir edición del polígono con validación previa y feedback visual.
Precondiciones	Herramientas de dibujo habilitadas; validadores activos.
Datos de entrada	Nuevo polígono válido para aoi-123.
Componentes involucrados	Front-end Web (Visor AOI), **API Gateway**, **AOIService**, **GeometryValidator**, **AOIRepository**, **AuditLog**
Pasos	1. Abrir AOI en modo edición de geometría 2. Ajustar vértices y confirmar 3. Guardar cambios 4. Verificar feedback de éxito en UI 5. Verificar que el visor renderiza el polígono actualizado
Resultado esperado	Geometría actualizada y visible.
Postcondiciones	AOI consistente en backend y mapa.
Prioridad	Media
Tipo	Aceptación
Caso de Uso	CU-04
Requisitos relacionados	RF-04, RNF-05
Criterios de aceptación	Validación clara; no se guardan geometrías inválidas.

ID	TC-A-04-03
Nombre	Denegación por permisos (UI)
Objetivo	Mostrar mensaje claro y bloqueo de controles cuando no hay permisos de edición.
Precondiciones	Usuario sin permiso para editar el AOI seleccionado.
Datos de entrada	Cualquier cambio sobre aoi-123.
Componentes involucrados	Front-end Web (Visor/Listado AOI), **API Gateway**, **AOIService**, **OwnershipPolicy/RBAC**, **AuditLog**
Pasos	1. Abrir AOI con usuario sin permisos 2. Intentar modificar nombre/geometría 3. Guardar cambios 4. Verificar mensaje “Permiso insuficiente” 5. Verificar que el AOI no se actualiza en UI ni backend
Resultado esperado	Operación denegada, UX clara.
Postcondiciones	Auditoría de denegación; estado inalterado.
Prioridad	Media

Tipo	Aceptación
Caso de Uso	CU-04
Requisitos relacionados	RF-04, RNF-02, RNF-05
Criterios de aceptación	No expone detalles internos; controles deshabilitados o acción bloqueada.

CU-05 Eliminar área de interés (AOI)

Pruebas Unitarias

ID	TC-U-05-01
Nombre	Eliminación válida (soft delete — camino feliz)
Objetivo	Marcar un AOI como eliminado sin perder historial (soft delete).
Precondiciones	aoiId existente y pertenece al currentUser o el rol autoriza borrado.
Datos de entrada	id=aoi-123.
Componentes involucrados	CUT: AOIService — **Dobles/colaboradores:** AOIRepository (mock), OwnershipPolicy (stub), AuditLog (mock)
Pasos	1. Invocar AOIService.deleteAOI(id) 2. Verificar AOIRepository.softDelete(id) → ok 3. Verificar AuditLog.log('aoi_delete', id, metadata)
Resultado esperado	Retorna ok=true.
Postcondiciones	AOI en estado deleted=true o status=DELETED; consultas normales no lo listan.
Prioridad	Alta
Tipo	Unitario
Caso de Uso	CU-05
Requisitos relacionados	RF-05, RNF-06
Criterios de aceptación	No se elimina físicamente el registro; operaciones derivadas omiten AOIs borrados.

ID	TC-U-05-02
Nombre	AOI inexistente
Objetivo	Responder NOT_FOUND cuando el aoiId no existe o ya está borrado.
Precondiciones	aoiId no presente o deleted=true.
Datos de entrada	id=aoi-XYZ.
Componentes involucrados	CUT: AOIService — **Dobles/colaboradores:** AOIRepository (mock), AuditLog (mock)

Pasos	1. Invocar AOIService.deleteAOI(idInexistente) 2. Verificar AOIRepository.softDelete(idInexistente) → not found 3. Verificar AuditLog.log('aoi_delete_failed', idInexistente, reason='NOT_FOUND')
Resultado esperado	Error NOT_FOUND.
Postcondiciones	Sin cambios en DB.
Prioridad	Media
Tipo	Unitario (Errores)
Caso de Uso	CU-05
Requisitos relacionados	RF-05, RNF-06
Criterios de aceptación	Mensaje consistente; no hay efectos laterales.

ID	TC-U-05-03
Nombre	Falta de permisos (owner/RBAC)
Objetivo	Impedir borrado si el usuario no es dueño ni tiene rol con privilegio.
Precondiciones	aoiId existe pero ownerId != currentUser y rol sin AOI_DELETE.
Datos de entrada	id=aoi-123.
Componentes involucrados	CUT: AOIService — **Dobles/colaboradores:** AOIRepository (mock), OwnershipPolicy (stub), AuditLog (mock)
Pasos	1. Invocar AOIService.deleteAOI(id) 2. Verificar OwnershipPolicy.check(aoi.ownerId, currentUser) → no autorizado 3. Verificar que no se llama a AOIRepository.softDelete 4. Verificar AuditLog.log('aoi_delete_denied', id, reason='FORBIDDEN')
Resultado esperado	Error FORBIDDEN.
Postcondiciones	AOI sin cambios.
Prioridad	Media
Tipo	Unitario (Seguridad)
Caso de Uso	CU-05
Requisitos relacionados	RF-05, RNF-02
Criterios de aceptación	No se filtran detalles internos; sin persistencia.

Pruebas de Integración

ID	TC-I-05-01
Nombre	DELETE /aoi/{id} (feliz)
Objetivo	Validar borrado (soft) desde API hasta DB y auditoría.
Precondiciones	AOI existente del usuario; API y servicio activos.
Datos de entrada	DELETE /aoi/aoi-123.

Componentes involucrados	API Gateway (REST), **AOIService**, **AOIRepository** (MySQL), **OwnershipPolicy**, **AuditLog**
Pasos	1. Realizar DELETE /aoi/{id} 2. Verificar 200 OK con ok=true 3. Verificar en DB flag deleted=true o status=DELETED 4. Verificar AuditLog aoi_delete con id y requestId
Resultado esperado	Borrado lógico aplicado; respuesta 200.
Postcondiciones	AOI no visible en listados por defecto.
Prioridad	Alta
Tipo	Integración
Caso de Uso	CU-05
Requisitos relacionados	RF-05, RNF-01, RNF-06
Criterios de aceptación	Latencia bajo umbral; lecturas posteriores no devuelven el AOI.

ID	TC-I-05-02
Nombre	Protección de integridad referencial (Suscripciones)
Objetivo	Verificar que suscripciones ligadas al AOI quedan **inactivas** o **re-asignadas** según política.
Precondiciones	SubscriptionRepository con suscripciones activas para aoi-123.
Datos de entrada	DELETE /aoi/aoi-123.
Componentes involucrados	API Gateway, **AOIService**, **AOIRepository**, **SubscriptionRepository**, **AuditLog**
Pasos	1. Ejecutar DELETE /aoi/{id} 2. Verificar respuesta 200 OK 3. Verificar en DB que las suscripciones ligadas quedan status=INACTIVE (o la política definida) 4. Verificar auditoría aoi_delete y subscription_auto_update
Resultado esperado	Suscripciones no quedan “huérfanas”; política aplicada.
Postcondiciones	Sistema consistente; sin disparar alertas para AOI borrado.
Prioridad	Alta
Tipo	Integración
Caso de Uso	CU-05
Requisitos relacionados	RF-05, RF-15, RNF-03, RNF-06
Criterios de aceptación	No hay referencias activas a AOI borrado; auditoría doble presente.

ID	TC-I-05-03
Nombre	Denegación por permisos (API)
Objetivo	Confirmar que el backend rechaza el DELETE cuando el usuario no es owner ni tiene rol.

Precondiciones	AOI existe; usuario sin privilegios.
Datos de entrada	DELETE /aoi/aoi-123.
Componentes involucrados	API Gateway, **AOIService**, **OwnershipPolicy**, **AOIRepository**, **AuditLog**
Pasos	1. Realizar DELETE /aoi/{id} con usuario sin permisos 2. Verificar 403 Forbidden 3. Verificar que el registro en DB no cambió 4. Verificar AuditLog aoi_delete_denied
Resultado esperado	Denegado; sin persistencia.
Postcondiciones	Auditoría de denegación.
Prioridad	Media
Tipo	Integración (Seguridad)
Caso de Uso	CU-05
Requisitos relacionados	RF-05, RNF-02
Criterios de aceptación	No fuga de información sensible; estado intacto.

Pruebas de aceptación

ID	TC-A-05-01
Nombre	Eliminar AOI desde UI (confirmación)
Objetivo	Validar UX de eliminación con diálogo de confirmación.
Precondiciones	Usuario logueado; AOI visible; permisos de borrado.
Datos de entrada	id=aoi-123.
Componentes involucrados	Front-end Web (Listado/Visor AOI), **API Gateway (REST)**, **AOIService**, **AOIRepository**, **AuditLog**
Pasos	1. Abrir listado/visor y seleccionar AOI 2. Click en “Eliminar” 3. Confirmar en el diálogo modal 4. Verificar notificación de éxito 5. Verificar que el AOI ya no aparece en el listado por defecto
Resultado esperado	AOI ya no visible; feedback claro.
Postcondiciones	AOI en estado borrado; auditoría presente.
Prioridad	Alta
Tipo	Aceptación
Caso de Uso	CU-05
Requisitos relacionados	RF-05, RNF-05
Criterios de aceptación	Mensaje claro; accesible; sin errores en consola.

ID	TC-A-05-02
Nombre	Borrado bloqueado por permisos (UX)

Objetivo	Mostrar mensaje claro y bloquear acción para usuario sin permisos.
Precondiciones	Usuario sin permiso AOI_DELETE.
Datos de entrada	id=aoi-123.
Componentes involucrados	Front-end Web (Listado/Visor AOI), **API Gateway**, **AOIService**, **OwnershipPolicy/RBAC**, **AuditLog**
Pasos	1. Abrir AOI con usuario sin permisos 2. Intentar eliminar 3. Verificar mensaje “Permiso insuficiente” 4. Verificar que el AOI continúa visible sin cambios 5. Verificar registro de auditoría de intento denegado
Resultado esperado	Acción denegada; estado intacto.
Postcondiciones	Auditoría de denegación.
Prioridad	Media
Tipo	Aceptación
Caso de Uso	CU-05
Requisitos relacionados	RF-05, RNF-05, RNF-02
Criterios de aceptación	UX coherente; no expone detalles internos.

ID	TC-A-05-03
Nombre	Efecto en suscripciones (feedback UI)
Objetivo	Informar al usuario qué ocurre con sus suscripciones ligadas al AOI borrado.
Precondiciones	Existen suscripciones activas sobre aoi-123.
Datos de entrada	id=aoi-123.
Componentes involucrados	Front-end Web (Suscripciones), **API Gateway**, **AOIService**, **SubscriptionRepository**, **AuditLog**
Pasos	1. Eliminar AOI desde UI 2. Abrir panel de suscripcione 3. Verificar que las suscripciones vinculadas aparecen como inactivas o re-asignadas (según política) 4. Verificar notificación informativa en UI 5. Verificar que no se pueden crear alertas sobre AOI borrado
Resultado esperado	UI refleja correctamente el efecto; sin alertas futuras sobre el AOI eliminado.
Postcondiciones	Consistencia funcional percibida por el usuario.
Prioridad	Media
Tipo	Aceptación
Caso de Uso	CU-05
Requisitos relacionados	RF-05, RF-15, RNF-05
Criterios de aceptación	Mensajes claros; no hay opciones incoherentes habilitadas.

CU-06 Ingestar producto SST (L2)

Pruebas Unitarias

ID	TC-U-06-01
Nombre	Ingesta SST (ventana válida — camino feliz)
Objetivo	Listar productos L2 de SST de la ventana configurada, descargar, registrar escenas y auditar.
Precondiciones	Configuración de ventana activa (p. ej., últimas 24 h); RemoteCatalog, StorageService y SceneRepository disponibles.
Datos de entrada	window="PT24H", varname="SST".
Componentes involucrados	CUT: IngestionSSTService — **Dobles/colaboradores:** RemoteCatalog (stub), StorageService (mock/spy), SceneRepository (mock), AuditLog (mock), Clock (stub), IdempotencyStore (stub)
Pasos	1. Invocar IngestionSSTService.run() 2. Verificar RemoteCatalog.listL2("SST", window) → files[] 3. Verificar StorageService.download(files[]) → storedURIs[] 4. Verificar SceneRepository.registerScenes(meta, storedURIs[]) 5. Verificar AuditLog.log('sst_ingest', count)
Resultado esperado	Escenas registradas con sus URIs; auditoría correcta.
Postcondiciones	Nuevos Scene(SST) almacenados con metadatos (acqTime, bbox, cobertura).
Prioridad	Alta
Tipo	Unitario
Caso de Uso	CU-06
Requisitos relacionados	RF-06, RNF-01, RNF-03, RNF-06
Criterios de aceptación	Conteo de escenas coincide con files[]; sin duplicados en la ventana.

ID	TC-U-06-02
Nombre	Idempotencia (reintento sin duplicar)
Objetivo	Evitar registros duplicados si la ingesta se ejecuta nuevamente sobre la misma ventana.
Precondiciones	IdempotencyStore contiene hashes/ids de escenas ya ingeridas.
Datos de entrada	files[] repetidos respecto a la corrida anterior.

Componentes involucrados	CUT: IngestionSSTService — **Dobles/colaboradores:** RemoteCatalog (stub), IdempotencyStore (stub), SceneRepository (mock), AuditLog (mock)
Pasos	1. Invocar IngestionSSTService.run() con files[] ya procesados 2. Verificar filtro por IdempotencyStore (skip existentes) 3. Verificar que SceneRepository.registerScenes no recibe duplicados 4. Verificar AuditLog.log('sst_ingest', countProcesados) con count solo de nuevos
Resultado esperado	Se procesan solo escenas nuevas; no hay duplicados.
Postcondiciones	Estado consistente; auditoría del diferencial.
Prioridad	Alta
Tipo	Unitario
Caso de Uso	CU-06
Requisitos relacionados	RF-06, RNF-03, RNF-06
Criterios de aceptación	Insert/Upsert idempotente verificado; conteo correcto.

ID	TC-U-06-03
Nombre	Tolerancia a fallos (descarga parcial)
Objetivo	Registrar adecuadamente errores de descarga y continuar con el resto.
Precondiciones	StorageService.download simula fallo intermitente en un subconjunto de files[].
Datos de entrada	files[] con 20% de fallos simulados.
Componentes involucrados	CUT: IngestionSSTService — **Dobles/colaboradores:** RemoteCatalog (stub), StorageService (mock), SceneRepository (mock), AuditLog (mock), RetryPolicy (stub)
Pasos	1. Invocar IngestionSSTService.run() 2. Forzar fallos intermitentes en StorageService.download 3. Aplicar RetryPolicy (p. ej., 3 reintentos exponenciales) 4. Registrar escenas que finalmente descargaron 5. Verificar AuditLog.log('sst_ingest', countOK, countFail)
Resultado esperado	Se registra el mayor número posible de escenas; fallos auditados.
Postcondiciones	Escenas exitosas disponibles; errores catalogados para re-proceso.
Prioridad	Media
Tipo	Unitario (Resiliencia)
Caso de Uso	CU-06

Requisitos relacionados	RF-06, RNF-01, RNF-04
Criterios de aceptación	Reintentos aplicados; métricas de éxito/fallo consistentes.

Pruebas de Integración

ID	TC-I-06-01
Nombre	Integración con catálogo remoto (S3/HTTPS/OPeNDAP)
Objetivo	Validar la compatibilidad del RemoteCatalog con el endpoint real y la selección por ventana.
Precondiciones	Endpoint remoto accesible; credenciales válidas si aplica.
Datos de entrada	window=PT24H, varname=SST.
Componentes involucrados	IngestionSSTService, **RemoteCatalog** (real o sandbox), **StorageService** (sandbox), **SceneRepository** (DB), **AuditLog**
Pasos	1. Ejecutar IngestionSSTService.run() 2. Verificar respuesta de RemoteCatalog.listL2 con metadatos mínimos (acqTime, bbox, tamaño) 3. Verificar descargas en StorageService (paths/URIs válidos) 4. Verificar registros en SceneRepository (SST) con metadatos y checks de cobertura
Resultado esperado	Ingesta completa desde el catálogo remoto; escenas disponibles.
Postcondiciones	Rásteres/archivos en storage; escenas indexadas para consultas posteriores.
Prioridad	Alta
Tipo	Integración
Caso de Uso	CU-06
Requisitos relacionados	RF-06, RNF-01, RNF-03
Criterios de aceptación	Sin timeouts; volumen esperado de escenas según ventana.

ID	TC-I-06-02
Nombre	Throughput y paralelismo de descarga
Objetivo	Validar el desempeño del StorageService con descargas concurrentes.
Precondiciones	Ancho de banda de prueba; maxConcurrency configurado.
Datos de entrada	files[] (N ≥ 50).
Componentes involucrados	IngestionSSTService, **StorageService** (descarga concurrente), **SceneRepository**, **AuditLog**, **MetricsCollector**

Pasos	1. Ejecutar la ingesta con maxConcurrency configurado 2. Medir tiempo total y por archivo 3. Verificar que no hay throttling/ban (respetar rate-limit del proveedor) 4. Registrar métricas (p50/p95 latencias, MB/s)
Resultado esperado	Throughput dentro de umbrales; sin errores por concurrencia.
Postcondiciones	Métricas almacenadas para capacity planning.
Prioridad	Media
Tipo	Integración (Rendimiento)
Caso de Uso	CU-06
Requisitos relacionados	RF-06, RNF-01, RNF-03
Criterios de aceptación	p95 dentro de SLA; tasa de errores < 1%.

ID	TC-I-06-03
Nombre	Idempotencia multi-nodo (scheduler distribuido)
Objetivo	Evitar doble procesamiento si dos instancias corren la misma ventana.
Precondiciones	Dos nodos del scheduler ejecutan run() para la misma ventana; IdempotencyStore compartido.
Datos de entrada	window=PT24H, varname=SST.
Componentes involucrados	IngestionSSTService (x2), **RemoteCatalog**, **IdempotencyStore** (compartido), **SceneRepository**, **AuditLog**, **DistributedLock**
Pasos	1. Disparar run() en dos nodos casi simultáneamente 2. Verificar coordinación via DistributedLock (lease/heartbeat) 3. Verificar que solo una instancia registra cada escena 4. Verificar auditoría sin duplicados
Resultado esperado	Registros únicos por escena; sin carreras.
Postcondiciones	Lista de escenas sin duplicidad; lock liberado correctamente.
Prioridad	Alta
Tipo	Integración (Concurrencia)
Caso de Uso	CU-06
Requisitos relacionados	RF-06, RNF-03, RNF-04
Criterios de aceptación	0 duplicados; lock renovado/lanzado sin deadlocks.

Pruebas de aceptación

ID	TC-A-06-01
Nombre	Ingesta ejecutada por Scheduler (operación diaria)
Objetivo	Confirmar que la ingesta se dispara automáticamente según el cron configurado y deja trazabilidad.
Precondiciones	Scheduler activo (cron diario o cada N horas); monitoreo habilitado.
Datos de entrada	Job programado para SST.
Componentes involucrados	Scheduler, **IngestionSSTService**, **RemoteCatalog**, **StorageService**, **SceneRepository**, **AuditLog**, **Monitoring/Alerts**
Pasos	1. Esperar/forzar disparo del job de ingesta 2. Verificar métricas y logs del job con conteo de escenas procesadas 3. Verificar nuevas escenas en visor/herramientas internas (si aplica) 4. Verificar alertas/healthchecks en caso de fallo
Resultado esperado	Job ejecutado; escenas nuevas disponibles; observabilidad correcta.
Postcondiciones	Historial de ejecuciones consultable para auditoría.
Prioridad	Alta
Tipo	Aceptación (Operación)
Caso de Uso	CU-06
Requisitos relacionados	RF-06, RNF-04, RNF-03
Criterios de aceptación	Sin incidentes; SLA de ventana cumplido.

ID	TC-A-06-02
Nombre	Reprocesamiento manual de ventana
Objetivo	Permitir re-ejecutar la ingesta para una ventana concreta sin duplicar.
Precondiciones	UI/CLI de operación habilitada para re-proceso; idempotencia activa.
Datos de entrada	window=2025-10-20T00:00Z/2025-10-21T00:00Z.
Componentes involucrados	Operator UI/CLI, **IngestionSSTService**, **RemoteCatalog**, **IdempotencyStore**, **SceneRepository**, **AuditLog**
Pasos	1. Solicitar re-proceso de ventana específica 2. Verificar ejecución exitosa 3. Validar que no se insertan escenas duplicadas 4. Verificar auditoría del re-proceso con rango temporal
Resultado esperado	Re-proceso limpio; estado consistente.

Postcondiciones	Escenas completas para la ventana; logs del re-proceso.
Prioridad	Media
Tipo	Aceptación
Caso de Uso	CU-06
Requisitos relacionados	RF-06, RNF-03, RNF-06
Criterios de aceptación	Diferencial aplicado; tiempos aceptables.

ID	TC-A-06-03
Nombre	Degradación controlada del proveedor
Objetivo	Mantener operación con retries/backoff y notificar si el proveedor remoto degrada el servicio.
Precondiciones	Simulación de respuestas 5xx/timeout intermitentes del catálogo o storage.
Datos de entrada	Job de ingesta estándar.
Componentes involucrados	Scheduler, **IngestionSSTService**, **RemoteCatalog**, **StorageService**, **AuditLog**, **Monitoring/Alerts**, **RetryPolicy**
Pasos	1. Ejecutar job con fallos intermitentes del proveedor 2. Verificar aplicación de backoff y número máximo de reintentos 3. Verificar que se procesa el subconjunto posible 4. Verificar alerta/incident si error > umbral definido
Resultado esperado	Operación parcialmente exitosa; notificación de incidente si corresponde.
Postcondiciones	Tareas pendientes marcadas para retry; visibilidad en panel de monitoreo.
Prioridad	Media
Tipo	Aceptación (Resiliencia/Operación)
Caso de Uso	CU-06
Requisitos relacionados	RF-06, RNF-04, RNF-01
Criterios de aceptación	SLO no se incumple sostenidamente; alertas a tiempo.

CU-07 Ingestar producto Chl-a (L2)

Pruebas Unitarias

ID	TC-U-07-01
Nombre	Ingesta Chl-a (ventana válida — camino feliz)
Objetivo	Listar productos L2 de Chl-a en la ventana configurada, descargar, registrar escenas y auditar.

Precondiciones	Ventana configurada (p. ej., últimas 24 h); RemoteCatalog, StorageService y SceneRepository disponibles.
Datos de entrada	window="PT24H", varname="CHLA".
Componentes involucrados	CUT: IngestionChlaService — **Dobles/colaboradores:** RemoteCatalog (stub), StorageService (mock/spy), SceneRepository (mock), AuditLog (mock), Clock (stub), IdempotencyStore (stub)
Pasos	1. Invocar IngestionChlaService.run() 2. Verificar RemoteCatalog.listL2("CHLA", window) → files[] 3. Verificar StorageService.download(files[]) → storedURIs[] 4. Verificar SceneRepository.registerScenes(meta, storedURIs[]) con type="CHLA" 5. Verificar AuditLog.log('chla_ingest', count)
Resultado esperado	Escenas CHLA registradas con URIs; auditoría correcta.
Postcondiciones	Nuevos Scene(CHLA) con metadatos (acqTime, bbox, cobertura).
Prioridad	Alta
Tipo	Unitario
Caso de Uso	CU-07
Requisitos relacionados	RF-07, RNF-01, RNF-03, RNF-06
Criterios de aceptación	Conteo de escenas coincide con files[]; sin duplicados.

ID	TC-U-07-02
Nombre	Enmascarado por calidad (flags QA/OC)
Objetivo	Registrar solo escenas/tiles cuya fracción válida supere el umbral de calidad (enmascarando nubes/glint).
Precondiciones	Política de calidad definida (p. ej., validFraction ≥ 0.6); QualityMasker disponible.
Datos de entrada	files[] con distintos validFraction.
Componentes involucrados	CUT: IngestionChlaService — **Dobles/colaboradores:** RemoteCatalog (stub con QA), QualityMasker (stub), StorageService (mock), SceneRepository (mock), AuditLog (mock)
Pasos	1. Invocar IngestionChlaService.run() 2. Aplicar QualityMasker sobre metadatos/flags de cada file 3. Filtrar escenas con validFraction < threshold 4. Registrar únicamente las escenas que cumplen 5. Verificar AuditLog.log('chla_ingest', countOK, countFiltered)
Resultado esperado	Solo se registran escenas que cumplen el umbral de calidad.
Postcondiciones	Escenas válidas disponibles para PFZ; baja señal ruidosa.
Prioridad	Media
Tipo	Unitario (Reglas de calidad)

Caso de Uso	CU-07
Requisitos relacionados	RF-07, RNF-01, RNF-06
Criterios de aceptación	countFiltered correcto; SceneRepository no recibe descartadas.

ID	TC-U-07-03
Nombre	Idempotencia (reintento sin duplicar)
Objetivo	Evitar duplicados al re-ejecutar la ingesta sobre la misma ventana.
Precondiciones	IdempotencyStore con hashes/ids de escenas CHLA ya ingeridas.
Datos de entrada	files[] repetidos respecto de corrida previa.
Componentes involucrados	CUT: IngestionChlaService — **Dobles/colaboradores:** RemoteCatalog (stub), IdempotencyStore (stub), SceneRepository (mock), AuditLog (mock)
Pasos	1. Invocar IngestionChlaService.run() con files[] previamente procesados 2. Filtrar por IdempotencyStore 3. Confirmar que SceneRepository.registerScenes no recibe duplicados 4. Verificar AuditLog.log('chla_ingest', countSoloNuevos)
Resultado esperado	Solo se procesan nuevas escenas; auditoría diferencial.
Postcondiciones	Estado consistente; sin duplicidad.
Prioridad	Alta
Tipo	Unitario
Caso de Uso	CU-07
Requisitos relacionados	RF-07, RNF-03, RNF-06
Criterios de aceptación	Upsert idempotente verificado; conteos correctos.

Pruebas de Integración

ID	TC-I-07-01
Nombre	Integración con catálogo remoto CHLA (fuentes múltiples)
Objetivo	Validar RemoteCatalog contra endpoints reales (p. ej., Sentinel-3/OLCI, NOAA-20 VIIRS, Aqua/Terra MODIS) y selección por ventana.
Precondiciones	Endpoints accesibles (S3/HTTPS/OPeNDAP); credenciales cuando aplique.
Datos de entrada	window=PT24H, varname=CHLA.
Componentes involucrados	IngestionChlaService, **RemoteCatalog** (real/sandbox), **StorageService** (sandbox), **SceneRepository** (DB), **AuditLog**

Pasos	1. Ejecutar IngestionChlaService.run() 2. Verificar obtención de files[] con metadatos (sensor, acqTime, bbox, tamaño) 3. Descargar a StorageService y obtener URIs 4. Registrar escenas CHLA en SceneRepository con type="CHLA"
Resultado esperado	Ingesta exitosa desde múltiples fuentes; escenas disponibles.
Postcondiciones	Archivos en storage; escenas indexadas para PFZ.
Prioridad	Alta
Tipo	Integración
Caso de Uso	CU-07
Requisitos relacionados	RF-07, RNF-01, RNF-03
Criterios de aceptación	Volumen esperado por ventana; sin timeouts críticos.

ID	TC-I-07-02
Nombre	Rendimiento de descarga y parsers OC
Objetivo	Medir throughput con descargas concurrentes y parsing de variables oceánicas (Chl-a/flags).
Precondiciones	maxConcurrency configurado; parsers OC habilitados; ancho de banda de prueba.
Datos de entrada	files[] (N ≥ 50) con distribución horaria.
Componentes involucrados	IngestionChlaService, **StorageService** (concurrencia), **SceneRepository**, **AuditLog**, **MetricsCollector**
Pasos	1. Ejecutar ingesta con concurrencia configurada 2. Medir tiempo total y p95 por archivo 3. Verificar que parsing no se convierte en cuello de botella 4. Registrar métricas (MB/s, p50/p95, tasa de error)
Resultado esperado	Throughput dentro de SLA; parsers estables.
Postcondiciones	Métricas disponibles para capacity planning.
Prioridad	Media
Tipo	Integración (Rendimiento)
Caso de Uso	CU-07
Requisitos relacionados	RF-07, RNF-01, RNF-03
Criterios de aceptación	p95 bajo umbral; error rate < 1%.

ID	TC-I-07-03
Nombre	Idempotencia y locking distribuido
Objetivo	Asegurar que dos instancias concurrentes no duplican escenas en la misma ventana.

Precondiciones	Dos nodos ejecutan run() para misma ventana; IdempotencyStore compartido; DistributedLock activo.
Datos de entrada	window=PT24H, varname=CHLA.
Componentes involucrados	IngestionChlaService (x2), **RemoteCatalog**, **IdempotencyStore** (compartido), **SceneRepository**, **AuditLog**, **DistributedLock**
Pasos	1. Lanzar dos run() casi a la vez 2. Verificar que un nodo toma el lock y el otro espera/omite 3. Confirmar registros únicos por escena 4. Verificar auditoría sin duplicados
Resultado esperado	0 duplicados; coordinación correcta.
Postcondiciones	Lock liberado; lista de escenas consistente.
Prioridad	Alta
Tipo	Integración (Concurrencia)
Caso de Uso	CU-07
Requisitos relacionados	RF-07, RNF-03, RNF-04
Criterios de aceptación	Sin deadlocks; sin carreras; auditoría limpia.

Pruebas de aceptación

ID	TC-A-07-01
Nombre	Job programado Chl-a (operación)
Objetivo	Confirmar ejecución automática por cron y disponibilidad de nuevas escenas CHLA.
Precondiciones	Scheduler activo; monitoreo/alertas habilitados.
Datos de entrada	Job de ingesta para CHLA.
Componentes involucrados	Scheduler, **IngestionChlaService**, **RemoteCatalog**, **StorageService**, **SceneRepository**, **AuditLog**, **Monitoring/Alerts**
Pasos	1. Disparar/esperar ejecución del job 2. Verificar conteos de escenas procesadas en logs/Metrics 3. Corroborar nuevas escenas en herramientas internas/visor (si aplica) 4. Verificar alertas si error supera umbral
Resultado esperado	Job ejecutado; escenas nuevas disponibles; observabilidad correcta.
Postcondiciones	Historial de ejecuciones consultable.
Prioridad	Alta
Tipo	Aceptación (Operación)
Caso de Uso	CU-07
Requisitos relacionados	RF-07, RNF-04, RNF-03

Criterios de aceptación	SLA cumplido; sin incidentes críticos.

ID	TC-A-07-02
Nombre	Reprocesamiento de ventana (sin duplicar)
Objetivo	Re-ejecutar ingesta para un rango temporal específico garantizando idempotencia.
Precondiciones	UI/CLI operativa; idempotencia habilitada.
Datos de entrada	window=2025-10-20T00:00Z/2025-10-21T00:00Z.
Componentes involucrados	Operator UI/CLI, **IngestionChlaService**, **RemoteCatalog**, **IdempotencyStore**, **SceneRepository**, **AuditLog**
Pasos	1. Solicitar re-proceso para el rango 2. Verificar ejecución y conteo de “nuevos vs existentes” 3. Verificar SceneRepository sin duplicados 4. Verificar auditoría del re-proceso
Resultado esperado	Re-proceso limpio; estado consistente.
Postcondiciones	Escenas completas para la ventana; logs asociados.
Prioridad	Media
Tipo	Aceptación
Caso de Uso	CU-07
Requisitos relacionados	RF-07, RNF-03, RNF-06
Criterios de aceptación	Diferencial correcto; tiempos aceptables.

ID	TC-A-07-03
Nombre	Degradación del proveedor y resiliencia
Objetivo	Mantener operación con reintentos/backoff y notificar incidentes si hay fallos intermitentes del catálogo o storage.
Precondiciones	Simulación de respuestas 5xx/timeout del proveedor.
Datos de entrada	Job estándar de ingesta CHLA.
Componentes involucrados	Scheduler, **IngestionChlaService**, **RemoteCatalog**, **StorageService**, **AuditLog**, **Monitoring/Alerts**, **RetryPolicy**
Pasos	1. Ejecutar job con fallos intermitentes 2. Verificar aplicación de reintentos/backoff según política 3. Confirmar procesamiento parcial del set posible 4. Verificar generación de alerta/incident si error > umbral
Resultado esperado	Operación parcialmente exitosa; incidentes visibles si corresponde.
Postcondiciones	Pendientes marcados para retry; observabilidad completa.
Prioridad	Media
Tipo	Aceptación (Resiliencia/Operación)

Caso de Uso	CU-07
Requisitos relacionados	RF-07, RNF-04, RNF-01
Criterios de aceptación	SLO sostenido; alertas oportunas.

CU-08 Calcular índice PFZ

Pruebas Unitarias

ID	TC-U-08-01
Nombre	Cálculo PFZ (camino feliz)
Objetivo	Combinar SST y Chl-a con parámetros por especie y generar pfzRaster.
Precondiciones	Existen rásteres SST y CHLA para la ventana; especie soportada; RasterEngine operativo.
Datos de entrada	species="merluza", window=PT24H.
Componentes involucrados	CUT: PFZModelService — **Dobles/colaboradores:** SceneRepository (stub), SpeciesParamRepository (stub), RasterEngine (mock/spy), PFZRunRepository (mock), TileCache (mock), AuditLog (mock), Clock (stub)
Pasos	1. Invocar PFZModelService.compute(species, window) 2. Verificar SceneRepository.load("SST", window) → sstRaster 3. Verificar SceneRepository.load("CHLA", window) → chlaRaster 4. Verificar SpeciesParamRepository.getBySpecies(species) → params 5. Verificar RasterEngine.computePFZ(sstRaster, chlaRaster, params) → pfzRaster 6. Verificar PFZRunRepository.saveRun(meta, pfzRaster) → runId 7. Verificar TileCache.prewarm(runId)8. Verificar AuditLog.log('pfz_run', runId)
Resultado esperado	runId válido y pfzRaster generado.
Postcondiciones	Ejecución PFZ persistida y cache precalentado.
Prioridad	Alta
Tipo	Unitario
Caso de Uso	CU-08
Requisitos relacionados	RF-08, RF-09, RNF-01, RNF-03, RNF-06
Criterios de aceptación	Llamadas en orden; pfzRaster no nulo; prewarm ejecutado; auditoría registrada.

ID	TC-U-08-02
Nombre	Parámetros de especie inexistentes
Objetivo	Rechazar cálculo si la especie no tiene parámetros de modelo.
Precondiciones	SpeciesParamRepository no contiene la especie.
Datos de entrada	species="kraken", window=PT24H.
Componentes involucrados	CUT: PFZModelService — **Dobles/colaboradores:** SceneRepository (stub), SpeciesParamRepository (stub), AuditLog (mock)
Pasos	1. Invocar PFZModelService.compute("kraken", window) 2. Verificar SpeciesParamRepository.getBySpecies("kraken") → vacío 3. Verificar que no se invoca RasterEngine ni PFZRunRepository 4. Verificar AuditLog.log('pfz_run_failed', reason='SPECIES_UNSUPPORTED')
Resultado esperado	Error SPECIES_UNSUPPORTED.
Postcondiciones	Sin ejecución PFZ registrada; estado inalterado.
Prioridad	Alta
Tipo	Unitario (Reglas de negocio)
Caso de Uso	CU-08
Requisitos relacionados	RF-08, RNF-06
Criterios de aceptación	Mensaje claro; cero efectos colaterales.

ID	TC-U-08-02
Nombre	Parámetros de especie inexistentes
Objetivo	Rechazar cálculo si la especie no tiene parámetros de modelo.
Precondiciones	SpeciesParamRepository no contiene la especie.
Datos de entrada	species="kraken", window=PT24H.
Componentes involucrados	CUT: PFZModelService — **Dobles/colaboradores:** SceneRepository (stub), SpeciesParamRepository (stub), AuditLog (mock)

Pasos	1. Invocar PFZModelService.compute("kraken", window) 2. Verificar SpeciesParamRepository.getBySpecies("kraken") → vacío 3. Verificar que no se invoca RasterEngine ni PFZRunRepository 4. Verificar AuditLog.log('pfz_run_failed', reason='SPECIES_UNSUPPORTED')
Resultado esperado	Error SPECIES_UNSUPPORTED.
Postcondiciones	Sin ejecución PFZ registrada; estado inalterado.
Prioridad	Alta
Tipo	Unitario (Reglas de negocio)
Caso de Uso	CU-08
Requisitos relacionados	RF-08, RNF-06
Criterios de aceptación	Mensaje claro; cero efectos colaterales.

Pruebas de Integración

ID	TC-I-08-01
Nombre	Integración completa (SST+CHLA→PFZ→Run→Tiles)
Objetivo	Validar pipeline end-to-end: carga de insumos, cómputo, persistencia y precalentado de tiles.
Precondiciones	Existen escenas SST y CHLA recientes; especie soportada; componentes de persistencia y cache activos.
Datos de entrada	species="merluza", window=2025-10-24/2025-10-25.
Componentes involucrados	PFZModelService, **SceneRepository**, **SpeciesParamRepository**, **RasterEngine**, **PFZRunRepository** (DB), **TileCache**, **AuditLog**
Pasos	1. Ejecutar PFZModelService.compute(species, window) 2. Verificar lectura de rásteres SST/CHLA para la ventana 3. Verificar pfzRaster producido por RasterEngine 4. Verificar persistencia de PFZRun con metadatos (tiempos, especie, calidad) 5. Verificar TileCache.prewarm(runId) y que los endpoints de tiles responden 200
Resultado esperado	runId creado; tiles servibles; auditoría OK.
Postcondiciones	PFZRun consultable e integrado al flujo de publicación.
Prioridad	Alta
Tipo	Integración
Caso de Uso	CU-08
Requisitos relacionados	RF-08, RF-09, RNF-01, RNF-03

Criterios de aceptación	Tiempo total dentro de SLA; cobertura PFZ consistente con insumos.

ID	TC-I-08-02
Nombre	Rendimiento del modelo (p95)
Objetivo	Medir latencias del cómputo PFZ y asegurar p95 dentro de umbrales.
Precondiciones	Conjunto de pruebas con N ventanas; telemetría habilitada.
Datos de entrada	N corridas sobre ventanas recientes.
Componentes involucrados	PFZModelService, **RasterEngine**, **SceneRepository**, **PFZRunRepository**, **TileCache**, **MetricsCollector**
Pasos	1. Ejecutar N corridas sobre distintos rangos 2. Capturar métricas de tiempo del RasterEngine y del pipeline total 3. Verificar p50/p95 en límites acordados 4. Reportar throughput y tasas de error
Resultado esperado	p95 ≤ SLA; estabilidad del pipeline.
Postcondiciones	Métricas disponibles para capacity planning.
Prioridad	Media
Tipo	Integración (Rendimiento)
Caso de Uso	CU-08
Requisitos relacionados	RNF-01, RNF-03, RF-08
Criterios de aceptación	p95 dentro de umbrales; error rate < 1%.

ID	TC-I-08-03
Nombre	Integridad de valores (rango y NaN handling)
Objetivo	Validar que el pfzRaster respeta rangos [0..1] (o escala definida) y maneja nulos/máscaras.
Precondiciones	Insumos con celdas enmascaradas (nubes/glint) y valores extremos.
Datos de entrada	Ventana con mezcla de calidades.
Componentes involucrados	PFZModelService, **RasterEngine**, **SceneRepository**, **PFZRunRepository**
Pasos	1. Ejecutar compute(species, window) con insumos mixtos 2. Verificar que celdas sin datos producen NaN o nodata en PFZ 3. Verificar que el rango de PFZ se clampa/escala correctamente 4. Verificar persistencia sin corrupción de metadatos

Resultado esperado	PFZ dentro de rango; nodata consistente.
Postcondiciones	Resultado utilizable por visor/alertas.
Prioridad	Media
Tipo	Integración (Calidad de datos)
Caso de Uso	CU-08
Requisitos relacionados	RF-08, RNF-06
Criterios de aceptación	Validaciones de rango pasan; máscaras propagadas.

Pruebas de aceptación

ID	TC-A-08-01
Nombre	Generar PFZ y visualizar en visor
Objetivo	Confirmar que, tras el cálculo, el mapa PFZ es visible en el visor web.
Precondiciones	Insumos disponibles; capa PFZ registrada; visor operativo.
Datos de entrada	species="merluza", window=últimas 24h.
Componentes involucrados	PFZModelService, **PFZRunRepository**, **TileCache**, **WMTSRegistry**, **MapGateway**, **Front-end Web (Visor)**, **AuditLog**
Pasos	1. Ejecutar compute(species, window) 2. Abrir visor y habilitar capa “PFZ” 3. Verificar carga de tiles sin errores 4. Realizar zoom/pan y verificar respuesta fluida
Resultado esperado	PFZ visible y navegable en el visor.
Postcondiciones	Capa disponible para inspección y alertas.
Prioridad	Alta
Tipo	Aceptación
Caso de Uso	CU-08
Requisitos relacionados	RF-08, RF-09, RF-10, RNF-01, RNF-05
Criterios de aceptación	Latencia de tiles aceptable; sin errores visibles en UI/console.

ID	TC-A-08-02
Nombre	Reproducibilidad de corrida (misma ventana/params)
Objetivo	Garantizar que misma ventana y parámetros producen resultados equivalentes.
Precondiciones	Datos inmutables en storage; semilla/versión de modelo fija.
Datos de entrada	Dos ejecuciones con species y window idénticos.

Componentes involucrados	PFZModelService, **SceneRepository**, **RasterEngine**, **PFZRunRepository**, **TileCache**
Pasos	1. Ejecutar compute(species, window) #1 2. Ejecutar compute(species, window) #2 3. Comparar checksums/estadísticos de pfzRaster 4. Verificar equivalencia dentro de tolerancia
Resultado esperado	Resultados reproducibles (o diferencia ≤ tolerancia).
Postcondiciones	Corridas trazables con metadatos/versionado.
Prioridad	Media
Tipo	Aceptación
Caso de Uso	CU-08
Requisitos relacionados	RNF-06, RF-08
Criterios de aceptación	Hash/estadísticos coinciden; metadatos de versión presentes.

ID	TC-A-08-03
Nombre	Degradación y fallback de insumos
Objetivo	Validar comportamiento cuando falta una variable: abortar o usar ventana extendida (según política).
Precondiciones	Política configurada (requireBoth=false, extendWindow=+6h).
Datos de entrada	window con SST presente y CHLA ausente.
Componentes involucrados	PFZModelService, **SceneRepository**, **PolicyConfig**, **AuditLog**, **PFZRunRepository**
Pasos	1. Ejecutar compute(species, window) con déficit de CHLA 2. Aplicar política: extender ventana +6h para buscar CHLA 3. Si se completa, continuar; si no, registrar pfz_run_skipped 4. Verificar auditoría de la decisión
Resultado esperado	Cálculo exitoso con fallback, o “skip” auditado.
Postcondiciones	Estado consistente; trazabilidad de la política aplicada.
Prioridad	Media
Tipo	Aceptación (Operación/Política)
Caso de Uso	CU-08
Requisitos relacionados	RF-08, RNF-01, RNF-04
Criterios de aceptación	Política respetada; mensajes claros para operación.

CU-09 Publicar tiles/WMTS PFZ

Pruebas Unitarias

ID	TC-U-09-01
Nombre	Publicación exitosa (camino feliz)
Objetivo	Publicar un pfzRaster existente como capa en WMTS generando y precalentando tiles.
Precondiciones	runId válido; pfzRaster persistido en PFZRunRepository; estilo y leyenda definidos.
Datos de entrada	runId="pfz-2025-10-25-01", style="pfz_default", legend="pfz_default_legend.png".
Componentes involucrados	CUT: MapPublisher — **Dobles/colaboradores:** PFZRunRepository (mock), TileCache (mock/spy), WMTSRegistry (mock), AuditLog (mock)
Pasos	1. Invocar MapPublisher.publish(runId, style, legend) 2. Verificar PFZRunRepository.find(runId) → pfzRaste 3. Verificar TileCache.generate(pfzRaster) → ok 4. Verificar WMTSRegistry.registerLayer(runId, style, legend) → ok 5. Verificar AuditLog.log('wmts_publish', runId)
Resultado esperado	ok=true; capa registrada y tiles disponibles.
Postcondiciones	Capa PFZ publicada y visible para el visor.
Prioridad	Alta
Tipo	Unitario
Caso de Uso	CU-09
Requisitos relacionados	RF-09, RNF-01, RNF-03, RNF-06
Criterios de aceptación	Orden de invocación correcto; sin excepciones; auditoría creada.

ID	TC-U-09-02
Nombre	runId inexistente
Objetivo	Rechazar publicación cuando la corrida PFZ no existe.
Precondiciones	runId no presente en el repositorio.
Datos de entrada	runId="pfz-missing", style="pfz_default", legend="pfz_default_legend.png".
Componentes involucrados	CUT: MapPublisher — **Dobles/colaboradores:** PFZRunRepository (mock), TileCache (mock), WMTSRegistry (mock), AuditLog (mock)
Pasos	1. Invocar MapPublisher.publish(runIdInexistente, style, legend) 2. Verificar PFZRunRepository.find(runIdInexistente) → null 3. Verificar que no se llama a TileCache.generate ni a WMTSRegistry.registerLayer 4. Verificar AuditLog.log('wmts_publish_failed', runIdInexistente, reason='RUN_NOT_FOUND')
Resultado esperado	Error RUN_NOT_FOUND.
Postcondiciones	Sin cambios en cache/registro WMTS.

Prioridad	Alta
Tipo	Unitario (Errores)
Caso de Uso	CU-09
Requisitos relacionados	RF-09, RNF-06
Criterios de aceptación	Sin efectos laterales; mensaje/estado consistentes.

ID	TC-U-09-03
Nombre	Fallo al generar tiles (recuperable)
Objetivo	Manejar fallo de TileCache.generate con error controlado y auditoría.
Precondiciones	pfzRaster accesible; TileCache.generate lanza excepción transitoria.
Datos de entrada	runId válido, style/legend válidos.
Componentes involucrados	CUT: MapPublisher — **Dobles/colaboradores:** PFZRunRepository (mock), TileCache (mock con fallo), WMTSRegistry (mock), AuditLog (mock), RetryPolicy (stub)
Pasos	1. Invocar MapPublisher.publish(runId, style, legend) 2. Forzar excepción en TileCache.generate 3. Aplicar RetryPolicy (p. ej., 3 intentos) 4. Si persiste el fallo, verificar que no se llama a WMTSRegistry.registerLayer 5. Verificar AuditLog.log('wmts_publish_failed', runId, reason='TILE_GENERATION_ERROR')
Resultado esperado	Publicación abortada con error controlado; sin registro WMTS.
Postcondiciones	Corrida PFZ permanece sin capa publicada; incidente auditado.
Prioridad	Media
Tipo	Unitario (Resiliencia)
Caso de Uso	CU-09
Requisitos relacionados	RF-09, RNF-04, RNF-06
Criterios de aceptación	Retries aplicados; no queda capa inconsistente en WMTS.

Pruebas de Integración

ID	TC-I-09-01
Nombre	Pipeline publicación: Run→Tiles→WMTS
Objetivo	Validar la integración completa desde el runId hasta la capa registrada en WMTS.
Precondiciones	PFZRunRepository con pfzRaster; TileCache y WMTSRegistry operativos.

Datos de entrada	runId válido, style="pfz_default", legend="pfz_default_legend.png".
Componentes involucrados	MapPublisher, **PFZRunRepository** (DB), **TileCache**, **WMTSRegistry**, **AuditLog**
Pasos	1. Invocar publicación para runId 2. Validar generación de tiles (chequear directorio/cache y conteo por nivel z) 3. Registrar capa en WMTS con metadatos (nombre, estilo, leyenda, CRS, bounding box) 4. Verificar AuditLog con wmts_publish y correlación de request
Resultado esperado	Capa PFZ registrada y tiles generados.
Postcondiciones	Capa resolvible por WMTSRegistry.resolve("PFZ").
Prioridad	Alta
Tipo	Integración
Caso de Uso	CU-09
Requisitos relacionados	RF-09, RNF-01, RNF-03
Criterios de aceptación	Tiles completos por niveles configurados; WMTS responde 200 a GetCapabilities.

Campo	**Valor**
ID	TC-I-09-02
Nombre	Validación de estilo y leyenda
Objetivo	Asegurar que la capa se registra con estilo/leyenda válidos y accesibles.
Precondiciones	Repositorio de estilos y assets accesible; style y legend existentes.
Datos de entrada	style="pfz_blue_red", legend="pfz_blue_red_legend.png".
Componentes involucrados	MapPublisher, **WMTSRegistry**, **StyleRepository/AssetStore** (si aplica), **AuditLog**
Pasos	1. Publicar con style/legend específicos 2. Verificar que WMTSRegistry.registerLayer valida y enlaza assets 3. Probar acceso HTTP a la leyenda (URL pública) y consistencia con el estilo 4. Verificar GetMap de un tile de muestra con render esperado
Resultado esperado	Capa con estilo correcto y leyenda accesible.
Postcondiciones	Estilo y leyenda visibles en el visor/clients.
Prioridad	Media
Tipo	Integración (Presentación)
Caso de Uso	CU-09
Requisitos relacionados	RF-09, RF-10, RNF-05
Criterios de aceptación	Leyenda retorna 200; render coincide con rampa definida.

ID	TC-I-09-03
Nombre	Re-publicación (replace/overwrite) de una corrida
Objetivo	Soportar re-publicar una corrida (p. ej., cambio de estilo) sin dejar residuos.
Precondiciones	Corrida previamente publicada; política replace=true.
Datos de entrada	runId existente, style="pfz_contrast".
Componentes involucrados	MapPublisher, **PFZRunRepository**, **TileCache**, **WMTSRegistry**, **AuditLog**
Pasos	1. Invocar publicación con replace=true (config/política) 2. Verificar invalidación de tiles antiguos (cache bust/invalidate) 3. Registrar capa con el nuevo estilo en WMTS (actualizar metadatos) 4. Verificar auditoría wmts_republish
Resultado esperado	Capa actualizada sin inconsistencias; tiles coherentes.
Postcondiciones	Visor consume la versión nueva; sin artefactos obsoletos.
Prioridad	Media
Tipo	Integración
Caso de Uso	CU-09
Requisitos relacionados	RF-09, RNF-03, RNF-01
Criterios de aceptación	Cache invalidado; GetCapabilities refleja nuevo estilo.

Pruebas de aceptación

ID	TC-A-09-01
Nombre	Ver PFZ publicado en el visor
Objetivo	Confirmar que la capa publicada es visible y operable en el visor web.
Precondiciones	Capa registrada en WMTS; MapGateway y front-end operativos.
Datos de entrada	layer="PFZ" (último runId publicado).
Componentes involucrados	Front-end Web (Visor), **MapGateway**, **WMTSRegistry**, **TileCache**
Pasos	1. Abrir visor y habilitar capa “PFZ” 2. Verificar que WMTSRegistry.resolve("PFZ") entrega la URL 3. Hacer zoom/pan y confirmar carga fluida de tiles 4. Comprobar coherencia visual con la leyenda
Resultado esperado	Capa PFZ visible y navegable; leyenda correcta.

Postcondiciones	Experiencia de usuario consistente.
Prioridad	Alta
Tipo	Aceptación
Caso de Uso	CU-09
Requisitos relacionados	RF-09, RF-10, RNF-05, RNF-01
Criterios de aceptación	p95 de carga de tile dentro de SLA; sin errores en consola.

ID	TC-A-09-02
Nombre	GetCapabilities y metadatos de capa
Objetivo	Verificar que la capa aparece en GetCapabilities con metadatos completos (estilo, CRS, bbox, tiempo).
Precondiciones	WMTS operativo y registrado.
Datos de entrada	Solicitudes GetCapabilities y GetTile.
Componentes involucrados	WMTSRegistry, **MapGateway**, **Front-end/Tester**
Pasos	1. Realizar GetCapabilities y localizar la capa PFZ 2. Verificar presencia de estilo, leyenda, CRS y TimeDimension (si aplica) 3. Solicitar GetTile para un z/x/y de prueba 4. Validar consistencia entre capabilities y tile servido
Resultado esperado	Metadatos completos y tiles consistentes.
Postcondiciones	Interoperabilidad garantizada con clientes WMTS.
Prioridad	Media
Tipo	Aceptación
Caso de Uso	CU-09
Requisitos relacionados	RF-09, RNF-05
Criterios de aceptación	Validación OK con clientes externos (qGIS/leaflet).

ID	TC-A-09-03
Nombre	Manejo de errores visible al usuario
Objetivo	Mostrar mensajes claros cuando la publicación falla y prevenir estados inconsistentes en UI.
Precondiciones	Simulación de fallo en TileCache o WMTSRegistry.
Datos de entrada	Acción “Publicar PFZ” en UI de operación.
Componentes involucrados	Front-end Operación, **MapPublisher API**, **AuditLog**, **Monitoring/Alerts**

Pasos	1. Disparar publicación desde UI 2. Simular fallo en generación de tiles o registro WMTS 3. Verificar mensaje claro en UI y trazabilidad del error 4. Confirmar que no queda capa “a medias” en el visor
Resultado esperado	Feedback claro; sin residuos de capa fallida.
Postcondiciones	Incidente registrado y notificado; posibilidad de reintento.
Prioridad	Media
Tipo	Aceptación
Caso de Uso	CU-09
Requisitos relacionados	RF-09, RNF-05, RNF-04
Criterios de aceptación	UI no deja botones en estado indeterminado; logs/alertas disponibles.

CU-10 Visualizar mapa PFZ

Pruebas Unitarias

ID	TC-U-10-01
Nombre	Resolver capa PFZ (camino feliz)
Objetivo	Devolver la URL WMTS de la capa “PFZ” registrada.
Precondiciones	WMTSRegistry contiene entrada para “PFZ”.
Datos de entrada	layerName="PFZ".
Componentes involucrados	CUT: MapGateway — **Dobles/colaboradores:** WMTSRegistry (mock)
Pasos	1. Invocar MapGateway.resolve("PFZ") 2. Verificar WMTSRegistry.resolve("PFZ") 3. Validar que MapGateway retorna wmtsURL no vacío
Resultado esperado	URL WMTS válida.
Postcondiciones	Ninguna.
Prioridad	Alta
Tipo	Unitario
Caso de Uso	CU-10
Requisitos relacionados	RF-10, RNF-06
Criterios de aceptación	Un único llamado a WMTSRegistry.resolve; sin excepciones.

ID	TC-U-10-02
Nombre	Capa inexistente

Objetivo	Retornar error controlado cuando la capa no está registrada.
Precondiciones	“PFZ” no existe en WMTSRegistry.
Datos de entrada	layerName="PFZ".
Componentes involucrados	CUT: MapGateway — **Dobles/colaboradores:** WMTSRegistry (mock)
Pasos	1. Invocar MapGateway.resolve("PFZ") 2. Forzar WMTSRegistry.resolve("PFZ") → null/not found 3. Verificar que MapGateway devuelve error LAYER_NOT_FOUND
Resultado esperado	Error LAYER_NOT_FOUND.
Postcondiciones	Ninguna.
Prioridad	Media
Tipo	Unitario (Errores)
Caso de Uso	CU-10
Requisitos relacionados	RF-10, RNF-06
Criterios de aceptación	Mensaje consistente; sin NPE/throw no controlado.

ID	TC-U-10-03
Nombre	Construcción de path de tiles (cliente)
Objetivo	Generar paths {z}/{x}/{y} correctos para solicitudes de tile PFZ.
Precondiciones	Esquema de rutas activo en TileCache.
Datos de entrada	z=6, x=20, y=35.
Componentes involucrados	CUT: UserUI (helper de mapa) — **Dobles/colaboradores:** ninguno (función pura)
Pasos	1. Invocar UserUI.buildTilePath("PFZ", z, x, y) 2. Verificar resultado /tiles/PFZ/6/20/35 3. Validar que no agrega querystring cuando no hay parámetros
Resultado esperado	Path correcto y estable.
Postcondiciones	Ninguna.
Prioridad	Media
Tipo	Unitario
Caso de Uso	CU-10
Requisitos relacionados	RF-10, RNF-05
Criterios de aceptación	Salida determinística y sin espacios/caracteres inválidos.

Pruebas de Integración

ID	TC-I-10-01

Nombre	Visor: habilitar capa PFZ y cargar primer tile
Objetivo	Validar flujo UserUI → MapGateway → WMTSRegistry → TileCache para un tile.
Precondiciones	Capa “PFZ” publicada y registrada; TileCache operativo.
Datos de entrada	layer="PFZ", z=5, x=10, y=12.
Componentes involucrados	UserUI, **MapGateway**, **WMTSRegistry**, **TileCache**
Pasos	1. UserUI.enableLayer("PFZ") 2. MapGateway.resolve("PFZ") → wmtsURL 3. UserUI solicita GET /tiles/PFZ/5/10/12 4. TileCache responde 200 con imagen de tile
Resultado esperado	Tile recibido (HTTP 200, imagen válida).
Postcondiciones	Capa visible en visor.
Prioridad	Alta
Tipo	Integración
Caso de Uso	CU-10
Requisitos relacionados	RF-10, RNF-01, RNF-05
Criterios de aceptación	Latencia dentro de SLA; sin errores en logs.

ID	TC-I-10-02
Nombre	Cache hit/miss y precalentado
Objetivo	Verificar comportamiento de TileCache ante primer acceso (miss) y siguientes (hit).
Precondiciones	TileCache con métricas habilitadas; PFZ recién publicado.
Datos de entrada	Secuencia de requests a mismo tile y a tiles vecinos.
Componentes involucrados	UserUI, **TileCache**, **WMTSRegistry/MapGateway**
Pasos	1. Solicitar tile PFZ (z/x/y) por primera vez → miss 2. Repetir solicitud → hit 3. Solicitar tiles vecinos (mosaico) para probar prewarm/batch 4. Verificar métricas de hit ratio y tiempos
Resultado esperado	Primer acceso miss; siguientes hit; latencias mejoran.
Postcondiciones	Cache caliente para el viewport.
Prioridad	Media
Tipo	Integración (Rendimiento)
Caso de Uso	CU-10
Requisitos relacionados	RNF-01, RF-10
Criterios de aceptación	Hit ratio esperado; p95 tras calentamiento dentro de SLA.

ID	TC-I-10-03

Nombre	Manejo de tile inexistente/CRS
Objetivo	Asegurar respuesta controlada si se solicita un z/x/y fuera de cobertura o CRS erróneo.
Precondiciones	Capa “PFZ” válida; validaciones de TileCache activas.
Datos de entrada	z=22 (por encima del máximo) o CRS inesperado.
Componentes involucrados	UserUI, **TileCache**, **MapGateway**
Pasos	1. Solicitar GET /tiles/PFZ/22/… (nivel no soportado) 2. Verificar respuesta 404/400 controlada 3. Registrar evento de validación en logs/metrics
Resultado esperado	Error controlado; no hay caída del servicio.
Postcondiciones	Estado del cache inalterado.
Prioridad	Media
Tipo	Integración (Robustez)
Caso de Uso	CU-10
Requisitos relacionados	RNF-04, RNF-06
Criterios de aceptación	Mensaje claro; no se generan tiles corruptos.

Pruebas de aceptación

ID	TC-A-10-01
Nombre	Habilitar capa PFZ desde UI (UX)
Objetivo	Confirmar que el usuario puede activar la capa y visualizarla sin fricción.
Precondiciones	UI operativa; capa PFZ publicada; conectividad estable.
Datos de entrada	Acción “Activar PFZ” en el visor.
Componentes involucrados	Front-end Web (Visor), **MapGateway**, **WMTSRegistry**, **TileCache**
Pasos	1. Abrir visor y activar “PFZ” 2. Verificar aparición de la capa sobre el mapa base 3. Pan y zoom moderados para provocar carga de nuevos tiles 4. Observar legend/estilo si está disponible
Resultado esperado	Capa visible y fluida en navegación.
Postcondiciones	Visor queda con capa activa.
Prioridad	Alta
Tipo	Aceptación
Caso de Uso	CU-10
Requisitos relacionados	RF-10, RNF-05, RNF-01
Criterios de aceptación	Sin errores visibles; FPS/latencia aceptables.

ID	TC-A-10-02

Nombre	Conectividad degradada (UX)
Objetivo	Mostrar feedback claro ante timeouts/lentitud y permitir reintento.
Precondiciones	Simulación de red lenta o intermitente.
Datos de entrada	Activación de capa y navegación.
Componentes involucrados	Front-end Web (Visor), **TileCache**, **MapGateway**
Pasos	1. Activar “PFZ” con red degradada 2. Simular timeout en carga de algunos tiles 3. Verificar placeholders/reintentos controlados en UI 4. Confirmar que la UI permite reintentar/recargar
Resultado esperado	UX robusta; no bloquea la sesión; reintentos visibles.
Postcondiciones	Estado del visor estable; sin bloqueos.
Prioridad	Media
Tipo	Aceptación (Resiliencia UX)
Caso de Uso	CU-10
Requisitos relacionados	RNF-05, RNF-04
Criterios de aceptación	Mensajes claros; no cuelgues; navegación posible.

ID	TC-A-10-03
Nombre	Coherencia visual con leyenda
Objetivo	Validar que los colores/valores del mapa coinciden con la leyenda de la capa PFZ.
Precondiciones	Leyenda publicada; estilo estático conocido.
Datos de entrada	Vista de un área con gradientes PFZ.
Componentes involucrados	Front-end Web (Visor), **TileCache**, **WMTSRegistry/MapGateway**
Pasos	1. Activar “PFZ” y abrir panel de leyenda 2. Inspeccionar gradientes/umbrales visibles en el mapa 3. Comparar con ticks/colores de la leyenda 4. Validar correspondencia visual y textual
Resultado esperado	Mapa y leyenda consistentes.
Postcondiciones	Experiencia interpretativa correcta para el usuario.
Prioridad	Media
Tipo	Aceptación (Presentación)
Caso de Uso	CU-10
Requisitos relacionados	RF-10, RNF-05
Criterios de aceptación	Sin desalineaciones notorias; leyenda accesible.

CU-11 Visualizar mapa SST

Pruebas Unitarias

ID	TC-U-11-01
Nombre	Resolver capa SST (camino feliz)
Objetivo	Devolver la URL WMTS de la capa “SST” registrada.
Precondiciones	WMTSRegistry contiene entrada para “SST”.
Datos de entrada	layerName="SST".
Componentes involucrados	CUT: MapGateway — **Dobles/colaboradores:** WMTSRegistry (mock)
Pasos	1. Invocar MapGateway.resolve("SST") 2. Verificar WMTSRegistry.resolve("SST") 3. Validar que MapGateway retorna wmtsURL no vacío
Resultado esperado	URL WMTS válida.
Postcondiciones	Ninguna.
Prioridad	Alta
Tipo	Unitario
Caso de Uso	CU-11
Requisitos relacionados	RF-11, RNF-06
Criterios de aceptación	Una sola invocación a WMTSRegistry.resolve; sin excepciones.

ID	TC-U-11-02
Nombre	Capa SST inexistente
Objetivo	Retornar error controlado cuando la capa no está registrada.
Precondiciones	“SST” no existe en WMTSRegistry.
Datos de entrada	layerName="SST".
Componentes involucrados	CUT: MapGateway — **Dobles/colaboradores:** WMTSRegistry (mock)
Pasos	1. Invocar MapGateway.resolve("SST") 2. Forzar WMTSRegistry.resolve("SST") → null/not found 3. Verificar que MapGateway devuelve error LAYER_NOT_FOUND
Resultado esperado	Error LAYER_NOT_FOUND.
Postcondiciones	Ninguna.
Prioridad	Media
Tipo	Unitario (Errores)
Caso de Uso	CU-11
Requisitos relacionados	RF-11, RNF-06
Criterios de aceptación	Mensaje consistente; sin NPE ni throws no controlados.

ID	TC-U-11-03
Nombre	Construcción de path de tiles SST (cliente)
Objetivo	Generar paths {z}/{x}/{y} correctos para solicitudes de tile SST.
Precondiciones	Esquema de rutas activo en TileCache.
Datos de entrada	z=6, x=18, y=40.
Componentes involucrados	CUT: UserUI (helper de mapa) — **Dobles/colaboradores:** ninguno (función pura)
Pasos	1. Invocar UserUI.buildTilePath("SST", z, x, y) 2. Verificar resultado /tiles/SST/6/18/40 3. Validar que no agrega querystring cuando no hay parámetros
Resultado esperado	Path correcto y estable.
Postcondiciones	Ninguna.
Prioridad	Media
Tipo	Unitario
Caso de Uso	CU-11
Requisitos relacionados	RF-11, RNF-05
Criterios de aceptación	Salida determinística y sin caracteres inválidos.

Pruebas de Integración

ID	TC-I-11-01
Nombre	Visor: habilitar capa SST y cargar primer tile
Objetivo	Validar flujo UserUI → MapGateway → WMTSRegistry → TileCache para un tile SST.
Precondiciones	Capa “SST” publicada y registrada; TileCache operativo.
Datos de entrada	layer="SST", z=5, x=9, y=11.
Componentes involucrados	UserUI, **MapGateway**, **WMTSRegistry**, **TileCache**
Pasos	1. UserUI.enableLayer("SST") 2. MapGateway.resolve("SST") → wmtsURL 3. UserUI solicita GET /tiles/SST/5/9/11 4. TileCache responde 200 con imagen de tile
Resultado esperado	Tile recibido (HTTP 200, imagen válida).
Postcondiciones	Capa visible en visor.
Prioridad	Alta
Tipo	Integración
Caso de Uso	CU-11
Requisitos relacionados	RF-11, RNF-01, RNF-05
Criterios de aceptación	Latencia dentro de SLA; sin errores en logs.

ID	TC-I-11-02
Nombre	Cache hit/miss y precalentado (SST)
Objetivo	Verificar comportamiento de TileCache ante primer acceso (miss) y siguientes (hit) con SST.
Precondiciones	TileCache con métricas habilitadas; SST recién publicado.
Datos de entrada	Secuencia de requests a mismo tile y a tiles vecinos.
Componentes involucrados	UserUI, **TileCache**, **WMTSRegistry/MapGateway**
Pasos	1. Solicitar tile SST (z/x/y) por primera vez → miss 2. Repetir solicitud → hit 3. Solicitar tiles vecinos (mosaico) para probar prewarm/batch 4. Verificar métricas de hit ratio y tiempos
Resultado esperado	Primer acceso miss; siguientes hit; latencias mejoran.
Postcondiciones	Cache caliente para el viewport.
Prioridad	Media
Tipo	Integración (Rendimiento)
Caso de Uso	CU-11
Requisitos relacionados	RNF-01, RF-11
Criterios de aceptación	Hit ratio esperado; p95 tras calentamiento dentro de SLA.

ID	TC-I-11-03
Nombre	Manejo de tile inexistente/CRS (SST)
Objetivo	Asegurar respuesta controlada si se solicita un z/x/y fuera de cobertura o CRS erróneo para SST.
Precondiciones	Capa “SST” válida; validaciones de TileCache activas.
Datos de entrada	z=22 (por encima del máximo) o CRS inesperado.
Componentes involucrados	UserUI, **TileCache**, **MapGateway**
Pasos	1. Solicitar GET /tiles/SST/22/… (nivel no soportado) 2. Verificar respuesta 404/400 controlada 3. Registrar evento de validación en logs/metrics
Resultado esperado	Error controlado; no hay caída del servicio.
Postcondiciones	Estado del cache inalterado.
Prioridad	Media
Tipo	Integración (Robustez)
Caso de Uso	CU-11
Requisitos relacionados	RNF-04, RNF-06
Criterios de aceptación	Mensaje claro; no se generan tiles corruptos.

Pruebas de aceptación

ID	TC-A-11-01
Nombre	Habilitar capa SST desde UI (UX)
Objetivo	Confirmar que el usuario puede activar la capa SST y visualizarla sin fricción.
Precondiciones	UI operativa; capa SST publicada; conectividad estable.
Datos de entrada	Acción “Activar SST” en el visor.
Componentes involucrados	Front-end Web (Visor), **MapGateway**, **WMTSRegistry**, **TileCache**
Pasos	1. Abrir visor y activar “SST” 2. Verificar aparición de la capa sobre el mapa base 3. Pan y zoom moderados para provocar carga de nuevos tiles 4. Observar legend/estilo si está disponible
Resultado esperado	Capa visible y fluida en navegación.
Postcondiciones	Visor queda con capa SST activa.
Prioridad	Alta
Tipo	Aceptación
Caso de Uso	CU-11
Requisitos relacionados	RF-11, RNF-05, RNF-01
Criterios de aceptación	Sin errores visibles; FPS/latencia aceptables.

ID	TC-A-11-02
Nombre	Conectividad degradada (UX SST)
Objetivo	Mostrar feedback claro ante timeouts/lentitud y permitir reintento para SST.
Precondiciones	Simulación de red lenta o intermitente.
Datos de entrada	Activación de capa y navegación.
Componentes involucrados	Front-end Web (Visor), **TileCache**, **MapGateway**
Pasos	1. Activar “SST” con red degradada 2. Simular timeout en carga de algunos tiles 3. Verificar placeholders/reintentos controlados en UI 4. Confirmar que la UI permite reintentar/recargar
Resultado esperado	UX robusta; no bloquea la sesión; reintentos visibles.
Postcondiciones	Estado del visor estable; sin bloqueos.
Prioridad	Media
Tipo	Aceptación (Resiliencia UX)
Caso de Uso	CU-11
Requisitos relacionados	RNF-05, RNF-04
Criterios de aceptación	Mensajes claros; no cuelgues; navegación posible.

ID	TC-A-11-03
Nombre	Coherencia visual con leyenda (SST)
Objetivo	Validar que los colores/valores del mapa SST coinciden con la leyenda configurada.
Precondiciones	Leyenda publicada; estilo estático conocido para SST.
Datos de entrada	Vista de un área con gradientes térmicos visibles.
Componentes involucrados	Front-end Web (Visor), **TileCache**, **WMTSRegistry/MapGateway**
Pasos	1. Activar “SST” y abrir panel de leyenda 2. Inspeccionar gradientes de temperatura en el mapa 3. Comparar con ticks/colores de la leyenda 4. Validar correspondencia visual y textual
Resultado esperado	Mapa y leyenda consistentes.
Postcondiciones	Interpretación correcta por el usuario.
Prioridad	Media
Tipo	Aceptación (Presentación)
Caso de Uso	CU-11
Requisitos relacionados	RF-11, RNF-05
Criterios de aceptación	Sin desalineaciones; leyenda accesible.

CU-12 Visualizar mapa Chl-A

Pruebas Unitarias

ID	TC-U-12-01
Nombre	Resolver capa CHLA (camino feliz)
Objetivo	Devolver la URL WMTS de la capa “CHLA” registrada.
Precondiciones	WMTSRegistry contiene entrada para “CHLA”.
Datos de entrada	layerName="CHLA".
Componentes involucrados	CUT: MapGateway — **Dobles/colaboradores:** WMTSRegistry (mock)
Pasos	1. Invocar MapGateway.resolve("CHLA") 2. Verificar WMTSRegistry.resolve("CHLA") 3. Validar que MapGateway retorna wmtsURL no vacío
Resultado esperado	URL WMTS válida.
Postcondiciones	Ninguna.
Prioridad	Alta
Tipo	Unitario

Caso de Uso	CU-12
Requisitos relacionados	RF-12, RNF-06
Criterios de aceptación	Una sola invocación; sin excepciones.

Campo	**Valor**
ID	TC-U-12-02
Nombre	Capa CHLA inexistente
Objetivo	Retornar error controlado cuando la capa no está registrada.
Precondiciones	“CHLA” no existe en WMTSRegistry.
Datos de entrada	layerName="CHLA".
Componentes involucrados	CUT: MapGateway — **Dobles/colaboradores:** WMTSRegistry (mock)
Pasos	1. Invocar MapGateway.resolve("CHLA") 2. Forzar WMTSRegistry.resolve("CHLA") → null/not found 3. Verificar que MapGateway devuelve error LAYER_NOT_FOUND
Resultado esperado	Error LAYER_NOT_FOUND.
Postcondiciones	Ninguna.
Prioridad	Media
Tipo	Unitario (Errores)
Caso de Uso	CU-12
Requisitos relacionados	RF-12, RNF-06
Criterios de aceptación	Mensaje consistente; sin NPE/throws no controlados.

ID	TC-U-12-03
Nombre	Construcción de path de tiles CHLA (cliente)
Objetivo	Generar paths {z}/{x}/{y} correctos para solicitudes de tile CHLA.
Precondiciones	Esquema de rutas activo en TileCache.
Datos de entrada	z=6, x=22, y=37.
Componentes involucrados	CUT: UserUI (helper de mapa) — **Dobles/colaboradores:** ninguno (función pura)
Pasos	1. Invocar UserUI.buildTilePath("CHLA", z, x, y) 2. Verificar resultado /tiles/CHLA/6/22/37 3. Validar que no agrega querystring cuando no hay parámetros
Resultado esperado	Path correcto y estable.
Postcondiciones	Ninguna.
Prioridad	Media
Tipo	Unitario
Caso de Uso	CU-12
Requisitos relacionados	RF-12, RNF-05

Criterios de aceptación	Salida determinística y sin caracteres inválidos.

Pruebas de Integración

ID	TC-I-12-01
Nombre	Visor: habilitar capa CHLA y cargar primer tile
Objetivo	Validar flujo UserUI → MapGateway → WMTSRegistry → TileCache para un tile CHLA.
Precondiciones	Capa “CHLA” publicada y registrada; TileCache operativo.
Datos de entrada	layer="CHLA", z=5, x=11, y=14.
Componentes involucrados	UserUI, **MapGateway**, **WMTSRegistry**, **TileCache**
Pasos	1. UserUI.enableLayer("CHLA") 2. MapGateway.resolve("CHLA") → wmtsURL 3. UserUI solicita GET /tiles/CHLA/5/11/14 4. TileCache responde 200 con imagen de tile
Resultado esperado	Tile recibido (HTTP 200, imagen válida).
Postcondiciones	Capa visible en visor.
Prioridad	Alta
Tipo	Integración
Caso de Uso	CU-12
Requisitos relacionados	RF-12, RNF-01, RNF-05
Criterios de aceptación	Latencia dentro de SLA; sin errores en logs.

ID	TC-I-12-02
Nombre	Cache hit/miss y precalentado (CHLA)
Objetivo	Verificar comportamiento de TileCache ante primer acceso (miss) y siguientes (hit) con CHLA.
Precondiciones	TileCache con métricas habilitadas; CHLA recién publicado.
Datos de entrada	Secuencia de requests a mismo tile y a tiles vecinos.
Componentes involucrados	UserUI, **TileCache**, **WMTSRegistry/MapGateway**
Pasos	1. Solicitar tile CHLA (z/x/y) por primera vez → miss 2. Repetir solicitud → hit 3. Solicitar tiles vecinos (mosaico) para probar prewarm/batch 4. Verificar métricas de hit ratio y tiempos
Resultado esperado	Primer acceso miss; siguientes hit; latencias mejoran.
Postcondiciones	Cache caliente para el viewport.
Prioridad	Media
Tipo	Integración (Rendimiento)
Caso de Uso	CU-12

Requisitos relacionados	RNF-01, RF-12
Criterios de aceptación	Hit ratio esperado; p95 tras calentamiento dentro de SLA.

ID	TC-I-12-03
Nombre	Manejo de tile inexistente/CRS (CHLA)
Objetivo	Asegurar respuesta controlada si se solicita un z/x/y fuera de cobertura o CRS erróneo para CHLA.
Precondiciones	Capa “CHLA” válida; validaciones de TileCache activas.
Datos de entrada	z=22 (por encima del máximo) o CRS inesperado.
Componentes involucrados	UserUI, **TileCache**, **MapGateway**
Pasos	1. Solicitar GET /tiles/CHLA/22/… (nivel no soportado) 2. Verificar respuesta 404/400 controlada 3. Registrar evento de validación en logs/metrics
Resultado esperado	Error controlado; no hay caída del servicio.
Postcondiciones	Estado del cache inalterado.
Prioridad	Media
Tipo	Integración (Robustez)
Caso de Uso	CU-12
Requisitos relacionados	RNF-04, RNF-06
Criterios de aceptación	Mensaje claro; no se generan tiles corruptos.

Pruebas de aceptación

ID	TC-A-12-01
Nombre	Habilitar capa CHLA desde UI (UX)
Objetivo	Confirmar que el usuario puede activar la capa CHLA y visualizarla sin fricción.
Precondiciones	UI operativa; capa CHLA publicada; conectividad estable.
Datos de entrada	Acción “Activar CHLA” en el visor.
Componentes involucrados	Front-end Web (Visor), **MapGateway**, **WMTSRegistry**, **TileCache**
Pasos	1. Abrir visor y activar “CHLA” 2. Verificar aparición de la capa sobre el mapa base 3. Pan y zoom moderados para provocar carga de nuevos tiles 4. Observar legend/estilo si está disponible
Resultado esperado	Capa visible y fluida en navegación.
Postcondiciones	Visor queda con capa CHLA activa.
Prioridad	Alta
Tipo	Aceptación

Caso de Uso	CU-12
Requisitos relacionados	RF-12, RNF-05, RNF-01
Criterios de aceptación	Sin errores visibles; FPS/latencia aceptables.

ID	TC-A-12-02
Nombre	Conectividad degradada (UX CHLA)
Objetivo	Mostrar feedback claro ante timeouts/lentitud y permitir reintento para CHLA.
Precondiciones	Simulación de red lenta o intermitente.
Datos de entrada	Activación de capa y navegación.
Componentes involucrados	Front-end Web (Visor), **TileCache**, **MapGateway**
Pasos	1. Activar “CHLA” con red degradada 2. Simular timeout en carga de algunos tiles 3. Verificar placeholders/reintentos controlados en UI 4. Confirmar que la UI permite reintentar/recargar
Resultado esperado	UX robusta; no bloquea la sesión; reintentos visibles.
Postcondiciones	Estado del visor estable; sin bloqueos.
Prioridad	Media
Tipo	Aceptación (Resiliencia UX)
Caso de Uso	CU-12
Requisitos relacionados	RNF-05, RNF-04
Criterios de aceptación	Mensajes claros; no cuelgues; navegación posible.

ID	TC-A-12-03
Nombre	Coherencia visual con leyenda (CHLA)
Objetivo	Validar que los colores/valores del mapa CHLA coinciden con la leyenda configurada.
Precondiciones	Leyenda publicada; estilo estático conocido para CHLA.
Datos de entrada	Vista de un área con gradientes de concentración visibles.
Componentes involucrados	Front-end Web (Visor), **TileCache**, **WMTSRegistry/MapGateway**
Pasos	1. Activar “CHLA” y abrir panel de leyenda 2. Inspeccionar gradientes/umbrales visibles en el mapa 3. Comparar con ticks/colores de la leyenda 4. Validar correspondencia visual y textual
Resultado esperado	Mapa y leyenda consistentes.
Postcondiciones	Interpretación correcta por el usuario.
Prioridad	Media
Tipo	Aceptación (Presentación)

Caso de Uso	CU-12
Requisitos relacionados	RF-12, RNF-05
Criterios de aceptación	Sin desalineaciones; leyenda accesible.

CU-13 Consultar Celda

Pruebas Unitarias

ID	TC-U-13-01
Nombre	Inspección exitosa (camino feliz)
Objetivo	Devolver valores PFZ, SST y CHLA y metadatos para una celda válida (lat, lon, date).
Precondiciones	Existe PFZRun para la date; repositorios responden en rango; proyección y resolución válidas.
Datos de entrada	lat=-45.20, lon=-62.75, date=2025-10-24.
Componentes involucrados	CUT: IdentifyService — **Dobles/colaboradores:** PFZRunRepository (mock), SceneRepository (mock)
Pasos	1. Invocar IdentifyService.identify(lat, lon, date) 2. Verificar PFZRunRepository.getPFZ(date, lat, lon) → pfzValue, meta 3. Verificar SceneRepository.getSST(date, lat, lon) → sst 4. Verificar SceneRepository.getCHLA(date, lat, lon) → chla 5. Construir payload {pfz, sst, chla, meta} y retornar
Resultado esperado	Respuesta con {pfz, sst, chla, meta} completa.
Postcondiciones	Ninguna (consulta pura).
Prioridad	Alta
Tipo	Unitario
Caso de Uso	CU-13
Requisitos relacionados	RF-13, RNF-01, RNF-06
Criterios de aceptación	Todos los repos invocados una sola vez; types y formato correctos.

ID	TC-U-13-02
Nombre	Fecha sin corrida PFZ
Objetivo	Devolver error controlado cuando no hay PFZRun para la fecha solicitada.
Precondiciones	No existe corrida PFZ válida para date.
Datos de entrada	lat=-44.00, lon=-60.00, date=2025-10-10.

Componentes involucrados	CUT: IdentifyService — **Dobles/colaboradores:** PFZRunRepository (mock)
Pasos	1. Invocar IdentifyService.identify(lat, lon, date) 2. PFZRunRepository.getPFZ(...) → null/NOT_FOUND 3. No consultar SST/CHA 4. Retornar error PFZ_NOT_AVAILABLE con recomendación (e.g., “Seleccione otra fecha”)
Resultado esperado	Error PFZ_NOT_AVAILABLE.
Postcondiciones	Ninguna.
Prioridad	Media
Tipo	Unitario (Errores)
Caso de Uso	CU-13
Requisitos relacionados	RF-13, RNF-06
Criterios de aceptación	No hay llamadas innecesarias a otros repos; mensaje claro.

ID	TC-U-13-03
Nombre	Coordenadas fuera de cobertura
Objetivo	Responder controladamente cuando el punto solicitado está fuera del bbox del ráster o en nodata.
Precondiciones	lat, lon fuera de cobertura de PFZ/SST/CHLA para date.
Datos de entrada	lat=-30.00, lon=-40.00, date=2025-10-24.
Componentes involucrados	CUT: IdentifyService — **Dobles/colaboradores:** PFZRunRepository (mock), SceneRepository (mock)
Pasos	1. Invocar IdentifyService.identify(lat, lon, date) 2. PFZRunRepository.getPFZ(...) → nodata/out_of_bounds 3. SceneRepository.getSST(...) → nodata/out_of_bounds 4. SceneRepository.getCHLA(...) → nodata/out_of_bounds 5. Retornar OUT_OF_COVERAGE con metadata de cobertura
Resultado esperado	Error OUT_OF_COVERAGE o payload con null/nodata consistente.
Postcondiciones	Ninguna.
Prioridad	Media
Tipo	Unitario (Bordes de dominio)
Caso de Uso	CU-13
Requisitos relacionados	RF-13, RNF-06
Criterios de aceptación	Señalización de nodata clara y uniforme.

Pruebas de Integración

ID	TC-I-13-01
Nombre	Flujo completo identify (PFZ+SST+CHLA)
Objetivo	Validar la interacción IdentifyService → PFZRunRepository/SceneRepository y el payload final.
Precondiciones	PFZRun vigente; SST/CHLA disponibles para la date.
Datos de entrada	lat=-45.00, lon=-62.00, date=2025-10-24.
Componentes involucrados	IdentifyService, **PFZRunRepository** (DB), **SceneRepository** (DB/Store), **Map/JSON Serializer**
Pasos	1. Ejecutar IdentifyService.identify(lat, lon, date) 2. Leer PFZ en coordenadas → valor numérico + metadata (runId, time, quality) 3. Leer SST y CHLA para el mismo punto y fecha 4. Validar payload {pfz, sst, chla, meta} con esquema
Resultado esperado	Payload completo y consistente.
Postcondiciones	Ninguna.
Prioridad	Alta
Tipo	Integración
Caso de Uso	CU-13
Requisitos relacionados	RF-13, RNF-01, RNF-06
Criterios de aceptación	Tiempo total dentro de SLA; tipos correctos; sin campos faltantes.

ID	TC-I-13-02
Nombre	Rendimiento p95 del identify
Objetivo	Medir latencias en N solicitudes concurrentes y asegurar p95 dentro de umbral.
Precondiciones	Telemetría activa; datos en cache/caliente.
Datos de entrada	1 000 requests distribuidas (coordenadas reales en AOIs).
Componentes involucrados	IdentifyService, **PFZRunRepository**, **SceneRepository**, **MetricsCollector**, **Cache (si aplica)**
Pasos	1. Disparar N requests concurrentes a identify 2. Medir p50/p95 total y por backend (PFZ/SST/CHLA) 3. Verificar uso de cache si está habilitado 4. Registrar métricas y comparar con SLA
Resultado esperado	p95 ≤ SLA; tasas de error < 1%.
Postcondiciones	Métricas disponibles para capacity planning.
Prioridad	Media
Tipo	Integración (Rendimiento)
Caso de Uso	CU-13

Requisitos relacionados	RNF-01, RNF-03
Criterios de aceptación	Sin timeouts; colas controladas; estabilidad bajo carga.

ID	TC-I-13-03
Nombre	Consistencia temporal (mismo timestamp)
Objetivo	Asegurar que PFZ, SST y CHLA provienen de la misma ventana/timestamp o se etiqueta el desfase.
Precondiciones	Insumos con timestamps cercanos pero no idénticos; política de tolerancia ±6 h.
Datos de entrada	date=2025-10-24T12:00Z.
Componentes involucrados	IdentifyService, **PFZRunRepository**, **SceneRepository**, **PolicyConfig**
Pasos	1. Ejecutar identify con date específica 2. Recuperar PFZ/SST/CHLA y sus timestamps reales 3. Validar que el desfase ≤ tolerancia; si no, marcar temporal_mismatch=true en meta 4. Retornar payload con etiquetas
Resultado esperado	Datos coherentes o marcados con temporal_mismatch.
Postcondiciones	Transparencia temporal para consumidor.
Prioridad	Media
Tipo	Integración (Calidad de datos)
Caso de Uso	CU-13
Requisitos relacionados	RF-13, RNF-06
Criterios de aceptación	Política aplicada; etiquetas presentes si corresponde.

Pruebas de aceptación

ID	TC-A-13-01
Nombre	Inspección desde el visor (UX)
Objetivo	Permitir al usuario clickear en el mapa y ver PFZ, SST y CHLA con metadatos en un panel.
Precondiciones	Visor operativo; capas publicadas; Identify endpoint disponible.
Datos de entrada	Click en lat=-45.10, lon=-62.40, date=seleccionada.
Componentes involucrados	Front-end Web (Visor), **IdentifyService API**, **PFZRunRepository**, **SceneRepository**

Pasos	1. Click sobre el mapa en el punto seleccionado 2. UI invoca IdentifyService con lat/lon/date 3. Mostrar panel con PFZ, SST, CHLA y metadatos (runId, hora, calidad) 4. Permitir copiar valores y enlace a “Ver leyenda”
Resultado esperado	Panel informativo con datos coherentes y legibles.
Postcondiciones	Nada persistido.
Prioridad	Alta
Tipo	Aceptación
Caso de Uso	CU-13
Requisitos relacionados	RF-13, RNF-05
Criterios de aceptación	Tiempos de respuesta aceptables; accesibilidad de la UI (teclado/lectores).

ID	TC-A-13-02
Nombre	Mensajes claros cuando no hay datos
Objetivo	Mostrar feedback entendible si no existen datos para la fecha o punto (sin errores técnicos).
Precondiciones	Fecha fuera de rango o punto out_of_coverage.
Datos de entrada	Click en área sin cobertura.
Componentes involucrados	Front-end Web (Visor), **IdentifyService API**
Pasos	1. Click en zona sin datos 2. API retorna PFZ_NOT_AVAILABLE u OUT_OF_COVERAGE 3. UI muestra mensaje claro y sugerencia (p. ej. “Mueva el mapa o cambie la fecha”) 4. No mostrar placeholders engañosos
Resultado esperado	Mensaje útil; sin ruido visual.
Postcondiciones	Ninguna.
Prioridad	Media
Tipo	Aceptación (UX/Errores)
Caso de Uso	CU-13
Requisitos relacionados	RF-13, RNF-05
Criterios de aceptación	Texto no técnico; estilos consistentes con el sistema.

ID	TC-A-13-03
Nombre	Integridad de unidades y formato

Objetivo	Asegurar que los valores presentados usan unidades y formatos correctos (p. ej., °C para SST, mg·m⁻³ para CHLA).
Precondiciones	UI con internacionalización; metadatos de unidades disponibles.
Datos de entrada	Punto típico dentro de cobertura.
Componentes involucrados	Front-end Web (Visor), **IdentifyService**, **Metadatos de estilo/leyenda**
Pasos	1. Click en mapa para inspección 2. Verificar que SST muestra "°C" y CHLA "mg·m⁻³" (o unidad definida) 3. Confirmar que PFZ aparece en escala 0–1 (o % si configurado) 4. Validar formatos numéricos y localización (separadores decimales)
Resultado esperado	Unidades y formatos correctos y consistentes con la leyenda.
Postcondiciones	Comprensión correcta por parte del usuario.
Prioridad	Media
Tipo	Aceptación (Presentación/Calidad)
Caso de Uso	CU-13
Requisitos relacionados	RF-13, RNF-05
Criterios de aceptación	Coincidencia con definiciones de estilo; sin ambigüedades.

CU-14 Configurar suscripción de alertas

Pruebas Unitarias

ID	TC-U-14-01
Nombre	Crear/actualizar suscripción (camino feliz)
Objetivo	Crear o actualizar una suscripción válida asociada a un AOI del usuario.
Precondiciones	aoiId existente y propiedad del usuario; parámetros válidos (species, threshold, freq).
Datos de entrada	species="merluza", threshold=0.75, aoiId="aoi-123", freq="DAILY@09:00Z".
Componentes involucrados	CUT: SubscriptionService — **Dobles/colaboradores:** AOIRepository (mock), SubscriptionRepository (mock), AuditLog (mock), Validator (stub)

Pasos	1. Invocar SubscriptionService.createOrUpdate(species, threshold, aoiId, freq) 2. Verificar AOIRepository.exists(aoiId, currentUser) → ok 3. Validar parámetros con Validator (threshold en [0..1], species soportada, freq válida) 4. Verificar SubscriptionRepository.save(subscription) → subId 5. Verificar AuditLog.log('sub_upsert', subId)
Resultado esperado	Retorna ok(subId) con datos persistidos.
Postcondiciones	Suscripción almacenada o actualizada; auditoría registrada.
Prioridad	Alta
Tipo	Unitario
Caso de Uso	CU-14
Requisitos relacionados	RF-14, RF-15, RNF-06
Criterios de aceptación	Validaciones completas; única escritura en repositorio.

ID	TC-U-14-02
Nombre	AOI inexistente o sin pertenencia
Objetivo	Rechazar la suscripción si el AOI no existe o no pertenece al usuario.
Precondiciones	AOIRepository.exists(aoiId, userId) devuelve false.
Datos de entrada	species="merluza", threshold=0.6, aoiId="aoi-XXX", freq="DAILY@09:00Z".
Componentes involucrados	CUT: SubscriptionService — **Dobles/colaboradores:** AOIRepository (mock), SubscriptionRepository (mock), AuditLog (mock)
Pasos	1. Invocar SubscriptionService.createOrUpdate(...) 2. AOIRepository.exists(...) → false 3. Verificar que no se llama a SubscriptionRepository.save 4. Verificar AuditLog.log('sub_upsert_denied', reason='AOI_NOT_OWNED')
Resultado esperado	Error AOI_NOT_OWNED o NOT_FOUND.
Postcondiciones	Sin cambios en suscripciones.
Prioridad	Alta
Tipo	Unitario (Seguridad/Propiedad)
Caso de Uso	CU-14
Requisitos relacionados	RF-14, RF-15, RNF-02
Criterios de aceptación	Sin escritura accidental; auditoría de intento denegado.

ID	TC-U-14-03
Nombre	Validación de parámetros inválidos
Objetivo	Rechazar valores fuera de rango o formatos inválidos (threshold, freq, species).
Precondiciones	Validator con reglas activas; species debe existir en catálogo.
Datos de entrada	species="desconocida", threshold=1.5, aoiId="aoi-123", freq="DAILYY" (mal formado).
Componentes involucrados	CUT: SubscriptionService — **Dobles/colaboradores:** AOIRepository (mock), Validator (stub), AuditLog (mock)
Pasos	1. Invocar SubscriptionService.createOrUpdate(...) 2. AOIRepository.exists(...) → ok 3. Validator detecta species inválida, threshold fuera de [0..1], freq inválida 4. Verificar que no se llama a SubscriptionRepository.save 5. Verificar AuditLog.log('sub_upsert_failed', reason='VALIDATION')
Resultado esperado	Error VALIDATION con detalle de campos.
Postcondiciones	Sin suscripción creada.
Prioridad	Media
Tipo	Unitario (Validación)
Caso de Uso	CU-14
Requisitos relacionados	RF-15, RNF-05, RNF-06
Criterios de aceptación	Mensajes claros por campo; sin side-effects.

Pruebas de Integración

ID	TC-I-14-01
Nombre	API REST: POST /subscriptions (crear)
Objetivo	Validar el flujo API→Service→Repos con una creación válida.
Precondiciones	Usuario autenticado; AOI existente y del usuario.
Datos de entrada	POST /subscriptions {species, threshold, aoiId, freq}.
Componentes involucrados	API Gateway (REST), **SubscriptionService**, **AOIRepository** (DB), **SubscriptionRepository** (DB), **AuditLog**
Pasos	1. Hacer POST con payload válido 2. API valida esquema y llama SubscriptionService 3. Repos validan AOI y luego persisten suscripción 4. API responde 201 con subId
Resultado esperado	201 Created + subId.
Postcondiciones	Suscripción disponible para motor de alertas.
Prioridad	Alta

Tipo	Integración
Caso de Uso	CU-14
Requisitos relacionados	RF-14, RF-15, RNF-01
Criterios de aceptación	Esquema JSON válido; trazabilidad en auditoría.

ID	TC-I-14-02
Nombre	API REST: PUT /subscriptions/{id} (actualizar)
Objetivo	Validar actualización de parámetros (threshold/freq) y preservación de integridad.
Precondiciones	Existe subId del usuario; motor de alertas consumirá cambios.
Datos de entrada	PUT /subscriptions/sub-001 {threshold=0.82, freq="HOURLY@02"}.
Componentes involucrados	API Gateway, **SubscriptionService**, **SubscriptionRepository** (DB), **AuditLog**
Pasos	1. Enviar PUT con payload válido 2. Service valida y persiste cambios 3. Confirmar 200 OK y contenido actualizado 4. Auditoría sub_upsert con subId
Resultado esperado	200 OK; cambios visibles en DB.
Postcondiciones	Cambios efectivos para próximas corridas de alerta.
Prioridad	Media
Tipo	Integración
Caso de Uso	CU-14
Requisitos relacionados	RF-14, RF-15, RNF-06
Criterios de aceptación	Sin duplicar suscripciones; idempotencia en PUT.

ID	TC-I-14-03
Nombre	Seguridad: acceso cruzado denegado
Objetivo	Impedir crear/editar suscripción sobre AOI de otro usuario.
Precondiciones	aoiId pertenece a otro usuario.
Datos de entrada	POST /subscriptions {aoiId="aoi-de-otro"}.
Componentes involucrados	API Gateway, **SubscriptionService**, **AOIRepository**, **AuditLog**
Pasos	1. Intentar POST con aoiId ajeno 2. AOIRepository.exists(...) → false para currentUser 3. API responde 403 Forbidden 4. Auditoría sub_upsert_denied
Resultado esperado	403 Forbidden.

Postcondiciones	Sin cambios en DB.
Prioridad	Alta
Tipo	Integración (Seguridad)
Caso de Uso	CU-14
Requisitos relacionados	RF-14, RNF-02
Criterios de aceptación	No fuga de información sobre AOI ajeno.

Pruebas de aceptación

ID	TC-A-14-01
Nombre	Configurar suscripción desde UI (camino feliz)
Objetivo	Permitir al usuario crear/editar una suscripción desde el panel con feedback claro.
Precondiciones	UI operativa; AOI del usuario; catálogo de especies y frecuencias cargado.
Datos de entrada	species="merluza", threshold=0.7, aoiId="aoi-123", freq="DAILY@09:00Z".
Componentes involucrados	Front-end Web (Panel de Suscripciones), **API Gateway**, **SubscriptionService**, **AOIRepository/SubscriptionRepository**, **AuditLog**
Pasos	1. Abrir panel y completar formulario con species/threshold/freq/AOI 2. Enviar formulario y esperar confirmación 3. Ver en listado la nueva suscripción con estado “Activa” 4. Editar threshold y guardar cambios
Resultado esperado	Suscripción creada/actualizada con confirmaciones visibles.
Postcondiciones	Suscripción lista para motor de alertas.
Prioridad	Alta
Tipo	Aceptación (UX)
Caso de Uso	CU-14
Requisitos relacionados	RF-14, RF-15, RNF-05
Criterios de aceptación	Mensajes claros; validaciones en campo; accesibilidad.

ID	TC-A-14-02
Nombre	Validaciones UX (umbral y frecuencia)
Objetivo	Guiar al usuario con validaciones en tiempo real (rangos y formato).
Precondiciones	Front-end con validadores; tooltips/ayuda habilitados.

Datos de entrada	threshold=-0.2 o 1.2, freq="DAILYY".
Componentes involucrados	Front-end Web, **API Gateway**, **SubscriptionService**
Pasos	1. Ingresar threshold fuera de [0..1] → mensaje inline 2. Ingresar freq mal formada → mensaje inline 3. Corregir valores y reintentar envío 4. Confirmar creación exitosa
Resultado esperado	Errores visibles y comprensibles; no se envía payload inválido.
Postcondiciones	Solo se persisten valores válidos.
Prioridad	Media
Tipo	Aceptación (Usabilidad)
Caso de Uso	CU-14
Requisitos relacionados	RNF-05, RF-15
Criterios de aceptación	Zero submits inválidos; mensajes localizados.

ID	TC-A-14-03
Nombre	Transparencia de efecto (qué alertas se dispararán)
Objetivo	Mostrar al usuario un resumen de qué condiciones dispararán alertas antes de guardar.
Precondiciones	UI con previsualización; existe PFZ reciente para el AOI.
Datos de entrada	Configuración candidate de suscripción.
Componentes involucrados	Front-end Web (Preview), **SubscriptionService (simulación opcional)**, **PFZRunRepository**
Pasos	1. Completar formulario y abrir previsualización 2. Ver estimación de incidencias por rango temporal reciente 3. Confirmar guardado 4. Ver mensaje con próximo ciclo de evaluación según freq
Resultado esperado	Usuario entiende el impacto de la suscripción.
Postcondiciones	Configuración guardada con expectativas claras.
Prioridad	Media
Tipo	Aceptación (UX/Transparencia)
Caso de Uso	CU-14
Requisitos relacionados	RF-15, RNF-05
Criterios de aceptación	Previsualización coherente; sin promesas inconsistentes.

CU-15 Recibir alerta PFZ

Pruebas Unitarias

ID	TC-U-15-01
Nombre	Generación y envío de alertas (camino feliz)
Objetivo	Procesar suscripciones activas, intersectar AOIs con el último PFZ, crear alertas y enviarlas.
Precondiciones	Existen suscripciones activas; PFZRunRepository.getLatestRun() retorna runId; umbrales superados.
Datos de entrada	now=2025-10-26T09:00Z.
Componentes involucrados	CUT: AlertEngine — **Dobles/colaboradores:** SubscriptionRepository (stub), PFZRunRepository (stub), AOIRepository (stub), AlertRepository (mock), NotificationService (mock/spy), AuditLog (mock)
Pasos	1. Invocar AlertEngine.process() 2. Verificar SubscriptionRepository.listActive() → subs[] 3. Verificar PFZRunRepository.getLatestRun() → runId 4. Verificar AOIRepository.intersect(runId, subs[]) → hits[] (PFZ ≥ threshold) 5. Verificar AlertRepository.createAlerts(hits[]) → alerts[] 6. Verificar NotificationService.send(alerts[]) → queued7. Verificar AuditLog.log('alerts_sent', count=alerts.length)
Resultado esperado	Alertas creadas y encoladas para envío.
Postcondiciones	Registros de alertas disponibles para consulta y auditoría.
Prioridad	Alta
Tipo	Unitario
Caso de Uso	CU-15
Requisitos relacionados	RF-14, RF-15, RNF-01, RNF-02, RNF-06
Criterios de aceptación	Orden correcto de llamadas; conteos coherentes; sin excepciones.

ID	TC-U-15-02
Nombre	Sin coincidencias (no se dispara ninguna alerta)
Objetivo	Confirmar que no se crean ni envían alertas si el PFZ no supera umbrales en ningún AOI.
Precondiciones	AOIRepository.intersect() devuelve hits=[].
Datos de entrada	Ejecución estándar del motor.

Componentes involucrados	CUT: AlertEngine — **Dobles/colaboradores:** SubscriptionRepository (stub), PFZRunRepository (stub), AOIRepository (stub), AlertRepository (mock), NotificationService (mock), AuditLog (mock)
Pasos	1. Invocar AlertEngine.process() 2. Obtener subs[] y runId 3. AOIRepository.intersect(...) → hits=[] 4. Verificar que no se llama a AlertRepository.createAlerts ni a NotificationService.send 5. Verificar AuditLog.log('alerts_sent', count=0)
Resultado esperado	Cero alertas creadas; auditoría con count=0.
Postcondiciones	Sin cambios en estado de alertas.
Prioridad	Media
Tipo	Unitario (Reglas)
Caso de Uso	CU-15
Requisitos relacionados	RF-14, RF-15, RNF-06
Criterios de aceptación	Ausencia de side-effects; log coherente.

ID	TC-U-15-03
Nombre	Fallo transitorio al notificar (reintentos)
Objetivo	Reintentar envíos cuando NotificationService.send() falla transitoriamente y registrar resultado.
Precondiciones	NotificationService.send() falla con error recuperable en el primer intento.
Datos de entrada	alerts[] con N destinatarios.
Componentes involucrados	CUT: AlertEngine — **Dobles/colaboradores:** SubscriptionRepository (stub), PFZRunRepository (stub), AOIRepository (stub), AlertRepository (mock), NotificationService (mock con fallo transitorio), RetryPolicy (stub), AuditLog (mock)
Pasos	1. Invocar AlertEngine.process() 2. Generar alerts[] normalmente 3. Forzar fallo transitorio en NotificationService.send(alerts[]) 4. Aplicar RetryPolicy (p. ej., 3 intentos con backoff) 5. En segundo/tercer intento, retornar queued 6. Verificar AuditLog.log('alerts_sent', countOK, countRetry)
Resultado esperado	Alertas finalmente encoladas; reintentos registrados.
Postcondiciones	Sin alertas perdidas; métricas de reintento.
Prioridad	Media
Tipo	Unitario (Resiliencia)

Caso de Uso	CU-15
Requisitos relacionados	RNF-04, RNF-01, RF-15
Criterios de aceptación	Máximo de reintentos respetado; backoff aplicado.

Pruebas de Integración

ID	TC-I-15-01
Nombre	End-to-end: Subs→PFZ→Intersección→Alertas→Notificación
Objetivo	Validar el pipeline completo desde suscripciones activas hasta notificaciones encoladas.
Precondiciones	Hay subs[] activos; PFZRun reciente; AOIs con PFZ ≥ umbral.
Datos de entrada	Ejecución programada del motor.
Componentes involucrados	AlertEngine, **SubscriptionRepository** (DB), **PFZRunRepository** (DB), **AOIRepository** (DB/GIS), **AlertRepository** (DB), **NotificationService** (cola real/sandbox), **AuditLog**, **MetricsCollector**
Pasos	1. Ejecutar AlertEngine.process() 2. Recuperar subs[] y runId 3. Calcular hits[] mediante AOIRepository.intersect 4. Persistir alerts[] en AlertRepository 5. Enviar a NotificationService y verificar en cola 6. Verificar AuditLog y métricas de throughput
Resultado esperado	Alertas creadas y visibles en cola de notificación.
Postcondiciones	Alertas disponibles para worker de entrega (email/push).
Prioridad	Alta
Tipo	Integración
Caso de Uso	CU-15
Requisitos relacionados	RF-14, RF-15, RNF-01, RNF-03
Criterios de aceptación	Tiempos de proceso dentro de SLA; consistencia de conteos.

ID	TC-I-15-02
Nombre	Escalado por lotes (batching) y límites de proveedor
Objetivo	Respetar límites del servicio de notificaciones usando batches y rate limiting.
Precondiciones	Límite del proveedor, p. ej., 100 msg/min; muchas alertas en alerts[].
Datos de entrada	alerts[] con N ≫ 100.
Componentes involucrados	AlertEngine, **AlertRepository**, **NotificationService** (real/sandbox con rate-limit), **RateLimiter/Batcher**, **AuditLog**, **MetricsCollector**

Pasos	1. Generar alerts[] masivo 2. Enviar en lotes respetando rate-limit 3. Verificar que no hay errores 429/Throttling 4. Registrar métricas de lotes y latencias
Resultado esperado	Envíos escalados sin violar límites; colas sanas.
Postcondiciones	Todos los mensajes encolados; sin bloqueos.
Prioridad	Media
Tipo	Integración (Rendimiento/Escalabilidad)
Caso de Uso	CU-15
Requisitos relacionados	RNF-01, RNF-03, RF-15
Criterios de aceptación	0 errores por rate-limit; throughput estable.

ID	TC-I-15-03
Nombre	Deduplicación y anti-rebote (debounce)
Objetivo	Evitar enviar múltiples alertas iguales a la misma suscripción dentro de la ventana de gracia.
Precondiciones	Política cooldown=24h; condiciones PFZ persistentes.
Datos de entrada	Corridas alert-engine sucesivas en menos de 24h con mismo hit.
Componentes involucrados	AlertEngine, **AlertRepository** (consulta de últimos envíos), **NotificationService**, **AuditLog**, **PolicyConfig**
Pasos	1. Ejecutar process() #1 → crea y envía alerta A 2. Ejecutar process() #2 dentro de cooldown con mismo hit 3. Verificar que no se cree nueva alerta 4. Registrar AuditLog.log('alerts_skipped_dedup', subId)
Resultado esperado	Una sola alerta por ventana de gracia.
Postcondiciones	Historial limpio; usuarios no spammeados.
Prioridad	Alta
Tipo	Integración (Política)
Caso de Uso	CU-15
Requisitos relacionados	RF-15, RNF-05, RNF-06
Criterios de aceptación	Sin duplicados; logs claros de skip.

Pruebas de aceptación

ID	TC-A-15-01
Nombre	Recepción de alerta por el usuario (UX)
Objetivo	Validar que el usuario recibe la alerta con información clara del AOI, especie, umbral y hora.

Precondiciones	Suscripción activa; notificación habilitada (email/push).
Datos de entrada	Alerta generada para aoi-123, species="merluza", pfz>=0.8.
Componentes involucrados	NotificationService (canal real/sandbox), **Cliente del usuario** (email/app), **AlertRepository**
Pasos	1. Disparar motor y generar alerta 2. Confirmar entrega en el canal (email/push) 3. Verificar que el mensaje incluye AOI, especie, valor PFZ, umbral, fecha/hora y link a visor 4. Abrir link y validar que apunta a la vista del AOI
Resultado esperado	Notificación legible y accionable.
Postcondiciones	Usuario informado; posible navegación al visor.
Prioridad	Alta
Tipo	Aceptación
Caso de Uso	CU-15
Requisitos relacionados	RF-15, RNF-05
Criterios de aceptación	Contenido y formato correctos; enlaces válidos.

ID	TC-A-15-02
Nombre	Preferencias del usuario (canal/horario silencioso)
Objetivo	Respetar canal preferido y ventana de silencio configurada por el usuario.
Precondiciones	Usuario con channel=email y quietHours=22:00–07:00 local.
Datos de entrada	Alerta generada a las 23:00 local.
Componentes involucrados	AlertEngine, **NotificationService**, **UserPrefsRepository** (si aplica), **Scheduler/DelayQueue**, **AuditLog**
Pasos	1. Generar alerta dentro de quiet hours 2. Verificar que se encola con entrega diferida al final de la ventana 3. Confirmar que no se envía inmediatamente 4. Registrar AuditLog.log('alert_delayed_quiet_hours', alertId)
Resultado esperado	Alerta diferida; política de silencio respetada.
Postcondiciones	Entrega programada; no se molesta al usuario.
Prioridad	Media
Tipo	Aceptación (Política/UX)
Caso de Uso	CU-15
Requisitos relacionados	RNF-05, RNF-02
Criterios de aceptación	Sin envíos durante quiet hours; entrega al salir del rango.

ID	TC-A-15-03
Nombre	Trazabilidad y auditoría de alertas

Objetivo	Poder auditar el ciclo de vida de cada alerta: generación, encolado, entrega/fracaso.
Precondiciones	Auditoría habilitada; IDs correlacionados (runId, subId, alertId).
Datos de entrada	Alerta típica de producción.
Componentes involucrados	AlertRepository, **NotificationService**, **AuditLog**, **Observability (logs/metrics/traces)**
Pasos	1. Generar alerta y capturar alertId 2. Consultar auditoría por alertId → eventos created, queued, `delivered
Resultado esperado	Trazabilidad completa y exportable.
Postcondiciones	Evidencia lista para compliance/QA.
Prioridad	Media
Tipo	Aceptación (Operación/Compliance)
Caso de Uso	CU-15
Requisitos relacionados	RNF-02, RNF-06, RNF-04
Criterios de aceptación	Logs correlacionados; export sin datos faltantes.

CU-16 Mostrar historial PFZ

Pruebas Unitarias

ID	TC-U-16-01
Nombre	Consulta de corridas por rango y AOI (camino feliz)
Objetivo	Recuperar corridas PFZ dentro de un rango temporal filtrando por AOI.
Precondiciones	Existen PFZRun en el rango; aoiId válido.
Datos de entrada	range="2025-10-01/2025-10-25", aoiId="aoi-123".
Componentes involucrados	CUT: HistoryService — **Dobles/colaboradores:** PFZRunRepository (mock)
Pasos	1. Invocar HistoryService.query(range, aoiId) 2. Verificar PFZRunRepository.find(range, aoiId) → runs[] 3. Formatear respuesta con metadatos (runId, window, species, quality) 4. Retornar runs[]
Resultado esperado	Lista runs[] con corridas del rango.
Postcondiciones	Ninguna.
Prioridad	Alta
Tipo	Unitario
Caso de Uso	CU-16
Requisitos relacionados	RF-16, RNF-06

Criterios de aceptación	Única llamada al repositorio; esquema correcto.

ID	TC-U-16-02
Nombre	Validación de rango (formato y límites)
Objetivo	Rechazar rangos mal formados o fuera de límites de retención.
Precondiciones	Política historyRetention=180d; validador de fechas activo.
Datos de entrada	range="2025-10-33/2025-11-01" o range > retention.
Componentes involucrados	CUT: HistoryService — **Dobles/colaboradores:** DateRangeValidator (stub), AuditLog (mock)
Pasos	1. Invocar HistoryService.query(range, aoiId) 2. Validar formato y límites con DateRangeValidator 3. Detectar error de formato o violación de retención 4. Registrar AuditLog.log('history_query_rejected', reason) 5. Retornar error INVALID_RANGE
Resultado esperado	Error INVALID_RANGE.
Postcondiciones	Sin acceso al repositorio.
Prioridad	Alta
Tipo	Unitario (Validación)
Caso de Uso	CU-16
Requisitos relacionados	RF-16, RNF-06
Criterios de aceptación	Mensaje claro; cero side-effects.

ID	TC-U-16-03
Nombre	Paginación y ordenamiento
Objetivo	Soportar page/size/sort devolviendo la porción correcta y orden estable.
Precondiciones	Muchas corridas en el rango; estrategia sort=runTime DESC.
Datos de entrada	range, aoiId, page=2, size=20, sort="-runTime".
Componentes involucrados	CUT: HistoryService — **Dobles/colaboradores:** PFZRunRepository (mock)
Pasos	1. Invocar HistoryService.query(range, aoiId, page, size, sort)2. Verificar PFZRunRepository.find(...) con límites y orden3. Construir payload con items[], total, page, size4. Retornar página solicitada
Resultado esperado	Página 2 con 20 items ordenados DESC por runTime.
Postcondiciones	Ninguna.
Prioridad	Media

Tipo	Unitario (Comportamiento de lista)
Caso de Uso	CU-16
Requisitos relacionados	RF-16, RNF-05
Criterios de aceptación	Ítems y metadatos de paginación correctos.

Pruebas de Integración

ID	TC-I-16-01
Nombre	Query end-to-end y render en UI
Objetivo	Validar UserUI → HistoryService → PFZRunRepository y render de resultados.
Precondiciones	Corridas existentes; AOI válido del usuario.
Datos de entrada	range="2025-09-01/2025-10-25", aoiId="aoi-123".
Componentes involucrados	UserUI, **HistoryService**, **PFZRunRepository** (DB), **Serializer/Presenter**
Pasos	1. En UI ingresar rango y AOI 2. UI invoca HistoryService.query(range, aoiId) 3. Repositorio devuelve runs[] 4. UI presenta tabla/lista con metadatos y acceso a detalle
Resultado esperado	Lista visible y navegable en la UI.
Postcondiciones	Ninguna.
Prioridad	Alta
Tipo	Integración
Caso de Uso	CU-16
Requisitos relacionados	RF-16, RNF-05, RNF-01
Criterios de aceptación	Latencia bajo SLA; sin errores de serialización.

ID	TC-I-16-02
Nombre	Exportación desde histórico (GeoTIFF/GeoJSON)
Objetivo	Encadenar consulta y exportación de corridas seleccionadas.
Precondiciones	ExportService operativo; permisos del usuario.
Datos de entrada	Selección de runs[] en la UI, format="GeoTIFF".
Componentes involucrados	UserUI, **HistoryService**, **PFZRunRepository** (DB), **ExportService**, **StorageService**, **AuditLog**

Pasos	1. Consultar histórico y seleccionar corridas 2. UI invoca ExportService.export(runs[], format) 3. ExportService lee datos de PFZRunRepository 4. Escribir artefacto en StorageService y devolver URL 5. Registrar auditoría de export
Resultado esperado	URL de descarga de export válido.
Postcondiciones	Artefacto disponible en storage.
Prioridad	Media
Tipo	Integración
Caso de Uso	CU-16
Requisitos relacionados	RF-16, RF-21, RNF-01
Criterios de aceptación	Archivo abrible en herramientas GIS; metadatos correctos.

ID	TC-I-16-03
Nombre	Seguridad y aislamiento por usuario/rol
Objetivo	Evitar que un usuario consulte histórico de AOIs ajenos; respetar RBAC.
Precondiciones	Usuarios A y B; AOIs de A y B; políticas RBAC activas.
Datos de entrada	Usuario A consulta aoiId de B.
Componentes involucrados	UserUI, **HistoryService**, **PFZRunRepository** (DB), **Auth/RBAC**
Pasos	1. Autenticar como usuario A 2. Invocar query con aoiId de B 3. RBAC valida pertenencia/permiso y bloquea acceso 4. Retornar 403/FORBIDDEN
Resultado esperado	Acceso denegado de forma controlada.
Postcondiciones	Sin filtraciones de metadatos.
Prioridad	Alta
Tipo	Integración (Seguridad)
Caso de Uso	CU-16
Requisitos relacionados	RNF-02, RF-16
Criterios de aceptación	Logs de intento; sin enumeración de recursos.

Pruebas de aceptación

ID	TC-A-16-01
Nombre	Consulta y análisis visual en la UI
Objetivo	Permitir al usuario explorar corridas históricas y visualizar indicadores básicos.

Precondiciones	UI con filtros y panel de métricas; datos existentes.
Datos de entrada	Filtros por fecha, especie y AOI.
Componentes involucrados	Front-end Web (Histórico), **HistoryService API**, **PFZRunRepository**
Pasos	1. Aplicar filtros y ejecutar búsqueda 2. Mostrar lista paginada con metadatos claves 3. Renderizar gráfico/indicadores (conteo, calidad promedio) 4. Permitir abrir detalle de una corrida seleccionada
Resultado esperado	Exploración fluida y entendible.
Postcondiciones	Ninguna.
Prioridad	Alta
Tipo	Aceptación (UX)
Caso de Uso	CU-16
Requisitos relacionados	RF-16, RNF-05
Criterios de aceptación	Sin cuelgues; tiempos y navegación acordes.

ID	TC-A-16-02
Nombre	Export desde histórico (flujo usuario)
Objetivo	El usuario descarga datos PFZ del rango consultado en formatos soportados.
Precondiciones	Exportación habilitada; storage accesible.
Datos de entrada	Selección de runs; format="GeoTIFF" o format="GeoJSON".
Componentes involucrados	Front-end Web (Histórico), **ExportService API**, **PFZRunRepository**, **StorageService**
Pasos	1. Seleccionar corridas y solicitar export 2. Esperar confirmación y recibir URL 3. Descargar archivo y abrir en herramienta externa 4. Validar contenido y metadatos básicos
Resultado esperado	Descarga válida y utilizable.
Postcondiciones	Artefacto queda disponible por tiempo configurado.
Prioridad	Media
Tipo	Aceptación
Caso de Uso	CU-16
Requisitos relacionados	RF-21, RF-16, RNF-01
Criterios de aceptación	Enlaces válidos; integridad de archivo.

ID	TC-A-16-03
Nombre	Mensajería ante ausencia de resultados

Objetivo	Mostrar mensajes claros cuando no hay corridas en el rango solicitado.
Precondiciones	Rango sin datos.
Datos de entrada	range fuera de periodos procesados.
Componentes involucrados	Front-end Web (Histórico), **HistoryService API**
Pasos	1. Ejecutar búsqueda con rango vacío 2. API retorna runs=[] 3. UI muestra estado “Sin resultados” con sugerencias (ajustar fechas/criterios) 4. Evitar placeholders confusos
Resultado esperado	Feedback útil y orientador.
Postcondiciones	Ninguna.
Prioridad	Media
Tipo	Aceptación (UX)
Caso de Uso	CU-16
Requisitos relacionados	RNF-05
Criterios de aceptación	Mensaje no técnico; consistencia con el diseño del sistema.

CU-17 Habilitar nueva fuente satelital

Pruebas Unitarias

ID	TC-U-17-01
Nombre	Alta de fuente satelital (camino feliz)
Objetivo	Registrar una nueva fuente satelital verificando conectividad y credenciales antes de persistir.
Precondiciones	Endpoint alcanzable; credenciales válidas; variables soportadas por el sistema.
Datos de entrada	platform="Sentinel-3", sensor="OLCI", vars="SST,CHLA", endpoint="https://catalog.example/api"
Componentes involucrados	CUT: ProductService — **Dobles/colaboradores:** ConnectorService (mock), ProductRepository (mock), AuditLog (mock), Validator (stub)
Pasos	1. Invocar ProductService.addSource(platform, sensor, vars, endpoint) 2. Validar parámetros con Validator (plataforma/sensor/vars/URL) 3. Verificar ConnectorService.test(endpoint, creds) → ok 4. Verificar ProductRepository.save(productSource) → productId 5. Verificar AuditLog.log('product_add', productId)
Resultado esperado	ok(productId) y fuente en estado enabled=true.

Postcondiciones	Fuente registrada para ingestión.
Prioridad	Alta
Tipo	Unitario
Caso de Uso	CU-17
Requisitos relacionados	RF-17, RNF-06, RNF-02
Criterios de aceptación	Orden de llamadas correcto; datos persistidos consistentes.

ID	TC-U-17-02
Nombre	Conectividad fallida o credenciales inválidas
Objetivo	Impedir el alta cuando ConnectorService.test falla y registrar auditoría de rechazo.
Precondiciones	Endpoint inaccesible o 401/403.
Datos de entrada	Configuración válida salvo credenciales/endpoints.
Componentes involucrados	CUT: ProductService — **Dobles/colaboradores:** ConnectorService (mock con fallo), ProductRepository (mock), AuditLog (mock)
Pasos	1. Invocar ProductService.addSource(...) 2. Forzar ConnectorService.test(...) → error 3. Verificar que no se llama a ProductRepository.save 4. Verificar AuditLog.log('product_add_failed', reason='CONNECTIVITY
Resultado esperado	Error SOURCE_CONNECTIVITY_FAILED o AUTH_FAILED.
Postcondiciones	No se crea la fuente; estado inalterado.
Prioridad	Alta
Tipo	Unitario (Errores)
Caso de Uso	CU-17
Requisitos relacionados	RF-17, RNF-04, RNF-02
Criterios de aceptación	Sin efectos laterales; mensaje claro.

ID	TC-U-17-03
Nombre	Validación de variables no soportadas
Objetivo	Rechazar alta si vars incluye productos desconocidos por el modelo (p. ej., SALINITY).
Precondiciones	Catálogo de variables soportadas (SST, CHLA) configurado.
Datos de entrada	vars="SST,SALINITY".

Componentes involucrados	CUT: ProductService — **Dobles/colaboradores:** Validator (stub), AuditLog (mock)
Pasos	1. Invocar ProductService.addSource(..., vars="SST,SALINITY") 2. Validator detecta variable no soportada 3. Verificar que no se llama a ConnectorService ni a ProductRepository 4. Verificar AuditLog.log('product_add_failed', reason='UNSUPPORTED_VAR')
Resultado esperado	Error UNSUPPORTED_VAR con detalle.
Postcondiciones	Sin registro de fuente.
Prioridad	Media
Tipo	Unitario (Validación)
Caso de Uso	CU-17
Requisitos relacionados	RF-17, RNF-06
Criterios de aceptación	Mensaje de validación específico; sin side-effects.

Pruebas de Integración

ID	TC-I-17-01
Nombre	API Admin: POST /products (alta)
Objetivo	Validar flujo AdminUI/API → ProductService → ConnectorService → ProductRepository.
Precondiciones	Usuario administrador autenticado; role con permiso para alta.
Datos de entrada	POST /products {platform, sensor, vars, endpoint, creds}.
Componentes involucrados	AdminUI/API Gateway, **ProductService**, **ConnectorService** (real/sandbox), **ProductRepository** (DB), **AuditLog**
Pasos	1. Enviar POST con payload válido 2. ProductService valida y testea conectividad 3. Persistir fuente en ProductRepository 4. Responder 201 con productId y estado enabled=true
Resultado esperado	Fuente creada y lista para ingestión.
Postcondiciones	Fuente visible en catálogo de productos.
Prioridad	Alta
Tipo	Integración
Caso de Uso	CU-17
Requisitos relacionados	RF-17, RNF-02, RNF-01
Criterios de aceptación	201 Created; auditoría y trazas presentes.

ID	TC-I-17-02
Nombre	Hardening: sanitización de endpoint y manejo de TLS
Objetivo	Asegurar que URLs se normalizan y se requiere TLS válido.
Precondiciones	Endpoint con certificados válidos/expirados; redirecciones 301/302.
Datos de entrada	endpoint="http://catalog…" y endpoint="https://catalog…" (cert válido y no válido).
Componentes involucrados	ProductService, **ConnectorService** (HTTP client real/sandbox), **SecurityConfig/TrustStore**, **AuditLog**
Pasos	1. Probar alta con http:// → forzar rechazo o upgrade a https:// según política 2. Probar https:// con cert inválido → rechazo controlado 3. Probar https:// con cert válido → aceptación 4. Verificar auditorías de cada decisión
Resultado esperado	Solo se acepta https:// con TLS válido.
Postcondiciones	Config de fuente segura.
Prioridad	Media
Tipo	Integración (Seguridad)
Caso de Uso	CU-17
Requisitos relacionados	RNF-02, RNF-04
Criterios de aceptación	Rechazos claros; logs de seguridad completos.

ID	TC-I-17-03
Nombre	Registro y descubrimiento en el Scheduler
Objetivo	Verificar que una fuente habilitada es descubierta por el Scheduler para ingestión.
Precondiciones	Scheduler activo; ProductRepository con la fuente enabled=true.
Datos de entrada	productId recientemente creado.
Componentes involucrados	Scheduler, **ProductRepository** (DB), **IngestionSSTService/IngestionChlaService** (según vars), **AuditLog**
Pasos	1. Scheduler consulta fuentes habilitadas 2. Descubre la nueva fuente por productId 3. Programa job de ingestión correspondiente (SST/CHLA) 4. Registrar evento scheduler_register_product en auditoría

Resultado esperado	Job de ingestión programado para la nueva fuente.
Postcondiciones	Fuente integrada al ciclo operativo.
Prioridad	Media
Tipo	Integración
Caso de Uso	CU-17
Requisitos relacionados	RF-17, RNF-03
Criterios de aceptación	El job aparece y corre en la próxima ventana planificada.

Pruebas de aceptación

ID	TC-A-17-01
Nombre	Alta desde consola de administración (UX)
Objetivo	Permitir a un operador registrar una fuente con feedback paso a paso (validación, prueba, alta).
Precondiciones	AdminUI operativo; usuario con permisos.
Datos de entrada	Plataforma, sensor, variables, endpoint y credenciales.
Componentes involucrados	AdminUI, **API Gateway**, **ProductService**, **ConnectorService**, **ProductRepository**, **AuditLog**
Pasos	1. Completar formulario y ejecutar “Probar conexión” 2. Ver feedback de prueba (latencia, auth, versión API) 3. Confirmar “Registrar” 4. Ver en listado la fuente enabled=true
Resultado esperado	Alta exitosa con mensajes claros.
Postcondiciones	Fuente disponible para Scheduler.
Prioridad	Alta
Tipo	Aceptación (UX)
Caso de Uso	CU-17
Requisitos relacionados	RF-17, RNF-05
Criterios de aceptación	Validaciones inline; errores no técnicos comprensibles.

ID	TC-A-17-02
Nombre	Observabilidad y auditoría de alta
Objetivo	Confirmar que el alta deja trazas y auditorías consultables para compliance.
Precondiciones	Observabilidad activa (logs/metrics/traces).
Datos de entrada	Alta típica de fuente.

Componentes involucrados	AuditLog, **Observability Stack** (logs/metrics/traces), **ProductService**
Pasos	1. Ejecutar alta de fuente 2. Consultar auditoría por productId → eventos test_ok, saved, enabled 3. Ver métricas (latencia de test) 4. Confirmar traza distribuida con correlación
Resultado esperado	Evidencia completa del proceso.
Postcondiciones	Material disponible para auditorías internas.
Prioridad	Media
Tipo	Aceptación (Compliance/Operación)
Caso de Uso	CU-17
Requisitos relacionados	RNF-02, RNF-06
Criterios de aceptación	IDs correlacionados y exportables.

ID	TC-A-17-03
Nombre	Prueba funcional de ingestión posterior al alta
Objetivo	Verificar que, luego del alta, se pueden listar y descargar productos L2 de la nueva fuente.
Precondiciones	Fuente recién creada; catálogos con escenas disponibles.
Datos de entrada	Ventana de tiempo reciente y filtros por variable (SST/CHLA).
Componentes involucrados	AdminUI/Operación, **Scheduler/Runner**, **RemoteCatalog**, **StorageService**, **SceneRepository**, **AuditLog**
Pasos	1. Forzar corrida de ingestión para la fuente recién alta 2. Listar escenas L2 en el rango 3. Descargar y registrar escenas en SceneRepository 4. Verificar auditoría `sst_ingest`
Resultado esperado	Ingestión efectiva con escenas accesibles.
Postcondiciones	Datos listos para pipeline PFZ.
Prioridad	Alta
Tipo	Aceptación (Funcional)
Caso de Uso	CU-17
Requisitos relacionados	RF-06, RF-07, RF-17, RNF-01
Criterios de aceptación	Conteos coinciden; sin errores en descarga/registro.

CU-18 Administrar usuarios

Pruebas Unitarias

ID	TC-U-18-01
Nombre	Actualización de datos básicos (camino feliz)
Objetivo	Modificar nombre, estado y roles permitidos de un usuario existente respetando reglas de validación y auditoría.
Precondiciones	userId existente; admin con permiso USER_EDIT; payload válido.
Datos de entrada	userId="u-102", changes={name:"Operador Sur", status:"ACTIVE", roles:["OPERATOR"]}
Componentes involucrados	CUT: UserAdminService — **Dobles/colaboradores:** UserRepository (mock), Validator (stub), AuthorizationService (stub), AuditLog (mock)
Pasos	1. Invocar UserAdminService.update(userId, changes) 2. Verificar AuthorizationService.hasPermission(admin,"USER_EDIT") → ok 3. Verificar UserRepository.findById(userId) → user 4. Validar changes con Validator (status/roles válidos) 5. Verificar UserRepository.updateUser(userId, changes) → ok 6. Verificar AuditLog.log('admin_user_update', userId, diff=changes)
Resultado esperado	ok=true; usuario actualizado.
Postcondiciones	Estado persistido y traza de auditoría.
Prioridad	Alta
Tipo	Unitario
Caso de Uso	CU-18
Requisitos relacionados	RF-19, RNF-02, RNF-06
Criterios de aceptación	Secuencia exacta; sin escrituras redundantes; auditoría presente.

ID	TC-U-18-02
Nombre	Intento de escalamiento de privilegios no autorizado
Objetivo	Bloquear la asignación de roles superiores cuando el admin no tiene permiso para ello.
Precondiciones	Admin con USER_EDIT pero sin ROLE_GRANT_ADMIN; changes.roles=["ADMIN"].
Datos de entrada	userId="u-103", changes={roles:["ADMIN"]}

Componentes involucrados	CUT: UserAdminService — **Dobles/colaboradores:** AuthorizationService (stub), UserRepository (mock), AuditLog (mock)
Pasos	1. Invocar UserAdminService.update(userId, changes) 2.AuthorizationService.hasPermission(admin,"ROLE_GRANT_ADMIN") → false 3. Verificar que no se llama a UserRepository.updateUser 4. Verificar AuditLog.log('admin_user_update_denied', userId, reason='INSUFFICIENT_PRIVILEGES')
Resultado esperado	Error FORBIDDEN o INSUFFICIENT_PRIVILEGES.
Postcondiciones	Usuario sin cambios.
Prioridad	Alta
Tipo	Unitario (Seguridad)
Caso de Uso	CU-18
Requisitos relacionados	RF-19, RNF-02
Criterios de aceptación	Sin side-effects; mensaje claro.

ID	TC-U-18-03
Nombre	Campos inmutables y validación de entrada
Objetivo	Rechazar cambios en id, email y datos con formato inválido.
Precondiciones	Usuario existente; validador activo para email/formato.
Datos de entrada	changes={id:"otro", email:"correo--mal", status:"X"}
Componentes involucrados	CUT: UserAdminService — **Dobles/colaboradores:** Validator (stub), UserRepository (mock), AuditLog (mock)
Pasos	1. Invocar UserAdminService.update(userId, changes) 2. Validator detecta campos inmutables (id, email) y status inválido 3. Verificar que no se llama a UserRepository.updateUser 4. Verificar AuditLog.log('admin_user_update_failed', userId, reason='VALIDATION')
Resultado esperado	Error VALIDATION con detalle por campo.
Postcondiciones	Sin cambios en persistencia.
Prioridad	Media
Tipo	Unitario (Validación)
Caso de Uso	CU-18
Requisitos relacionados	RF-19, RNF-06, RNF-05
Criterios de aceptación	Mensajes por campo; ningún write en DB.

Pruebas de Integración

ID	TC-I-18-01
Nombre	API Admin: PUT /admin/users/{id} (camino feliz)
Objetivo	Validar AdminUI/API → UserAdminService → UserRepository con RBAC y auditoría.
Precondiciones	Token admin con USER_EDIT; usuario objetivo existe.
Datos de entrada	PUT /admin/users/u-102 {name,status,roles}
Componentes involucrados	AdminUI/API Gateway, **UserAdminService**, **AuthorizationService**, **UserRepository** (DB), **AuditLog**
Pasos	1. Enviar PUT con payload válido 2. API verifica token y USER_EDIT 3. Service valida y persiste cambios 4. API responde 200 con usuario actualizado 5. Auditoría admin_user_update registrada
Resultado esperado	200 OK; cambios persistidos.
Postcondiciones	Datos actualizados y auditados.
Prioridad	Alta
Tipo	Integración
Caso de Uso	CU-18
Requisitos relacionados	RF-19, RNF-02, RNF-06
Criterios de aceptación	Estructura JSON correcta; sin campos desconocidos.

ID	TC-I-18-02
Nombre	Control de concurrencia (optimistic locking)
Objetivo	Evitar sobrescritura perdida usando version en el update.
Precondiciones	Usuario con version=7; dos admins editan simultáneamente.
Datos de entrada	PUT A con version=7; PUT B con version=7 después de A.
Componentes involucrados	API Gateway, **UserAdminService**, **UserRepository** (DB con version), **AuditLog**
Pasos	1. Admin A envía PUT con version=7 → éxito, version=8 2. Admin B envía PUT con version=7 → conflicto 3. Service retorna 409 CONFLICT con detalle de versión 4. Registrar admin_user_update_conflict
Resultado esperado	Segundo update falla con 409.
Postcondiciones	Persistencia consistente; sin pérdida de cambios.
Prioridad	Media
Tipo	Integración (Concurrencia)

Caso de Uso	CU-18
Requisitos relacionados	RNF-04, RNF-06
Criterios de aceptación	Manejo de 409 y recomendación de reintento/refresh.

Campo	**Valor**
ID	TC-I-18-03
Nombre	Seguridad: edición cruzada denegada
Objetivo	Impedir que un admin de dominio “A” edite usuarios del dominio “B”.
Precondiciones	Política multi-tenant por tenantId.
Datos de entrada	PUT /admin/users/u-555 (tenant B) por admin de tenant A.
Componentes involucrados	API Gateway, **AuthorizationService** (filtros por tenant), **UserAdminService**, **UserRepository**, **AuditLog**
Pasos	1. Resolver tenantId del token 2. Verificar que user.tenantId != token.tenantId 3. Rechazar con 403 Forbidden 4. Registrar admin_user_update_denied
Resultado esperado	403 Forbidden.
Postcondiciones	Sin lectura/escritura de datos de otro tenant.
Prioridad	Alta
Tipo	Integración (Seguridad/Multitenancy)
Caso de Uso	CU-18
Requisitos relacionados	RNF-02
Criterios de aceptación	No hay fuga de metadatos del usuario objetivo.

Pruebas de aceptación

ID	TC-A-18-01
Nombre	Edición desde consola de administración (UX)
Objetivo	Permitir cambiar nombre, estado y roles permitidos con validaciones y confirmaciones claras.
Precondiciones	Admin autenticado; usuario objetivo cargado en el formulario.
Datos de entrada	Cambios de name, status, roles permitidos.
Componentes involucrados	AdminUI, **API Gateway**, **UserAdminService**, **UserRepository**, **AuditLog**
Pasos	1. Abrir ficha de usuario y editar campos 2. Guardar y esperar confirmación visual 3. Ver reflejados los cambios en el listado 4. Revisar panel de actividad (auditoría)

Resultado esperado	Cambios visibles y consistentes; feedback positivo.
Postcondiciones	Auditoría consultable.
Prioridad	Alta
Tipo	Aceptación (UX)
Caso de Uso	CU-18
Requisitos relacionados	RF-19, RNF-05
Criterios de aceptación	Validaciones inline; mensajes no técnicos.

ID	TC-A-18-02
Nombre	Restricciones de roles en UI (evitar escalamiento)
Objetivo	La UI solo permite seleccionar roles que el admin puede otorgar.
Precondiciones	Mapeo grantableRoles por perfil del admin.
Datos de entrada	Intento de otorgar ADMIN sin permiso.
Componentes involucrados	AdminUI, **API Gateway**, **AuthorizationService**
Pasos	1. Abrir selector de roles 2. Ver que roles no otorgables aparecen deshabilitados/ocultos 3. Intentar enviar payload manipulando la red (devtools) 4. API responde 403 y UI muestra mensaje claro
Resultado esperado	No se puede escalar privilegios desde UI ni por API.
Postcondiciones	Integridad del modelo de permisos.
Prioridad	Alta
Tipo	Aceptación (Seguridad/UX)
Caso de Uso	CU-18
Requisitos relacionados	RNF-02, RNF-05
Criterios de aceptación	Controles de UI y de servidor coherentes.

ID	TC-A-18-03
Nombre	Manejo de conflictos de edición (UX)
Objetivo	Mostrar un flujo claro cuando se produce un conflicto de versiones durante la edición.
Precondiciones	Otro admin actualizó el usuario (version cambió).
Datos de entrada	Guardar cambios con version desactualizada.
Componentes involucrados	AdminUI, **API Gateway**, **UserAdminService**
Pasos	1. Intentar guardar cambios y recibir 409 2. UI muestra banner con opción “Recargar datos” 3. Recargar ficha con la última versión 4. Reaplicar cambios y guardar

Resultado esperado	Usuario logra guardar tras resolver el conflicto.
Postcondiciones	Datos consistentes; UX sin pérdida de trabajo excesiva.
Prioridad	Media
Tipo	Aceptación (UX/Concurrencia)
Caso de Uso	CU-18
Requisitos relacionados	RNF-05, RNF-04
Criterios de aceptación	Mensajes comprensibles; recuperación simple y guiada.

CU-19 Exportar PFZ

Pruebas Unitarias

ID	TC-U-19-01
Nombre	Export GeoTIFF con recorte (camino feliz)
Objetivo	Exportar corridas PFZ de un rango/área en formato GeoTIFF con CRS estándar.
Precondiciones	Hay corridas en el rango; AOI válido; CRS de salida EPSG:4326.
Datos de entrada	area=aoi-123, range=2025-10-01/2025-10-25, format="GeoTIFF".
Componentes involucrados	CUT: ExportService — **Dobles/colaboradores:** PFZRunRepository (mock), Clipper (mock), StorageService (mock), AuditLog (mock)
Pasos	1. Invocar ExportService.export(area, range, "GeoTIFF") 2. Verificar PFZRunRepository.collect(range, area) → rasters[] 3. Verificar Clipper.clip(rasters[], area) → subset 4. Verificar StorageService.write(subset, "GeoTIFF", "EPSG:4326", meta) → url 5. Verificar AuditLog.log('export', url)
Resultado esperado	Retorna downloadURL(url) válido.
Postcondiciones	Artefacto persistido y trazado en auditoría.
Prioridad	Alta
Tipo	Unitario
Caso de Uso	CU-19
Requisitos relacionados	RF-21, RF-16, RNF-01, RNF-06
Criterios de aceptación	Llamadas en orden, sin duplicados; CRS correcto.

ID	TC-U-19-02
Nombre	Formato no soportado
Objetivo	Rechazar export si format no es válido (p. ej., CSV).
Precondiciones	Catálogo de formatos: GeoTIFF, GeoJSON.

Datos de entrada	format="CSV".
Componentes involucrados	CUT: ExportService — **Dobles/colaboradores:** AuditLog (mock)
Pasos	1. Invocar ExportService.export(area, range, "CSV") 2. Validar formato contra catálogo 3. Rechazar con error UNSUPPORTED_FORMAT 4. Verificar AuditLog.log('export_failed', reason='UNSUPPORTED_FORMAT')
Resultado esperado	Error UNSUPPORTED_FORMAT.
Postcondiciones	Sin escritura en storage.
Prioridad	Media
Tipo	Unitario (Validación)
Caso de Uso	CU-19
Requisitos relacionados	RF-21, RNF-06
Criterios de aceptación	Mensaje claro; cero side-effects.

ID	TC-U-19-03
Nombre	Fallo de recorte/CRS
Objetivo	Manejar errores del Clipper (geometría inválida o reproyección fallida).
Precondiciones	AOI con geometría self-intersecting o CRS incompatible.
Datos de entrada	area=aoi-corrupta, format="GeoTIFF".
Componentes involucrados	CUT: ExportService — **Dobles/colaboradores:** PFZRunRepository (mock), Clipper (mock con error), AuditLog (mock)
Pasos	1. Invocar ExportService.export(area, range, "GeoTIFF") 2. PFZRunRepository.collect(...) → rasters[] 3. Clipper.clip(...) lanza `GEOMETRY_ERROR
Resultado esperado	Error controlado EXPORT_CLIP_FAILED.
Postcondiciones	Nada persistido.
Prioridad	Media
Tipo	Unitario (Errores)
Caso de Uso	CU-19
Requisitos relacionados	RNF-04, RNF-06
Criterios de aceptación	Sin fugas de recursos; logging adecuado.

Pruebas de Integración

ID	TC-I-19-01
Nombre	End-to-end: Collect→Clip→Write (GeoTIFF)

Objetivo	Validar el flujo completo con dependencias reales/sandbox.
Precondiciones	Storage accesible; rasters disponibles; AOI válido.
Datos de entrada	range=2025-10-05/2025-10-20, area=aoi-123, format="GeoTIFF".
Componentes involucrados	ExportService, **PFZRunRepository** (DB), **Clipper** (lib GIS), **StorageService** (S3/FS), **AuditLog**
Pasos	1. Ejecutar ExportService.export(...) 2. Collect de rasters PFZ por rango/área 3. Clip geoespacial del mosaico 4. Write en storage y obtener URL 5. Verificar auditoría y que el archivo se descarga correctamente
Resultado esperado	URL de descarga válido; archivo abrible en QGIS.
Postcondiciones	Artefacto disponible por TTL configurado.
Prioridad	Alta
Tipo	Integración
Caso de Uso	CU-19
Requisitos relacionados	RF-21, RF-16, RNF-01
Criterios de aceptación	Integridad de archivo, tamaño razonable, metadatos correctos.

ID	TC-I-19-02
Nombre	Export GeoJSON (vectores derivados)
Objetivo	Validar export en GeoJSON (isocontornos/umbrales PFZ si aplica).
Precondiciones	Pipeline de vectorización habilitado (opcional).
Datos de entrada	format="GeoJSON", threshold=0.7.
Componentes involucrados	ExportService, **PFZRunRepository**, **Clipper/Vectorizer**, **StorageService**, **AuditLog**
Pasos	1. Collect de rasters 2. Vectorizer genera polígonos por umbral 3. Clip por AOI 4. Write GeoJSON en storage y devolver URL
Resultado esperado	GeoJSON válido (FeatureCollection).
Postcondiciones	Archivo usable en GIS/visores web.
Prioridad	Media
Tipo	Integración
Caso de Uso	CU-19
Requisitos relacionados	RF-21, RNF-05
Criterios de aceptación	Propiedades y CRS correctos; tamaño razonable.

ID	TC-I-19-03
Nombre	Rendimiento y límite de tamaño
Objetivo	Asegurar p95 y tamaño máximo de export bajo límites configurados.
Precondiciones	Datos voluminosos; límites maxArea, maxDuration, maxBytes.
Datos de entrada	Solicitud grande cercana a límites.
Componentes involucrados	ExportService, **PFZRunRepository**, **Clipper**, **StorageService**, **MetricsCollector**, **PolicyConfig**, **AuditLog**
Pasos	1. Ejecutar export con parámetros al límite 2. Medir latencias totales y por etapa (collect/clip/write) 3. Verificar que el tamaño final ≤ maxBytes o activar compactación 4. Registrar métricas y auditoría
Resultado esperado	SLA de rendimiento cumplido y tamaño dentro del límite.
Postcondiciones	Export disponible o rechazo por política con mensaje claro.
Prioridad	Media
Tipo	Integración (Rendimiento/Política)
Caso de Uso	CU-19
Requisitos relacionados	RNF-01, RNF-03, RF-21
Criterios de aceptación	p95 dentro del umbral; sin timeouts.

Pruebas de aceptación

ID	TC-A-19-01
Nombre	Exportar desde la UI (flujo completo)
Objetivo	Permitir al usuario seleccionar área/rango/formato y descargar el archivo.
Precondiciones	UI operativa; permisos del usuario; storage accesible.
Datos de entrada	area=AOI seleccionado, range, format="GeoTIFF".
Componentes involucrados	Front-end Web (Histórico/Export), **ExportService API**, **PFZRunRepository**, **StorageService**, **AuditLog**
Pasos	1. Seleccionar AOI, rango y formato 2. Confirmar export y esperar notificación de listo 3. Recibir downloadURL y descargar 4. Abrir en QGIS/visor y validar contenido
Resultado esperado	Archivo descargado y utilizable.
Postcondiciones	Link activo hasta TTL configurado.
Prioridad	Alta
Tipo	Aceptación
Caso de Uso	CU-19

Requisitos relacionados	RF-21, RNF-05
Criterios de aceptación	UX clara; mensajes no técnicos; sin enlaces rotos.

ID	TC-A-19-02
Nombre	Metadatos y proyección del archivo
Objetivo	Verificar que el artefacto contiene metadatos (CRS, rango temporal, especie, resolución).
Precondiciones	Export generado correctamente.
Datos de entrada	Archivo descargado.
Componentes involucrados	Herramienta GIS del usuario, **ExportService** (definiciones de meta)
Pasos	1. Abrir archivo en GIS 2. Verificar CRS EPSG:4326 o el definido 3. Revisar metadatos (runId/range/species/resolution) 4. Confirmar alineación con leyenda/estilo esperado
Resultado esperado	Metadatos completos y correctos.
Postcondiciones	Confianza en la interpretabilidad del archivo.
Prioridad	Media
Tipo	Aceptación (Calidad de datos)
Caso de Uso	CU-19
Requisitos relacionados	RF-21, RNF-05, RNF-06
Criterios de aceptación	Sin inconsistencias entre meta y contenido.

ID	TC-A-19-03
Nombre	Manejo de errores y mensajes al usuario
Objetivo	Mostrar mensajes claros cuando la exportación falla por formato, tamaño o geometría.
Precondiciones	Política de límites activa; validadores en back y front.
Datos de entrada	Export con área excesiva o formato no soportado.
Componentes involucrados	Front-end Web, **ExportService API**, **PolicyConfig**
Pasos	1. Solicitar export que viola un límite o con formato inválido 2. API responde error `UNSUPPORTED_FORMAT
Resultado esperado	Feedback útil; operación exitosa tras corrección.
Postcondiciones	Sin artefactos huérfanos en storage.
Prioridad	Media
Tipo	Aceptación (UX/Errores)
Caso de Uso	CU-19
Requisitos relacionados	RNF-05, RNF-04
Criterios de aceptación	Mensajes consistentes; sin “stacktraces” en UI.

Definición de base de datos para el sistema.

El proceso de normalización se aplicó para estructurar las tablas de manera eficiente, eliminando dependencias de datos que podrían causar anomalías en la inserción, actualización y eliminación. Se ha logrado una estructura que, al menos, cumple con la tercera forma normal (3NF).

Fundamentos del Diseño del Modelo de Base de Datos Relacional (DER) para SIZPOPE

El Diagrama de Entidad-Relación (DER) del Sistema de Atención de Emergencias (SATE) representa el esquema lógico de la capa de persistencia, constituyendo la traducción directa de las clases de entidad definidas en el Diagrama de Clases de Diseño (DCD). La estructura de este modelo ha sido definida para garantizar la integridad de los datos, la trazabilidad de la información y el soporte eficiente a todas las reglas de negocio identificadas en la etapa de análisis.

Coherencia y Trazabilidad con el Diseño Orientado a Objetos

La estructura del DER se basa en el principio de coherencia del modelo. Cada clase estereotipada como *entity* en el DCD (User, AOI, Scene, PFZrun, etc.) se mapea a una tabla principal en el DER. Esta correspondencia directa facilita la trazabilidad entre el código de la aplicación (los Controladores y Entidades del DCD) y la base de datos, asegurando que las operaciones definidas en las clases de control (ej. User.createUser()) encuentren el esquema de datos exacto y consistente que esperan.

Integridad Estructural a través de la Normalización

El esquema de la base de datos se ha diseñado adhiriéndose a los principios de Normalización, principalmente hasta la Tercera Forma Normal (3FN). Este proceso es crucial para la estabilidad del sistema, ya que logra dos objetivos fundamentales:

Minimización de la Redundancia: Almacenar datos solo una vez. Por ejemplo, la información de un usuario reside únicamente en la tabla User y es referenciada en alerts_subscription mediante su identificador único. Esto reduce el espacio de almacenamiento y elimina las inconsistencias que surgirían al actualizar un mismo dato en múltiples lugares.

Mantenimiento de la Consistencia: Las tablas están diseñadas para que los atributos dependan completamente de su Clave Primaria (PK). Esto previene anomalías de inserción, actualización y borrado, asegurando que cada fila de datos representa una entidad única y bien definida.

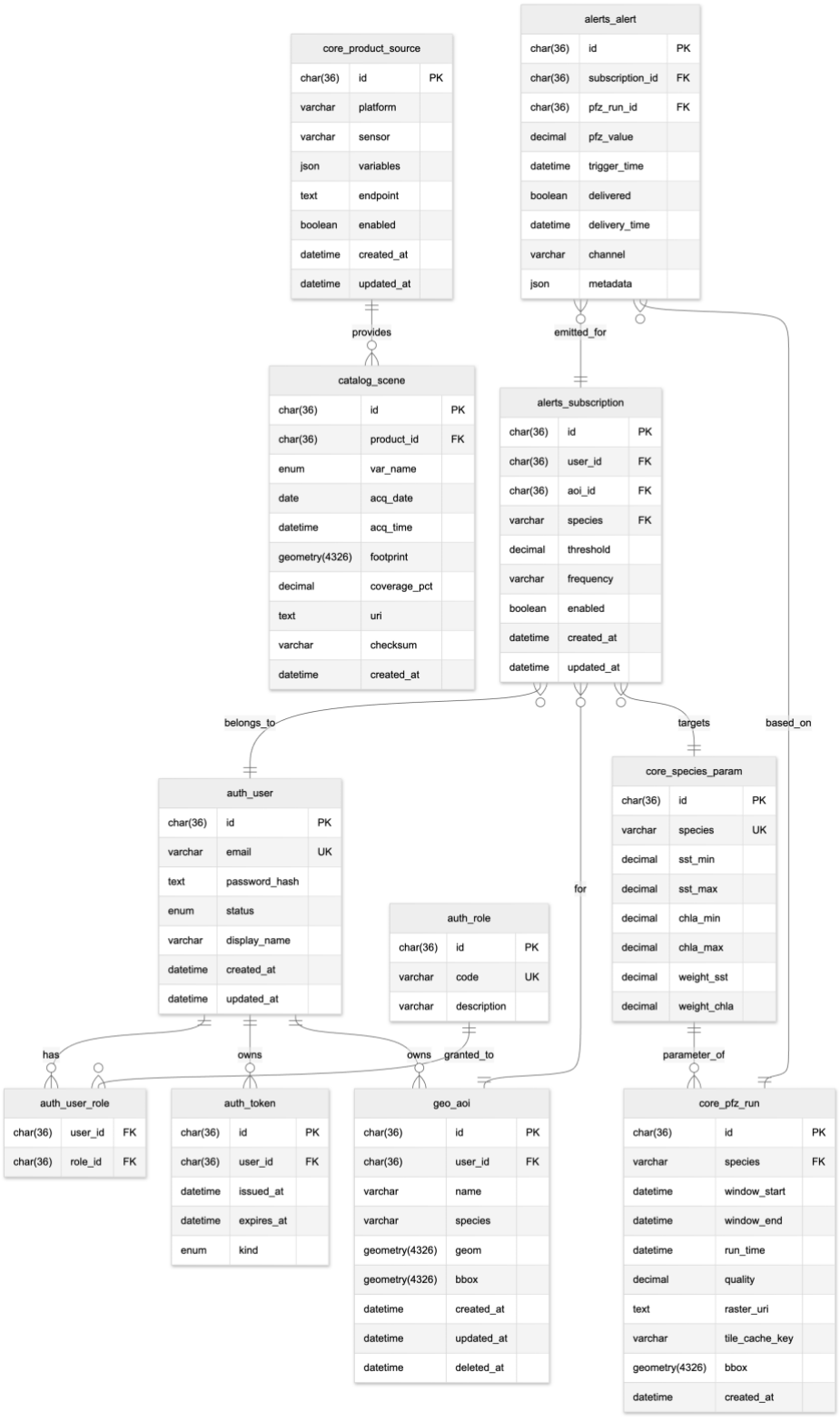
Aplicación de las Reglas de Negocio mediante Relaciones

Las asociaciones entre las tablas del DER son una manifestación directa de las relaciones lógicas de los Casos de Uso del SIZPOPE, impuestas mediante Claves Foráneas (FK). Estas claves garantizan la Integridad Referencial, lo que significa que una tabla relacionada solo puede hacer referencia a una fila existente en la tabla primaria, haciendo imposible la existencia de datos huérfanos o inválidos.

Esta estructura asegura que, incluso ante la complejidad transaccional de un sistema de emergencias, la

base de datos actuará como un repositorio de información consistente y altamente consultable.

Diagrama entidad-relación de la base de datos



Detalle de relaciones en diagrama entidad-relación

Usuario → Áreas de Interés (AOI)
1:N (uno a muchos). Un *usuario* (auth_user) puede crear múltiples *AOI* (geo_aoi), pero cada AOI pertenece a un único usuario (geo_aoi.user_id → auth_user.id). Si se borra el usuario, se eliminan sus AOI.

Usuario → Suscripciones
1:N. Un *usuario* puede tener varias *suscripciones* a alertas (alerts_subscription.user_id → auth_user.id). Al eliminar el usuario, se eliminan sus suscripciones.

AOI → Suscripciones
1:N. Una *AOI* puede estar asociada a múltiples *suscripciones* (distintas especies/umbrales/frecuencias) (alerts_subscription.aoi_id → geo_aoi.id). Si se borra la AOI, se eliminan sus suscripciones.

Especie (Parámetros) → Suscripciones
1:N. Una entrada de *parámetros por especie* (core_species_param.species) puede ser objetivo de múltiples *suscripciones* (alerts_subscription.species). Actualizaciones de especie se propagan

Especie (Parámetros) → Ejecuciones PFZ (PFZ Run)
1:N. Una configuración de *parámetros por especie* puede originar múltiples *ejecuciones del modelo PFZ* (core_pfz_run.species → core_species_param.species).

Producto Satelital → Escenas L2
1:N. Una *fuente de producto satelital* (core_product_source) puede proveer múltiples *escenas de catálogo* L2 (catalog_scene.product_id → core_product_source.id) para variables como *SST* o *CHLA*.

Ejecución PFZ → Alertas
1:N. Una *ejecución PFZ* (core_pfz_run) puede dar lugar a múltiples *alertas* (alerts_alert.pfz_run_id → core_pfz_run.id), en función de cuántas suscripciones resulten disparadas. Restricción: no se permite borrar la ejecución si hay alertas que la referencian (ON DELETE RESTRICT).

Suscripción → Alertas
1:N. Una *suscripción* puede generar múltiples *alertas* a lo largo del tiempo (alerts_alert.subscription_id → alerts_subscription.id). Cascada: al eliminar la suscripción, se eliminan sus alertas.

Usuario → Tokens
1:N. Un *usuario* puede tener múltiples *tokens* activos (acceso/refresh/verificación) (auth_token.user_id → auth_user.id). Cascada: si se borra el usuario, se eliminan sus tokens.

Usuario ↔ Rol (vía User_Role)
M:N (muchos a muchos). Un *usuario* puede poseer varios *roles*, y un *rol* puede asignarse a múltiples *usuarios*. Se materializa con la tabla puente auth_user_role (user_id, role_id).

Creación de las tablas MySQL.

Creación de la base y configuración inicial

```
-- ============================================================
-- SIZPOPE - MySQL 8.0+ Schema (InnoDB, utf8mb4, Spatial)  --
-- ============================================================

CREATE DATABASE IF NOT EXISTS sizpope
  DEFAULT CHARACTER SET utf8mb4
  DEFAULT COLLATE utf8mb4_general_ci;

USE sizpope;

SET NAMES utf8mb4;
SET foreign_key_checks = 0;
```

Creación de las tablas

auth_user

Usuarios del sistema. Email único, hash de contraseña, estado (PENDING/ACTIVE/SUSPENDED), timestamps. Base para ownership (AOI, suscripciones), tokens y auditoría.

```
CREATE TABLE IF NOT EXISTS auth_user (
  id            CHAR(36)     NOT NULL PRIMARY KEY DEFAULT (UUID()),
  email         VARCHAR(320) NOT NULL,
  password_hash TEXT         NOT NULL,
  status        ENUM('PENDING','ACTIVE','SUSPENDED') NOT NULL,
  display_name  VARCHAR(255),
  created_at    DATETIME(6)  NOT NULL DEFAULT CURRENT_TIMESTAMP(6),
  updated_at    DATETIME(6)  NOT NULL DEFAULT CURRENT_TIMESTAMP(6) ON UPDATE
CURRENT_TIMESTAMP(6),
  UNIQUE KEY ux_auth_user_email (email)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

auth_role

Catálogo de roles (p. ej., ADMIN, OPERATOR, VIEWER). Código único y descripción. Soporta RBAC.

```
CREATE TABLE IF NOT EXISTS auth_role (
  id          CHAR(36)    NOT NULL PRIMARY KEY DEFAULT (UUID()),
  code        VARCHAR(64) NOT NULL,
  description VARCHAR(255),
  UNIQUE KEY ux_auth_role_code (code)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

auth_user_role

Tabla puente M:N entre usuarios y roles. PK compuesta (user_id, role_id).

```
CREATE TABLE IF NOT EXISTS auth_user_role (
  user_id CHAR(36) NOT NULL,
  role_id CHAR(36) NOT NULL,
  PRIMARY KEY (user_id, role_id),
  CONSTRAINT fk_ur_user FOREIGN KEY (user_id) REFERENCES auth_user(id) ON DELETE CASCADE,
  CONSTRAINT fk_ur_role FOREIGN KEY (role_id) REFERENCES auth_role(id) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

auth_token

Tokens asociados a usuarios (ACCESS/REFRESH/VERIFY_EMAIL) con expiración.

```
CREATE TABLE IF NOT EXISTS auth_token (
  id         CHAR(36)    NOT NULL PRIMARY KEY DEFAULT (UUID()),
  user_id    CHAR(36)    NOT NULL,
  issued_at  DATETIME(6) NOT NULL DEFAULT CURRENT_TIMESTAMP(6),
  expires_at DATETIME(6) NOT NULL,
  kind       ENUM('ACCESS','REFRESH','VERIFY_EMAIL') NOT NULL,
  KEY ix_token_user (user_id),
  CONSTRAINT fk_token_user FOREIGN KEY (user_id) REFERENCES auth_user(id) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

geo_aoi

Áreas de interés definidas por el usuario (POLYGON SRID 4326). Índice espacial en geom, bbox generado, ownership por user_id, soft delete (deleted_at).

```
CREATE TABLE IF NOT EXISTS geo_aoi (
```

```
  id          CHAR(36)     NOT NULL PRIMARY KEY DEFAULT (UUID()),
  user_id     CHAR(36)     NOT NULL,
  name        VARCHAR(200) NOT NULL,
  species     VARCHAR(120),
  geom        POLYGON      NOT NULL SRID 4326,
  bbox        POLYGON      /*!80018 AS (ST_Envelope(geom)) STORED */ SRID 4326,
  created_at  DATETIME(6)  NOT NULL DEFAULT CURRENT_TIMESTAMP(6),
  updated_at  DATETIME(6)  NOT NULL DEFAULT CURRENT_TIMESTAMP(6) ON UPDATE
CURRENT_TIMESTAMP(6),
  deleted_at  DATETIME(6)  NULL,
  SPATIAL KEY sidx_geo_aoi_geom (geom),
  KEY ix_geo_aoi_user (user_id),
  CONSTRAINT fk_geo_aoi_user FOREIGN KEY (user_id) REFERENCES auth_user(id) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

core_product_source

Fuentes de productos satelitales (plataforma, sensor, variables JSON, endpoint). Único por (platform, sensor, endpoint). Controla habilitación de ingestión.

```
CREATE TABLE IF NOT EXISTS core_product_source (
  id          CHAR(36)     NOT NULL PRIMARY KEY DEFAULT (UUID()),
  platform    VARCHAR(80)  NOT NULL,
  sensor      VARCHAR(80)  NOT NULL,
  variables   JSON         NOT NULL,          -- ["SST","CHLA"]
  endpoint    TEXT         NOT NULL,
  enabled     BOOLEAN      NOT NULL DEFAULT TRUE,
  created_at  DATETIME(6)  NOT NULL DEFAULT CURRENT_TIMESTAMP(6),
  updated_at  DATETIME(6)  NOT NULL DEFAULT CURRENT_TIMESTAMP(6) ON UPDATE
CURRENT_TIMESTAMP(6),
  UNIQUE KEY ux_core_product (platform, sensor, endpoint(255))
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

core_species_param

Parámetros por especie para el modelo PFZ (rangos SST/CHLA y pesos que suman 1). species único; validaciones con CHECK.

```
CREATE TABLE IF NOT EXISTS core_species_param (
```

```
  id          CHAR(36)     NOT NULL PRIMARY KEY DEFAULT (UUID()),
  species     VARCHAR(120) NOT NULL,
  sst_min     DECIMAL(5,2) NOT NULL,
  sst_max     DECIMAL(5,2) NOT NULL,
  chla_min    DECIMAL(6,3) NOT NULL,
  chla_max    DECIMAL(6,3) NOT NULL,
  weight_sst  DECIMAL(4,3) NOT NULL,
  weight_chla DECIMAL(4,3) NOT NULL,
  CONSTRAINT ck_w_between CHECK (weight_sst  >= 0 AND weight_sst  <= 1),
  CONSTRAINT ck_c_between CHECK (weight_chla >= 0 AND weight_chla <= 1),
  CONSTRAINT ck_w_sum     CHECK (ROUND(weight_sst + weight_chla,3) = 1.000),
  UNIQUE KEY ux_species (species)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

core_pfz_run

Ejecuciones del modelo PFZ (ventana temporal, raster_uri, calidad, bbox con índice espacial). Índices por run_time y species. FK a core_species_param (ON UPDATE CASCADE).

```
CREATE TABLE IF NOT EXISTS core_pfz_run (
  id           CHAR(36)     NOT NULL PRIMARY KEY DEFAULT (UUID()),
  species      VARCHAR(120) NOT NULL,
  window_start DATETIME(6)  NOT NULL,
  window_end   DATETIME(6)  NOT NULL,
  run_time     DATETIME(6)  NOT NULL DEFAULT CURRENT_TIMESTAMP(6),
  quality      DECIMAL(4,2),
  raster_uri   TEXT         NOT NULL,
  tile_cache_key VARCHAR(200),
  bbox         POLYGON      SRID 4326 NULL,
  created_at   DATETIME(6)  NOT NULL DEFAULT CURRENT_TIMESTAMP(6),
  KEY ix_pfz_run_time (run_time),
  KEY ix_pfz_species (species),
  SPATIAL KEY sidx_pfz_bbox (bbox),
  CONSTRAINT fk_pfz_species FOREIGN KEY (species) REFERENCES core_species_param(species) ON
UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

catalog_scene

Escenas L2 ingeridas (SST/CHLA) por producto y fecha. footprint POLYGON con índice espacial; índice por (var_name, acq_date). FK a core_product_source (RESTRICT en delete).

```
CREATE TABLE IF NOT EXISTS catalog_scene (
  id           CHAR(36)     NOT NULL PRIMARY KEY DEFAULT (UUID()),
  product_id   CHAR(36)     NOT NULL,
  var_name     ENUM('SST','CHLA') NOT NULL,
  acq_date     DATE         NOT NULL,
  acq_time     DATETIME(6)  NULL,
  footprint    POLYGON      NOT NULL SRID 4326,
  coverage_pct DECIMAL(5,2) CHECK (coverage_pct >= 0 AND coverage_pct <= 100),
  uri          TEXT         NOT NULL,
  checksum     VARCHAR(128),
  created_at   DATETIME(6)  NOT NULL DEFAULT CURRENT_TIMESTAMP(6),
  SPATIAL KEY sidx_scene_footprint (footprint),
  KEY ix_scene_var_date (var_name, acq_date),
  CONSTRAINT fk_scene_product FOREIGN KEY (product_id) REFERENCES core_product_source(id) ON DELETE
RESTRICT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

alerts_subscription

Suscripciones de usuario por AOI y especie (umbral, frecuencia, enabled). Único (user_id, aoi_id, species). FKs a auth_user, geo_aoi (CASCADE) y core_species_param (ON UPDATE CASCADE).

```
CREATE TABLE IF NOT EXISTS alerts_subscription (
  id        CHAR(36)     NOT NULL PRIMARY KEY DEFAULT (UUID()),
  user_id   CHAR(36)     NOT NULL,
  aoi_id    CHAR(36)     NOT NULL,
  species   VARCHAR(120) NOT NULL,
  threshold DECIMAL(4,2) NOT NULL,
  frequency VARCHAR(64)  NOT NULL,
  enabled   BOOLEAN      NOT NULL DEFAULT TRUE,
  created_at DATETIME(6) NOT NULL DEFAULT CURRENT_TIMESTAMP(6),
```

```
  updated_at DATETIME(6)  NOT NULL DEFAULT CURRENT_TIMESTAMP(6) ON UPDATE
CURRENT_TIMESTAMP(6),
  UNIQUE KEY ux_sub (user_id, aoi_id, species),
  KEY ix_sub_user (user_id),
  KEY ix_sub_aoi (aoi_id),
  CONSTRAINT ck_threshold CHECK (threshold >= 0 AND threshold <= 1),
  CONSTRAINT fk_sub_user    FOREIGN KEY (user_id) REFERENCES auth_user(id) ON DELETE CASCADE,
  CONSTRAINT fk_sub_aoi     FOREIGN KEY (aoi_id)  REFERENCES geo_aoi(id)   ON DELETE CASCADE,
  CONSTRAINT fk_sub_species FOREIGN KEY (species) REFERENCES core_species_param(species) ON
UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

alerts_alert

Alertas emitidas (valor PFZ, tiempos, canal, metadata JSON). FKs a alerts_subscription (CASCADE) y core_pfz_run (RESTRICT). Índices por (subscription_id, trigger_time) y pfz_run_id.

```
CREATE TABLE IF NOT EXISTS alerts_alert (
  id              CHAR(36)    NOT NULL PRIMARY KEY DEFAULT (UUID()),
  subscription_id CHAR(36)    NOT NULL,
  pfz_run_id      CHAR(36)    NOT NULL,
  pfz_value       DECIMAL(4,2) NOT NULL,
  trigger_time    DATETIME(6)  NOT NULL DEFAULT CURRENT_TIMESTAMP(6),
  delivered       BOOLEAN      NOT NULL DEFAULT FALSE,
  delivery_time   DATETIME(6)   NULL,
  channel         VARCHAR(32)   DEFAULT 'EMAIL',
  metadata        JSON          NULL,
  KEY ix_alert_sub_time (subscription_id, trigger_time),
  KEY ix_alert_run (pfz_run_id),
  CONSTRAINT fk_alert_sub FOREIGN KEY (subscription_id) REFERENCES alerts_subscription(id) ON DELETE
CASCADE,
  CONSTRAINT fk_alert_run FOREIGN KEY (pfz_run_id)      REFERENCES core_pfz_run(id)      ON DELETE
RESTRICT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

ops_audit_log

Bitácora/auditoría: tipo de evento, referencia, actor, timestamp y detalle JSON. Índices por (event_type, at_time) y actor_user. FK opcional al usuario (SET NULL).

```
CREATE TABLE IF NOT EXISTS ops_audit_log (
  id          BIGINT       NOT NULL PRIMARY KEY AUTO_INCREMENT,
  event_type  VARCHAR(80)  NOT NULL,
  ref_id      CHAR(36)     NULL,
  actor_user  CHAR(36)     NULL,
  at_time     DATETIME(6)  NOT NULL DEFAULT CURRENT_TIMESTAMP(6),
  detail      JSON         NULL,
  KEY ix_audit_event_time (event_type, at_time DESC),
  KEY ix_audit_actor (actor_user),
  CONSTRAINT fk_audit_actor FOREIGN KEY (actor_user) REFERENCES auth_user(id) ON DELETE SET NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

ops_job_run

Ejecuciones de jobs operativos (ingestiones, motor de alertas, etc.). Estado (OK/ERROR/RUNNING), métricas JSON y error. Índice por (job_name, started_at).

```
CREATE TABLE IF NOT EXISTS ops_job_run (
  id          CHAR(36)    NOT NULL PRIMARY KEY DEFAULT (UUID()),
  job_name    VARCHAR(80) NOT NULL,
  started_at  DATETIME(6) NOT NULL DEFAULT CURRENT_TIMESTAMP(6),
  finished_at DATETIME(6) NULL,
  status      ENUM('OK','ERROR','RUNNING') NOT NULL,
  metrics     JSON NULL,
  error_msg   TEXT NULL,
  KEY ix_job_name_time (job_name, started_at DESC)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Al terminar la creacion de las tablas se ejecuta esta sentencia para activae (habilitar) la validación y el enforcement de claves foráneas en la sesión actual.

```
SET foreign_key_checks = 1;
```

Inserción, consulta y borrado de registros.

Inserción de registros

```
SET NAMES utf8mb4;
USE sizpope;
SET foreign_key_checks = 0;

/* ================ AUTH ================ */
INSERT INTO auth_user (id, email, password_hash, status, display_name, created_at, updated_at) VALUES
('00000000-0000-0000-0000-000000000001', 'admin@sizpope.ar',  '$2y$10$hash_admin',  'ACTIVE',   'Admin
SIZPOPE',     NOW(), NOW()),
('00000000-0000-0000-0000-000000000002', 'ana@sizpope.ar',    '$2y$10$hash_ana',    'ACTIVE',   'Ana Analista',
NOW(), NOW()),
('00000000-0000-0000-0000-000000000003', 'oscar@sizpope.ar',  '$2y$10$hash_oscar',  'ACTIVE',   'Oscar
Operador',    NOW(), NOW()),
('00000000-0000-0000-0000-000000000004', 'vale@sizpope.ar',   '$2y$10$hash_vale',   'PENDING',  'Valeria
Observ.',   NOW(), NOW());

INSERT INTO auth_role (id, code, description) VALUES
('00000000-0000-0000-0000-00000000AA01', 'ADMIN',   'Administrador del sistema'),
('00000000-0000-0000-0000-00000000AA02', 'ANALYST', 'Analista (operaciones y modelos)'),
('00000000-0000-0000-0000-00000000AA03', 'VIEWER',  'Solo lectura');

INSERT INTO auth_user_role (user_id, role_id) VALUES
('00000000-0000-0000-0000-000000000001', '00000000-0000-0000-0000-00000000AA01'),
('00000000-0000-0000-0000-000000000001', '00000000-0000-0000-0000-00000000AA02'),
('00000000-0000-0000-0000-000000000002', '00000000-0000-0000-0000-00000000AA02'),
('00000000-0000-0000-0000-000000000003', '00000000-0000-0000-0000-00000000AA02'),
('00000000-0000-0000-0000-000000000004', '00000000-0000-0000-0000-00000000AA03');

INSERT INTO auth_token (id, user_id, issued_at, expires_at, kind) VALUES
(UUID(), '00000000-0000-0000-0000-000000000001', NOW(), DATE_ADD(NOW(), INTERVAL 7 DAY), 'ACCESS'),
(UUID(), '00000000-0000-0000-0000-000000000002', NOW(), DATE_ADD(NOW(), INTERVAL 7 DAY), 'ACCESS'),
(UUID(), '00000000-0000-0000-0000-000000000003', NOW(), DATE_ADD(NOW(), INTERVAL 30 DAY),
'REFRESH'),
(UUID(), '00000000-0000-0000-0000-000000000004', NOW(), DATE_ADD(NOW(), INTERVAL 1 DAY),
'VERIFY_EMAIL');
```

```
/* ================ GEO (AOI) ================ */
```

```
/* Polígonos simples (lon lat) SRID 4326 */
INSERT INTO geo_aoi (id, user_id, name, species, geom, created_at, updated_at) VALUES
('00000000-0000-0000-0000-00000000A001','00000000-0000-0000-0000-000000000002','Golfo San Jorge','Merluza',
  ST_GeomFromText('POLYGON((-67 -45,-65 -45,-65 -47,-67 -47,-67 -45))',4326), NOW(), NOW()),
('00000000-0000-0000-0000-00000000A002','00000000-0000-0000-0000-000000000003','Bahía Blanca','Langostino',
  ST_GeomFromText('POLYGON((-62 -39,-60 -39,-60 -40,-62 -40,-62 -39))',4326), NOW(), NOW()),
('00000000-0000-0000-0000-00000000A003','00000000-0000-0000-0000-000000000002','Costa Esquel','Calamar',
  ST_GeomFromText('POLYGON((-67 -42,-64 -42,-64 -44,-67 -44,-67 -42))',4326), NOW(), NOW()),
('00000000-0000-0000-0000-00000000A004','00000000-0000-0000-0000-000000000004','Rawson Norte','Merluza',
  ST_GeomFromText('POLYGON((-66 -43.2,-64.5 -43.2,-64.5 -44,-66 -44,-66 -43.2))',4326), NOW(), NOW());
```

```
/* =============== CORE: Product Sources =============== */
INSERT INTO core_product_source (id, platform, sensor, variables, endpoint, enabled, created_at, updated_at)
VALUES
('00000000-0000-0000-0000-00000000P001','Sentinel-3','OLCI',
JSON_ARRAY('SST','CHLA'),'https://s3.olci.endpoint', TRUE, NOW(), NOW()),
('00000000-0000-0000-0000-00000000P002','NOAA-20',  'VIIRS',
JSON_ARRAY('SST','CHLA'),'https://noaa20.viirs.endpoint', TRUE, NOW(), NOW()),
('00000000-0000-0000-0000-00000000P003','Terra',    'MODIS',
JSON_ARRAY('SST','CHLA'),'https://terra.modis.endpoint', TRUE, NOW(), NOW());
```

```
/* =============== CORE: Parámetros por Especie =============== */
INSERT INTO core_species_param (id, species, sst_min, sst_max, chla_min, chla_max, weight_sst, weight_chla)
VALUES
('00000000-0000-0000-0000-00000000S001','Merluza',     6.0, 14.0, 0.200, 3.500, 0.60, 0.40),
('00000000-0000-0000-0000-00000000S002','Calamar',     8.0, 16.0, 0.150, 2.500, 0.50, 0.50),
('00000000-0000-0000-0000-00000000S003','Langostino', 10.0, 20.0, 0.100, 1.800, 0.55, 0.45);
```

```
/* =============== CATALOG: Escenas L2 =============== */
INSERT INTO catalog_scene (id, product_id, var_name, acq_date, acq_time, footprint, coverage_pct, uri, checksum,
created_at) VALUES
(UUID(),'00000000-0000-0000-0000-00000000P001','SST','2025-10-20','2025-10-20 12:00:00',
  ST_GeomFromText('POLYGON((-67 -44,-64 -44,-64 -46,-67 -46,-67 -44))',4326), 82.5,
's3://bucket/s3_olci/sst_20251020.tif','chk1', NOW()),
(UUID(),'00000000-0000-0000-0000-00000000P001','CHLA','2025-10-20','2025-10-20 12:03:00',
```

```
  ST_GeomFromText('POLYGON((-67 -44,-64 -44,-64 -46,-67 -46,-67 -44))',4326), 79.2,
's3://bucket/s3_olci/chla_20251020.tif','chk2', NOW()),
(UUID(),'00000000-0000-0000-0000-00000000P002','SST','2025-10-21','2025-10-21 11:50:00',
  ST_GeomFromText('POLYGON((-66 -43,-63 -43,-63 -45,-66 -45,-66 -43))',4326), 76.1,
's3://bucket/noaa20_viirs/sst_20251021.tif','chk3', NOW()),
(UUID(),'00000000-0000-0000-0000-00000000P002','CHLA','2025-10-21','2025-10-21 11:53:30',
  ST_GeomFromText('POLYGON((-66 -43,-63 -43,-63 -45,-66 -45,-66 -43))',4326), 74.0,
's3://bucket/noaa20_viirs/chla_20251021.tif','chk4', NOW()),
(UUID(),'00000000-0000-0000-0000-00000000P003','SST','2025-10-22','2025-10-22 10:40:00',
  ST_GeomFromText('POLYGON((-65.5 -43.5,-62.8 -43.5,-62.8 -45,-65.5 -45,-65.5 -43.5))',4326), 70.0,
's3://bucket/terra_modis/sst_20251022.tif','chk5', NOW()),
(UUID(),'00000000-0000-0000-0000-00000000P003','CHLA','2025-10-22','2025-10-22 10:44:00',
  ST_GeomFromText('POLYGON((-65.5 -43.5,-62.8 -43.5,-62.8 -45,-65.5 -45,-65.5 -43.5))',4326), 69.3,
's3://bucket/terra_modis/chla_20251022.tif','chk6', NOW());
```

```
/* =============== CORE: PFZ Runs =============== */
INSERT INTO core_pfz_run (id, species, window_start, window_end, run_time, quality, raster_uri, tile_cache_key,
bbox, created_at) VALUES
('00000000-0000-0000-0000-00000000R001','Merluza',   '2025-10-20 00:00:00','2025-10-20 23:59:59','2025-10-20
13:00:00', 0.92,
 's3://bucket/pfz/merluza_20251020.tif','pfz_mer_20251020',
 ST_GeomFromText('POLYGON((-67 -44,-64 -44,-64 -46,-67 -46,-67 -44))',4326), NOW()),
('00000000-0000-0000-0000-00000000R002','Calamar',   '2025-10-21 00:00:00','2025-10-21 23:59:59','2025-10-21
12:10:00', 0.88,
 's3://bucket/pfz/calamar_20251021.tif','pfz_cal_20251021',
 ST_GeomFromText('POLYGON((-66 -43,-63 -43,-63 -45,-66 -45,-66 -43))',4326), NOW()),
('00000000-0000-0000-0000-00000000R003','Langostino','2025-10-22 00:00:00','2025-10-22 23:59:59','2025-10-22
11:10:00', 0.90,
 's3://bucket/pfz/langostino_20251022.tif','pfz_lan_20251022',
 ST_GeomFromText('POLYGON((-65.5 -43.5,-62.8 -43.5,-62.8 -45,-65.5 -45,-65.5 -43.5))',4326), NOW());
```

```
/* =============== ALERTS: Subscriptions =============== */
INSERT INTO alerts_subscription (id, user_id, aoi_id, species, threshold, frequency, enabled, created_at, updated_at)
VALUES
('00000000-0000-0000-0000-00000000B001','00000000-0000-0000-0000-000000000002','00000000-0000-0000-
0000-00000000A001','Merluza',    0.70,'DAILY@09:00Z', TRUE, NOW(), NOW()),
```

```
('00000000-0000-0000-0000-00000000B002','00000000-0000-0000-0000-000000000003','00000000-0000-0000-0000-00000000A002','Langostino', 0.65,'DAILY@09:00Z', TRUE, NOW(), NOW()),
('00000000-0000-0000-0000-00000000B003','00000000-0000-0000-0000-000000000002','00000000-0000-0000-0000-00000000A003','Calamar',    0.60,'DAILY@12:00Z', TRUE, NOW(), NOW()),
('00000000-0000-0000-0000-00000000B004','00000000-0000-0000-0000-000000000004','00000000-0000-0000-0000-00000000A004','Merluza',    0.75,'DAILY@09:00Z', TRUE, NOW(), NOW()),
('00000000-0000-0000-0000-00000000B005','00000000-0000-0000-0000-000000000003','00000000-0000-0000-0000-00000000A001','Merluza',    0.80,'DAILY@18:00Z', TRUE, NOW(), NOW());
```

```
/* =============== ALERTS: Alerts (disparadas por PFZ runs) =============== */
INSERT INTO alerts_alert (id, subscription_id, pfz_run_id, pfz_value, trigger_time, delivered, delivery_time, channel, metadata) VALUES
('00000000-0000-0000-0000-00000000L001','00000000-0000-0000-0000-00000000B001','00000000-0000-0000-0000-00000000R001',0.78,'2025-10-20 13:20:00',TRUE, '2025-10-20 13:20:05','EMAIL', JSON_OBJECT('aoi','Golfo SJ')),
('00000000-0000-0000-0000-00000000L002','00000000-0000-0000-0000-00000000B005','00000000-0000-0000-0000-00000000R001',0.81,'2025-10-20 13:21:00',TRUE, '2025-10-20 13:21:04','EMAIL', JSON_OBJECT('aoi','Golfo SJ')),
('00000000-0000-0000-0000-00000000L003','00000000-0000-0000-0000-00000000B003','00000000-0000-0000-0000-00000000R002',0.69,'2025-10-21 12:20:00',FALSE, NULL,'EMAIL', JSON_OBJECT('aoi','Costa Esquel')),
('00000000-0000-0000-0000-00000000L004','00000000-0000-0000-0000-00000000B002','00000000-0000-0000-0000-00000000R003',0.77,'2025-10-22 11:30:00',TRUE, '2025-10-22 11:30:06','EMAIL', JSON_OBJECT('aoi','Bahía Blanca')),
('00000000-0000-0000-0000-00000000L005','00000000-0000-0000-0000-00000000B004','00000000-0000-0000-0000-00000000R001',0.83,'2025-10-20 13:25:00',TRUE, '2025-10-20 13:25:03','EMAIL',
JSON_OBJECT('aoi','Rawson Norte')),
('00000000-0000-0000-0000-00000000L006','00000000-0000-0000-0000-00000000B001','00000000-0000-0000-0000-00000000R002',0.62,'2025-10-21 12:35:00',FALSE, NULL,'EMAIL', JSON_OBJECT('retry',true));
```

```
/* =============== OPS: Auditoría y Jobs =============== */
INSERT INTO ops_audit_log (event_type, ref_id, actor_user, at_time, detail) VALUES
('register','00000000-0000-0000-0000-000000000001','00000000-0000-0000-0000-000000000001','2025-10-19 09:00:00', JSON_OBJECT('email','admin@sizpope.ar')),
('aoi_create','00000000-0000-0000-0000-00000000A001','00000000-0000-0000-0000-000000000002','2025-10-20 09:30:00', JSON_OBJECT('name','Golfo San Jorge')),
('pfz_run','00000000-0000-0000-0000-00000000R001','00000000-0000-0000-0000-000000000001','2025-10-20 13:00:00', JSON_OBJECT('species','Merluza')),
```

```
('alerts_sent','00000000-0000-0000-0000-00000000R001','00000000-0000-0000-0000-000000000001','2025-10-20 13:26:00', JSON_OBJECT('count',3)),
('pfz_run','00000000-0000-0000-0000-00000000R002','00000000-0000-0000-0000-000000000001','2025-10-21 12:10:00', JSON_OBJECT('species','Calamar')),
('alerts_sent','00000000-0000-0000-0000-00000000R003','00000000-0000-0000-0000-000000000001','2025-10-22 11:40:00', JSON_OBJECT('count',1));

INSERT INTO ops_job_run (id, job_name, started_at, finished_at, status, metrics, error_msg) VALUES
(UUID(),'ingest_sst','2025-10-20 11:50:00','2025-10-20 12:10:00','OK',    JSON_OBJECT('files',2,'bytes',104857600), NULL),
(UUID(),'ingest_chla','2025-10-21 11:45:00','2025-10-21 12:05:00','OK',   JSON_OBJECT('files',2,'bytes',73400320), NULL),
(UUID(),'alert_engine','2025-10-20 13:18:00','2025-10-20 13:26:00','OK',  JSON_OBJECT('alerts',3), NULL);

SET foreign_key_checks = 1;
```

Consulta de registros

Últimas alertas por usuario/AOI/especie (estado más reciente):

Devuelve, por suscripción activa, la última alerta emitida (si existe) y su estado.

```
/* Parámetros
   SET @p_user_id = '00000000-0000-0000-0000-000000000002'; -- opción: limitar a un usuario
*/
SELECT
  s.id                AS subscription_id,
  u.email               AS user_email,
  aoi.name              AS aoi_name,
  s.species,
  s.threshold,
  s.frequency,
  s.enabled,
  al.id               AS last_alert_id,
  al.pfz_value,
  al.trigger_time,
  al.delivered,
```

```
  al.channel
FROM alerts_subscription s
JOIN auth_user u      ON u.id  = s.user_id
JOIN geo_aoi aoi      ON aoi.id = s.aoi_id
LEFT JOIN alerts_alert al
  ON al.subscription_id = s.id
  AND al.trigger_time = (
    SELECT MAX(al2.trigger_time)
    FROM alerts_alert al2
    WHERE al2.subscription_id = s.id
  )
WHERE (@p_user_id IS NULL OR s.user_id = @p_user_id)
ORDER BY u.email, aoi.name, s.species;
```

Cobertura de escenas L2 por producto/variable en un rango de fechas:

Resumen operativo de disponibilidad (cantidad y cobertura promedio) por fuente y variable.

```
/* Parámetros
   SET @p_from = DATE('2025-10-20');
   SET @p_to   = DATE('2025-10-22');
*/
SELECT
  ps.platform,
  ps.sensor,
  cs.var_name,
  COUNT(*)                        AS scenes_count,
  ROUND(AVG(cs.coverage_pct), 2)      AS avg_coverage_pct,
  MIN(cs.acq_date)                AS first_date,
  MAX(cs.acq_date)                 AS last_date
FROM catalog_scene cs
JOIN core_product_source ps ON ps.id = cs.product_id
WHERE cs.acq_date BETWEEN @p_from AND @p_to
GROUP BY ps.platform, ps.sensor, cs.var_name
ORDER BY ps.platform, ps.sensor, cs.var_name;
```

Detección espacial: PFZ runs que intersectan con AOIs del usuario y superan umbral de su suscripción

Encuentra ejecuciones PFZ que caen en el rango de tiempo y tocan el AOI del usuario (usando ST_Intersects sobre bbox del PFZ y geom del AOI). Útil para validar por qué se disparó una alerta.

```
/* Parámetros
   SET @p_user_id = '00000000-0000-0000-0000-000000000002';
   SET @p_from_ts = TIMESTAMP('2025-10-20 00:00:00');
   SET @p_to_ts   = TIMESTAMP('2025-10-22 23:59:59');
*/
SELECT
  u.email                    AS user_email,
  aoi.name                   AS aoi_name,
  s.species,
  pr.id                      AS pfz_run_id,
  pr.run_time,
  pr.window_start, pr.window_end,
  pr.raster_uri,
  pr.quality,
  s.threshold,
  ST_AsText(aoi.geom)        AS aoi_wkt,
  ST_AsText(pr.bbox)         AS run_bbox_wkt
FROM alerts_subscription s
JOIN auth_user u     ON u.id     = s.user_id
JOIN geo_aoi aoi     ON aoi.id   = s.aoi_id
JOIN core_pfz_run pr ON pr.species = s.species
WHERE s.enabled = TRUE
  AND s.user_id = @p_user_id
  AND pr.run_time BETWEEN @p_from_ts AND @p_to_ts
  AND pr.bbox IS NOT NULL
  AND ST_Intersects(pr.bbox, aoi.geom) = 1
ORDER BY pr.run_time DESC, aoi.name;
```

Borrado de registros

```
USE sizpope;

DELETE FROM alerts_alert;
DELETE FROM alerts_subscription;

DELETE FROM catalog_scene;

DELETE FROM core_pfz_run;
DELETE FROM core_species_param;
DELETE FROM core_product_source;

DELETE FROM geo_aoi;

DELETE FROM ops_audit_log;
DELETE FROM ops_job_run;

DELETE FROM auth_user_role;
DELETE FROM auth_token;
DELETE FROM auth_role;
DELETE FROM auth_user;
```

Fundamentos de Codificación y Diseño Orientado a Objetos (POO)

El **Sistema de Identificación de Zonas Potenciales de Pesca (SIZPOPE)** ha sido diseñado e implementado en **Java** con un enfoque riguroso de **Programación Orientada a Objetos (POO)**, garantizando una arquitectura modular, escalable y mantenible para un entorno de misión crítica. La POO se materializa en una estructura de clases y objetos que encapsulan datos y comportamiento del dominio (usuarios, áreas de interés, parámetros por especie, ejecuciones PFZ, suscripciones y alertas), fomentando la reutilización y la claridad del diseño.

Los principios de **encapsulamiento, herencia, polimorfismo y abstracción** se aplican para modelar y orquestar los casos de uso de SIZPOPE. En particular, las **entidades** (User, AOI, SpeciesParam, PFZRun, Subscription, Alert, ProductSource) representan el estado del dominio; las **clases de control** (AuthService, AOIService, PFZModelService, MapPublisher, AlertEngine, ExportService) coordinan la lógica de negocio y las reglas; y los **objetos de frontera** (UserRepository, SceneRepository, PFZRunRepository, WMTSRegistry, TileCache, RemoteCatalog, StorageService) aíslan la infraestructura (persistencia, catálogos remotos, publicación de mosaicos/tiles y caché). Este reparto de responsabilidades asegura **alta cohesión** y **bajo**

acoplamiento, permitiendo, por ejemplo, habilitar una nueva fuente satelital o cambiar el backend de mapas sin afectar el núcleo de negocio.

Además de los principios de la POO, SIZPOPE se beneficia de capacidades intrínsecas del ecosistema Java:

Portabilidad (WORA): la ejecución en la **JVM** permite desplegar SIZPOPE en múltiples plataformas (desde servidores on-premise hasta entornos distribuidos en la nube) sin cambios en el código.

Gestión automática de memoria (GC): reduce el riesgo de fugas y simplifica el manejo de estructuras volumétricas (rasters, colecciones de escenas), aumentando la estabilidad del sistema.

Robustez y manejo de excepciones: el tipado estático y el modelo de excepciones facilitan la contención de fallos (E/S, red, tiempo de espera de catálogos remotos) y su trazabilidad vía AuditLog.

Multihilo (multithreading): es clave para la **ingestión concurrente** de productos L2 (SST/CHL-a), el **cálculo PFZ** en paralelo por ventana temporal/especie, el **precalentamiento de tiles** y el **disparo de alertas**, todo sin bloquear la interacción del usuario.

Ecosistema y bibliotecas: el acceso a APIs maduras (JDBC, HTTP, JSON, librerías geoespaciales y ráster) acelera la conectividad con bases de datos, almacenamiento de objetos y motores de mosaicos, además de soportar algoritmos para detección de frentes térmicos e integración SST/Chl-a.

En cuanto a **tipos de datos y modelado**, se emplean tipos primitivos (p. ej., boolean para flags como enabled, int/double para umbrales y métricas) y **tipos de referencia** para entidades ricas (String, List, Map, y clases de dominio como AOI o PFZRun). La creación y uso de objetos en la **capa de presentación** (p. ej., WebUI, MapGateway) y en la **capa de negocio** (p. ej., new PFZModelService(...), new AOIService(...)) garantiza que los controladores operen siempre con modelos coherentes y validados (por ejemplo, AOI con geometría en SRID 4326 y Subscription con threshold $\in [0,1]$). Los **constructores parametrizados** (p. ej., public Subscription(UUID userId, UUID aoiId, String species, double threshold, String frequency)) obligan a inicializar cada objeto en un estado válido desde su creación, previniendo inconsistencias y mejorando la legibilidad.

Las **estructuras de control** dan forma al flujo de la aplicación. Las condicionales if/else se emplean para validar credenciales en AuthService, verificar permisos por rol (RBAC) en UserAdminService, y aplicar reglas de negocio (p. ej., determinar si un PFZRun califica para notificaciones según umbral y AOI). Los bucles for/while se usan para **iterar resultados** de catálogos remotos durante la ingestión (RemoteCatalog.listL2(...)), procesar **tiles** a diferentes niveles de zoom en MapPublisher/TileCache, recorrer **suscripciones activas** en AlertEngine y generar exportaciones en lotes en ExportService. La elección entre for o while responde a si el número de iteraciones es conocido (p. ej., niveles de zoom) o depende de una condición (p. ej., lectura de streams o cursores paginados).

La interfaz principal se ofrece mediante un **visor web GIS** (Dashboard), punto de acceso a las funciones clave: gestión de usuarios y roles (panel de administración), alta/edición de **Áreas de Interés (AOI)**, configuración de **suscripciones** por especie/umbral/frecuencia, visualización de **capas** (PFZ, SST, Chl-a) vía WMTSRegistry/TileCache, **inspección de celdas** con metadatos y **descarga/exportación** en formatos

cartográficos estándar. Cada acción de la UI dispara métodos de los controladores que encapsulan la lógica de negocio (por ejemplo, al habilitar la capa PFZ en el mapa: MapGateway.enableLayer("PFZ") → WMTSRegistry.resolve("PFZ") → solicitud de tiles a TileCache). Esta **separación de responsabilidades** permite evolucionar la interfaz (estilos, widgets, flujos de usuario) sin comprometer los contratos de servicio ni la persistencia.

En síntesis, la adopción disciplinada de POO sobre Java, sumada a una **arquitectura en capas** con entidades, controles y fronteras bien definidas, ha permitido construir **SIZPOPE** como un sistema **robusto, eficiente y escalable**, preparado para incorporar nuevas especies, sensores satelitales y estilos cartográficos, manteniendo la confiabilidad de la información y la trazabilidad operativa en todo el flujo: ingestión → modelado PFZ → publicación → suscripciones/alertas → análisis y exportación.

A eso se llega con una aplicación correcta del **modelo Vista–Controlador (MVC)**, que en la siguiente tabla queda expresada para **SIZPOPE**:

Esta capa adicional explicita la frontera con infraestructura geoespacial y catálogos satelitales, habitual en SIZPOPE.

Capa	Paquete	Contenido
Presentación (UI)	sizpope.ui	WebApp, Dashboard, MapViewer (visor GIS), LoginView, AOIForm, SubscriptionForm, HistoryView, ExportDialog.
Negocio (Business)	sizpope.control	AuthService, AOIService, PFZModelService, MapPublisher, IdentifyService, AlertEngine, ExportService, UserAdminService, ProductService.
Modelo (Domain)	sizpope.domain	Entidades puras: User, AOI, SpeciesParam, PFZRun, Subscription, Alert, ProductSource, Scene.
Datos (Persistence)	sizpope.persistence	Repositorios/DAOs: UserRepository, AOIRepository, PFZRunRepository, SceneRepository, SpeciesParamRepository, SubscriptionRepository, AlertRepository, AuditLogRepository.
Integración (Infra/Gateways) *	sizpope.integration	Conectores y servicios externos: RemoteCatalog, TileCache, WMTSRegistry, StorageService, TokenService, MailService, ConnectorService, Clipper.

Esta arquitectura se adhiere rigurosamente al **principio de separación de responsabilidades**, fundamental en sistemas robustos y escalables. Mantiene una división clara entre presentación**,** lógica de negocio**,** modelo **e** infraestructura**.**

Capa de Presentación (UI): incluye el dashboard, formularios (alta/edición de AOI y suscripciones), el visor de mapas y los componentes de interacción (botones, filtros, leyendas). Su función es mostrar información y capturar entradas (p. ej., habilitar capa PFZ, dibujar un polígono AOI, descargar un GeoTIFF).

Capa de Lógica de Negocio (Controladores/Servicios): contiene las reglas y procesos que responden a las acciones del usuario: alta/validación de usuarios, CRUD de AOIs, ingestión y modelado PFZ, publicación de tiles, disparo de alertas, exportaciones. Aquí se implementan las operaciones CRUD y las validaciones de dominio (umbral $\in [0,1]$, geometrías SRID 4326, especies válidas).

Capa de Modelo (Dominio): define entidades inmutables o validadas y sus invariantes, sin dependencias de UI o infraestructura. Sirve de contrato estable entre controladores y persistencia.

Capa de Datos (Persistencia): encapsula acceso a base de datos y almacenes, mapea entidades a tablas y gestiona transacciones. Un cambio de motor (p. ej., ajustes en MySQL**)** no afecta a la lógica de negocio.

Capa de Integración (Gateways/Infra): desacopla catálogos satelitales, publicación WMTS y caché de tiles. Permite sustituir proveedores o endpoints sin modificar casos de uso.

Beneficios prácticos:

- **Facilidad de desarrollo:** UI y negocio evolucionan en paralelo. Cambios de estilo/mapa (Leaflet/OpenLayers) no tocan PFZModelService, y una optimización en TileCache no afecta a Dashboard.
- **Mantenimiento simplificado:** los defectos se localizan por capa. Si falla la eliminación de un AOI, se revisa AOIService.deleteAOI(id) y AOIRepository.softDelete(id); si cambia el texto de un botón, la modificación es solo en sizpope.ui.
- **Reusabilidad:** los métodos de servicio (p. ej., ExportService.export(...)) se reutilizan desde UI y desde tareas programadas, evitando duplicación y garantizando comportamientos consistentes.
- **Escalabilidad:** la segmentación por capas permite añadir nuevas fuentes satelitales (ProductService + ConnectorService) o nuevas especies (SpeciesParam) con impacto mínimo fuera de su capa.

En resumen, la **interacción directa** de la UI con servicios bien definidos y la **separación estricta de capas** son pilares que hacen a SIZPOPE eficiente, mantenible y listo para crecer (nuevas variables, sensores o estilos cartográficos) sin comprometer la estabilidad del núcleo del sistema.

Tratamiento y Manejo de Excepciones

El proyecto **SIZPOPE** implementa un **manejo robusto y proactivo de excepciones**, clave para la estabilidad operativa, la integridad de datos y la conformidad con los RNF (disponibilidad, seguridad, rendimiento y usabilidad). Este enfoque produce una UX predecible y un sistema resiliente frente a contingencias.

Capa de Datos (Persistencia: Repositorios/DAO)

Componentes: UserRepository, AOIRepository, PFZRunRepository, SceneRepository, SubscriptionRepository, AlertRepository, AuditLogRepository.

Tecnología: JDBC/JPA + *connection pool* (p. ej., HikariCP), MySQL/MariaDB.

Excepciones típicas y tratamiento

- SQLException / SQLIntegrityConstraintViolationException: violaciones FK/UK, consultas mal formadas, *deadlocks*.
  Acción: *rollback* transaccional, *retry* limitado con *backoff* si es seguro (idempotente), mapeo a error de dominio (p. ej., DuplicateEmailException), registro en ops_audit_log.
- DataAccessException (envoltorio propio): unifica errores JDBC y los estandariza para la capa de negocio.
- ClassNotFoundException (driver JDBC ausente) / ConfigurationException: durante *bootstrap*.
  Acción: fallo temprano controlado, *health check* en "rojo", alerta operativa.
- Validaciones espaciales (SRID/geom inválidas):
  Acción: rechazar con DomainValidationException y detalle (AOI inválida), sin cambiar estado.

Patrones

- Transacciones ACID en operaciones críticas (CU-01/03/04/05/18/19).
- *Retry* condicionado (solo lecturas o escrituras idempotentes).
- *Fail fast* en *bootstrap* si falta el driver/config.

Capa de Integración (Infra/Gateways)

Componentes: RemoteCatalog, StorageService, WMTSRegistry, TileCache, TokenService, MailService, ConnectorService, Clipper.

Excepciones típicas y tratamiento

- SocketTimeoutException / HttpClientErrorException / HttpServerErrorException: fallas en catálogos satelitales o publicación WMTS.
  Acción: *retry* con backoff, *circuit breaker* si persiste, *queue* diferida (CU-06/07/09).
- IOException (E/S en almacenamiento de objetos):
  Acción: reintento, validación de integridad (checksum), limpieza de artefactos parciales.
- JsonProcessingException (parsing de metadatos):
  Acción: descartar escena problemática y continuar lote; registrar *diagnóstico*.

Patrones

- Idempotencia en descargas/publicación (marcas por runId / tile_cache_key).
- *Bulkhead* para aislar hilos de ingestión/modelado de la UI.

Capa de Negocio (Servicios/Controladores de CU)

Componentes: AuthService, AOIService, PFZModelService, MapPublisher, IdentifyService, AlertEngine, ExportService, UserAdminService, ProductService, HistoryService.

Excepciones típicas y tratamiento

- DomainValidationException: reglas del dominio (umbral $\in [0,1]$, AOI vacía, especie desconocida).
  Acción: respuesta 400 (HTTP) con mensajes accionables; no mutar estado.

- AuthorizationException / AccessDeniedException (RBAC):
  Acción: 403; auditar intento con AuditLog.
- ModelComputationException (PFZ): raster faltante, parámetros inconsistentes.
  Acción: marcar *run* fallido, no publicar capas; reintento programado.
- ExportException: proyección/clip/serialización.
  Acción: cancelar export, ofrecer reintento; conservar *logs* técnicos.

Patrones

- Mapeo de excepciones a **código de error estándar** y **mensaje de negocio**.
- *Compensating actions* (p. ej., si falla publicación tras generar tiles, invalidar cache parcial).

Capa de Presentación (UI Web / API)

Componentes: WebApp/Dashboard, MapViewer, LoginView, AOIForm, SubscriptionForm, HistoryView, ExportDialog, API REST.

Propagación y manejo

- **Normalización de errores** desde servicios a API (p. ej., Problem+JSON): status, code, message, correlationId.
- **Notificaciones UI**: *toast/banners* con mensaje claro y acción sugerida (reintentar, editar AOI, revisar credenciales).
- **Flujos alternativos vinculados a CU**:
  - **CU-01 Registrar Usuario**: si DuplicateEmailException → informar email en uso y sugerir recuperación.
  - **CU-04 Editar AOI**: si DomainValidationException → resaltar geometría inválida y bloquear "Guardar".
  - **CU-09 Publicar tiles**: si WMTSRegistry falla → aviso de demora y reintento automático en *background*.
  - **CU-19 Exportar PFZ**: si ExportException → ofrecer nuevo formato/CRS y reintentar.

Patrones

- Mensajes **no técnicos** para el usuario; detalles técnicos sólo en *logs* y correlationId.
- Desactivar acciones mientras hay reintentos; *spinners* y estados vacíos informativos.
-

Con este diseño, SIZPOPE **falla de forma controlada**: captura, clasifica, registra, informa y—cuando es seguro—**recupera** mediante reintentos e idempotencia. Así sostiene la **continuidad operativa**, preserva la **integridad** y mantiene una **experiencia de usuario predecible**, alineada con los RNF y los flujos alternativos de los casos de uso.

Aplicación de los Pilares de POO

El diseño arquitectónico del **Sistema de Identificación de Zonas Potenciales de Pesca (SIZPOPE)** se apoya en los cuatro principios esenciales de la Programación Orientada a Objetos (POO). Su aplicación disciplinada garantiza modularidad**,** mantenibilidad **y** escalabilidad a largo plazo. Estos pilares no solo estructuran el código, también facilitan el trabajo colaborativo y la adaptación a nuevos sensores, especies o estilos cartográficos.

Encapsulamiento

Se aplica mediante **modificadores de acceso** y **APIs explícitas**. Los atributos del modelo permanecen privados y solo se exponen a través de métodos controlados, asegurando que toda mutación respete las invariantes de dominio.

- Ejemplos:
    - En User, campos sensibles como passwordHash y roles son privados; su actualización pasa por validaciones de políticas (complejidad de password, RBAC).
    - En AOI, la geometría (geom) se encapsula para garantizar **SRID=4326** y evitar polígonos inválidos; métodos de fábrica validan y normalizan el WKT/GeoJSON.
    - En Subscription, threshold se restringe a [0,1] y species debe existir en SpeciesParam; los *setters* rechazan estados inconsistentes.

Este enfoque evita estados inválidos, reduce superficie de error y endurece la seguridad del sistema.

Abstracción

SIZPOPE separa responsabilidades por capas y **oculta detalles de infraestructura** tras interfaces estables.

- **Negocio vs. Datos**: AOIService.updateAOI(...) abstrae la persistencia real (AOIRepository). Quien usa el servicio **no necesita saber** si se escribe en MySQL/MariaDB, si hay *pooling* o si se usa caché.
- **Cómputo vs. Fuentes**: PFZModelService.compute(...) consume SceneRepository y SpeciesParamRepository sin conocer cómo se descargan o indexan las escenas (esa complejidad vive en RemoteCatalog, StorageService).
- **Mapeo vs. Publicación**: MapPublisher.publish(runId, ...) trabaja contra TileCache/WMTSRegistry con un contrato simple; el backend (WMTS, CDN, estilos) es intercambiable.

La abstracción permite sustituir proveedores satelitales, motores de tiles o almacenes sin tocar la lógica de negocio.

Herencia (propuesta de diseño)

Se utiliza con moderación, privilegiando composición e interfaces. Aun así, hay escenarios donde una jerarquía aporta claridad y reutilización.

- **Observación satelital**
  Clase base Scene con atributos comunes: sceneId, acqTime, footprint, coverage, uri.
  Subclases especializadas:
    - SSTScene (derivada de Scene) con metadatos térmicos específicos.

    - ChlaScene (derivada de Scene) con metadatos bio-ópticos.
      Esto evita duplicación y permite extender a nuevas variables (p. ej., TurbidityScene) sin romper el contrato.
- **Resultado de modelo**
  Clase base ModelRun con runId, ventana temporal y bbox.
  Subclase PFZRun agrega métricas y *artefactos* (p. ej., rasterUri, tileCacheKey).
  De ser necesario, futuras variantes (p. ej., un modelo alternativo de idoneidad) heredarían la base común.

Polimorfismo (propuesta de diseño)

Permite tratar objetos distintos bajo un **contrato común**, facilitando componentes reutilizables.

- **Estrategias de cómputo PFZ**
  Interfaz ComputePFZStrategy con compute(sstRaster, chlaRaster, params).
  Implementaciones:
    - WeightedWindowStrategy (promedios ponderados por especie).
    - FrontDetectionStrategy (realza frentes térmicos antes de combinar).
      PFZModelService selecciona la estrategia adecuada sin *ifs* dispersos; nuevas estrategias se agregan sin modificar el cliente (OCP).
- **Visualización/Identificación**
  Interfaz Inspectable con LatLon sampleAt(lat, lon, date) o MapFeature toFeature().
  Tanto PFZRun como Scene pueden implementarla: el **visor** consume Inspectable para "inspeccionar celda" sin saber si el origen es PFZ, SST o Chl-a.
- **Renderización en el mapa**
  Interfaz RenderableLayer con wmtsLayerDescriptor().
  Capas PFZLayer, SSTLayer, ChlaLayer proveen su descriptor; MapGateway las trata uniformemente al habilitar/ocultar.

Con estas aplicaciones de los pilares, SIZPOPE mantiene **alta cohesión**, **bajo acoplamiento** y puntos de extensión claros: incorporar una nueva especie, sensor o algoritmo no implica reescribir el sistema, solo agregar o sustituir implementaciones detrás de interfaces bien definidas.

Uso de Estructuras de Datos Avanzadas

Si bien las colecciones básicas de Java como java.util.List y java.util.ArrayList son herramientas fundamentales y ampliamente utilizadas para la gestión de datos comunes como listas de usuarios y eventos en muchas secciones del sistema, la complejidad y los requisitos específicos de ciertos módulos demandan la implementación de estructuras de datos y algoritmos más sofisticados para optimizar el rendimiento, la eficiencia y la funcionalidad. A continuación, se detallan las estructuras y algoritmos específicos empleados en módulos clave:
Cola (Queue): Esta estructura de datos es indispensable en el Módulo de Notificación, donde se 91 encarga de la gestión eficiente y ordenada de los envíos salientes. Los reportes validados, que requieren ser procesados y enviados a los destinatarios pertinentes, se colocan en una Cola FIFO (First-In, First-Out). Esta elección asegura que las alertas se despachen exactamente en el orden en que fueron confirmadas,

garantizando una comunicación consistente y sin saltos temporales. La implementación de una cola asegura que no haya sobrecarga del sistema de envío y que cada notificación sea manejada de manera individual y secuencial.

Lista Enlazada (LinkedList): La flexibilidad y eficiencia de las Listas Enlazadas son aprovechadas internamente en la implementación del Módulo GIS (Sistema de Información Geográfica). Su uso principal reside en la gestión dinámica de la superposición de capas en el mapa. Una LinkedList permite añadir, eliminar o reordenar capas de manera eficiente sin la necesidad de reestructurar grandes bloques de memoria, lo cual es crucial en entornos gráficos donde las capas pueden activarse o desactivarse constantemente por el usuario o por lógicas internas del sistema, optimizando la visualización y el rendimiento interactivo del mapa.

Algoritmos de Búsqueda: Para asegurar una recuperación de datos rápida y eficiente, el sistema incorpora dos tipos principales de algoritmos de búsqueda: ○ Búsqueda Secuencial: Este algoritmo se utiliza en los controladores para localizar usuarios por ID o filtrar eventos por fecha en las consultas a la base de datos cuando los datos no garantizan un ordenamiento previo o el volumen de datos es manejable. Aunque menos eficiente para grandes volúmenes de datos, es robusto y simple de implementar. ○ Búsqueda Binaria: En situaciones donde los datos pueden garantizarse como ordenados (por ejemplo, por ID de usuario o fecha de evento), se emplea la búsqueda binaria. Este algoritmo reduce significativamente el tiempo de búsqueda al dividir repetidamente el espacio de búsqueda por la mitad, lo que resulta en una eficiencia logarítmica (O(log n)), esencial para bases de datos con un gran número de registros y para responder rápidamente a las consultas del usuario.

Algoritmos de Ordenación: La presentación de datos al usuario final a menudo requiere una ordenación específica para facilitar la comprensión y el análisis. Para esto, se consideran algoritmos de ordenación avanzados:

QuickSort o MergeSort: Estos algoritmos se pueden utilizar para ordenar los resultados del histórico de alertas (correspondiente al Caso de Uso CU-11) antes de mostrarlos al usuario. La ordenación puede realizarse por criterios como la fecha de la alerta (ascendente o descendente) o por la severidad (crítica, alta, media, baja). La elección entre QuickSort o MergeSort dependerá de factores como el tamaño de los datos, la estabilidad requerida (MergeSort es estable, QuickSort no siempre lo es) y el entorno de ejecución, pero ambos ofrecen un rendimiento promedio de O(n log n), lo que los hace ideales para conjuntos de datos grandes. La implementación de estos algoritmos garantiza que el usuario siempre vea la información más relevante y organizada de manera intuitiva.

Repositorio:

url: https://github.com/sebasanfilippo/Seminario-Practica-2025

Bibliografía

- **Siglo XXI.** (2025). Modulo 1-4 - Lectura 1-4. Análisis y Diseño de Software
- **Siglo XXI.** (2024). Modulo 1-4 - Lectura 1-4. Bases de Datos
- **Siglo XXI.** (2025). Modulo 1-2 - Lectura 1-2. Seminario de Practica Profesional
- (**N.d.) NOAA CLASS** Retrieved September 30, 2025, from https://www.ospo.noaa.gov/products/ocean/

- **(N.d.) Catalogos CONAE** Retrieved September 30, 2025, from https://catalogos.conae.gov.ar/catalogo/otrosCatalogos.html
- **Kendall, K., & Kendall, J. (2011)**. Análisis y diseño de sistemas. Pearson Education