SEMINARIO DE PRÁCTICA DE INFORMÁTICA

INF275-11265

Trabajo Práctico 4

Módulo 4

Alumno: Emiliano Andrés Baum

Legajo: VINFO15483

Titular Experto: Ana Carolina Ferreyra

Titular Disciplinar: Pablo Alejandro Virgolini

Fecha de entrega: 16/11/2025

Período de cursada: 2025

Modalidad: Individual

Índice

Sistema de Alerta Temprana de Emergencias (SATE)

Introducción

El Sistema de Alerta Temprana de Emergencias (SATE) es una solución informática diseñada para colaborar frente a situaciones de emergencias y mitigar los daños en términos sociales, ambientales y económicos, proporcionando información crucial en tiempo casi real a organismos gubernamentales y no gubernamentales. La recurrencia de incendios forestales o inundaciones en distintas regiones del país y el mundo resalta la necesidad de herramientas tecnológicas que permitan una respuesta rápida y coordinada. Actualmente, la información sobre posibles focos de incendio o inundaciones puede ser fragmentada y de difícil acceso, lo que retrasa la toma de decisiones y la movilización de recursos.

Este proyecto se propone desarrollar una plataforma de **software libre** y **código abierto** que centralice y visualice datos satelitales y de otras fuentes públicas, transformándolos en alertas geo-referenciadas. De esta manera, se busca optimizar la gestión de emergencias, permitiendo una intervención temprana frente a catástrofes y minimizar los daños ambientales y materiales. El **SATE** servirá como una herramienta estratégica para la prevención y la respuesta, fortaleciendo la capacidad de los equipos de emergencia para proteger comunidades y ecosistemas y que, a diferencia de otras herramientas disponibles en el mercado, ponga a disposición del conjunto de la sociedad una herramienta libre, gratuita en su uso y soportada por la comunidad del software libre.

La recurrencia cada vez mayor de fenómenos asociados al cambio climático producto de la acción humana y ciclos naturales a nivel nacional e internacional subraya la urgente necesidad de herramientas tecnológicas avanzadas. Estos fenómenos no solo destruyen vastas extensiones de ecosistemas, sino que también ponen en peligro vidas humanas, infraestructuras y medios de subsistencia. *En la actualidad, uno de los mayores desafíos radica en la fragmentación y la difícil accesibilidad de la información sobre posibles focos de incendio o inundaciones. Esta carencia retrasa significativamente la toma de decisiones críticas y la movilización de recursos, lo que a menudo resulta en una propagación incontrolada del fuego y mayores pérdidas.*

En este contexto, el proyecto **SATE** propone el desarrollo de una plataforma de software libre y código abierto. Su función principal será la centralización y visualización de datos provenientes de diversas fuentes. Entre estas fuentes se incluyen datos satelitales, que ofrecen una perspectiva amplia y continua del territorio, así como otra información pública relevante que pueda indicar la presencia o el riesgo de incendios. El sistema procesará esta información para generar alertas geo-referenciadas, lo que significa que cada aviso estará asociado a una ubicación geográfica precisa. Esta precisión es fundamental para dirigir los esfuerzos de respuesta de manera eficiente.

Al optimizar la gestión de emergencias, el **SATE** facilitará una intervención temprana y coordinada. La capacidad de detectar y localizar focos de incendio o inundaciones en sus etapas iniciales es crucial para

contener su expansión y minimizar los daños ambientales y materiales. La plataforma no solo actuará como una herramienta reactiva, sino que también tendrá un papel estratégico en la prevención, al permitir la identificación de áreas de alto riesgo y la implementación de medidas preventivas.

Justificación

El Sistema de Alerta Temprana de Emergencias (**SATE**) se justifica por la urgente necesidad de modernizar y optimizar la gestión de emergencias de manera masiva frente a catástrofes naturales. La falta de una herramienta centralizada, accesible, de código abierto y gratuita que proporcione datos en tiempo casi real sobre nuestro territorio nacional impacta directamente en la capacidad de respuesta de los organismos encargados de actuar frente a estas situaciones.

El proyecto es de vital importancia por las siguientes razones:

- Desde el **punto de vista social** el sistema satisfará la necesidad de contar con una fuente de información única, precisa y en tiempo real sobre la ubicación y posible propagación de los focos de calor o el alcance o desarrollo de una inundación. Esto permitirá a los organismos gubernamentales, brigadas de bomberos y la sociedad civil acceder a datos cruciales que actualmente son dispersos, difíciles de interpretar y, a menudo, de acceso restringido. Al ofrecer alertas personalizadas, el sistema también cubrirá la necesidad de los usuarios de ser notificados proactivamente sobre riesgos en sus áreas de interés.
- Desde el punto de vista **tecnológico**, el proyecto integra y pone en práctica conceptos avanzados como la automatización del consumo de datos de fuentes externas (APIs), el procesamiento de grandes volúmenes de datos georreferenciados y su persistencia en bases de datos relacionales (MySQL). El desarrollo de un prototipo en Java y la aplicación del Proceso Unificado de Desarrollo (PUD) demostrará la capacidad de construir una solución robusta, escalable y mantenible.
- El **impacto social** del **SATE** es significativo. Los incendios forestales y otras catástrofes naturales no solo causan pérdidas económicas y ecológicas, sino que también ponen en riesgo vidas humanas. Una respuesta temprana, facilitada por la información del sistema, puede reducir drásticamente el tiempo de reacción, lo que se traduce en una menor extensión del fuego, menos daños materiales y, lo más importante, la protección de las comunidades. Lo mismo se podría indicar en el caso de las inundaciones.

En esencia, el **SATE** se podría convertir en un pilar fundamental para fortalecer la capacidad de los equipos de emergencia, equipándolos con los medios necesarios para proteger comunidades y ecosistemas vulnerables gracias a un aspecto distintivo y de gran valor como lo es su compromiso con el modelo de software libre. A diferencia de otras herramientas disponibles en el mercado, esta plataforma pondrá a disposición de toda la sociedad una solución gratuita en su uso y soportada activamente por la comunidad global del software libre. Esta filosofía no solo democratiza el acceso a tecnología vital para la gestión de desastres, sino que también fomenta la colaboración y la mejora continua, asegurando que el **SATE** evolucione y se adapte a las necesidades cambiantes en la lucha contra los incendios forestales.

Definiciones del proyecto y del sistema

En esta sección detallaremos cuestiones vinculadas a el alcance, objetivo y una descripción general del

Sistema de Alerta Temprana de Emergencias.

Definiciones del Proyecto

Alcance

El proyecto abarca el análisis, diseño, desarrollo de un prototipo funcional (utilizando Java y MySQL), y despliegue inicial de un sistema de alerta temprana. Se incluirá la integración de fuentes de datos satelitales (como los EOS 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 alertas.

Objetivos del Proyecto:

- Diseñar y desarrollar un sistema informático que procesa datos satelitales y visualice datos de focos de calor (anomalías térmicas) e inundaciones (cuerpos de agua).
- Implementar una base de datos MySQL para gestionar los datos de usuarios, sus ubicaciones de interés y el histórico de alertas validadas.
- Desarrollar un prototipo en lenguaje Java que consuma datos de una fuente externa y los almacene de manera persistente en la base de datos.
- Crear una interfaz de usuario básica que muestre los eventos descritos en un mapa geo-referenciado.
- Definir una arquitectura de sistema escalable que permita la adición de nuevas fuentes de datos y funcionalidades a futuro.

Definiciones del Sistema (SATE)

Descripción General

El **SATE** será una sistema orientado a servicios que, mediante el procesamiento de datos satelitales de acceso público, identificará anomalías térmicas y cuerpos de agua. La detección de estas anomalías es algo que no recae en el software, sino que lo realizan los procesadores y son parte de los productos estándar de los procesadores. 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 correo electrónico.

Funcionamiento

Consumo de datos: Un módulo del sistema se conectará a fuentes de datos satelitales (como las de la NASA, ESA, CONAE) para obtener información sobre anomalías térmicas y cuerpos de agua.

- **Procesamiento:** Los datos brutos serán procesados para filtrar y validar los eventos, utilizando criterios como la intensidad y la persistencia de la señal.
- **Persistencia:** La información de los eventos validados asi como la información de los usuarios y sus preferencias se almacenará en una base de datos relacional (MySQL).
- **Alerta y Visualización:** El sistema activará alertas para los usuarios cuyas áreas de interés coincidan

con los nuevos eventos. Simultáneamente, se actualizará el mapa accesible a través de la interfaz web, mostrando la ubicación, hora y otros datos relevantes de cada uno de los eventos.

Se pueden utilizar productos MODIS y VIIRS Level 2 provenientes de satélites abiertos como la serie EOS o JPSS. También se pueden utilizar fuentes de los satélites SAOCOM 1A y 1B, los cuales son operados por la agencia espacial Argentina (CONAE). Los datos de los satélites EOS/JPSS también son adquiridos por CONAE en su estación ubicada en Falda del Cañete y pueden ser accedidos rápidamente, brindando un valor agregado fundamental comparado con obtenerlos vía NASA-FIRMS, otra fuente de datos pública dedicada a focos de calor.

Elicitación

La fase de elicitación es crucial para el éxito del proyecto, ya que nos permite comprender en profundidad el dominio del problema y las necesidades de los usuarios. Para ello, se aplicará una combinación de técnicas de elicitación que permitan asegurar la obtención de *requerimientos funcionales* y *no funcionales* de manera integral. El objetivo es ir más allá de la simple descripción de lo que el sistema debe hacer, para entender el porqué y el cómo de cada funcionalidad.

Métodos de Elicitación

Los métodos que se utilizarán fueron seleccionados por su idoneidad para un proyecto que tendra diferentes tipos de usuarios y un componente técnico avanzado-especializado:

- **Entrevistas con Expertos y Stakeholders Clave:** Se llevarán a cabo entrevistas semi-estructuradas [1]con figuras clave en el campo de la gestión de emergencias y la ciencia de datos. Esto incluye a:
    - **Representantes de Organismos Gubernamentales y de Emergencia:** Bomberos, miembros de Defensa Civil y personal de organismos como la CONAE. El foco estará en entender su flujo de trabajo actual, los desafíos que enfrentan para obtener información y qué tipo de datos les serían más útiles para la toma de decisiones
    - **Profesionales en el dominio del negocio:** Especialistas en geografía, meteorología y el estudio de los incendios e inundaciones. Su conocimiento técnico es vital para comprender la naturaleza de los datos satelitales y cómo deben ser procesados.
    - **Potenciales Usuarios Finales:** Miembros de la comunidad en zonas de riesgo que pueden ofrecer una perspectiva valiosa sobre cómo el sistema podría servirles para su seguridad personal.
- **Análisis de Documentos Existentes:** Se revisarán documentos técnicos, informes de incidentes previos, regulaciones gubernamentales sobre gestión de emergencias y la documentación de las APIs satelitales de agencias como la NASA, la ESA y la CONAE. Este análisis nos permitirá conocer las fuentes de información disponibles y las limitaciones técnicas que podría tener su uso.
- **Observación de Procesos:** Se buscará la oportunidad de observar cómo los organismos de emergencia actuales monitorean y responden a los eventos. Esta técnica es invaluable para descubrir requerimientos que los usuarios podrían no mencionar durante una entrevista. Por ejemplo, se podría observar cómo un analista usa múltiples pantallas o plataformas para

[1] Presentan un grado mayor de flexibilidad que las estructuradas, debido a que parten de preguntas planeadas, que pueden ajustarse a los entrevistados. Su ventaja es la posibilidad de adaptarse a los sujetos con enormes posibilidades para motivar al interlocutor, aclarar términos, identificar ambigüedades y reducir formalismos. Bravo-Garcia, 2013

correlacionar datos de diferentes fuentes, lo que justificaría la necesidad de una plataforma unificada.

- **Análisis de Sistemas Análogos (Benchmarking[2]):** Se estudiarán plataformas similares existentes, tanto comerciales como de código abierto, para identificar las mejores prácticas, las funcionalidades clave y las áreas de oportunidad. El objetivo no es copiar, sino aprender de sus éxitos y errores para ofrecer una propuesta de valor única y superior, especialmente en lo que respecta a la simplicidad y el acceso libre.

A partir de estos métodos no solo se facilitará la recolección de una lista de requisitos, sino que también permitirá una comprensión holística del problema de negocio, asegurando que la solución SATE sea efectiva, útil y factible.

Conocimiento del negocio

Para comprender la dinámica del negocio y los procesos de gestión de emergencias, se realizará un análisis detallado.

Actores Clave:

- *Organismos Gubernamentales:* Defensa Civil, Ministerios de Ambiente, Secretarías de Emergencia.
- *Organismos No Gubernamentales:* ONGs medioambientales, fundaciones.
- *Bomberos:* Voluntarios y profesionales.
- *Público en general:* Personas que viven en zonas de riesgo.

Flujo de Trabajo Actual:

- *Detección:* Se realiza a través de llamadas de emergencia, avistamientos visuales, o por la recepción de informes de satélites (a menudo en formatos crudos o que requieren conocimientos técnicos para su interpretación).
- *Validación:* Se confirma el punto de interés a través de llamadas o envío de equipos al lugar.
- *Coordinación:* La información se transmite por radio o teléfono a los equipos en campo.
- *Respuesta:* Se movilizan los recursos para combatir el evento.

Problemas y Desafíos:

- *Tiempo de respuesta lento:* La información no se centraliza ni se distribuye eficientemente.
- *Falta de coordinación:* Diferentes organismos pueden tener información fragmentada o poca interoperabilidad, lo que dificulta una respuesta unificada.
- *Acceso a la información:* Los datos satelitales son complejos y no están al alcance de todos los actores.
- *Adaptación al nuevo sistema:* si bien solo seria necesaria una navegador, adaptarse al nuevo

[2] Los puntos de referencia (*benchmarks*) son estándares predefinidos, mientras que el *benchmarking* es el proceso en el cual se establecen esos estándares. Para definir los puntos de referencia, debes comparar tu trabajo con otra línea de base. https://asana.com/es/resources/benchmarking

sistema requiere una capacitación y de mostrar las ventajas de su uso.

Propuesta de solución

La propuesta es desarrollar el **Sistema de Alerta Temprana Emergencias (SATE)**, un sistema informático que automatizará y mejorará el flujo de trabajo actual. La solución se basará en una arquitectura orientada a servicios que permitirá la integración de diferentes fuentes de datos y la visualización en una interfaz amigable.

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

La capa de datos será responsable de la persistencia, el acceso y la gestión de toda la información del sistema SATE. Se construirá sobre una base de datos **relacional** robusta, como MySQL, que es ampliamente utilizada, bien documentada y ofrece las capacidades necesarias para un proyecto de esta envergadura.

La elección de una base de datos relacional se justifica por su capacidad para manejar la estructura de datos compleja de manera íntegra. La información del proyecto se puede organizar en tablas relacionadas, lo que garantiza la consistencia e integridad de los datos, algo fundamental para un sistema de emergencias.

Los datos se estructurarán de la siguiente manera:

- **Gestión de Usuarios:** Se crearán tablas para almacenar información de los usuarios, sus roles y los permisos de acceso. Esto incluye datos como nombre, correo electrónico, credenciales de acceso y la asignación de roles de **Usuario** y **Administrador**.
- **Gestión de Eventos y Datos Geoespaciales:** La información clave sobre los eventos (incendios, inundaciones) y los reportes manuales se almacenará en tablas específicas. Para manejar los datos de geolocalización, se aprovecharán las funciones **geoespaciales nativas** de la base de datos. MySQL cuenta con tipos de datos como **POINT**[3], **LINESTRING**[4] y **POLYGON**[5], y ofrece un conjunto de funciones SQL para realizar consultas eficientes sobre información geo-referenciada.

[3] Point Reference
[4] Linestring
[5] Poligon

- **Gestión de Áreas de Interés y Notificaciones:** Las áreas que los usuarios definan en el mapa se guardarán también con tipos de datos geoespaciales. Esto permitirá que la capa de negocio realice consultas espaciales (como "¿qué eventos se encuentran dentro de esta área?") de forma directa en la base de datos, lo que optimiza la lógica de las notificaciones.

De esta manera, la capa de datos proporciona una solución unificada y consistente para almacenar tanto los datos estructurados tradicionales como la información geo-referenciada, cumpliendo con el requisito de utilizar una base de datos puramente relacional.

Capa de negocio

El **núcleo funcional del sistema** (el ***core*** del negocio) se desarrollará por completo en **Java**. Esta capa servirá como el cerebro del sistema, orquestando todas las operaciones y asegurando que la lógica de negocio se aplique de manera consistente y eficiente.

Sus responsabilidades clave incluyen:

- **Consumo y procesamiento de datos:** Se encargará de obtener datos de APIs externas (como las de la CONAE o NASA) y de procesar esa información en bruto para identificar los "eventos" (focos de calor o cuerpos de agua).
- **Lógica de negocio:** Validará la información entrante y ejecutará las reglas de negocio, como la validación de reportes manuales por parte del administrador o el cruce de datos geoespaciales para determinar si un evento ha ocurrido dentro de las áreas de interés definidas por los usuarios.
- **Gestión de la persistencia:** Aunque la base de datos es la que almacena los datos, la capa de negocio gestionará todas las interacciones con ella, tanto para guardar nueva información como para recuperar datos históricos o de usuario.
- **Comunicación con la capa de presentación:** Funcionará como un servicio o API que la interfaz de usuario (la **Capa de Presentación**) consumirá. Esto garantiza que la lógica del negocio esté desacoplada del frontend, permitiendo la escalabilidad y la futura expansión del sistema a múltiples plataformas (web, móvil, etc.).

Capa de presentación

Se implementará una **interfaz de usuario de escritorio** desarrollada en Java. Esta capa será la cara visible del sistema y se encargará de todas las interacciones con el usuario.

Sus responsabilidades clave incluyen:

- **Visualización de datos geo-referenciados[6]:** Mostrará el mapa y los eventos (incendios, inundaciones, reportes manuales) de manera clara y precisa. Para lograr esto, se utilizarán bibliotecas de visualización de mapas que permitan la representación de puntos y polígonos sobre un mapa base.
- **Gestión de la interacción:** Capturará las acciones del usuario, como el dibujo de áreas de interés, la selección de eventos o el ingreso de reportes manuales, y enviará estas solicitudes a la **capa de negocio** para su procesamiento.
- **Comunicación con la capa de negocio:** Esta capa actuará como un cliente que consume los servicios

[6] Existen varias librerías como **GeoTools** o **JTS** para datos geoespaciales mientras que para mapas tenemos **JMapViewer** u **OpenMap**.

que expone la capa de negocio. Esto garantiza una arquitectura limpia, donde la lógica de la presentación está separada de la lógica del negocio.

- **Manejo de la lógica de presentación:** Será responsable de la representación visual, la navegación entre pantallas y de asegurar una experiencia de usuario fluida e intuitiva.

Ventajas de la solución

- **Automatización**: La solución reduce significativamente la carga de trabajo manual, ya que el sistema automatiza la detección de eventos (como focos de calor o inundaciones) a través del consumo continuo de datos satelitales. Esto permite que el personal se enfoque en la gestión de emergencias en lugar de en la recolección de información.
- **Información en tiempo casi real (Near Real Time[7])**: El sistema acelera el tiempo de respuesta ante un evento al proporcionar alertas oportunas y geo-referenciadas. La capacidad de reaccionar rápidamente ante una emergencia es crítica para minimizar los daños y proteger vidas.
- **Mejora de la coordinación**: Al servir como una fuente única y centralizada de información validada, el sistema facilita la comunicación y la coordinación entre distintos organismos y equipos de emergencia. Esto evita la fragmentación de datos y asegura que todos los actores trabajen con la misma información.
- **Accesibilidad y democratización**: Al ser un proyecto de software libre y código abierto, la solución es accesible para una mayor cantidad de usuarios y organizaciones, sin las barreras económicas de las herramientas comerciales. Esto permite que más personas se beneficien de la información para la prevención y la seguridad.
- **Escalabilidad y extensibilidad**: La arquitectura modular del sistema, basada en capas, facilita la integración de nuevas funcionalidades en el futuro. Esto incluye, por ejemplo, módulos de predicción de la propagación del fuego o la incorporación de nuevos tipos de eventos, lo que asegura la vida útil y la relevancia continua del proyecto.Inicio del análisis

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 SATE. El análisis se articula en dos dimensiones clave: los **Requerimientos Funcionales**, que detallan el qué del sistema (las acciones y servicios que **SATE** 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.

[7] Los productos GPM casi en tiempo real (GMI y DPR) suelen estar disponibles a las pocas horas de la observación. NASA

Requerimientos Funcionales

Requisitos Funcionales del Sistema	Descripción
RF-01	El sistema debe permitir el registro de nuevos usuarios con su información de contacto y ubicación.
RF-02	El sistema debe permitir a los usuarios definir una o varias áreas de interés en el mapa.
RF-03	El sistema debe procesar datos satelitales para publicar focos de calor georeferenciados .
RF-04	El sistema debe procesar datos satelitales para publicar cuerpos de agua (inundaciones) georeferenciados.
RF-05	El sistema debe enviar notificaciones por correo electrónico a los usuarios cuando se detecte un evento dentro de sus áreas de interés.
RF-06	El sistema debe permitir a los usuarios reportar una anomalía térmica - foco de calor (incendio) manualmente (entrada manual) y que el mismo sea distinguible de los reportados por satélite.
RF-07	El sistema debe permitir a los usuarios reportar un cuerpo de agua (inundación) manualmente (entrada manual) y que el mismo sea distinguible de los reportados por satélite.
RF-08	El sistema debe ofrecer una vista histórica de los eventos que fueron validados por el administrador.
RF-09	El sistema debe permitir a un administrador validar, editar y eliminar eventos (focos de calor o cuerpos de agua) reportados manualmente antes de que se hagan públicos. Esto garantiza la calidad y veracidad de la información.
RF-10	El sistema debe permitir a los usuarios elegir si desean recibir notificaciones solo sobre focos de calor, solo sobre inundaciones, o sobre ambos tipos de eventos.
RF-11	El sistema debe mostrar información detallada sobre cada evento (por ejemplo, fecha y hora de detección, fuente de datos, y en el caso de reportes manuales, el usuario que lo reportó).
RF-12	El sistema debe ser capaz de integrar datos de múltiples fuentes satelitales (por ejemplo, de diferentes agencias como la NASA, CONAE o la ESA) para una mayor cobertura y precisión.
RF-13	El sistema de administrador debe permitir al administrador la gestión de los usuarios del sistema

Requerimientos No Funcionales

Requisitos No Funcionales del Sistema	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". (Rendimiento)
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.(Seguridad)
RNF-03	El sistema debe ser capaz de manejar un aumento en el volumen de datos satelitales y en la cantidad de usuarios.(Escalabilidad)
RNF-04	El sistema debe estar disponible 24/7, ya que los incendios pueden ocurrir en cualquier momento.(Disponibilidad)
RNF-05	La interfaz de usuario debe ser intuitiva y fácil de usar, incluso para personal no técnico.(Usabilidad)
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.

Casos de uso

El inicio del análisis se centra en la **definición de los casos de uso del sistema**, que describen las interacciones de los usuarios y el/los administradores con el mismo.

Caso de Uso 1: Registrarse en el Sistema	
CU-01	Registrarse en el Sistema
Actor(es)	Usuario No Registrado, Sistema
Referencias	RF-01, RNF-02
Descripción	Este caso de uso describe el proceso mediante el cual un nuevo usuario crea una cuenta en el sistema para acceder a todas las funcionalidades.
Precondiciones	● El usuario no tiene una cuenta activa en el sistema. ● El sistema está disponible y operativo

Flujo Normal	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, ubicación (opcional), y una contraseña. 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. 7. El sistema informa al actor que el registro se ha completado con éxito.
Flujos Alternativos	1. **Correo Electrónico ya en uso**: Si el correo electrónico ya existe, el sistema muestra un mensaje de error y no permite la creación de la cuenta. 2. **Campos Incompletos:** Si falta algún dato obligatorio, el sistema resalta los campos faltantes y solicita que se completen.
Postcondiciones	El actor ahora es un **Usuario Registrado** y puede iniciar sesión en el sistema.
Excepciones	● No llega el correo electrónico de registro. Opción de nuevo envío.

Caso de Uso 2: Autenticarse en el Sistema	
CU-02	Autenticarse en el Sistema
Actor(es)	Usuario Registrado, Sistema
Referencias	RF-01, RNF-02
Descripción	Este caso de uso describe el proceso mediante el cual un usuario se autentica en el sistema para acceder a todas las funcionalidades.
Precondiciones	● El usuario tiene una cuenta activa en el sistema. ● El sistema está disponible y operativo
Flujo Normal	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	3. **Correo Electrónico ya en uso**: Si el correo electrónico ya existe, el sistema muestra un mensaje de error y no permite la creación de la cuenta. 4. **Campos Incompletos:** Si falta algún dato obligatorio, el sistema resalta los campos faltantes y solicita que se completen.
Postcondiciones	El actor ahora es un **Usuario Registrado** y puede iniciar sesión en el sistema.
Excepciones	● Usuario o contraseña invalido. ● Recupero de contraseña

Caso de Uso 3: Visualizar Mapa de Alertas	
CU-03	Visualizar Mapa de Alertas
Actor(es)	Usuario Autenticado, Sistema
Referencias	RF-01, RF-02, RF-03, RNF-02, RNF-04, RNF-05
Descripción	Este caso de uso permite a un usuario autenticado ver en un mapa los eventos o posibles eventos detectados por el sistema.
Precondiciones	● El usuario ha iniciado sesión. ● El sistema ha procesado datos satelitales y tiene eventos para mostrar.
Flujo Normal	1. El actor navega a la sección del mapa de alertas. 2. El sistema consulta la base de datos de eventos y recupera la información más reciente. 3. El sistema superpone los puntos de los eventos en un mapa georreferenciado en la interfaz de usuario. 4. El actor puede hacer clic en un foco para ver detalles, como la hora de detección, la fuente de datos y la intensidad. 5. El sistema actualiza el mapa periódicamente para

	mostrar nuevos eventos.
Flujos Alternativos	**No hay eventos:** Si la base de datos no contiene eventos recientes, el sistema muestra un mensaje indicando que no hay alertas activas en el momento.
Postcondiciones	El actor ahora es un **Usuario Registrado** y puede iniciar sesión en el sistema.
Excepciones	-

Caso de Uso 4: Definir Área de Interés	
CU-04	Definir Área de Interés
Actor(es)	Usuario Autenticado, Sistema
Referencias	RF-01,RF-02,RF-03, RNF-05
Descripción	Este caso de uso permite al usuario personalizar su experiencia al seleccionar una o más zonas geográficas de las que desea recibir alertas específicas.
Precondiciones	● El usuario ha iniciado sesión. ● La funcionalidad de geolocalización en el navegador está habilitada..
Flujo Normal	1. El actor navega a la sección de configuración de alertas. 2. El sistema presenta un mapa interactivo con herramientas de dibujo. 3. El actor utiliza las herramientas para dibujar un polígono o seleccionar un radio alrededor de una ubicación. 4. El actor asigna un nombre a la nueva área de interés (ej., "Localidad X", "Parque Nacional"). 5. El actor confirma la creación del área de interés. 6. El sistema guarda las coordenadas geográficas del área y el nombre asociado en la base de datos **MySQL**, vinculándolas a la cuenta del usuario. 7. El sistema muestra un mensaje de confirmación.
Flujos Alternativos	1. **Límite de Áreas Alcanzado:** Si el sistema tiene un límite de áreas de interés por usuario, el sistema notifica al actor que no puede añadir más.

Postcondiciones	El usuario tiene una o más áreas de interés definidas, las cuales se usarán para enviar alertas personalizadas..
Excepciones	El sistema no envía notificaciones para el área seleccionada

Caso de Uso 5: Alerta de Evento	
CU-05	Alerta de Evento
Actor(es)	Usuario Registrado, Sistema
Referencias	RF-01, RF-10, RNF-04
Descripción	Describe el proceso automático por el cual el sistema notifica a un usuario cuando se detecta un nuevo foco de calor o inundación dentro de una de sus áreas de interés previamente definidas.
Precondiciones	● El usuario tiene al menos un área de interés definida. ● El sistema ha detectado un nuevo foco de calor. ● El sistema ha detectado un nuevo cuerpo de agua.
Flujo Normal	1. El sistema detecta un nuevo foco de calor a través del procesamiento de datos satelitales. 2. El sistema valida que el foco o cuerpo de agua sea genuino. 3. El sistema ejecuta un proceso de geocercado[8], comparando las coordenadas del nuevo evento con las áreas de interés guardadas en la base de datos de los usuarios. 4. Si el evento se encuentra dentro de un área de interés de un usuario, el sistema genera una notificación. 5. El sistema envía un correo electrónico al usuario con los detalles de la alerta (ubicación del evento, hora, área afectada). 6. El sistema muestra un mensaje de confirmación.
Flujos Alternativos	**Foco fuera de las áreas de interés:** Si el nuevo evento no coincide con ninguna área de interés, el sistema no envía una alerta personalizada, pero lo muestra en el mapa general.
Postcondiciones	El usuario ha sido notificado proactivamente de un riesgo potencial.

[8] https://en.wikipedia.org/wiki/Geofence

Excepciones	El sistema no detecta el evento y la alerta no es enviada.

Caso de Uso 6: Reportar Evento Manualmente	
CU-06	Reportar Evento Manualmente
Actor(es)	Usuario Registrado
Referencias	RF-01, RF-06, RF-07, RF-09, RF-10, RNF-05
Descripción	Este caso de uso permite a los usuarios con conocimiento directo de un incendio (ej. brigadistas, bomberos voluntarios, vecinos) ingresar la ubicación de un foco o inundación para que se integre al sistema y se muestre en el mapa.
Precondiciones	● El usuario ha iniciado sesión.
Flujo Normal	1. El actor navega hacia la función de reporte manual. 2. El sistema presenta una interfaz para ingresar datos del foco (ubicación en el mapa, coordenadas, tipo de catástrofe). 3. El actor introduce los datos y, opcionalmente, una descripción o fotos. 4. El actor confirma el reporte. 5. El sistema valida los datos y guarda el reporte en la base de datos (por ejemplo, en una tabla de "Reportes Manuales"). 6. El reporte queda pendiente de validación por un administrador. Una vez validado, se superpone en el mapa general. 7. El sistema informa al actor que el reporte ha sido recibido.
Flujos Alternativos	**Datos incompletos:** El sistema no permite enviar el reporte si faltan datos obligatorios, como la ubicación.
Postcondiciones	La información de un foco de incendio o inundación ha sido agregada al sistema para su posterior visualización y validación.
Excepciones	-

Caso de Uso 7: Gestionar Usuarios

CU-07	Gestionar Usuarios
Actor(es)	Administrador del Sistema
Referencias	RF-01, RF-13, RNF-03
Descripción	Este caso de uso permite a un administrador gestionar las cuentas de usuario, incluyendo la creación, modificación y eliminación de perfiles. Esto es crucial para mantener la integridad y la seguridad del sistema.
Precondiciones	● El usuario administrador ha iniciado sesión.
Flujo Normal	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 (activar, desactivar). 6. El administrador puede eliminar la cuenta de un usuario. 7. El sistema confirma los cambios realizados.
Flujos Alternativos	**No hay usuarios registrados:** 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	-

Caso de Uso 8: Validar Reportes Manuales	
CU-08	Validar Reportes Manuales
Actor(es)	Administrador del Sistema
Referencias	RF-01, RF-06, RF-07, RF-09, RNF-03
Descripción	Este caso de uso describe el proceso mediante el cual un administrador verifica la veracidad de los reportes de incendios enviados manualmente por los usuarios antes de que se muestran en el mapa público.

Precondiciones	● El administrador ha iniciado sesión en el sistema. ● Existen reportes manuales pendientes de validación.
Flujo Normal	1. El administrador accede al panel de reportes pendientes. 2. El sistema muestra una lista de los reportes enviados por los usuarios, con detalles como la ubicación y la hora. 3. El administrador revisa la información de cada reporte, usando herramientas externas si es necesario para verificar la autenticidad (por ejemplo, cruzar los datos con otras fuentes). 4. El administrador tiene la opción de **aprobar** o **rechazar** el reporte. 5. Si el reporte es aprobado, el sistema lo integra a la base de datos de focos de calor y lo hace visible en el mapa para todos los usuarios. 6. Si el reporte es rechazado, el sistema lo descarta.
Flujos Alternativos	**No hay reportes pendientes:** El sistema informa al administrador que no hay reportes manuales para validar.
Postcondiciones	Los reportes manuales han sido procesados, y los que son válidos se han añadido al sistema.
Excepciones	La alerta no impacta en el sistema. No se envian notificaciones.

Caso de Uso 9: Integrar Nueva Fuente de Datos	
CU9	Integrar Nueva Fuente de Datos
Actor(es)	Administrador del Sistema
Referencias	RF-01, RF-12, RNF-03
Descripción	Este caso de uso se centra en la capacidad de escalar el sistema al permitir que un administrador conecte una nueva fuente de datos (por ejemplo, un nuevo servicio de la ESA, CONAE o privada).
Precondiciones	● El administrador tiene los permisos para configurar fuentes de datos. ● Existe una nueva API o fuente de datos disponible para ser integrada
Flujo Normal	1. El administrador accede a la sección de configuración de fuentes de datos.

	2. El sistema muestra un formulario para añadir una nueva fuente. 3. El administrador ingresa la URL de la API, las credenciales de acceso (si son necesarias), y otros parámetros de configuración. 4. El sistema realiza una prueba de conexión a la nueva fuente de datos. 5. Si la conexión es exitosa, el sistema guarda la configuración y habilita el módulo de consumo de datos para la nueva fuente. 6. El sistema comienza a procesar los datos de la nueva fuente.
Flujos Alternativos	**No hay reportes pendientes:** El sistema informa al administrador que no hay reportes manuales para validar.
Postcondiciones	El sistema está configurado para consumir datos de una nueva fuente, mejorando la cobertura y la precisión de la detección.
Excepciones	La nueva fuente no puede agregarse

Caso de Uso 10: Suscribirse a Notificaciones	
CU10	Suscribirse a Notificaciones Específicas
Actor(es)	Usuario Registrado
Referencias	RF-01, RF-05, RF-10, RNF-04, RNF-05
Descripción	Este caso de uso describe cómo los usuarios pueden suscribirse para recibir notificaciones de eventos.
Precondiciones	● El usuario ha iniciado sesión.
Flujo Normal	1. El actor navega a la sección de preferencias de notificaciones. 2. El sistema presenta opciones de suscripción (por ejemplo, notificaciones para eventos confirmados por validación, alertas en un rango de distancia específico, etc.). 3. El actor selecciona las opciones que le interesan. 4. El actor guarda los cambios. 5. El sistema actualiza las preferencias del usuario en la base de datos **MySQL**.
Flujos Alternativos	**No se guardan los cambios:** El sistema muestra un mensaje de error si no puede guardar las preferencias.

Postcondiciones	Las preferencias de notificación del usuario han sido actualizadas, y solo recibirá las alertas que cumplen con sus criterios.
Excepciones	El sistema no envía las notificaciones. Los eventos no corresponden al área de interés solicitada por el usuario.

Caso de Uso 11: Consultar Histórico de Alertas	
CU-11	Consultar Histórico de Alertas
Actor(es)	Usuario Registrado
Referencias	RF-01, RF-11, RNF-04
Descripción	Este caso de uso permite a los usuarios revisar el historial de alertas y eventos en un período de tiempo determinado para analizar la recurrencia o la tendencia en una zona específica.
Precondiciones	● El usuario ha iniciado sesión. ● El sistema tiene un registro histórico de eventos.
Flujo Normal	1. El actor navega a la sección de historial de alertas. 2. El sistema presenta una interfaz con opciones de búsqueda, como un rango de fechas y una ubicación geográfica. 3. El actor ingresa los criterios de búsqueda. 4. El sistema consulta la base de datos (MySQL) para recuperar los datos históricos que coinciden con los criterios. 5. El sistema muestra los resultados en un mapa y/o una tabla, permitiendo al usuario ver la evolución de los incendios en el tiempo.
Flujos Alternativos	**No hay datos en el período seleccionado:** El sistema muestra un mensaje indicando que no se encontraron eventos para la fecha o ubicación especificadas.
Postcondiciones	El usuario ha accedido a información histórica relevante para el análisis de tendencias.
Excepciones	El histórico se encuentra vacío.

Matriz de trazabilidad de requerimientos

ID Requisito	Requisito Funcional/No Funcional	Caso de Uso	Clases de Dominio	Artefacto de Implementación (Prototipo)
RF-01	El sistema debe permitir el registro de nuevos usuarios.	CU-01: Registrarse en el Sistema	Usuario	Módulo de Registro (Formulario y lógica Java)
RF-02	El sistema debe permitir a los usuarios definir áreas de interés.	CU-04: Definir Área de Interés	Usuario, ÁreaDeInteres	Módulo de Mapa Interactivo
RF-03	El sistema debe procesar datos satelitales de anomalías térmicas.	N/A (Proceso de fondo)	Evento, FuenteDeDatos	Módulo de Consumo de API Externa (Clase Java)
RF-04	El sistema debe procesar datos de inundaciones satelitales.	N/A (Proceso de fondo)	Evento, FuenteDeDatos	Módulo de Consumo de API Externa (Clase Java)
RF-05	El sistema debe enviar notificaciones por correo.	CU-05: Recibir Alerta de Incendio	Usuario, Notificacion, Evento	Módulo de Notificación (Lógica Java)
RF-06	El sistema debe permitir el reporte manual de incendios.	CU-06: Reportar Foco Manualmente	ReporteManual , Usuario	Formulario de Reporte Manual
RF-07	El sistema debe permitir el reporte manual de inundaciones.	CU-06: Reportar Foco Manualmente	ReporteManual , Usuario	Formulario de Reporte Manual
RF-08	El sistema debe ofrecer una vista histórica de eventos.	CU-10: Consultar Histórico de Alertas	Evento, Administrador	Módulo de Historial y Filtros
RNF-01	Rendimiento: Latencia mínima.	Todos los CU de visualización y procesamiento.	Evento, FuenteDeDatos	Optimización de consultas a la base de datos
RNF-02	Seguridad: Autenticación y autorización.	CU2: Iniciar Sesión, CU7: Gestionar Usuarios	Usuario, Administrador	Módulo de Autenticación y Perfiles
RNF-03	Escalabilidad: Manejar aumento de datos y usuarios.	Todos los CU.	Todas las Clases	Arquitectura del sistema (Java, MySQL)
RNF-04	Disponibilidad: Sistema 24/7.	Todos los CU.	Todas las Clases	Estrategia de despliegue y monitoreo del servidor

RNF-05	Usabilidad: Interfaz intuitiva.	Todos los CU con interacción de usuario.	N/A (UX/UI)	Diseño de interfaz de usuario
RF-09	Gestión: Validar/eliminar reportes manuales.	CU-07: Gestionar Usuarios, CU-08: Validar Reportes Manuales	ReporteManual, Administrador	Módulo de Administración
RF-10	Suscripción: Elegir tipo de eventos.	CU-09: Suscribirse a Notificaciones Específicas	Usuario, Notificación	Perfil de Usuario y Opciones
RF-11	Detalle: Mostrar información detallada de eventos.	CU-03: Visualizar Mapa de Alertas	Evento	Vistas de detalle en la interfaz del mapa
RF-12	Múltiples Fuentes: Integrar datos de múltiples fuentes.	CU-08: Integrar Nueva Fuente de Datos Satelitales	FuenteDeDatos, Administrador	Módulo de Configuración del Administrador
RNF-06	Interoperabilidad: Tener una API.	N/A (API externa)	Evento, Notificacion	Módulo de API pública
RNF-07	Compatibilidad: Navegadores y dispositivos.	Todos los CU con interacción de usuario.	N/A	Diseño web responsivo
RNF-08	Mantenibilidad: Código modular y documentado.	N/A (Práctica de desarrollo)	Todas las Clases	Estándares de programación y documentación

Diagrama de Casos de Uso

Este diagrama te proporciona una visión general y completa del comportamiento del sistema, mostrando quién (actores) interactúa con cada funcionalidad (casos de uso) y cómo estas funcionalidades se relacionan entre sí.

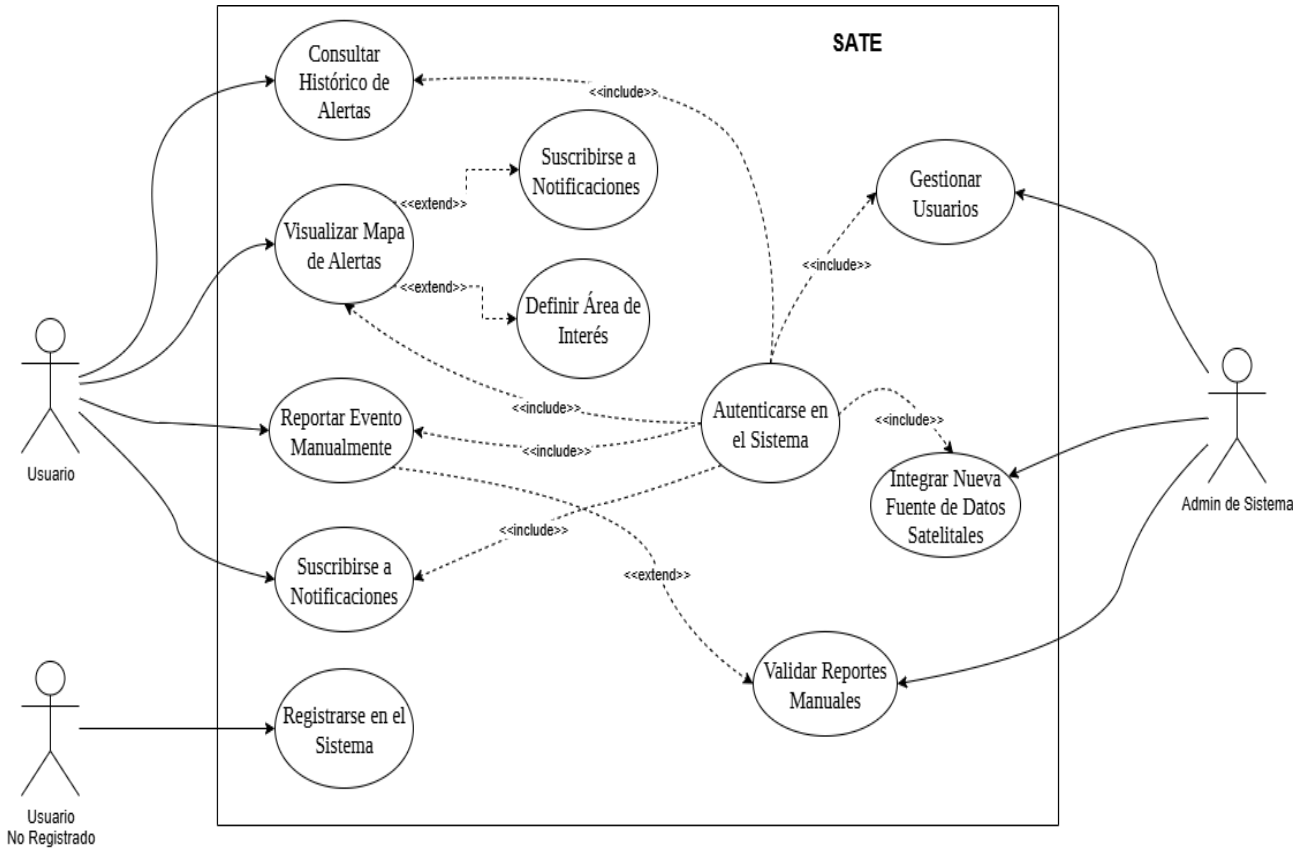


Diagrama de Dominio

Este diagrama representa el modelo conceptual de negocio del Sistema de Atención de Emergencias (SATE). Muestra las entidades principales del proyecto, sus atributos y las relaciones que existen entre ellas, sirviendo como fundamental para el diseño del sistema.

Diagrama de dominio para el SATE

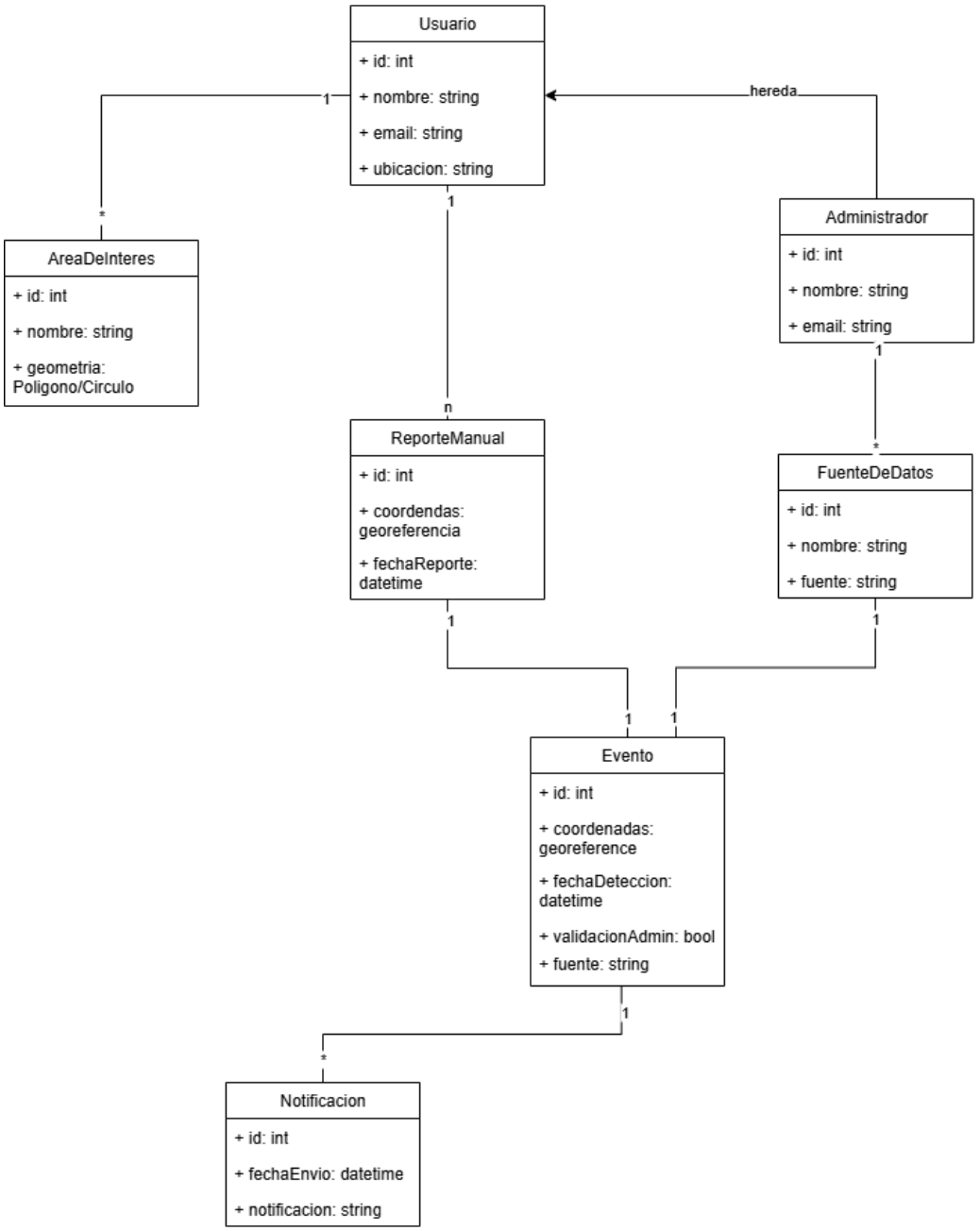


Diagrama de Secuencia

A continuación presento el diagrama de secuencia para la definición de un área de interés y suscribirse a notificaciones para dicha área.

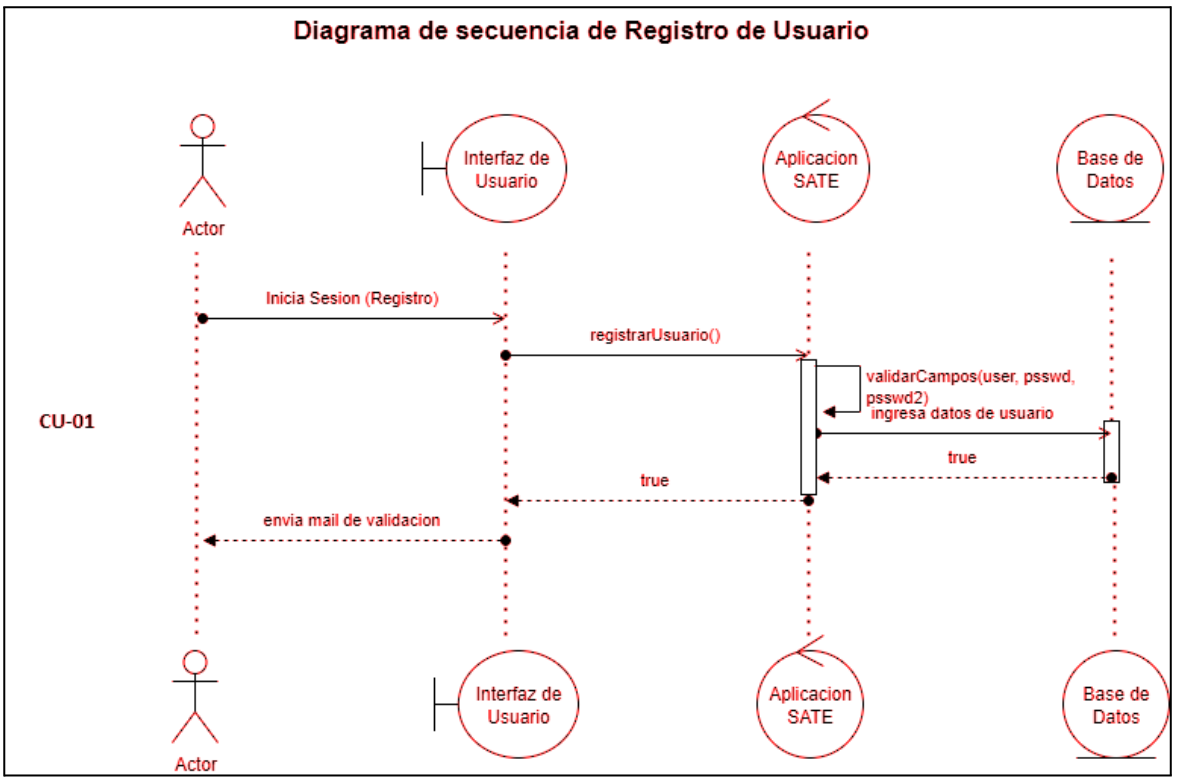


Diagrama de secuencia de Registro de Usuario
CU-01
Actor
Interfaz de Usuario
Aplicacion SATE
Base de Datos
Inicia Sesion (Registro)
registrarUsuario()
validarCampos(user, psswd, psswd2)
ingresa datos de usuario
true
true
envia mail de validacion
Actor
Interfaz de Usuario
Aplicacion SATE
Base de Datos

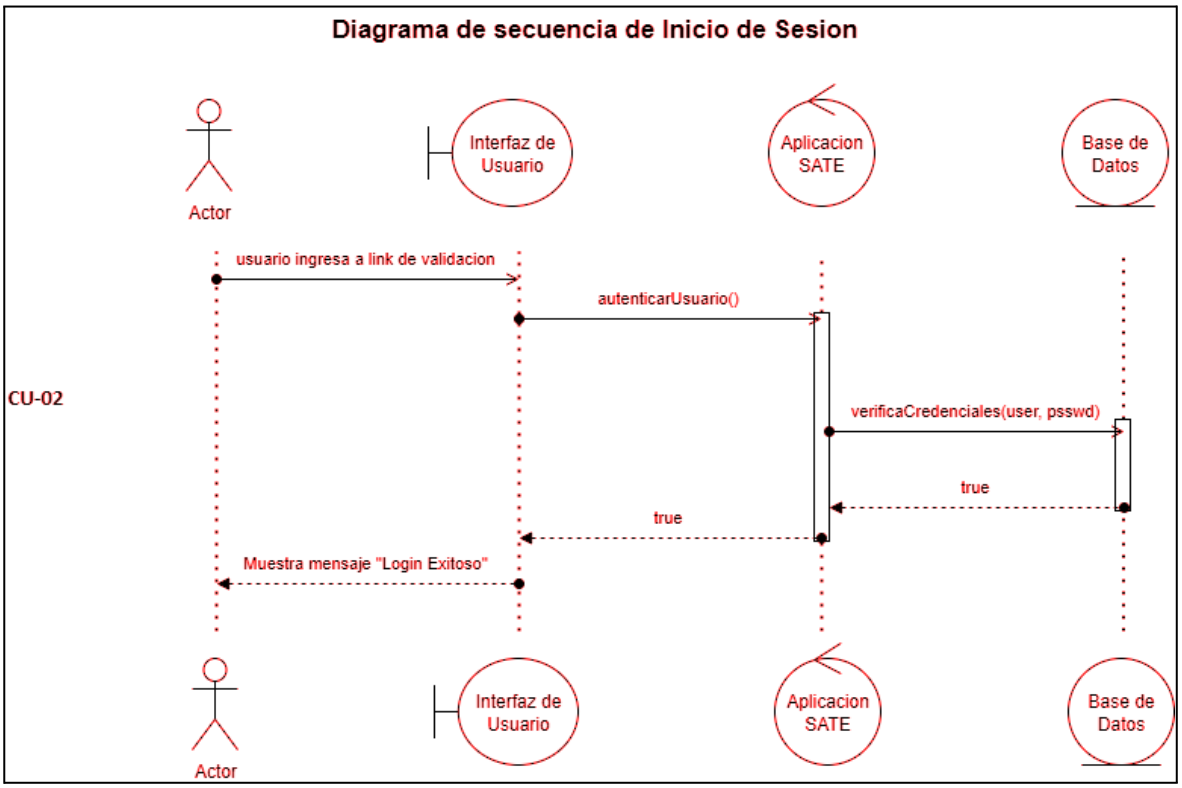


Diagrama de secuencia de Inicio de Sesion
CU-02
Actor
Interfaz de Usuario
Aplicacion SATE
Base de Datos
usuario ingresa a link de validacion
autenticarUsuario()
verificaCredenciales(user, psswd)
true
true
Muestra mensaje "Login Exitoso"
Actor
Interfaz de Usuario
Aplicacion SATE
Base de Datos

Diagrama de secuencia de Autenticacion de usuario

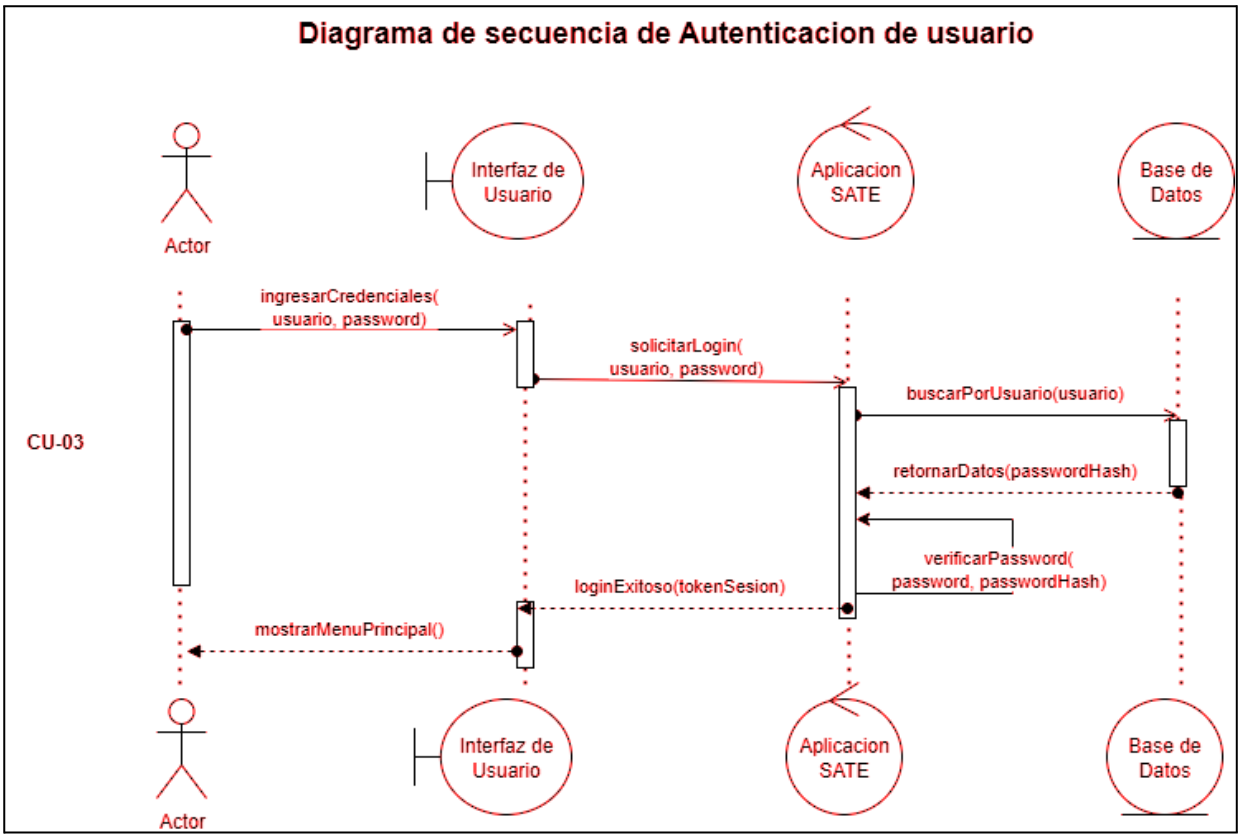


Diagrama de secuencia del caso de uso Definir Área de Interés

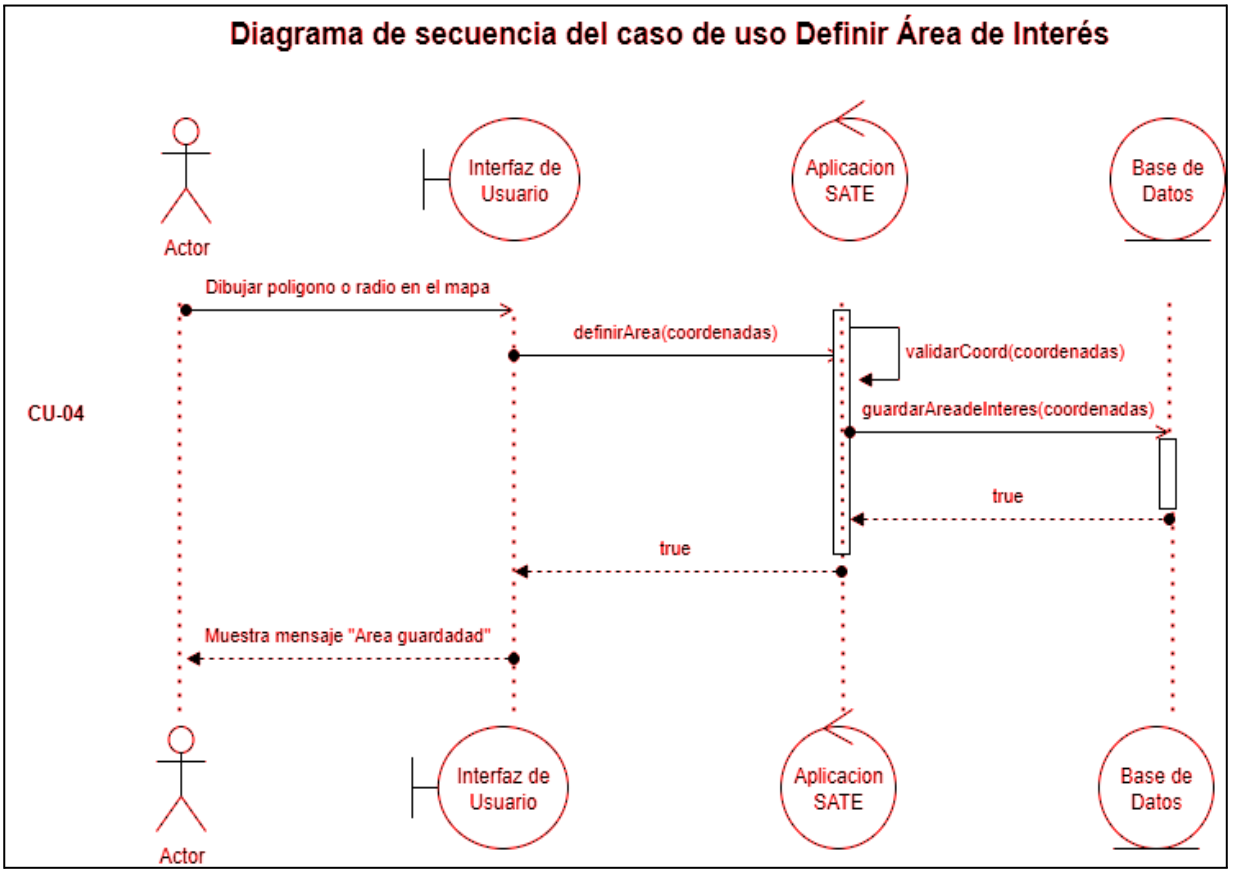


Diagrama de secuencia de Alerta de Evento

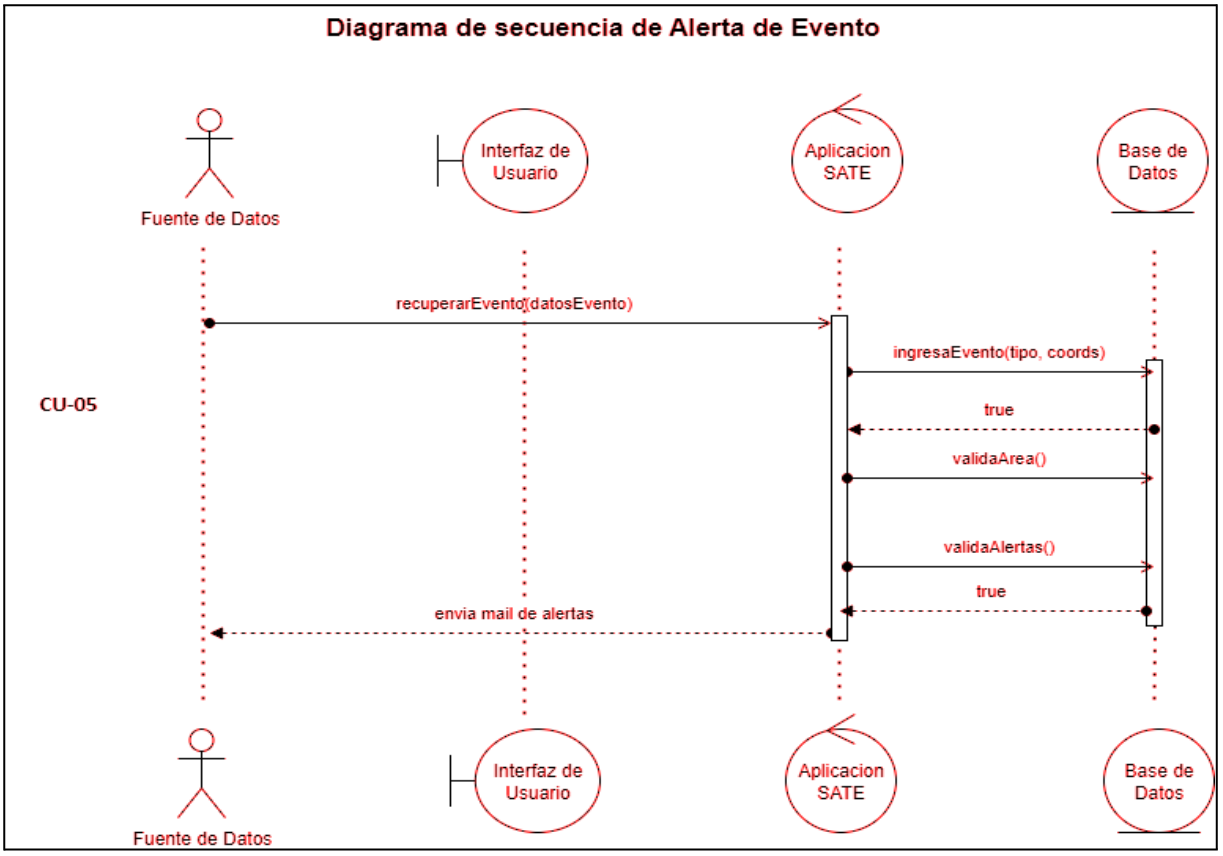


Diagrama de secuencia Reportar Evento Manualmente

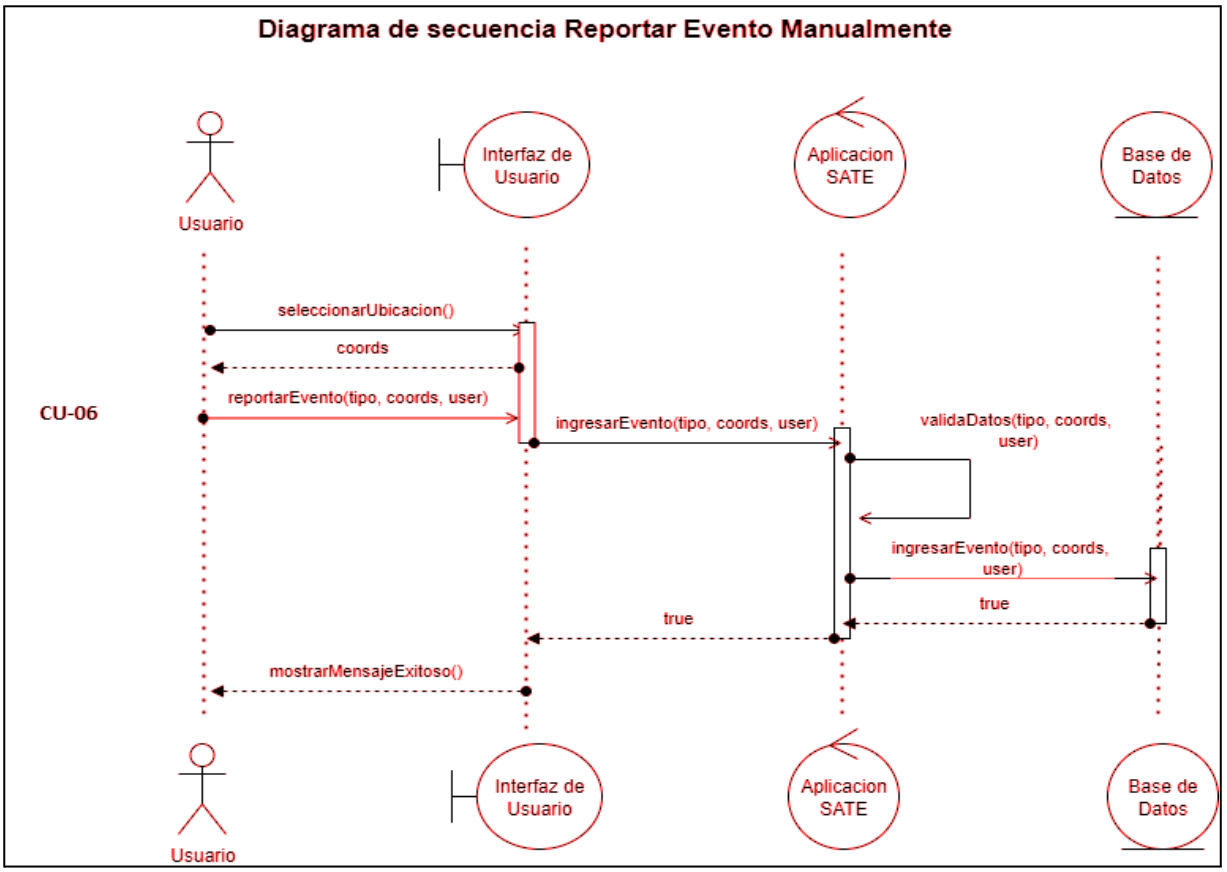


Diagrama de secuencia Gestionar Usuarios

CU-07

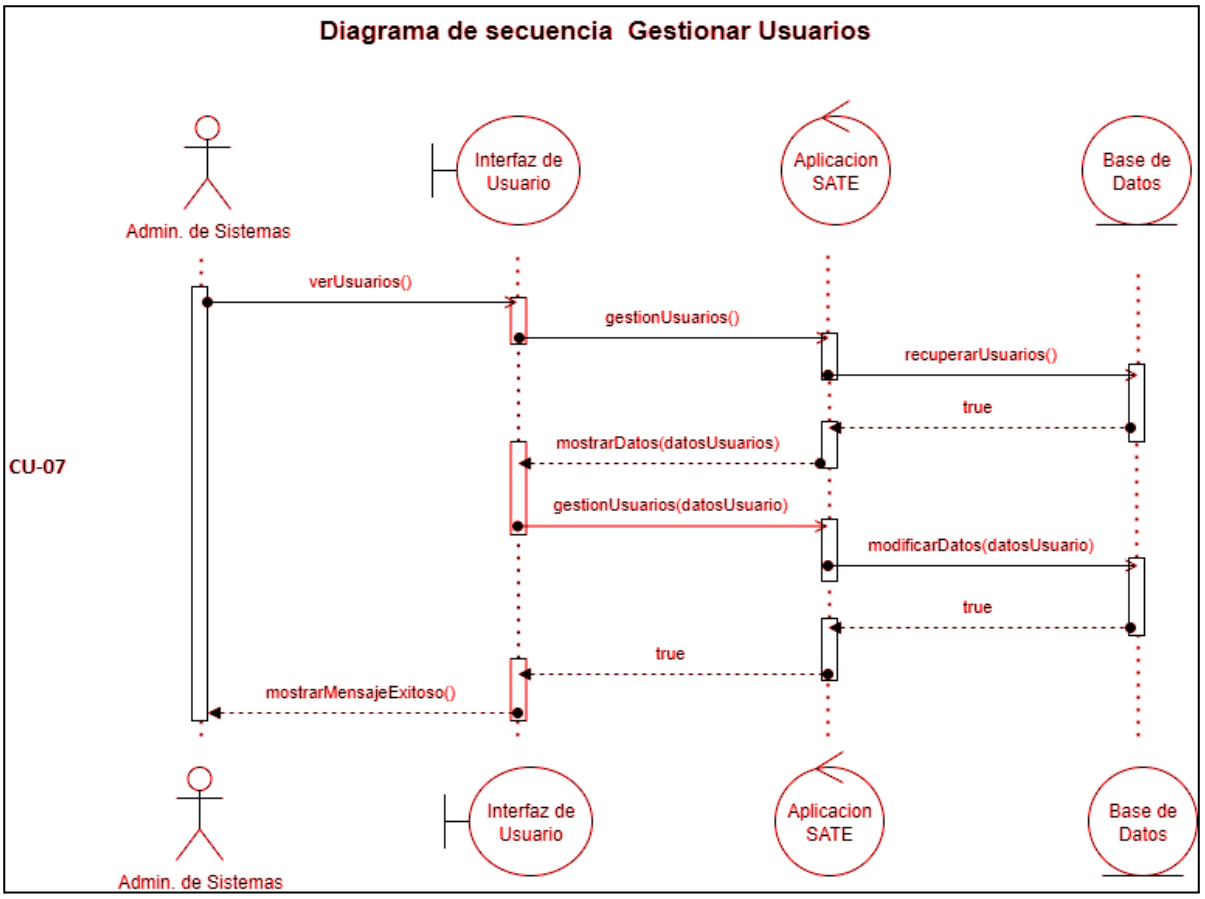
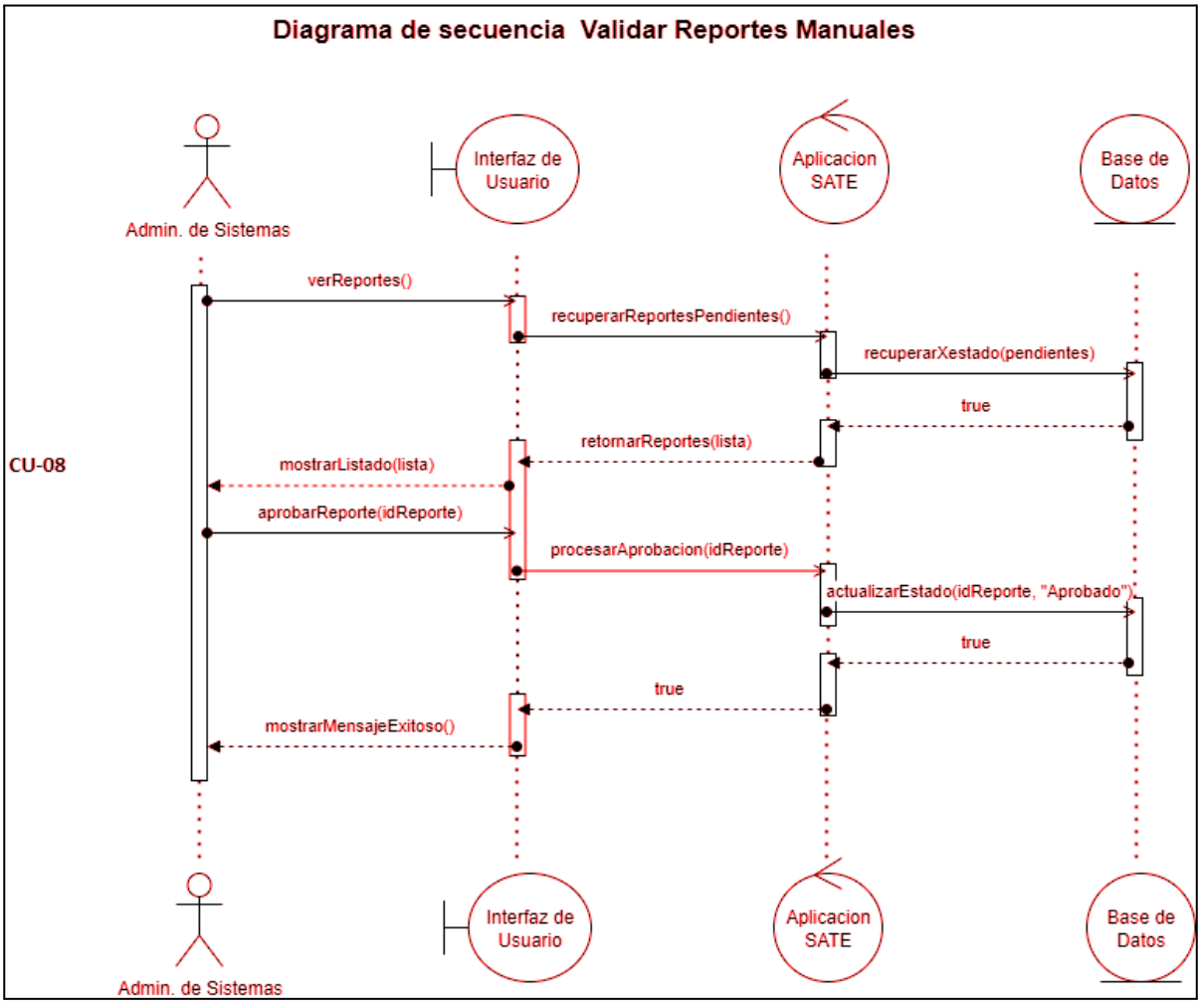


Diagrama de secuencia Validar Reportes Manuales

CU-08



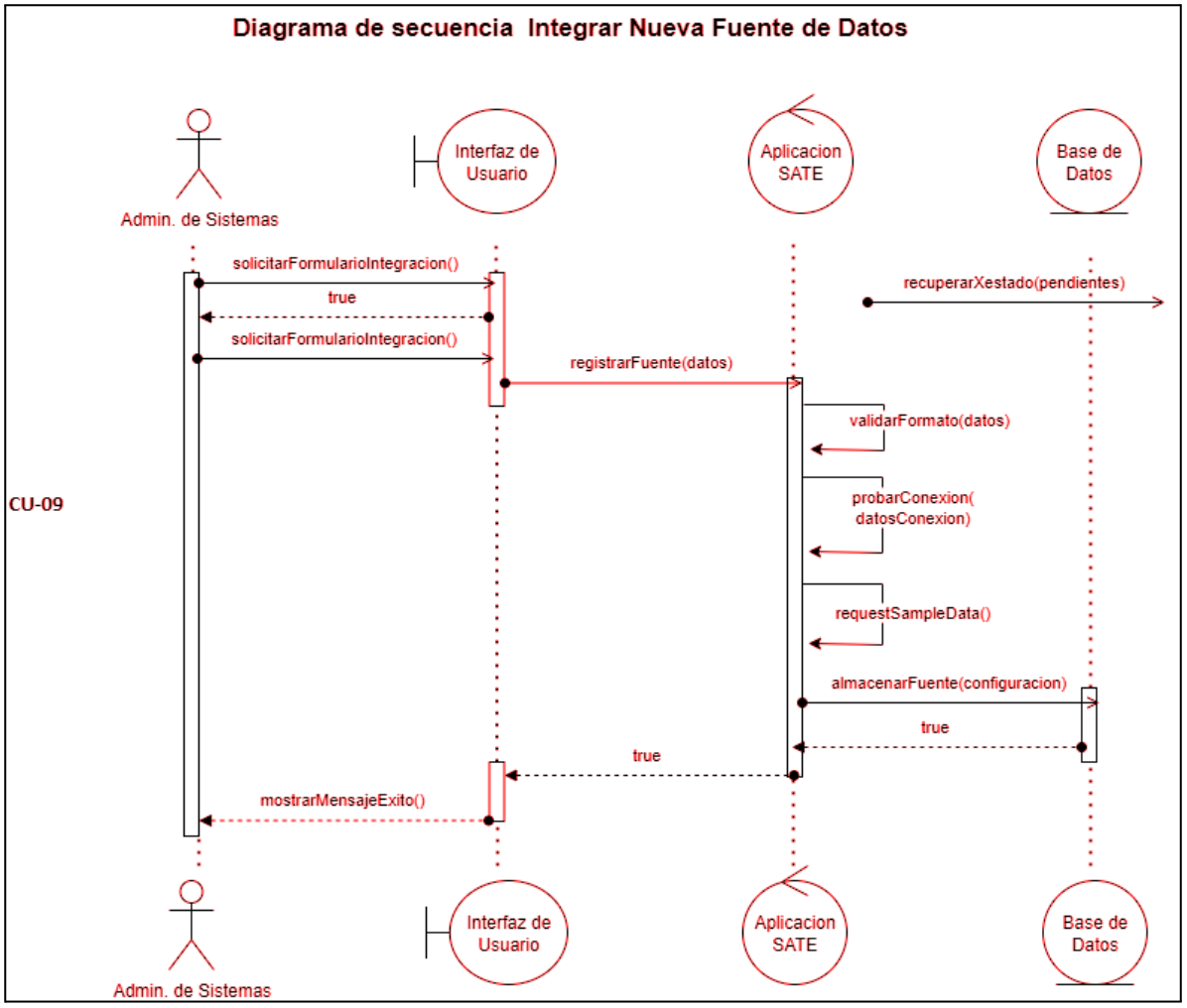


Diagrama de secuencia Integrar Nueva Fuente de Datos
CU-09
Admin. de Sistemas
Interfaz de Usuario
Aplicacion SATE
Base de Datos
solicitarFormularioIntegracion()
true
solicitarFormularioIntegracion()
registrarFuente(datos)
recuperarXestado(pendientes)
validarFormato(datos)
probarConexion(datosConexion)
requestSampleData()
almacenarFuente(configuracion)
true
true
mostrarMensajeExito()
Admin. de Sistemas
Interfaz de Usuario
Aplicacion SATE
Base de Datos

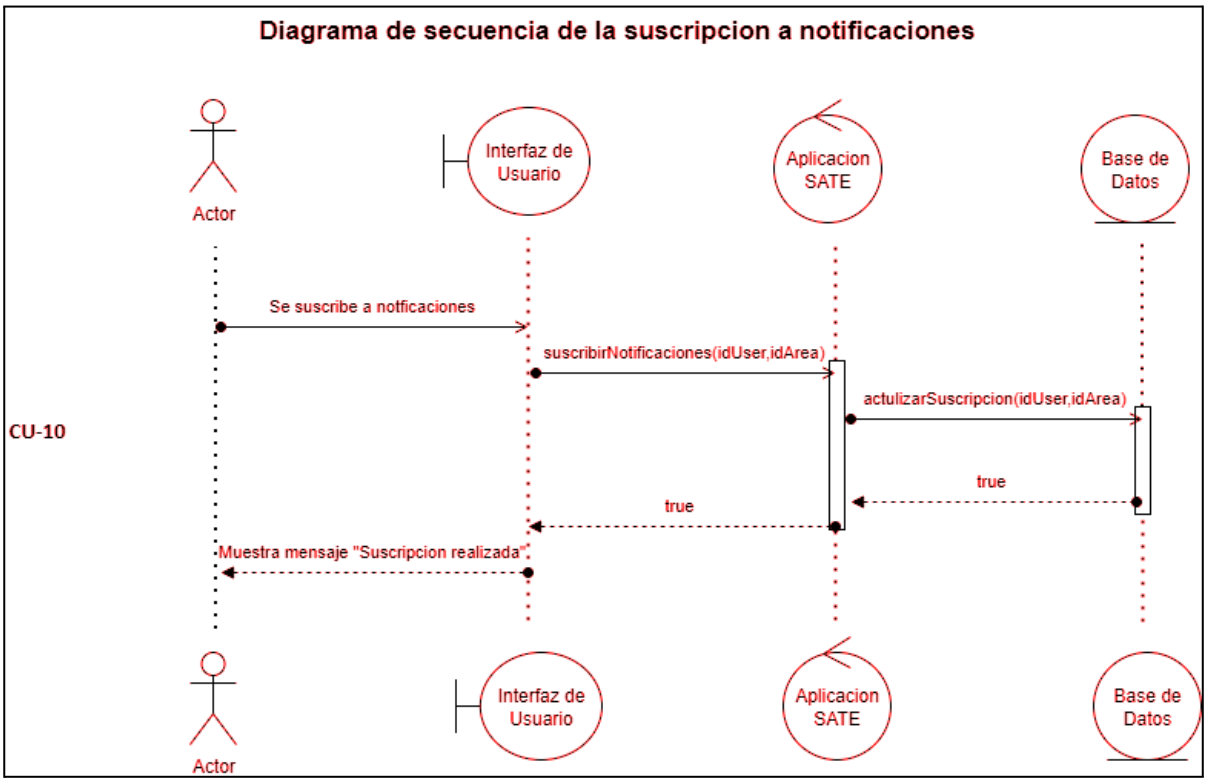
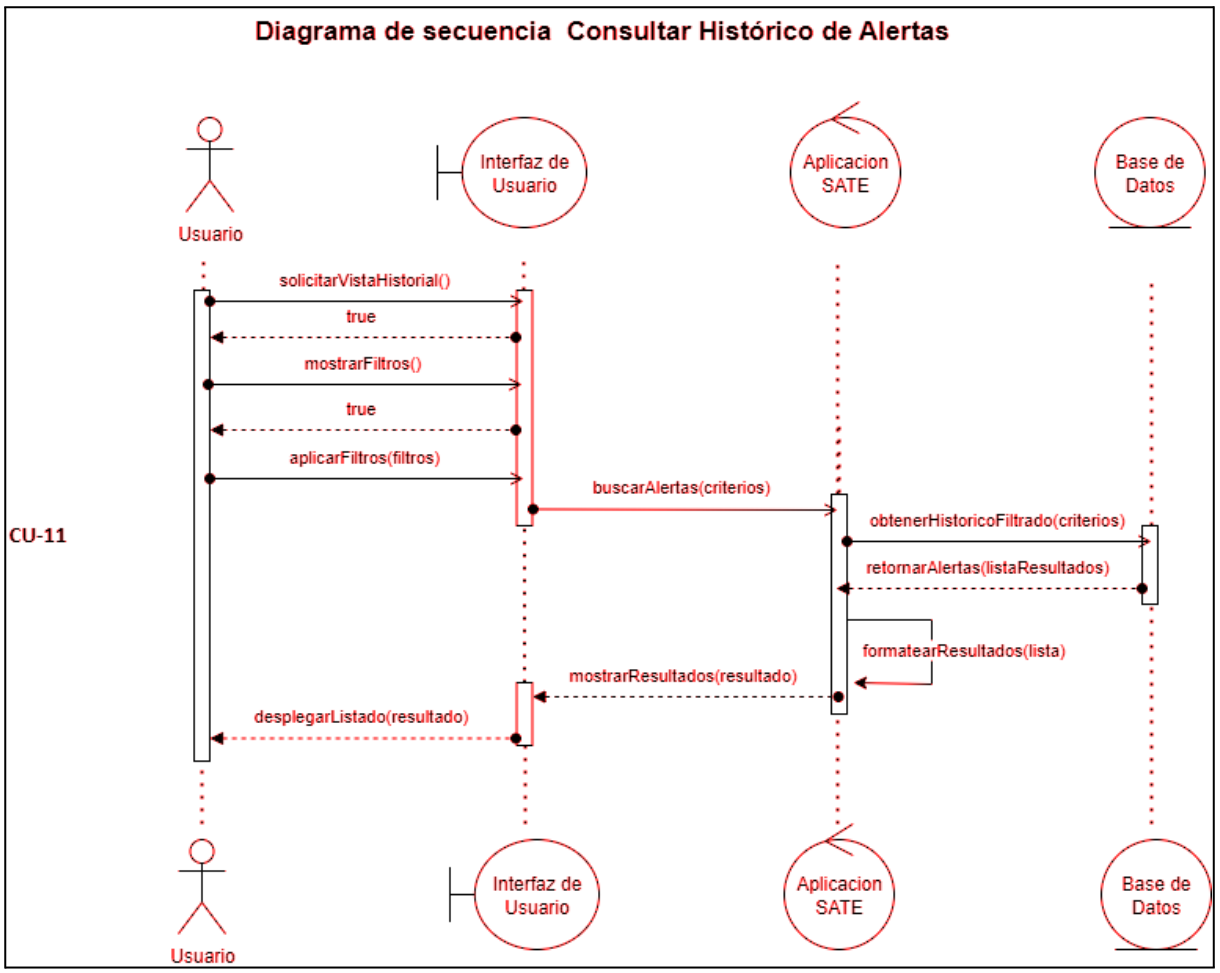


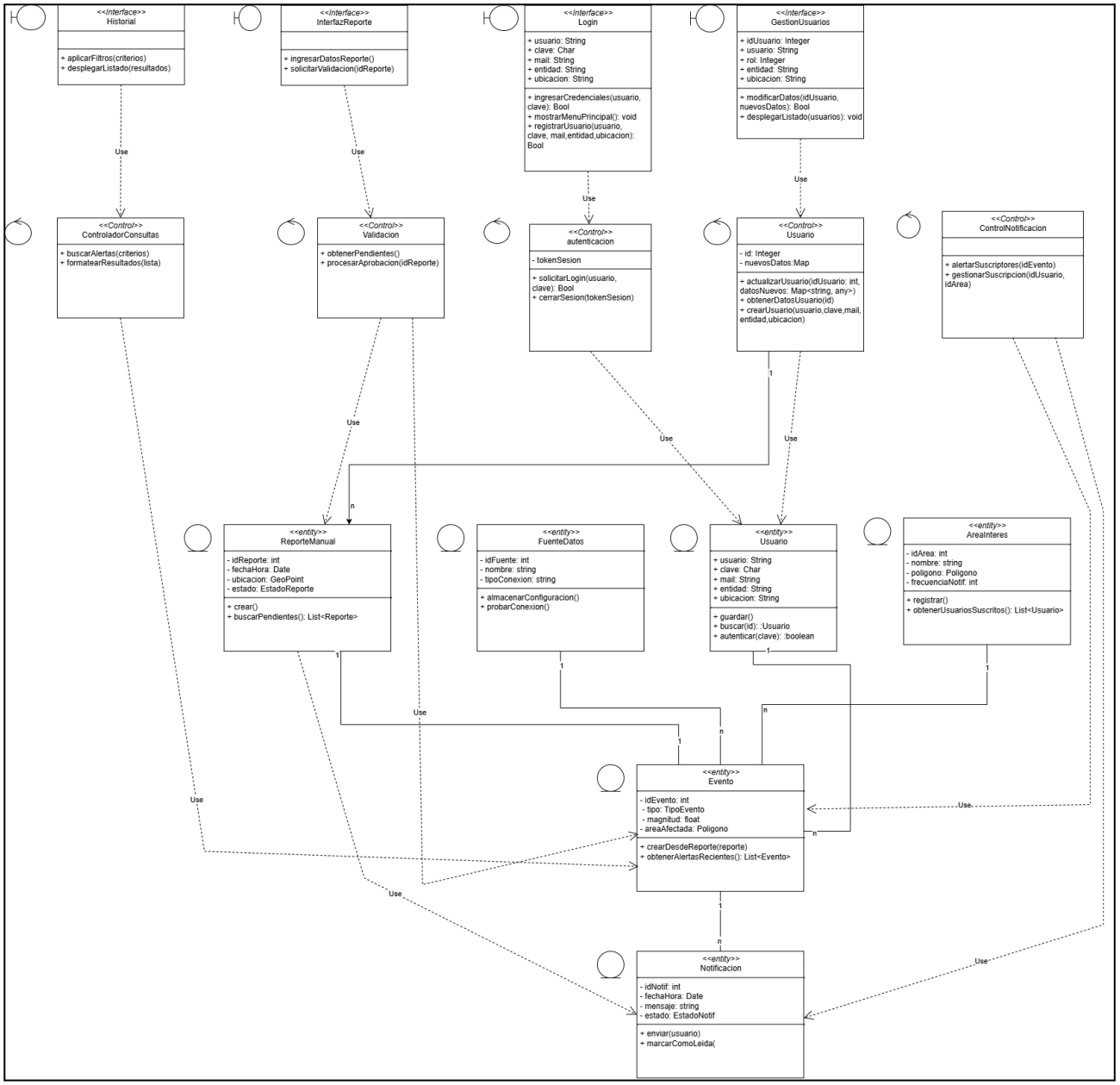
Diagrama de secuencia de la suscripcion a notificaciones
CU-10
Actor
Interfaz de Usuario
Aplicacion SATE
Base de Datos
Se suscribe a notficaciones
suscribirNotificaciones(idUser,idArea)
actulizarSuscripcion(idUser,idArea)
true
true
Muestra mensaje "Suscripcion realizada"
Actor
Interfaz de Usuario
Aplicacion SATE
Base de Datos



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)**, las clases se han categorizado mediante estereotipos (*<<entity>>*, *<<control>>*, *<<boundary>>*) para garantizar 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 SATE.



Etapa de Implementación

Modelo de pruebas

El objetivo general de la prueba es garantizar que el Sistema SATE 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)

Caso de Uso 1: Registrarse en el Sistema

El objetivo de esta prueba es asegurar la correcta creación de una cuenta de usuario, incluyendo la validación de campos, la verificación de unicidad del correo electrónico, la encriptación de la contraseña y la persistencia de datos en la base de datos.

Nivel: Pruebas Unitarias

Objetivo	Asegurar que los métodos de validación de negocio y las funciones de criptografía (simuladas) funcionen correctamente de forma aislada.
Contexto	Se prueban métodos esenciales de la Capa de Negocio, asumiendo que las clases **Usuario** y **ControladorUsuario** implementan la lógica de validación.

ID Caso de Prueba	Componente/ Método	Precondiciones	Pasos de Prueba	Resultado Esperado
PU-USU-001	ControladorUsuario.validarEmail(email)	El email de entrada es nuevo@sate.com.	Llamar al método validarEmail() con el email de prueba.	El método **retorna true** si la dirección tiene un formato válido (contiene @ y un dominio, siguiendo la lógica de la expresión regular definida).

PU-PASS-002	Usuario.encriptarContraseña(password)	Contraseña de entrada: MiClaveSegura123.	Invocar el método que realiza el *hashing* de la contraseña.	El método retorna una **cadena de *hash*** (simulación de SHA-256/bcrypt) de longitud fija y que no coincide con la cadena de entrada original.
PU-DATOS-003	ControladorUsuario.validarCampos(datos)	Se pasan datos donde el campo apellido está vacío (null o "").	Llamar al método que verifica la completitud de los campos obligatorios.	El método retorna **false** y lanza una excepción o un código de error que indica **"Campos Incompletos"** (Flujo Alternativo).

Nivel: Pruebas de Integración

Objetivo	Verificar la comunicación entre la Capa de Negocio (ControladorUsuario) y la Capa de Datos (DBConnector) para asegurar la unicidad y la persistencia del nuevo registro.
Contexto	Se valida el flujo de persistencia que involucra la inserción y la consulta de existencia del email antes de insertar, utilizando los métodos de DBConnector.

ID Caso de Prueb	Componentes Integrados	Precondiciones	Pasos de Prueba	Resultado Esperado

a				
PI-REG-DB-001	ControladorUsuario, DBConnector	La conexión a la BBDD está activa. El email test@sate.com no existe previamente en la tabla Usuario.	1. ControladorUsuario.registrarUsuario(...) es invocado. 2. Se invoca DBConnector.verificarEmailExistente(). 3. Se invoca DBConnector.insertarUsuario().	1. verificarEmailExistente() retorna **false**. 2. insertarUsuario() retorna **true**. El registro existe en la tabla Usuario y el campo contraseña está encriptado.
PI-REG-DB-002	ControladorUsuario, DBConnector	La conexión a la BBDD está activa. El email existente@sate.com ya se encuentra en la tabla Usuario.	1. ControladorUsuario.registrarUsuario(...) es invocado. 2. Se invoca DBConnector.verificarEmailExistente().	1. verificarEmailExistente() retorna **true**. 2. El ControladorUsuario lanza una excepción (o retorna un código de error) de **"Correo Electrónico ya en uso"** (Flujo Alternativo).

Nivel: Pruebas de Aceptación

Objetivo	Confirmar que el proceso de registro completo (CU-01) es usable para el **Usuario No Registrado** y que las notificaciones del sistema (Sistema) funcionan como se espera (Flujo Normal y Excepciones).
Contexto	Se simulan las acciones de un usuario nuevo en la interfaz RegisterScreen y se valida la respuesta del sistema al cumplir o fallar con los criterios de los campos.

ID Caso de Prueba	Escenario de Negocio	Precondiciones	Pasos de Prueba (Realizados por Usuario Final/Stakeholder)	Resultado Esperado (Validación de Negocio)
PA-FLUJO-001	Registro Exitoso	La cuenta no existe. El sistema de correo (simulado) está activo.	1. Acceder a la RegisterScreen. 2. Llenar todos los campos con datos válidos y únicos. 3. Presionar "Registrarse".	El sistema muestra un **mensaje de éxito** (Postcondición). El usuario recibe un **correo electrónico de confirmación** (Verificación del Flujo Normal).
PA-FLUJO-002	Validación de Unicidad	El email test@sate.com ya existe en la BBDD.	1. Acceder a la RegisterScreen. 2. Intentar registrar una nueva cuenta usando test@sate.com. 3. Presionar "Registrarse".	El sistema **no permite el registro** y muestra un **mensaje de error claro** indicando **"El correo electrónico ya se encuentra registrado"** (Flujo Alternativo).
PA-FLUJO-003	Obligatoriedad de Campos	El sistema tiene habilitada la validación de campos obligatorios.	1. Acceder a la RegisterScreen. 2. Dejar el campo **"Contraseña"** vacío. 3. Presionar "Registrarse".	El sistema **resalta el campo faltante** (Contraseña) y muestra una advertencia sin procesar el envío de datos (Flujo Alternativo).

Caso de Uso 2: Autenticarse en el Sistema

El objetivo de esta prueba es garantizar la seguridad y el acceso funcional al sistema, verificando que solo los **Usuarios Registrados** puedan iniciar sesión utilizando credenciales válidas y que los flujos alternativos (errores de credenciales y campos incompletos) sean gestionados correctamente.

Nivel: Pruebas Unitarias

Objetivo	Asegurar que los métodos de validación de entrada (campos obligatorios) y el método de comparación de contraseñas *hash* funcionen de forma aislada.
Contexto	Se prueban métodos de bajo nivel en las clases Usuario y ControladorUsuario antes de interactuar con la Capa de Datos.

ID Caso de Prueba	Componente/ Método	Precondiciones	Pasos de Prueba	Resultado Esperado
PU-LOGIN-001	ControladorUsuario.validarLogin(datos)	Los datos de entrada contienen la contraseña como null o cadena vacía ("").	Llamar al método validarLogin() con campos incompletos.	El método retorna **false** o lanza una excepción/código de error que indica **"Campos Incompletos"** (Flujo Alternativo/Excepción).
PU-PASS-002	Usuario.verificarContraseña(input, storedHash)	input (contraseña ingresada) es "Clave123"; storedHash es el *hash* de "Clave123".	Llamar al método de comparación de *hashes*.	El método **retorna true**, confirmando la coincidencia de la contraseña ingresada con el *hash* almacenado.

PU-PASS-003	Usuario.verificarContraseña(input, storedHash)	input (contraseña ingresada) es "ClaveErronea"; storedHash es el *hash* de "ClaveCorrecta".	Llamar al método de comparación de *hashes*.	El método **retorna false**, indicando un fallo de autenticación (**Usuario o contraseña inválido** - Excepción).

Nivel: Pruebas de Integración

Objetivo	Verificar la comunicación entre la Capa de Presentación, la Capa de Negocio (ControladorUsuario) y la Capa de Datos (DBConnector) para el proceso de autenticación.
Contexto	Se valida la secuencia completa de la verificación de credenciales, desde que la Capa de Presentación envía la solicitud hasta que se confirma la existencia del usuario y la validez de la contraseña.

ID Caso de Prueba	Componentes Integrados	Precondiciones	Pasos de Prueba	Resultado Esperado
PI-AUTH-DB-001	LoginScreen, ControladorUsuario, DBConnector	El email test@sate.com existe en la DB con un *hash* de contraseña válido.	1. La LoginScreen envía test@sate.com y la contraseña correcta a ControladorUsuario.autenticar(). 2. ControladorUsuario consulta al DBConnector.	DBConnector recupera el *hash*. ControladorUsuario verifica el *hash* y retorna el objeto **Usuario** (o un token de sesión).

PI-AUTH-DB-002	LoginScreen, ControladorUsuario, DBConnector	El email inexistente@sate.com no existe en la tabla Usuario.	1. La LoginScreen envía el email inexistente y una contraseña cualquiera a ControladorUsuario.autenticar(). 2. ControladorUsuario consulta al DBConnector.	DBConnector retorna null o un código de "usuario no encontrado". ControladorUsuario lanza una excepción de **"Usuario o contraseña inválido"** (Excepción).
PI-AUTH-DB-003	LoginScreen, ControladorUsuario, DBConnector	El email test@sate.com existe, pero la contraseña ingresada es incorrecta.	1. ControladorUsuario.autenticar() recupera el *hash* del email. 2. La función de verificarContraseña() compara el *hash* y falla.	ControladorUsuario lanza una excepción de **"Usuario o contraseña inválido"** (Excepción), **sin especificar** si falló el usuario o la contraseña por motivos de seguridad.

Nivel: Pruebas de Aceptación

Eje	Descripción
Objetivo	Confirmar que el flujo de autenticación es rápido, seguro y lleva al usuario a la interfaz correcta según su rol (por ejemplo, AdminScreen o MainDashboard).
Contexto	Se simula el uso por parte de usuarios con diferentes roles para validar la segregación de acceso y la usabilidad de la interfaz LoginScreen.

ID Caso de Prueba	Escenario de Negocio	Precondiciones	Pasos de Prueba (Realizados por Usuario Final/Stakeholder)	Resultado Esperado (Validación de Negocio)
PA-AUTH-001	Inicio de Sesión Exitoso (Rol Básico)	El usuario ciudadano@sate.com está registrado y activo.	1. Acceder a la LoginScreen. 2. Ingresar credenciales correctas. 3. Presionar "Iniciar Sesión".	El sistema **cierra la LoginScreen** y abre la vista principal del **MainDashboard** (Flujo Normal).
PA-AUTH-002	Inicio de Sesión Exitoso (Rol Admin)	El usuario admin@sate.com está registrado con rol Administrador.	1. Acceder a la LoginScreen. 2. Ingresar credenciales correctas. 3. Presionar "Iniciar Sesión".	El sistema **cierra la LoginScreen** y abre la vista del **AdminScreen** (Validación de la lógica de roles).
PA-AUTH-003	Credenciales Inválidas	El usuario ingresa un email registrado, pero la contraseña es incorrecta.	1. Ingresar email correcto y contraseña incorrecta. 2. Presionar "Iniciar Sesión".	El sistema **no avanza de pantalla** y muestra un **único mensaje de error** ("Usuario o contraseña inválido") sin exponer el motivo exacto del fallo (Excepción).

PA-AUTH-004	Usabilidad de la Pantalla	Usuario de prueba.	1. Acceder a la LoginScreen. 2. No ingresar nada en el campo contraseña y hacer clic en el botón de "Iniciar Sesión".	El sistema **resalta** el campo de la contraseña y muestra un mensaje solicitando completar el dato, **sin lanzar un error de conexión** (Validación de la Capa de Presentación).

Caso de Uso 3: Visualizar Mapa de Alertas

El Caso de Uso 3 (CU-03: Visualizar Mapa de Alertas) es fundamental, ya que valida el corazón geoespacial del sistema y la Capa de Presentación.
El objetivo de esta prueba es asegurar la correcta recuperación de eventos desde la base de datos y su representación gráfica precisa y eficiente en el componente GIS, cumpliendo con los requisitos de performance (RNF-04).

Nivel: Pruebas Unitarias

Objetivo	Asegurar que la entidad Evento pueda exponer sus coordenadas para el dibujo del mapa y que el componente gráfico pueda manejar escenarios de datos nulos.
Contexto	Pruebas aisladas en la entidad Evento y en el GISMapPanel (simulado) antes de la interacción con el controlador o la BBDD.

ID Caso de Prueba	Componente/ Método	Precondiciones	Pasos de Prueba	Resultado Esperado

PU-EVT-001	Evento.getCoordenadas()	Se instancia un objeto Evento donde areaAfectada es POINT(-65.01 -40.50) y tiene el método de extracción de coordenadas.	Llamar al método getCoordenadas() del objeto Evento.	El método retorna un *array* de strings ["-65.01", "-40.50"], listo para ser consumido por la lógica de dibujo del mapa.
PU-GIS-002	GISMapPanel.setEventos(list)	El GISMapPanel es inicializado. Se le pasa una lista de Evento completamente vacía.	Llamar a setEventos([]) y forzar el repintado (paintComponent()).	El panel no dibuja marcadores y muestra la cadena **"No hay eventos activos para mostrar."** (Validación del Flujo Alternativo).

Nivel: Pruebas de Integración

Objetivo	Verificar el flujo de datos completo para la visualización: desde la solicitud de la interfaz hasta la recuperación y transferencia de los datos de la base de datos.
Contexto	Se enfoca en la interacción entre la Capa de Presentación (MainDashboard), la Capa de Negocio (ControladorEvento) y la Capa de Datos (DBConnector) para obtener los eventos.

ID Caso de	Componentes Integrados	Precondiciones	Pasos de Prueba	Resultado Esperado

Prueba				
PI-VIEW-001	MainDashboard, ControladorEvento, DBConnector	La BBDD contiene 50 registros de Evento activos. La conexión está activa.	El MainDashboard invoca ControladorEvento.obtenerEventosActivos() al cargar.	ControladorEvento retorna la lista completa (50 objetos Evento). El MainDashboard recibe la lista y la pasa con éxito al GISMapPanel.
PI-VIEW-002	DBConnector, ControladorEvento	Se simula un error de conectividad o una caída temporal de la BBDD.	El MainDashboard invoca ControladorEvento.obtenerEventosActivos().	DBConnector lanza una SQLException. ControladorEvento debe capturar la excepción y retornar una lista vacía, informando el error a la Capa de Presentación para que muestre un mensaje de **"Error de carga de datos"**.
PI-VIEW-003	MainDashboard, ControladorEvento, GISMapPanel	ControladorEvento retorna una lista de 5 eventos con coordenadas válidas.	1. MainDashboard recibe la lista y la pasa a GISMapPanel.setEventos(). 2. Se verifica que la tabla (JTable) esté poblada.	Los datos de la tabla deben coincidir con la lista. El GISMapPanel debe invocar repaint() y la simulación gráfica de los 5 marcadores debe ser visible

				en el panel.

Nivel: Pruebas de Aceptación

Objetivo	Validar que la interfaz de usuario es efectiva para la toma de decisiones, que el sistema cumple con el RNF-04 (Rendimiento) y que la interacción con los elementos GIS es funcional.
Contexto	Se prueban los flujos de interacción del usuario con el MainDashboard y la funcionalidad de filtrado dinámico.

ID Caso de Prueba	**Escenario de Negocio**	**Precondiciones**	**Pasos de Prueba (Realizados por Usuario Final/Stakeholder)**	**Resultado Esperado (Validación de Negocio)**
PA-GIS-001	Velocidad de Carga (RNF-04)	Usuario autenticado. La BBDD tiene 100 eventos activos.	1. El usuario accede a la sección de mapa (MainDashboard). 2. Se mide el tiempo transcurrido hasta que **todos** los 100 marcadores están dibujados en el mapa.	El tiempo de carga y dibujo **no debe superar los 3 segundos** (Validación crítica del RNF-04, que requiere una respuesta ágil).

PA-GIS-002	Flujo de Visualización Detallada	El mapa está cargado con múltiples marcadores.	1. El actor hace **clic** sobre un marcador de evento en el GISMapPanel.	La interfaz muestra un cuadro de diálogo o un panel adyacente que contiene al menos 3 campos de detalle (ej: "Hora de Detección", "Fuente", "Intensidad") correspondientes a ese evento (Flujo Normal, paso 4).
PA-GIS-003	Filtrado Dinámico y Coherencia (RF-03)	El MainDashboard está cargado con eventos de tipo A y B. El filtro está en "Todos".	1. El actor selecciona el filtro por "Tipo A" en los controles. 2. Presiona el botón "Aplicar Filtros".	La JTable solo muestra eventos de Tipo A. El **GISMapPanel refleja este cambio inmediatamente**, eliminando los marcadores de Tipo B (Validación de la coherencia entre las vistas y el RF-03).
PA-GIS-004	Carga en Escenario de Calma	El ControladorEvento confirma que no hay eventos activos en la BBDD.	1. El usuario accede al MainDashboard.	El GISMapPanel ocupa el espacio de la ventana pero muestra de forma clara, visible y legible el mensaje: **"No hay alertas activas en el momento."** (Validación del Flujo Alternativo).

Caso de Uso 4: Definir Área de Interés

El Caso de Uso 4 (CU-04: Definir Área de Interés) es crucial para la personalización del servicio de alertas. Requiere una validación rigurosa de la funcionalidad GIS (dibujo de polígonos) y la gestión de las reglas de negocio (límite de áreas).
El objetivo de esta prueba es asegurar que un usuario autenticado pueda definir y persistir áreas geográficas personalizadas, que las herramientas de dibujo GIS funcionen correctamente y que se cumplan las reglas de límite por usuario.

Nivel: Pruebas Unitarias

Objetivo	Asegurar la correcta generación del formato geoespacial WKT (POLYGON) a partir de las coordenadas de dibujo, y validar las reglas de negocio aisladas, como el límite de áreas.
Contexto	Se prueban los métodos de la entidad o clase utilitaria encargada del manejo de áreas geográficas (AreaInteres o lógica dentro del ControladorUsuario).

ID Caso de Prueba	Componente/ Método	Precondiciones	Pasos de Prueba	Resultado Esperado
PU-AREA-001	AreaInteres.generarWKT(coords)	El *input* son 4 pares de Lat/Lon válidos que definen un polígono cerrado (ej: [lon1 lat1, lon2 lat2, lon3 lat3, lon1 lat1]).	Llamar al método generarWKT() con los pares de coordenadas del polígono.	El método retorna la **cadena WKT válida** en el formato POLYGON((lon1 lat1, lon2 lat2, lon3 lat3, lon1 lat1)).
PU-LIMITE-002	ControladorUsuario.verificarLimiteAreas(user)	El usuario de prueba ya tiene definido el número máximo permitido de áreas de interés (regla: 5 áreas).	Llamar al método que verifica el límite de áreas para el usuario.	El método **retorna false** (o lanza una excepción de negocio) indicando que se alcanzó el límite.

PU-COORD-003	AreaInteres.validarGeometria(wkt)	El WKT de entrada es inválido (ej: solo dos puntos, no un polígono cerrado).	Invocar el validador de geometría.	El método **retorna false**, indicando que la geometría no cumple con el requisito de ser un polígono válido (RNF-05).

Nivel: Pruebas de Integración

Objetivo	Verificar el flujo de persistencia completo: desde la Capa de Negocio (ControladorUsuario) que recibe el WKT, hasta la Capa de Datos (DBConnector) que lo inserta, asegurando el vínculo con el usuario.
Contexto	Se valida la interacción entre el controlador y la base de datos para la gestión de las áreas de interés.

ID Caso de Prueba	Componentes Integrados	Precondiciones	Pasos de Prueba	Resultado Esperado
PI-AREA-DB-001	ControladorUsuario, DBConnector	Conexión activa. El objeto AreaInteres contiene un POLYGON WKT válido y un nombre.	1. ControladorUsuario.guardarAreaInteres(area, user) es invocado. 2. DBConnector.insertarAreaInteres() es invocado.	DBConnector retorna **true** (éxito en la inserción). El nuevo registro en la tabla de la BBDD está **vinculado al ID del usuario** y el campo de coordenadas

				contiene el WKT.
PI-AREA-DB-002	ControladorUsuario, DBConnector	El usuario ID=201 tiene 4 áreas guardadas previamente en la BBDD.	1. ControladorUsuario.obtenerAreasDeInteres(ID=201) es invocado. 2. La Capa de Negocio procesa la información devuelta por DBConnector.	El ControladorUsuario retorna una **Lista (List<AreaInteres>) con exactamente 4 objetos** y sus respectivas coordenadas WKT.
PI-AREA-DB-003	ControladorUsuario, DBConnector	DBConnector simula un error de constraint (ej: nombre de área de interés duplicado para el mismo usuario).	ControladorUsuario.guardarAreaInteres(area, user) es invocado.	DBConnector lanza una SQLException. ControladorUsuario la captura y la propaga como un error de negocio que indica **"El nombre del área ya está en uso"**.

Nivel: Pruebas de Aceptación

Objetivo	Confirmar la usabilidad de las herramientas de dibujo GIS, la claridad de la respuesta del sistema al guardar, y la correcta aplicación del Flujo Alternativo (Límite de Áreas).
Contexto	Se prueban las interacciones del usuario final en la interfaz gráfica (simulación del mapa interactivo) para la creación de un área.

ID Caso de Prueba	Escenario de Negocio	Precondiciones	Pasos de Prueba (Realizados por Usuario Final/Stakeholder)	Resultado Esperado (Validación de Negocio)
PA-AREA-001	Creación de Polígono Exitoso	Usuario autenticado. El usuario no ha alcanzado el límite de áreas.	1. Navegar a la sección de configuración de alertas. 2. Usar las herramientas de dibujo para trazar un **polígono de 5 vértices**. 3. Ingresar un nombre ("Mi Empresa"). 4. Presionar "Guardar Área".	El sistema muestra un **mensaje de confirmación** ("Área guardada con éxito"). El polígono dibujado se **mantiene visible** en el mapa y aparece en la lista de áreas del usuario.
PA-AREA-002	Validación de Límite Alcanzado	Usuario autenticado. El usuario tiene **5 áreas** guardadas (límite máximo simulado).	1. Navegar a la sección de configuración. 2. Intentar dibujar una sexta área y presionar "Guardar Área".	El sistema **no guarda el área** y muestra un mensaje de advertencia claro en la interfaz: **"Límite de Áreas Alcanzado: 5/5. No puede añadir más."** (Validación del Flujo Alternativo).
PA-AREA-003	Usabilidad de Herramientas GIS	Usuario autenticado.	1. El actor selecciona la herramienta de dibujo (polígono). 2. El actor intenta presionar el botón "Guardar Área" **antes de cerrar el polígono** (antes de que la geometría sea válida).	El botón **"Guardar Área"** permanece **deshabilitado** o el sistema muestra un mensaje de error que guía al usuario a **cerrar la figura** antes de la confirmación.

Caso de Uso 5: Alerta de Evento

El Caso de Uso 5 (CU-05: Alerta de Evento) se centra en la lógica de backend y el proceso automático de geocercado para enviar alertas personalizadas. Es crítico para validar la funcionalidad geoespacial (RF-10) y el rendimiento (RNF-04).

El objetivo de esta prueba es asegurar la ejecución correcta y eficiente del proceso de geocercado, garantizando que solo los usuarios cuya área de interés es intersectada por un nuevo evento reciban una notificación.

Nivel: Pruebas Unitarias

Objetivo	Asegurar la precisión de la lógica de geocercado (punto-en-polígono) y la función de envío de notificaciones de forma aislada.
Contexto	Se prueban métodos utilitarios críticos, como el servicio de geocercado y el servicio de notificación (ambos simulados en la Capa de Negocio).

ID Caso de Prueba	Componente/ Método	Precondiciones	Pasos de Prueba	Resultado Esperado
PU-GEO-001	GeocercadoService.isInside(eventWKT, areaWKT)	eventWKT es un POINT con coordenadas interiores al areaWKT (POLYGON) de prueba.	Llamar a isInside() con coordenadas que caen **dentro** del polígono.	La función **retorna true**, confirmando la coincidencia del área de interés.
PU-GEO-002	GeocercadoService.isInside(eventWKT, areaWKT)	eventWKT es un POINT con coordenadas exteriores al areaWKT (POLYGON) de prueba.	Llamar a isInside() con coordenadas que caen **fuera** del polígono.	La función **retorna false**, indicando que el evento no intersecta el área de interés (Flujo Alternativo).

PU-NOT-003	NotificacionService.enviarAlerta(details, email)	El email de destino es válido. El detalle del evento incluye tipo, ubicación y hora.	Invocar enviarAlerta() con datos simulados y verificar el formato del mensaje.	El método **retorna true** (simulación de envío exitoso). El cuerpo del correo (simulado) incluye los detalles requeridos de la alerta (ubicación del evento, hora, área afectada).

Nivel: Pruebas de Integración

Objetivo	Verificar la ejecución secuencial del proceso de alerta: desde la detección del evento hasta la activación de la notificación, pasando por la consulta de áreas de interés.
Contexto	Se valida la interacción completa entre el ControladorEvento, el DBConnector y el servicio de Geocercado al detectar un nuevo evento.

ID Caso de Prueba	Componentes Integrados	Precondiciones	Pasos de Prueba	Resultado Esperado
PI-ALERT-001	ControladorEvento, DBConnector, GeocercadoService	BBDD contiene 10 áreas de interés. Se inyecta un nuevo Evento que	1. ControladorEvento.procesarNuevoEvento(evento) es invocado. 2. DBConnector recupera las 10 áreas.	El ControladorEvento identifica al **Usuario A** y **solo invoca** la función de envío de notificación para su email.

		coincide con el área del **Usuario A**.	3. El controlador itera y llama a GeocercadoService.isInside() 10 veces.	
PI-ALERT-002	ControladorEvento, DBConnector, NotificacionService	Evento detectado que **no** coincide con ninguna de las áreas guardadas en la BBDD.	1. ControladorEvento.procesarNuevoEvento() es invocado. 2. La lógica de geocercado finaliza sin encontrar coincidencias.	El sistema **no invoca** el NotificacionService.enviarAlerta() para ningún usuario. El evento se persiste en la tabla Evento y está listo para ser mostrado en el mapa general (Flujo Alternativo).
PI-ALERT-003	ControladorEvento, DBConnector	El DBConnector lanza una SQLException al intentar recuperar la lista de áreas de interés.	ControladorEvento.procesarNuevoEvento() es invocado.	El sistema registra el error en un *log* interno (**RNF-04** - el proceso no debe detenerse por una falla de *logging* o DB temporal) y **no envía notificaciones**, pero el evento debe ser guardado en la tabla Evento.

Nivel: Pruebas de Aceptación

Objetivo	Validar que el usuario reciba la alerta proactiva y que el sistema maneje los escenarios de coincidencia y no coincidencia de acuerdo con las expectativas del negocio (RF-10).
Contexto	Se utiliza un simulador de detección de eventos y se verifican las bandejas de entrada de los usuarios de prueba.

ID Caso de Prueba	Escenario de Negocio	Precondiciones	Pasos de Prueba (Realizados por Usuario Final/Stakeholder)	Resultado Esperado (Validación de Negocio)
PA-ALERTA-001	Notificación Proactiva Exitosa	El **Usuario A** tiene una AreaInteres definida. El sistema está funcionando.	1. Se registra la detección de un nuevo evento que **cae dentro** del área del Usuario A. 2. El actor verifica el correo electrónico del Usuario A.	El **Usuario A** recibe la alerta personalizada en un tiempo aceptable (validando RNF-04). La alerta incluye la hora, la ubicación y el tipo de riesgo.
PA-ALERTA-002	Foco Fuera de Área (No Notificación)	El **Usuario B** tiene una AreaInteres definida. Se registra un evento que **cae fuera** del área del Usuario B.	1. El actor registra la detección del evento. 2. Se verifica la bandeja de entrada del Usuario B.	El **Usuario B no recibe** la notificación personalizada. El evento sí debe ser visible en el **mapa general** del MainDashboard (Validación del Flujo Alternativo).
PA-ALERTA-003	Múltiples Coincidencias	Los Usuarios C y D tienen áreas de interés que **coinciden** con el mismo evento.	1. Se registra la detección del evento. 2. Se verifican las bandejas de entrada de C y D.	Tanto el **Usuario C** como el **Usuario D** reciben la alerta personalizada de forma independiente y concurrente (Validación del RF-10 y la escalabilidad del servicio).

PA-ALERTA-004	Fallo de Notificación (Excepción)	Se activa la excepción de envío de correo electrónico.	1. El sistema intenta enviar una alerta (simulando un fallo). 2. El actor revisa el estado del sistema.	La alerta **no llega** al usuario. El sistema registra el fallo del envío en su bitácora (log) y la función del ControladorEvento retorna un estado de "Fallo en el envío", pero el **evento se marca como procesado** internamente.

Caso de Uso 6: Reportar Evento Manualmente

El Caso de Uso 6 (CU-06: Reportar Evento Manualmente) valida el punto de entrada de información asistida por el usuario, siendo crucial para la integridad de los datos y la posterior validación por parte de un administrador.

El objetivo de esta prueba es asegurar que los usuarios puedan ingresar reportes de eventos de manera exitosa y que el sistema valide los datos requeridos, guardando el reporte en la base de datos con el estado inicial de **"Pendiente de Validación"**.

Nivel: Pruebas Unitarias

Objetivo	Asegurar que los métodos de la Capa de Presentación generen correctamente la geometría WKT (POINT) y que la Capa de Negocio pueda verificar los datos obligatorios antes de la persistencia.
Contexto	Pruebas aisladas en la lógica de **ReporteManual** y **ControladorEvento** para validar la estructura del dato y los campos requeridos.

ID Caso de Prueba	**Componente/ Método**	**Precondiciones**	**Pasos de Prueba**	**Resultado Esperado**

PU-REP-001	ReporteManualScreen.generarWKT()	Coordenadas de entrada válidas: Latitud = -34.60 y Longitud = -58.38.	Llamar al método de generación WKT con los datos.	El método retorna la cadena WKT **POINT(-58.38 -34.60)** (validando el formato WKT y el orden Lon/Lat).
PU-CTRL-002	ControladorEvento.validarReporte(data)	Los datos de entrada contienen la ubicación como null o vacía (""), pero el resto de campos están completos.	Llamar al método de validación de negocio.	El método retorna **false** o lanza una excepción/código de error que indica **"Ubicación obligatoria"** (Validación del Flujo Alternativo: Datos incompletos).
PU-REP-003	ReporteManual.asignarEstado()	Se inicializa un nuevo objeto ReporteManual para la base de datos.	Llamar a la lógica que asigna el estado inicial del reporte.	El objeto ReporteManual tiene el atributo estado asignado con el valor **"Pendiente de Validación"**.

Nivel: Pruebas de Integración

Objetivo	Verificar la secuencia completa de persistencia del reporte: desde que el ControladorEvento recibe el reporte hasta que DBConnector lo inserta en la tabla correcta con el estado de validación requerido.

Contexto	Se valida la interacción entre el controlador y la base de datos para la gestión del flujo normal y la persistencia de la geometría WKT.

ID	Componentes Integrados	Precondiciones	Pasos de Prueba	Resultado Esperado
PI-SAVE-001	ReporteManualScreen, ControladorEvento, DBConnector	El usuario ID=101 ha ingresado todos los datos obligatorios.	1. La ReporteManualScreen invoca ControladorEvento.guardarReporte(). 2. DBConnector.insertarReporteManual() es invocado.	DBConnector retorna **true** (éxito en la inserción). El registro en la BBDD existe, está **vinculado al ID del usuario 101** y tiene el estado **"Pendiente de Validación"**.
PI-SAVE-002	ControladorEvento, DBConnector	Se intenta guardar un reporte donde el tipo de catástrofe (foco o inundación) no es válido según los valores permitidos en la DB (error de *constraint*).	ControladorEvento.guardarReporte() es invocado. DBConnector intenta la inserción.	DBConnector lanza una SQLException (Violación de CHECK constraint). ControladorEvento la captura y la propaga como un error de negocio: **"Tipo de catástrofe inválido"**.

PI-SAVE-003	ReporteManualScreen, ControladorEvento	El usuario incluye una **descripción opcional** de 500 caracteres y no incluye fotos (Flujo Normal, paso 3).	ControladorEvento.guardarReporte() es invocado y procesa la descripción larga.	El reporte se guarda exitosamente. Al recuperar el reporte de la BBDD, el campo **descripción no está truncado** (se guarda la longitud máxima).

Nivel: Pruebas de Aceptación

Objetivo	Confirmar la usabilidad de la interfaz de reporte, la correcta aplicación del Flujo Alternativo (datos incompletos) y la confirmación visual de la recepción del reporte por parte del sistema.
Contexto	Se simula el uso por parte de un Usuario Registrado en la interfaz ReporteManualScreen.

ID Caso de Prueba	**Escenario de Negocio**	**Precondiciones**	**Pasos de Prueba (Realizados por Usuario Final/Stakeholder)**	**Resultado Esperado (Validación de Negocio)**
PA-REPORTE-001	Envío Exitoso y Confirmación	Usuario autenticado. Todos los campos obligatorios se completan (Coordenadas, Tipo).	1. Navegar a la función de reporte manual. 2. Ingresar todos los datos. 3. Presionar "Confirmar Reporte".	El sistema muestra un **mensaje de éxito** ("Reporte recibido y pendiente de validación"). La ventana de reporte se cierra o se reinicia (Flujo Normal, paso 7).

PA-REPORTE-002	Prevención por Datos Incompletos	Usuario autenticado. El usuario omite ingresar las coordenadas (ubicación).	1. Ingresar solo el tipo de catástrofe y la descripción. 2. Presionar "Confirmar Reporte".	El sistema **no envía el reporte** y muestra un mensaje de error o **resalta los campos faltantes** (coordenadas/ubicación), forzando al usuario a completarlos (Flujo Alternativo).
PA-REPORTE-003	Ubicación Visual en Mapa	El sistema presenta un mapa interactivo para ingresar la ubicación.	1. El actor utiliza la herramienta de **"seleccionar punto en el mapa"** en la interfaz. 2. El punto de clic se refleja automáticamente en los campos de Latitud/Longitud.	La Latitud y Longitud se **auto-completan** con precisión decimal después del clic, y el reporte se guarda exitosamente con la coordenada del mapa (Validación del Flujo Normal, paso 3).
PA-REPORTE-004	Estatus Inicial del Reporte	El reporte ha sido enviado y el usuario navega al MainDashboard.	1. Enviar un reporte exitosamente. 2. Un **Validador** (o Administrador) consulta la ValidationScreen.	El reporte recién ingresado **aparece visible** en la lista de la ValidationScreen con la etiqueta de estado **"Pendiente"**, esperando la acción de validación (Postcondición).

Caso de Uso 7: Gestionar Usuarios

El Caso de Uso 7 (CU-07: Gestionar Usuarios) es exclusivo del rol de Administrador del Sistema y es vital para la seguridad y el mantenimiento del sistema (RF-13, RNF-03).
El objetivo de esta prueba es asegurar que el Administrador del Sistema pueda realizar todas las operaciones CRUD (Crear, Consultar, Modificar, Eliminar) sobre las cuentas de usuario de manera segura y eficiente, manteniendo la integridad de los datos.

Nivel: Pruebas Unitarias

Objetivo	Asegurar que los métodos de negocio para las modificaciones de usuario funcionen correctamente, especialmente el cambio de roles y la activación/desactivación.
Contexto	Pruebas aisladas en la Capa de Negocio, dentro del ControladorUsuario, para verificar la lógica de los cambios de estado y rol.

ID	Componente/ Método	Precondiciones	Pasos de Prueba	Resultado Esperado
PU-ROL-001	ControladorUsuario.cambiarRol(id, nuevoRol)	El usuario ID=201 tiene el rol actual "Ciudadano". El nuevoRol es "Validador".	Llamar al método cambiarRol() con los nuevos parámetros.	El método retorna **true**. La propiedad rol del objeto Usuario interno se actualiza a "Validador".
PU-ESTADO-002	ControladorUsuario.cambiarEstado(id, activo)	El usuario ID=305 tiene el estado actual activo = true. Se llama con activo = false.	Llamar al método cambiarEstado() para desactivar la cuenta.	El método retorna **true**. La propiedad activo del objeto Usuario se establece en **false**.

PU-PERF-003	ControladorUsuario.validarEmailModificado(nuevoEmail)	El nuevoEmail (admin@sate.com) ya está asignado a otro usuario en el sistema.	Llamar al método de validación de unicidad de email (similar a CU-01).	El método retorna **false** y propaga un error que impide la modificación del perfil para evitar duplicidad de cuentas.

Nivel: Pruebas de Integración

Objetivo	Verificar la secuencia de persistencia y eliminación de usuarios, garantizando que el DBConnector ejecute las consultas de actualización y eliminación correctamente.
Contexto	Se valida la interacción completa entre el controlador de usuario y la base de datos para la gestión de las cuentas.

ID Caso de Prueba	Componentes Integrados	Precondiciones	Pasos de Prueba	Resultado Esperado
PI-DB-001	ControladorUsuario, DBConnector	El usuario ID=101 existe en la tabla.	1. ControladorUsuario.modificarPerfil(ID=101, ...) es invocado con un nuevo email. 2. DBConnector.actualizarUsuario() es invocado.	DBConnector retorna **true**. La fila del usuario ID=101 en la BBDD tiene el nuevo email y el timestamp de modificación actualizado.

PI-DB-002	ControladorUsuario, DBConnector	El usuario ID=500 existe en la tabla, pero tiene reportes manuales vinculados.	1. ControladorUsuario.eliminarUsuario(ID=500) es invocado. 2. DBConnector.eliminarUsuario() es invocado.	La operación debe ser exitosa, asumiendo una regla **ON DELETE CASCADE** o **SET NULL** para evitar fallos de integridad. El registro de ID=500 **desaparece** de la tabla Usuario.
PI-DB-003	ControladorUsuario, DBConnector	La tabla Usuario está vacía (cero registros).	ControladorUsuario.obtenerTodosLosUsuarios() es invocado al cargar el panel de administración.	DBConnector retorna un resultado vacío. ControladorUsuario retorna una **lista vacía** (List<Usuario>) a la Capa de Presentación (Validación del Flujo Alternativo).

Nivel: Pruebas de Aceptación

Objetivo	Confirmar que el Administrador puede gestionar de forma segura y clara las cuentas de usuario, y que la funcionalidad de desactivación (seguridad) es efectiva.
Contexto	Se simula el uso por parte del Administrador del Sistema en la interfaz AdminScreen para validar la usabilidad y las funciones de seguridad.

ID Caso de Prueba	Escenario de Negocio	Precondiciones	Pasos de Prueba (Realizados por Usuario	Resultado Esperado (Validación de Negocio)

			Final/Stakeholder)	
PA-ADM-001	Modificación de Rol y Perfil	El Administrador ha iniciado sesión. Existe el usuario UsuarioA.	1. Navegar al panel de gestión. 2. Seleccionar UsuarioA y modificar su nombre y cambiar su rol a "Validador". 3. Presionar "Guardar Cambios".	El sistema muestra **mensaje de confirmación**. La tabla se actualiza y el usuario modificado **accede a las funciones de ValidationScreen** en su próxima sesión.
PA-ADM-002	Desactivación de Cuenta	El usuario UsuarioB está "Activo".	1. El Administrador selecciona UsuarioB. 2. Presiona el botón "Desactivar/Activar" (cambiando el estado a "Desactivado").	El sistema confirma el cambio. UsuarioB **no puede iniciar sesión** con sus credenciales, recibiendo el mensaje: "Cuenta Desactivada. Contacte al administrador."
PA-ADM-003	Eliminación de Cuenta	El usuario UsuarioC existe y no es el administrador.	1. El Administrador selecciona UsuarioC. 2. Presiona el botón "Eliminar Cuenta". 3. Confirma la acción en el diálogo de seguridad.	La fila de UsuarioC **desaparece inmediatamente** de la tabla de la AdminScreen. Se muestra un mensaje de confirmación de "Cuenta eliminada".

PA-ADM-004	Carga Inicial y Visibilidad (RNF-03)	La tabla Usuario contiene 500 registros.	1. El Administrador navega al panel de gestión de usuarios.	La lista de usuarios (JTable) se **carga y muestra en su totalidad en un tiempo ágil** (Validación del RNF-03: Performance de consultas), y la interfaz muestra una barra de desplazamiento funcional.

Caso de Uso 8: Validar Reportes Manuales (CU-08)

El **Caso de Uso 8** (**CU-08: Validar Reportes Manuales**) es una funcionalidad de control de calidad crítica, exclusiva para el rol de Administrador, que garantiza que solo los datos verificados (RF-07) se integren al mapa público.
El objetivo de esta prueba es asegurar que el Administrador del Sistema pueda procesar los reportes manuales, eligiendo entre **Aprobar** (integrando el reporte como un Evento oficial) o **Rechazar** (descartándolo del flujo), manteniendo la integridad del sistema (RNF-03).

Nivel: Pruebas Unitarias

Objetivo	Asegurar que los métodos de negocio realicen correctamente la transición de estado de un reporte y la creación de una nueva entidad (Evento) al ser aprobado.
Contexto	Pruebas aisladas en el ControladorEvento para verificar la lógica de la aprobación/rechazo y la conversión de entidades.

ID	Componente/ Método	Precondiciones	Pasos de Prueba	Resultado Esperado
PU-APR-001	ControladorEvento.aprobarReporte(idReporte)	El ReporteManual existe con estado "Pendiente".	Llamar a aprobarReporte(id) simulando la conversión a Evento.	El método retorna **true**. Se invoca la lógica de creación de la entidad **Evento** (con la geolocalizació

				n WKT del reporte).
PU-RECH-002	ControladorEvento.rechazarReporte(idReporte)	El ReporteManual existe con estado "Pendiente".	Llamar a rechazarReporte(id) simulando la eliminación lógica o el cambio de estado.	El método retorna **true**. El reporte manual se marca como **"Rechazado"** (o es eliminado) sin generar una nueva entidad Evento.
PU-CTRL-003	ControladorEvento.obtenerPendientes()	Se invoca la función para recuperar los reportes pendientes de validación.	El método accede a la lógica que filtra los reportes por estado.	El método **retorna una lista de objetos ReporteManual** donde el atributo estado es **"Pendiente de Validación"**.

Nivel: Pruebas de Integración

Objetivo	Verificar la persistencia de los cambios de estado y la inserción del nuevo Evento en la base de datos tras la aprobación.
Contexto	Se valida la interacción entre el ControladorEvento y el DBConnector para la actualización (UPDATE) del reporte manual y la inserción (INSERT) del nuevo evento.

ID	Componentes Integrados	Precondiciones	Pasos de Prueba	Resultado Esperado
PI-APR-DB-001	ControladorEvento, DBConnector	El reporte ID=400 está "Pendiente" en la BBDD.	1. ControladorEvento.aprobarReporte(ID=400) es invocado. 2. DBConnector.actualizarEstadoReporte() y DBConnector.insertarEvento() son invocados.	1. El reporte ID=400 se actualiza a estado **"Aprobado"**. 2. Una nueva fila se inserta en la tabla **Evento** con la geolocalización y los detalles del reporte.
PI-RECH-DB-002	ControladorEvento, DBConnector	El reporte ID=500 está "Pendiente".	1. ControladorEvento.rechazarReporte(ID=500) es invocado. 2. DBConnector.actualizarEstadoReporte() es invocado.	La fila del reporte ID=500 se actualiza a estado **"Rechazado"** (o se elimina). **No se inserta ninguna fila** en la tabla Evento.
PI-LOAD-DB-003	ControladorEvento, DBConnector	La BBDD no tiene reportes con estado "Pendiente de Validación".	1. El AdminScreen invoca ControladorEvento.obtenerPendientes().	DBConnector retorna una lista vacía. ControladorEvento retorna una lista vacía (List<ReporteManual>) a la capa de presentación (Validación del Flujo Alternativo).

Nivel: Pruebas de Aceptación

Objetivo	Confirmar la seguridad del flujo (acceso solo para Admin), la claridad de la interfaz para la revisión y la propagación inmediata de los eventos aprobados al mapa público.

Contexto	Se simula el uso por parte del Administrador en la interfaz ValidationScreen (Vista 5) y se verifica el impacto en el MainDashboard (Mapa).

ID Caso de Prueba	**Escenario de Negocio**	**Precondiciones**	**Pasos de Prueba (Realizados por Usuario Final/Stakeholder)**	**Resultado Esperado (Validación de Negocio)**
PA-VAL-001	Flujo de Aprobación Completo	El Administrador ha iniciado sesión. Existe un reporte R-01 pendiente.	1. Acceder a la ValidationScreen. 2. Seleccionar R-01 y presionar **" Aprobar Reporte"**. 3. El Administrador navega al MainDashboard.	El reporte R-01 **desaparece de la lista** de pendientes. El evento **es visible inmediatamente** en el mapa (MainDashboard) para todos los usuarios.
PA-VAL-002	Flujo de Rechazo	Existe un reporte R-02 pendiente.	1. Acceder a la ValidationScreen. 2. Seleccionar R-02 y presionar **"Rechazar Reporte"**.	El reporte R-02 **desaparece de la lista** de pendientes. El evento **no aparece ni se visualiza** en el mapa público (Validación de la Postcondición de rechazo).
PA-VAL-003	Escenario "Sin Pendientes"	No hay reportes en la tabla Reportes Manuales con estado "Pendiente".	1. El Administrador accede al panel de reportes pendientes.	El sistema muestra un **mensaje claro** en la interfaz que indica: **"No hay reportes manuales pendientes de validación."** (Flujo Alternativo).

PA-VAL-004	Seguridad de Acceso	Un usuario con rol **"Validador"** intenta acceder al panel de reportes pendientes.	1. El Validador inicia sesión. 2. Intenta navegar a la ValidationScreen o la opción de menú.	El sistema **permite el acceso** (ya que el validador también gestiona reportes, según el diseño inicial), pero verifica que solo se muestren los datos necesarios para la validación y no funciones de administración general (Validación RF-01).

Caso de Uso 9: Integrar Nueva Fuente de Datos (CU-09)

El **Caso de Uso 9 (CU-09: Integrar Nueva Fuente de Datos)** es una funcionalidad de **escalabilidad y mantenimiento**, exclusiva del Administrador, vital para el **crecimiento futuro del sistema** (RF-12) y la optimización del rendimiento (RNF-03).
El objetivo de esta prueba es asegurar que el Administrador pueda configurar, probar y persistir los parámetros de conexión de una nueva fuente de datos externa (API), habilitando el procesamiento de datos subsecuente, y manejando las fallas de conexión de manera controlada.

Nivel: Pruebas Unitarias

Objetivo	Asegurar que los métodos de validación de la URL y las credenciales de la API funcionen correctamente de forma aislada.
Contexto	Pruebas aisladas en la Capa de Negocio, dentro de la lógica del ControladorConfiguracion, para verificar la integridad de los datos de conexión.

ID Caso de Prueba	Componente/ Método	Precondiciones	Pasos de Prueba	Resultado Esperado

PU-CONF-001	ControladorConfiguracion.validarURL(url)	La URL de entrada es inválida (ej: falta el protocolo http:// o https://).	Llamar al método validarURL() con la URL incorrecta.	El método retorna **false** y lanza una excepción indicando que el formato de la URL es incorrecto.
PU-CONF-002	ServicioAPI.probarConexion(url, credentials)	El método de prueba de conexión está implementado para devolver un *status code* HTTP. Se simula un status 200 OK.	Invocar probarConexion() con parámetros simulados de éxito.	El método retorna **true**, confirmando que la infraestructura base de la API es accesible.
PU-CONF-003	ServicioAPI.probarConexion(url, credentials)	Se simula una respuesta de error HTTP 401 (Acceso no autorizado) o 403.	Invocar probarConexion() con credenciales inválidas.	El método retorna **false** y la lógica de negocio debe interpretar esto como un fallo en la prueba de conexión.

Nivel: Pruebas de Integración

Objetivo	Verificar la secuencia de prueba de conexión, la persistencia de los parámetros y la activación del nuevo módulo de consumo de datos.
Contexto	Se valida la interacción completa entre el ControladorConfiguracion, el DBConnector y el servicio externo (ServicioAPI simulado).

ID Caso de Prueba	Componentes Integrados	Precondiciones	Pasos de Prueba	Resultado Esperado
PI-INTEG-DB-001	ControladorConfiguracion, ServicioAPI, DBConnector	El ServicioAPI pasa la prueba de conexión (simula status 200 OK).	1. ControladorConfiguracion.guardarFuente(data) es invocado. 2. DBConnector.insertarFuenteDatos() es invocado.	DBConnector retorna **true**. El registro de la nueva fuente se inserta en la BBDD con su URL, credenciales (encriptadas) y un estado de **"Activo"**.
PI-INTEG-FLOW-002	AdminScreen, ControladorConfiguracion, ServicioAPI	El administrador ingresa una URL válida, pero la prueba de conexión falla (ej: *timeout*).	1. El AdminScreen presiona "Probar Conexión". 2. ServicioAPI.probarConexion() falla. 3. Se invoca la lógica de la Capa de Presentación.	El AdminScreen muestra un **mensaje de error claro** ("Fallo en la prueba de conexión: Timeout"). El sistema **no guarda la configuración** en la base de datos.
PI-INTEG-FLOW-003	ControladorConfiguracion, ModuloConsumo	Una nueva fuente de datos ha sido guardada exitosamente en la DB.	1. ControladorConfiguracion.habilitarFuente(id) es invocado. 2. La lógica interna activa la clase o módulo encargado del consumo periódico de datos.	El método de inicio de consumo (ModuloConsumo.start()) se invoca con éxito, y el *log* del sistema registra **"Consumo iniciado para fuente X"**.

Nivel: Pruebas de Aceptación

Objetivo	Confirmar la usabilidad de la interfaz de configuración y la respuesta del sistema al validar y guardar la nueva fuente de datos, cumpliendo con el RF-12.
Contexto	Se simula el uso por parte del Administrador del Sistema en la interfaz de configuración de fuentes de datos.

ID Caso de Prueba	**Escenario de Negocio**	**Precondiciones**	**Pasos de Prueba (Realizados por Usuario Final/Stakeholder)**	**Resultado Esperado (Validación de Negocio)**
PA-CONF-001	Integración Completa Exitosa	El Administrador tiene los datos de conexión válidos (URL y *API Key*).	1. Navegar a la sección de configuración de fuentes. 2. Llenar el formulario con los datos. 3. Presionar "Probar Conexión" y luego "Guardar Configuración".	El sistema muestra **dos mensajes de éxito** ("Conexión Exitosa" y "Configuración Guardada"). El nombre de la nueva fuente aparece en la lista de fuentes activas (Validación del Flujo Normal).
PA-CONF-002	Fallo por Credenciales	El Administrador ingresa la URL correcta, pero una *API Key* intencionalmente errónea.	1. Llenar el formulario y presionar "Probar Conexión".	El sistema muestra un **mensaje de error específico** indicando **"Fallo de Autenticación (401/403). Verifique credenciales"**. La configuración no se guarda.

PA-CONF-003	Persistencia Segura	El Administrador guarda una fuente de datos con una credencial sensible.	1. Guardar la fuente de datos exitosamente. 2. El actor simula la consulta directa a la base de datos.	La credencial sensible (ej: *API Key*) **no debe ser visible en texto plano** en la tabla de la BBDD; debe estar **encriptada** (Validación de RNF-03: Seguridad).
PA-CONF-004	Activación de Consumo	La configuración de la nueva fuente ha sido guardada.	1. El actor verifica el *log* de eventos del sistema (simulado).	El *log* muestra una entrada que confirma el inicio del procesamiento de datos, con una marca de tiempo y el ID de la nueva fuente, por ejemplo: **[2025-10-05] Proceso ETL iniciado para Fuente: CONAE-NUEVO**.

Caso de Uso 10: Suscribirse a Notificaciones Específicas

El **Caso de Uso 10 (CU-10: Suscribirse a Notificaciones Específicas)** valida la capacidad del usuario para **personalizar** el flujo de alertas, un aspecto clave de la usabilidad y la eficiencia del sistema (RF-05, RNF-04).

El objetivo de esta prueba es garantizar que el usuario pueda seleccionar diversas opciones de suscripción (ej., por tipo de evento, por estado de validación), que estas preferencias se persistan correctamente y que la lógica de envío de alertas las respete estrictamente.

Nivel: Pruebas Unitarias

Objetivo	Asegurar que la entidad Usuario o la entidad de Preferencia pueda almacenar y exponer correctamente las opciones binarias y los rangos numéricos (ej., distancia).
Contexto	Pruebas aisladas en la Capa de Negocio, verificando la integridad de la estructura de datos de las preferencias de notificación.

ID Caso de Prueba	Componente/ Método	Precondiciones	Pasos de Prueba	Resultado Esperado
PU-PREF-001	Preferencia.setDistancia(valor)	El valor de entrada es un rango numérico válido (ej: 50.0 km).	Llamar al método setDistancia() con el valor de prueba.	El método retorna **true**. La propiedad rangoDistancia del objeto se almacena con el valor 50.0.
PU-PREF-002	Preferencia.setTipoEvento(tipo, estado)	Se intenta establecer una preferencia con un tipo de evento no permitido (ej: "Terremoto").	Llamar al método setTipoEvento() con el tipo inválido.	El método retorna **false** y lanza una excepción indicando que el tipo de evento es inválido (RNF-05).
PU-PREF-003	ControladorUsuario.validarPreferencias(prefs)	El usuario deselecciona todas las opciones de notificación (un estado válido).	Llamar al método de validación de preferencias.	El método retorna **true**, ya que deshabilitar todas las notificaciones es una configuración válida del usuario.

Nivel: Pruebas de Integración

Objetivo	Verificar la actualización de las preferencias de usuario en la base de datos y la correcta interacción de la lógica de envío de notificaciones con estos nuevos criterios.

Contexto	Se valida la secuencia completa de guardar las preferencias y cómo el proceso de geocercado (CU-05) utiliza estos datos.

ID Caso de Prueba	Componentes Integrados	Precondiciones	Pasos de Prueba	Resultado Esperado
PI-SAVE-001	ControladorUsuario, DBConnector	El usuario ID=300 tiene preferencias por defecto. Se envían nuevas preferencias (Solo eventos validados = TRUE).	1. ControladorUsuario.guardarPreferencias(ID=300, newPrefs) es invocado. 2. DBConnector.actualizarPreferencias() es invocado.	DBConnector retorna **true**. La fila del usuario ID=300 en la tabla de preferencias (o Usuario) se actualiza con el valor **soloValidados = 1**.
PI-NOTIF-002	ControladorEvento, Preferencia	El **Usuario A** tiene rangoDistancia = 5 km. Un nuevo evento ocurre a **3 km** de su área de interés.	El ControladorEvento procesa el nuevo evento, recuperando las preferencias del Usuario A.	El sistema compara la distancia de 3 km con el rango de 5 km, determina la **coincidencia** y **envía la notificación** (Validación de RF-10 y RF-05).

PI-NOTIF-003	ControladorEvento, Preferencia	El **Usuario B** tiene Solo eventos validados = TRUE. Ocurre un nuevo Reporte Manual (no validado).	El ControladorEvento procesa el nuevo reporte.	El sistema ignora al Usuario B en el flujo de envío de alertas. El usuario **no recibe la notificación**, pero el sistema no genera un error (Validación de la lógica de negocio).

Nivel: Pruebas de Aceptación

Objetivo	Confirmar la facilidad de uso de la interfaz de preferencias y que el sistema solo envíe las alertas que el usuario ha solicitado (Postcondición).
Contexto	Se simula el uso por parte de un Usuario Registrado en la interfaz de preferencias de notificación.

ID Caso de Prueba	**Escenario de Negocio**	**Precondiciones**	**Pasos de Prueba (Realizados por Usuario Final/Stakeholder)**	**Resultado Esperado (Validación de Negocio)**
PA-PREF-001	Suscripción Exitosa por Criterio	El usuario selecciona la opción "Recibir alertas solo para eventos Confirmados por validación".	1. Navegar a Preferencias. 2. Seleccionar la casilla "Solo Validados". 3. Presionar "Guardar	El sistema muestra un **mensaje de confirmación**. En la próxima alerta, el usuario **solo recibirá la notificación** si el evento ha pasado por CU-08

			Cambios".	(Validación).
PA-PREF-002	Suscripción por Rango de Distancia	El usuario establece el "Rango de Distancia" a **1 km** (el criterio más estricto).	1. Establecer el rango de distancia. 2. Presionar "Guardar Cambios".	El sistema guarda el rango. El usuario **no recibe notificaciones** de eventos que ocurren a 1.1 km o más de su área de interés (Validación de la precisión, RF-10).
PA-PREF-003	Fallo al Guardar Cambios	Se simula una interrupción de la conexión de la BBDD justo al presionar "Guardar Cambios".	1. El actor selecciona nuevas preferencias. 2. Presiona "Guardar Cambios".	El sistema muestra un **mensaje de error** claro que indica **"No se pudieron guardar las preferencias. Intente nuevamente."** (Flujo Alternativo). Las preferencias en la BBDD **no se modifican**.
PA-PREF-004	Usabilidad de Opciones	Usuario de prueba.	1. El actor navega a la interfaz de preferencias.	La interfaz presenta opciones de suscripción **claramente etiquetadas** (ej: *checkbox* para "Notificaciones por Correo Electrónico", *slider* o campo para "Rango en Km") y el botón de guardar es visible y funcional.

Caso de Uso 11: Consultar Histórico de Alertas

El **Caso de Uso 11 (CU-11: Consultar Histórico de Alertas)** es vital para el **análisis de tendencias** del sistema (RF-11) y requiere una validación estricta de la conexión a la base de datos especializada **MySQL** y el rendimiento (RNF-04).

El objetivo de esta prueba es asegurar la correcta conexión y recuperación de datos desde el **Histórico de Eventos** , verificando que los filtros de rango de fechas y ubicación funcionen con precisión, y que los resultados se muestran de forma eficiente en la interfaz.

Nivel: Pruebas Unitarias

Objetivo	Asegurar el correcto funcionamiento de la lógica de validación de rangos de fechas y la generación de *queries* geográficas antes de interactuar con la base de datos.
Contexto	Pruebas aisladas en los métodos del ControladorHistorico para validar los parámetros de búsqueda.

ID Caso de Prueba	Componente/ Método	Precondiciones	Pasos de Prueba	Resultado Esperado
PU-FILT-001	ControladorHistorico.validarRangoFechas(inicio, fin)	inicio = '2025-01-01', fin = '2024-01-01' (rango invertido).	Llamar al método con la fecha de inicio posterior a la fecha de fin.	Retorna **false** y lanza una excepción indicando que la fecha de inicio es posterior a la fecha de fin.
PU-FILT-002	ControladorHistorico.generarQueryGeo(area, radio)	El usuario selecciona un radio de búsqueda de **25 km** alrededor de un punto (Lat/Lon).	Llamar al método para construir la consulta geográfica para MySQL.	El método retorna la **cadena de consulta Geo-query** para buscar eventos dentro de un radio de 25 km del punto especificado.

Nivel: Pruebas de Integración

Objetivo	Verificar la conexión exitosa y el intercambio de datos entre la Capa de Negocio (ControladorHistorico) y la base de datos de históricos (**MySQL**).
Contexto	Se valida la secuencia de consulta, que es de alta criticidad debido a la tecnología no relacional utilizada para el histórico.

ID Caso de Prueba	Componentes Integrados	Precondiciones	Pasos de Prueba	Resultado Esperado
PI-SQL-001	ControladorHistorico, MySQLConnector	La conexión a MySQL está activa. Existe un evento histórico que coincide con los criterios.	1. ControladorHistorico.consultarHistorico(criterios) es invocado. 2. MySQLConnector ejecuta la consulta.	ControladorHistorico retorna una **lista de objetos EventoHistorico** (no vacía), y la consulta se ejecuta con éxito.
PI-SQL-002	ControladorHistorico, MySQLConnector	Se simula un fallo en la conexión a MySQL (ej: *timeout* o servicio caído).	ControladorHistorico.consultarHistorico() es invocado.	MySQLConnector lanza una excepción de conexión. ControladorHistorico la captura y retorna una lista vacía, propagando un mensaje de **"Error de Conexión a Histórico"** a la interfaz.

PI-SQL-003	ControladorHistorico, MySQLConnector	Se consulta un rango de fechas con **1000 eventos** históricos.	Se invoca ControladorHistorico.consultarHistorico(rango_masivo).	La consulta se completa dentro del límite de tiempo del **RNF-04** (ej: menos de 5 segundos), y el controlador retorna la lista completa sin *timeout* de conexión.

Nivel: Pruebas de Aceptación

Objetivo	Validar la usabilidad de la interfaz de búsqueda, la claridad de los datos históricos mostrados y el cumplimiento del RNF-04 (Rendimiento) en la visualización.
Contexto	Se simula el uso por parte del Usuario Registrado en la interfaz de consulta de históricos.

ID Caso de Prueba	Escenario de Negocio	Precondiciones	Pasos de Prueba (Realizados por Usuario Final/Stakeholder)	Resultado Esperado (Validación de Negocio)
PA-HIST-001	Búsqueda y Visualización por Fechas	Usuario autenticado. Se ingresa un rango de 6 meses con datos.	1. Navegar a Histórico. 2. Ingresar Rango de Fechas válido. 3. Presionar "Buscar".	El sistema despliega los resultados en una tabla y el mapa. El mapa debe mostrar la **evolución cronológica de los eventos** (ej: mediante un *slider* de tiempo) (Validación RF-11).

PA-HIST-002	Flujo Alternativo (Sin Resultados)	Usuario autenticado. Se ingresa un rango de fechas o una ubicación sin eventos históricos.	1. Ingresar Rango: 'Año 1990' y Ubicación: 'Mi ciudad'. 2. Presionar "Buscar".	El sistema **no muestra resultados** en el mapa/tabla y presenta un mensaje claro: **"No se encontraron eventos para los criterios especificados."** (Validación del Flujo Alternativo).
PA-HIST-003	Rendimiento del Despliegue (RNF-04)	Se recuperan 500 eventos históricos que cumplen los criterios.	1. El actor realiza una consulta que retorna 500 resultados. 2. Se mide el tiempo de renderizado.	El tiempo de respuesta de la interfaz (desde presionar Buscar hasta el despliegue de los 500 puntos en el mapa/tabla) **no supera los 5 segundos** (Validación de RNF-04).
PA-HIST-004	Usabilidad del Filtro Geográfico	Usuario de prueba.	1. El actor selecciona la opción de filtrar por "Área de Interés definida" o "Radio". 2. Ejecuta la consulta.	La interfaz **solo muestra los eventos históricos** que caen dentro del área seleccionada, probando que el filtro geográfico de la *query* de MySQL se aplica correctamente.

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 SATE

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 (Usuario, Evento, ReporteManual, AreaInteres, 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. Usuarios.crearUsuario()) 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 Usuario y es referenciada en ReporteManual o Notificacion 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 SATE, 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.

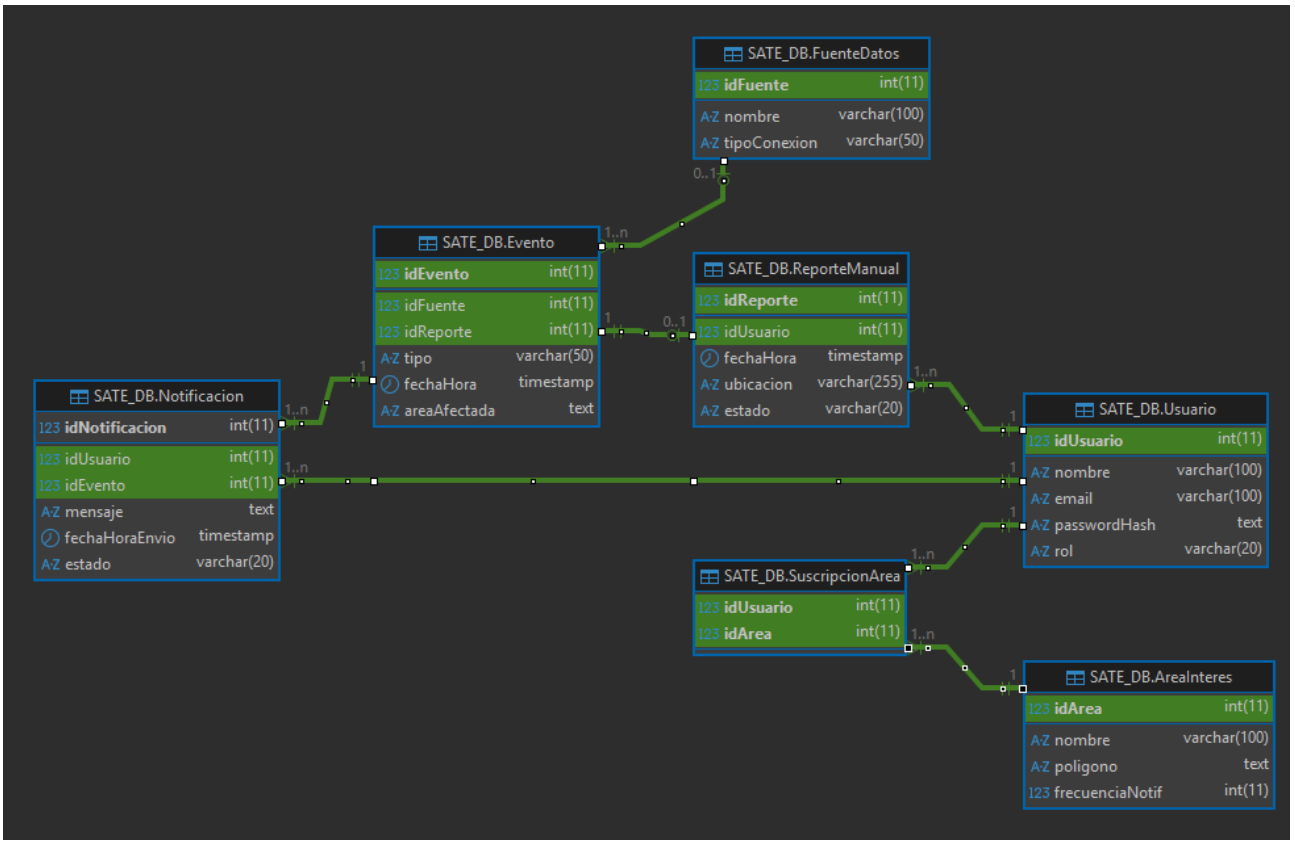
Por ejemplo:

- Relación Uno a Muchos (1:N): La relación entre Usuario y ReporteManual asegura, mediante una FK en la tabla de reportes, que todo reporte en el sistema esté siempre asociado a un único usuario responsable, apoyando directamente la funcionalidad de "Registrar Reporte" y la posterior trazabilidad de la autoría.
- Relación Uno a Uno (1:1): La relación entre ReporteManual y Evento (una vez validado) se implementa para reflejar la regla de que un reporte validado se convierte en un evento único y

viceversa, manteniendo la distinción clara entre la entrada del usuario y el evento oficial del sistema.

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



Explicación de las Relaciones

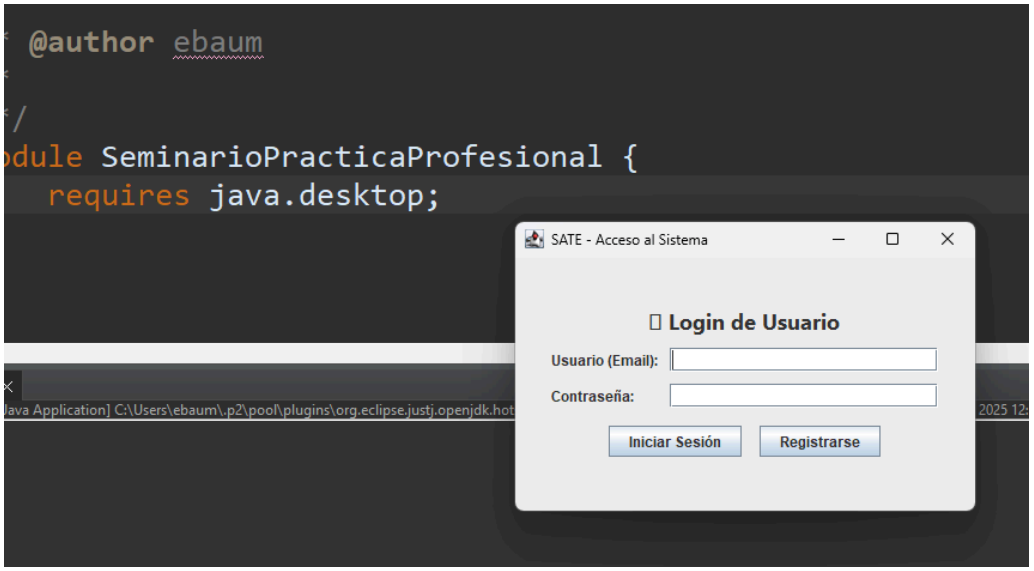
- **Usuario a SuscripcionesArea**: Una relación **uno a muchos (1:N)**. Un usuario puede definir múltiples áreas de interés, pero un área de interés pertenece a un único usuario.
- **Usuario a Reporte_Manual**: Una relación **uno a muchos (1:N)**. Un usuario puede realizar múltiples reportes manuales, pero cada reporte es hecho por un solo usuario.
- **Usuario a Notificaciones**: Una relación **uno a muchos (1:N)**. Un usuario puede recibir múltiples notificaciones, pero cada notificación está destinada a un único usuario.
- **Fuentes_Datos a Eventos**: Una relación **uno a muchos (1:N)**. Una fuente de datos (ej. CONAE) puede detectar múltiples eventos.
- **Reportes_Manuales a Eventos**: Una relación **uno a uno (1:1)**. Un reporte manual, una vez validado, se convierte en un único evento.
- **Eventos a Notificaciones**: Una relación **uno a muchos (1:N)**. Un evento puede generar múltiples notificaciones, una por cada usuario suscrito en su área de interés.

Prototipos de la Interfaz de Usuario SATE

La siguiente sección presenta el prototipo visual de la Capa de Presentación del Sistema SATE. Para garantizar una simulación realista del entorno operativo final, el diseño se ha modelado bajo la estética y los componentes de una aplicación de escritorio tradicional en Java Swing. Estos mock-ups ilustran los principales flujos de trabajo y la separación de vistas requerida por los diferentes perfiles de usuario (Ciudadano, Validador y Administrador), enfocándose en la usabilidad, la claridad de los datos geográficos (GIS) y la gestión centralizada del sistema.

Vista 1: Acceso y Autenticación de Usuario (Login Screen)

Esta es la primera interfaz que el usuario encuentra. Su diseño se enfoca en la simplicidad y la seguridad, siendo una ventana **modal** y centralizada. Permite a los usuarios ingresar sus credenciales para acceder al sistema. Incluye enlaces directos a la **recuperación de contraseña** y al **formulario de registro** para nuevos ciudadanos. La validación simulada en esta vista determina el rol del usuario y la pantalla de inicio correspondiente.

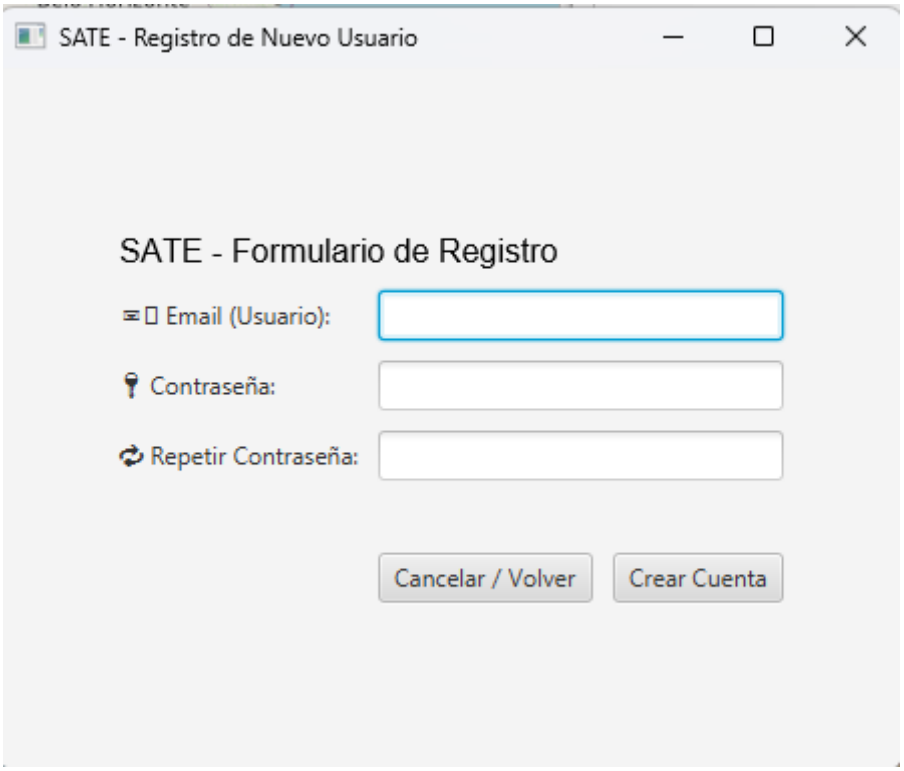


Vista 2: Registro de Nuevo Usuario (Register Screen)

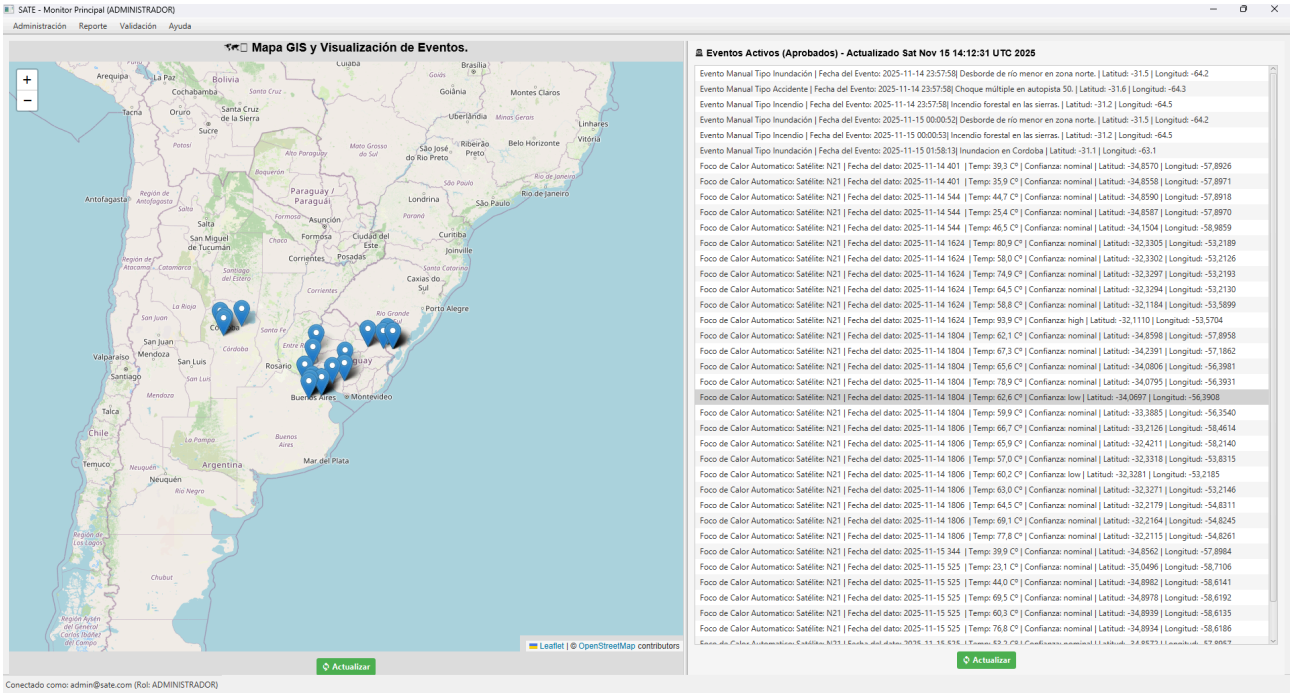
Interfaz destinada a usuarios externos o ciudadanos que desean obtener una cuenta para reportar incidencias. Captura los datos esenciales (nombre, email, contraseña) y utiliza un diseño limpio para reducir la fricción en la creación de una cuenta. Al completarse con éxito, el sistema asignará automáticamente el **rol de Usuario** al nuevo registro.

Vista 3: Monitor de Eventos y Dashboard Principal

Esta es la interfaz de trabajo central para los roles de Ciudadano y Validador. Utiliza un JSplitPane para dividir la pantalla en dos áreas clave:



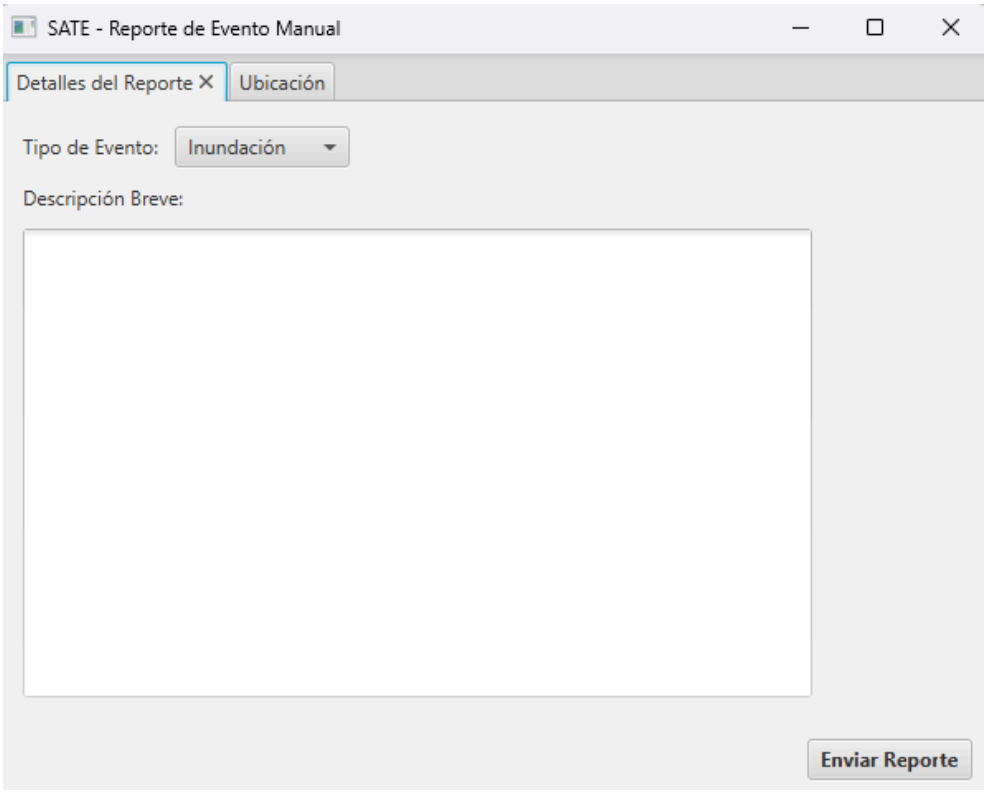
1. **Panel Derecho:** Controles de **filtrado** por tipo y estado, y una tabla (JTable) que lista los eventos recuperados de la base de datos (**Evento**).
2. **Panel Izquierdo:** El **Mapa GIS** simulado (**GISMapPanel**) que visualiza la ubicación exacta (puntos WKT) de los eventos activos, permitiendo una rápida identificación geográfica de las alertas. Incluye el botón "Nuevo Reporte Manual".

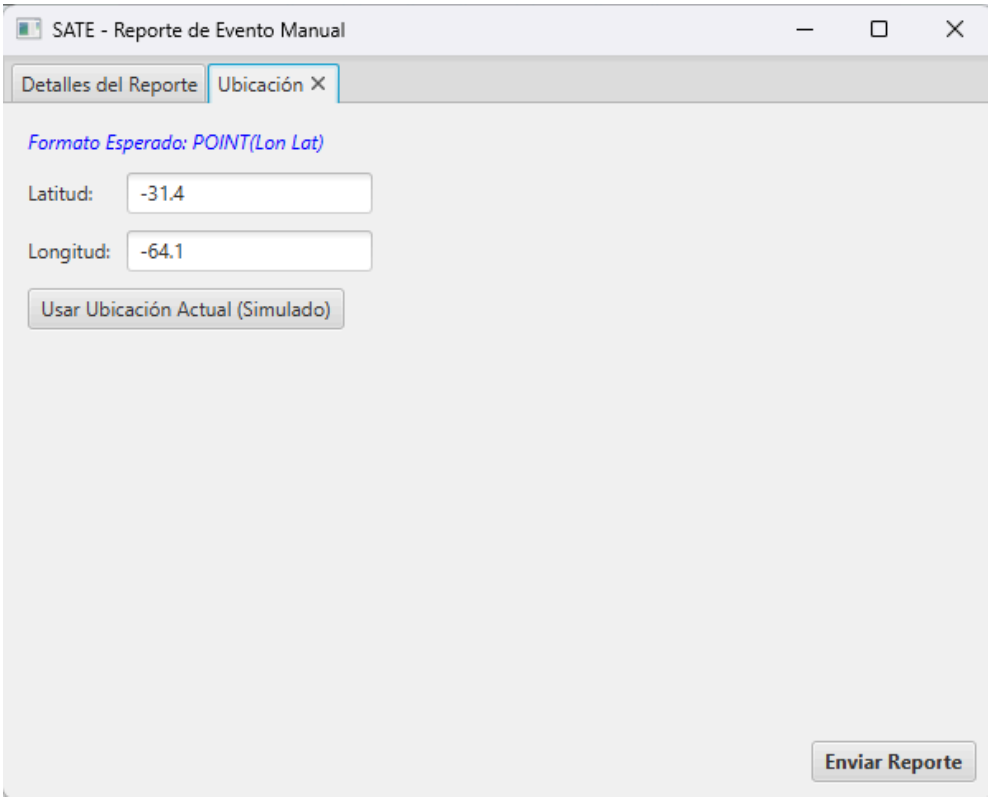


Vista 4: Registro de Nuevo Evento (Reporte Manual)

Formulario esencial para que los usuarios reporten una nueva incidencia. Utiliza un diseño de **pestañas**

(JTabbedPane) para organizar la información en secciones lógicas: "Detalles del Reporte" (tipo y descripción) y "Ubicación". La interfaz de ubicación solicita **Latitud y Longitud** en formato decimal, que el sistema internamente transforma al formato **WKT (POINT)** para su almacenamiento en la base de datos.

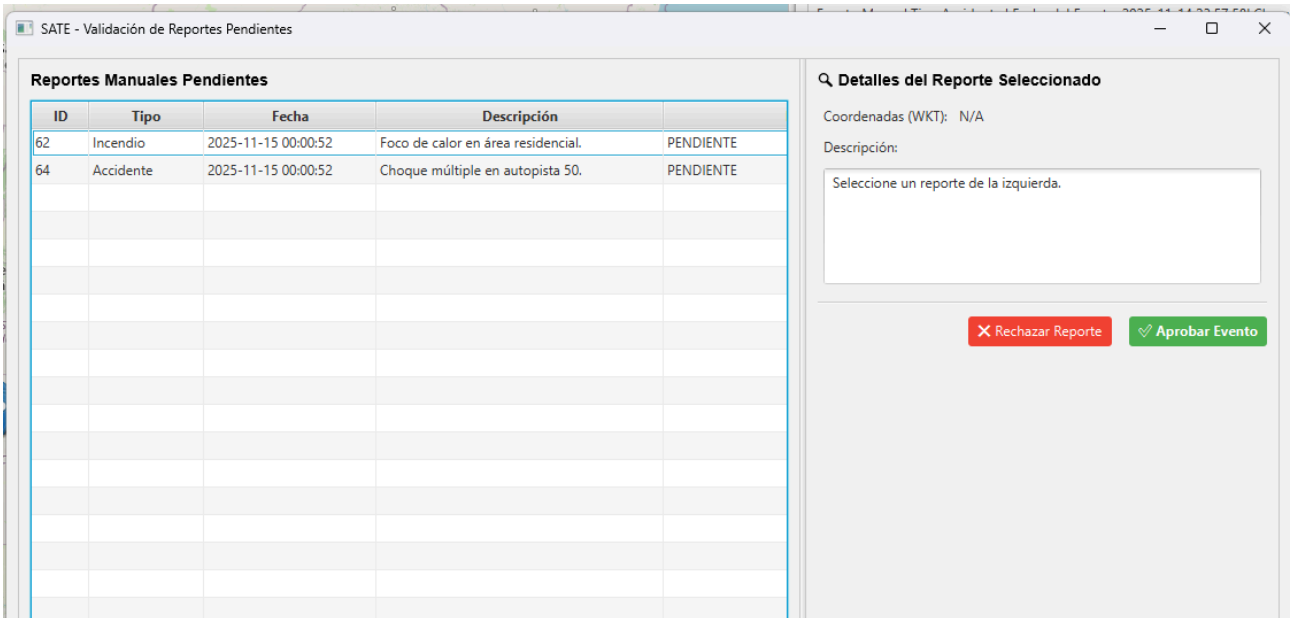




Vista 5: Validación y Aceptación de Reportes (Validation Screen)

Interfaz exclusiva para el rol de **Validador** o **Administrador**. Su propósito es gestionar los reportes manuales que se encuentran en estado **"Pendiente"**. La vista presenta una lista de reportes en espera y un panel de detalles. Permite al usuario **revisar** el contenido de la alerta y tomar una decisión binaria:

- **"Aprobar Evento":** Transforma el reporte manual en un evento oficial del sistema, activando el proceso de notificación.
- **"Rechazar Reporte":** Descarta el reporte por ser inválido o duplicado.



Fundamentos de Codificación y Diseño Orientado a Objetos (POO)

El Sistema de Alertas Tempranas de Eventos (SATE) ha sido meticulosamente desarrollado utilizando el lenguaje de programación Java, un pilar fundamental en el ámbito del software empresarial y las aplicaciones robustas. Durante su concepción e implementación, se han aplicado de manera rigurosa los paradigmas de la Programación Orientada a Objetos (POO), lo que ha permitido construir una arquitectura de software modular, escalable y de fácil mantenimiento.
La adhesión a la POO se manifiesta en la estructuración del SATE mediante clases y objetos que encapsulan datos y comportamientos relacionados, fomentando la reutilización de código y la claridad en el diseño. Conceptos como la herencia, el polimorfismo y la encapsulación han sido explotados para crear un sistema adaptable a diferentes tipos de eventos y fuentes de datos, garantizando una alta cohesión y bajo acoplamiento entre sus componentes.
Además de los principios de la POO, el desarrollo del SATE ha respetado y capitalizado las características fundamentales intrínsecas del lenguaje Java. Esto incluye, entre otros aspectos:

- **Portabilidad (Write Once, Run Anywhere - WORA):** La compilación a bytecode Java y su ejecución en la Máquina Virtual de Java (JVM) asegura que el SATE pueda operar sin modificaciones en una amplia gama de plataformas y sistemas operativos, desde servidores dedicados hasta entornos distribuidos.
- **Gestión Automática de Memoria (Garbage Collection):** Java libera a los desarrolladores de la gestión manual de memoria, reduciendo significativamente la probabilidad de errores como las fugas de memoria y optimizando la asignación y liberación de recursos. Esto contribuye a la estabilidad y fiabilidad del sistema a largo plazo.
- **Robustez y Manejo de Excepciones**: Java impone un estricto tipado y un robusto mecanismo de manejo de excepciones, lo que ha permitido al SATE gestionar de forma elegante y controlada situaciones anómalas o errores inesperados, minimizando interrupciones y garantizando la integridad de los datos. A su vez se almacena en un log los errores y el funcionamiento general.
- **Multihilo (Multithreading):** La capacidad de Java para gestionar hilos de ejecución concurrentes ha sido crucial para el SATE, permitiendo el procesamiento simultáneo de múltiples alertas, la monitorización continua de diversas fuentes de eventos y la ejecución eficiente de tareas en segundo plano sin bloquear la interfaz principal o el flujo de datos crítico.
- **Amplio Ecosistema y Bibliotecas:** El extenso ecosistema de Java y la disponibilidad de una vasta colección de APIs y bibliotecas (tanto nativas como de terceros) han agilizado el desarrollo y enriquecido las funcionalidades del SATE, desde la conectividad con bases de datos y sistemas de mensajería hasta la implementación de algoritmos complejos para la detección de patrones de eventos.

En **resumen**, la elección de Java y el uso de las buenas prácticas de la Programación Orientada a Objetos y la implementación inteligente de las características inherentes del lenguaje permitieron la creación de un Sistema de Alertas Tempranas de Eventos (SATE) robusto, eficiente, escalable y altamente confiable, capaz de cumplir con su propósito crítico de manera efectiva.
La **sintaxis** utilizada se adhiere a los estándares de Java, lo que facilita la comprensión y el manejo de los elementos del código. Para los identificadores y los estados de activación, se recurre a tipos de datos

primitivos, como int para valores numéricos enteros o boolean para representar estados de verdadero/falso. Por otro lado, cuando se requiere trabajar con estructuras de datos más complejas, como cadenas de texto (String), listas de elementos (List), o la creación de objetos personalizados como Usuario o Evento, se emplean tipos de referencia. Esto permite una mayor flexibilidad y capacidad para modelar información más elaborada dentro del sistema.
Más allá de la interfaz, la creación de objetos es igualmente crítica en la capa de negocio. Aquí, se materializan los conceptos centrales del dominio de la aplicación. Cuando un nuevo usuario se registra o se recupera información de uno existente, se invoca **new Usuario(...)**, creando una representación en memoria de esa entidad con sus atributos y comportamientos. Esta práctica garantiza que la lógica de negocio opere sobre modelos de datos coherentes y bien definidos. Asimismo, la orquestación de estas operaciones recae en los controladores, que también son instanciados dinámicamente. Por ejemplo**, new ControladorUsuario()** se encarga de gestionar las interacciones relacionadas con los usuarios, validando datos, coordinando con la capa de persistencia y actualizando el estado de la aplicación. Esta clara separación de responsabilidades, facilitada por la creación de objetos, contribuye a un diseño robusto y mantenible, donde cada componente cumple una función específica dentro del ecosistema del software.
En las clases de modelo, se hace uso de *constructores parametrizados*, como por ejemplo, **public Usuario(int id, String nombre, ...)**. Esta práctica es fundamental para garantizar que todos los objetos sean inicializados con un estado válido y completo desde el momento de su creación. Al requerir que los valores esenciales se pasen como argumentos durante la instanciación, se previene la creación de objetos con datos incompletos o inconsistentes, lo que podría llevar a errores en tiempo de ejecución o a un comportamiento impredecible de la aplicación. Esta estrategia de diseño fomenta la encapsulación y la inmutabilidad de los objetos, contribuyendo a un código más robusto y fácil de mantener. Además, facilita la detección temprana de errores de inicialización, ya que el compilador exigirá la provisión de todos los parámetros requeridos.
En el desarrollo de software, la implementación de estructuras de control es fundamental para definir el flujo lógico de una aplicación. Las estructuras condicionales (if/else) son pilares esenciales, ya que permiten ejecutar bloques de código específicos basándose en la evaluación de una condición booleana. Por ejemplo, en una LoginScreen, estas condicionales son cruciales para la validación de credenciales, verificando si el nombre de usuario y la contraseña ingresados coinciden con los registrados. De manera similar, en los controladores de una aplicación, las sentencias if/else son indispensables para la verificación de permisos por rol, asegurando que solo los usuarios con la autoridad adecuada puedan acceder a determinadas funcionalidades o recursos. Esto contribuye directamente a la seguridad y la integridad del sistema.

Por otro lado, las estructuras repetitivas (for/while) son igualmente vitales para la automatización y la eficiencia del código. Su principal función es la iteración sobre colecciones de datos, lo que se manifiesta de diversas maneras. Un caso común es la AdminScreen, donde un bucle for podría utilizarse para listar usuarios, recorriendo una array o una lista de objetos de usuario y mostrando la información relevante de cada uno. Asimismo, estas estructuras son imprescindibles para consultar registros de la base de datos. Un bucle while podría emplearse para procesar los resultados de una consulta, extrayendo y manipulando cada fila de datos hasta que no queden más registros. La elección entre for y while a menudo depende de si se conoce de antemano el número de iteraciones o si la iteración debe continuar mientras una condición específica sea verdadera, respectivamente. Ambas estructuras son indispensables para el manejo dinámico de datos y la automatización de tareas repetitivas en cualquier aplicación.
El sistema se articula en torno a un Dashboard Principal (MainDashboard), que sirve como el centro neurálgico de la interfaz de usuario. Desde este dashboard, los usuarios pueden acceder a diversas funcionalidades a través de botones de acción. Estos botones, tales como AdminScreen y ValidationScreen, no son meras opciones de navegación, sino que actúan como un menú gráfico de selección que encapsula operaciones específicas.

La **interacción dentro del sistema** está diseñada para ser directa, intuitiva y orientada a la acción, lo que simplifica la experiencia del usuario y optimiza la operatividad. Cada elemento interactivo en la interfaz de usuario, como los botones, está vinculado a una funcionalidad específica. Al pulsar un botón, el sistema no solo reacciona visualmente, sino que invoca directamente un método predefinido que reside en los controladores del sistema. Estos controladores son componentes clave de la arquitectura de la aplicación, responsables de gestionar la lógica de negocio y coordinar las respuestas a las interacciones del usuario.

Eso se logra por una correcta aplicación del modelo vista controlador que en la siguiente tabla queda expresada:

Capa	Paquete	Contenido
Presentación (UI)	sate.ui	SATEApp, LoginScreenFX, ValidationScreenFX, etc.
Negocio (Business)	sate.controlador	ControladorUsuario, ControladorEvento, ControladorNOAA.
Modelo (Domain)	sate.modelo	Clases puras de entidades (Usuario, Reporte, Evento, NOAAHotspot).
Datos (Persistence)	sate.datos	DBConnector, DAOs (UsuarioDAO).

Esta arquitectura se adhiere rigurosamente al principio de la separación de responsabilidades, una práctica fundamental en el diseño de software robusto y escalable. Garantiza una clara y definida distinción entre la capa de presentación y la capa de lógica de negocio.

- **Capa de Presentación:** Incluye elementos como el dashboard, los formularios, los botones y cualquier otro componente visual con el que el usuario interactúa directamente. Su función principal es mostrar información y capturar las entradas del usuario.
- **Capa de Lógica de Negocio (Controladores y sus Métodos):** Contiene las reglas y procesos que definen el comportamiento del sistema en respuesta a las acciones del usuario. Aquí se implementan las operaciones fundamentales como la creación, lectura, actualización y eliminación de datos, así como validaciones y cualquier otra lógica específica del dominio.

Esta clara separación no solo mejora la organización del código, sino que también ofrece beneficios significativos tanto en la fase de desarrollo como en el mantenimiento a largo plazo del sistema:

- **Facilidad de Desarrollo:** Los desarrolladores pueden trabajar en la interfaz de usuario y en la lógica de negocio de manera independiente, lo que permite un desarrollo paralelo y reduce la probabilidad de conflictos. Las modificaciones en la apariencia o el diseño no requieren cambios en la lógica subyacente, y viceversa.
- **Mantenimiento Simplificado:** Cuando surge la necesidad de actualizar o corregir errores, la modularidad del sistema permite identificar y aislar rápidamente la sección afectada. Si un error está en la lógica de eliminación de un usuario, se sabe exactamente dónde buscar (en el método eliminarUsuario() del controlador). De igual manera, si se desea cambiar el texto o el estilo de un botón, la modificación se limita a la capa de presentación.
- **Reusabilidad:** Los métodos en los controladores pueden ser diseñados para ser reutilizables, lo que evita la duplicación de código y promueve la consistencia en el comportamiento del sistema.
- **Escalabilidad:** Un sistema bien segmentado es más fácil de escalar. Se pueden añadir nuevas

funcionalidades o modificar las existentes con un impacto mínimo en otras partes del sistema.
En resumen, la interacción directa y la arquitectura basada en controladores con una clara separación de capas son pilares fundamentales para un sistema eficiente, mantenible y adaptable a futuras expansiones o cambios.

Tratamiento y Manejo de Excepciones

El proyecto SATE implementa un manejo robusto y proactivo de excepciones, fundamental para garantizar la estabilidad operativa del sistema, mantener la integridad de los datos y asegurar la coherencia con los Requerimientos No Funcionales (RNF) previamente establecidos. Este enfoque integral se traduce en una experiencia de usuario fluida y un sistema resiliente ante situaciones imprevistas.

Manejo de Excepciones por Capa:

- Capa de Datos (DBConnector): Si bien esta planteada no esta ejecutandose porque no esta resuelta como se termina de implementar la BBDD. En esta capa crítica, se utiliza una estrategia rigurosa de bloques try-catch para anticipar y gestionar errores que podrían comprometer la conexión o la manipulación de la base de datos. Los tipos de excepciones manejadas incluyen, pero no se limitan a:
    - SQLException: Esta excepción es crucial para detectar y gestionar problemas relacionados con la interacción con la base de datos, como fallos en la conexión, consultas SQL malformadas, violaciones de restricciones de integridad, o errores durante las transacciones. Un manejo adecuado de SQLException permite al sistema intentar recuperarse (por ejemplo, reintentar la conexión) o, en su defecto, fallar de manera controlada sin detener la aplicación abruptamente.
    - ClassNotFoundException: Esta excepción se captura cuando el sistema no puede localizar o cargar el controlador JDBC necesario para establecer la conexión con la base de datos. Su manejo es vital durante la fase de inicialización del sistema para asegurar que todos los componentes de acceso a datos estén disponibles.
      La implementación de estos bloques permite al sistema "fallar organizadamente". Esto significa que, en lugar de un colapso total, el sistema puede capturar el error, registrarlo para análisis posterior, y notificar adecuadamente al usuario o a otros módulos del sistema sobre la situación, manteniendo así la aplicación en un estado operativo, aunque con funcionalidades potencialmente limitadas.
- **Propagación y Manejo en la UI:** Las excepciones que son capturadas en la capa de datos y no pueden ser resueltas internamente se propagan de manera controlada hacia la capa de presentación. Este proceso es esencial para mantener al usuario informado y para guiarlo a través de flujos alternativos, en línea con los Casos de Uso definidos.
    - **Notificación al Usuario:** Una vez que una excepción llega a la Capa de Presentación (UI), el sistema utiliza mecanismos como JOptionPane.showMessageDialog para mostrar mensajes de error claros, concisos y comprensibles para el usuario final. Estos mensajes no solo informan sobre la falla, sino que también pueden sugerir acciones correctivas o indicar el estado actual del sistema.
    - **Flujos Alternativos en Casos de Uso:** La propagación de excepciones y su manejo en la UI están directamente vinculados a los "Flujos Alternativos" definidos en los Casos de Uso. Por ejemplo, si se produce un error al intentar guardar cambios (ej., CU-10: "No se guardan los cambios"), el sistema mostrará un mensaje descriptivo al usuario, permitiéndole entender la causa del problema y, si es posible, reintentar la operación o tomar una acción

alternativa. Esto asegura que, incluso en caso de error, la experiencia del usuario sea predecible y orientada a la solución.

En resumen, el sistema SATE no solo se enfoca en la detección de errores, sino que también implementa una estrategia integral para su manejo, propagación y notificación, garantizando la robustez, la usabilidad y la adherencia a los estándares de calidad definidos en sus

Aplicación de los Pilares de POO

El diseño arquitectónico del Sistema de Alerta Temprana de Eventos (SATE) se fundamenta en cuatro principios esenciales de la programación orientada a objetos (POO), cuya correcta y rigurosa implementación es crucial para garantizar la modularidad, la mantenibilidad y la escalabilidad del sistema a largo plazo. Estos pilares no solo estructuran el código, sino que también facilitan la colaboración entre desarrolladores y la adaptación a futuros requisitos.

- **Encapsulamiento:** Este principio se logra principalmente a través del uso estratégico de modificadores de acceso. En SATE, los atributos o el "estado" de los objetos del modelo son declarados como private, lo que significa que no pueden ser accedidos directamente desde fuera de la clase. En su lugar, se proporcionan métodos de acceso públicos (getters) para recuperar los valores de los atributos y métodos de mutación (setters) para modificarlos. Esto permite un control estricto sobre cómo se acceden y modifican los datos, protegiendo la integridad del objeto. Por ejemplo, en la clase Usuario, atributos sensibles como passwordHash (por razones de seguridad) y rol se mantienen privados. Su acceso y modificación están estrictamente controlados por métodos públicos específicos que pueden implementar validaciones o lógicas de negocio adicionales, evitando así manipulaciones directas y potencialmente inseguras. Este enfoque previene estados inconsistentes y mejora la seguridad general del sistema.
- **Abstracción:** La abstracción en SATE se implementa mediante la división del sistema en capas bien definidas, lo que permite a los desarrolladores interactuar con funcionalidades de alto nivel sin necesidad de comprender los detalles intrínsecos de su implementación subyacente. Un ejemplo claro de esto es la separación entre la **Capa de Negocio** y la **Capa de Datos**. La ControladorUsuario, que reside en la Capa de Negocio, actúa como una abstracción de la DBConnector en la Capa de Datos. Cuando un desarrollador invoca un método como ControladorUsuario.eliminarUsuario(id), no necesita saber si la operación subyacente se ejecuta en una base de datos MySQL, un clúster de Elasticsearch o cualquier otro sistema de almacenamiento de datos. Su única preocupación es la funcionalidad expuesta por el método: la eliminación de un usuario. Esta abstracción facilita la intercambiabilidad de las tecnologías de persistencia sin afectar la lógica de negocio y simplifica el desarrollo al ocultar la complejidad inherente de las operaciones de bajo nivel.
- **Herencia (Propuesta de Diseño):** La herencia es un mecanismo fundamental para establecer jerarquías de clases y promover la reutilización de código. En el contexto de SATE, se propone aplicar la herencia en el **Modelo de Eventos** para gestionar la diversidad de eventos que el sistema debe monitorear. Se diseñaría una clase base genérica Evento que encapsularía todos los atributos comunes a cualquier tipo de evento, como latitud, longitud y fecha. Posteriormente, subclases específicas como FocoCalor (para incendios o anomalías térmicas) y CuerpoDeAgua (para inundaciones o variaciones en niveles hídricos) heredarían estos atributos de la clase Evento. Además de los atributos heredados, cada subclase añadiría propiedades específicas que son relevantes únicamente para su tipo particular de evento (por ejemplo, temperatura para FocoCalor

o nivel para CuerpoDeAgua). Esto no solo reduce la duplicación de código, sino que también crea una estructura lógica y extensible para manejar nuevos tipos de eventos en el futuro.

- **Polimorfismo (Propuesta de Diseño):** El polimorfismo, que significa "muchas formas", permite que objetos de diferentes clases sean tratados como objetos de una clase común, respondiendo de manera diferente a la misma llamada a método. En SATE, el polimorfismo se implementaría mediante el uso de interfaces o la sobreescritura de métodos. Por ejemplo, se podría definir una interfaz IVisualizable que establecería un contrato para la representación visual de diferentes elementos del sistema. Tanto la clase Evento como la clase ReporteManual (que podría representar observaciones manuales de usuarios) podrían implementar esta interfaz. Al hacerlo, ambas clases estarían obligadas a definir un método obtenerCoordenadas(). Sin embargo, la implementación de obtenerCoordenadas() se comportaría de manera diferente según el objeto que la implemente: un Evento podría devolver las coordenadas de su origen, mientras que un ReporteManual podría devolver las coordenadas del lugar donde fue registrado por el usuario. Esto permite al sistema manejar objetos heterogéneos de manera uniforme a través de una interfaz común, facilitando la construcción de componentes de visualización que pueden interactuar con diferentes tipos de datos sin necesidad de conocer su tipo concreto en tiempo de compilación.

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 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.

Informe Funcional y Arquitectónico del Sistema SATE

Sistema: Sistema de Alerta Temprana de Eventos (SATE)

Arquitectura: Diseño por Capas (Modelo, Datos, Negocio, Presentación)

Tecnología: JavaFX, Java

Prerrequisitos para ejecutar la aplicación:

- Se deben cargar las librerias de JavaFX. Las mismas se encuentran en la carpeta *lib\openjfx-21.0.9_windows-x64_bin-sdk\javafx-sdk-21.0.9\lib*
- Se debe cargar la libreria mysql, la cual se encuentra en la carpeta *lib\mysql-connector-j-9.4.0\mysql-connector-j-9.4.0*
- *Se debe obtener una map key para poder realizar la consulta, la cual se obtiene desde esta url: https://firms.modaps.eosdis.nasa.gov/api/map_key/*

1. Funcionamiento General y Arquitectura

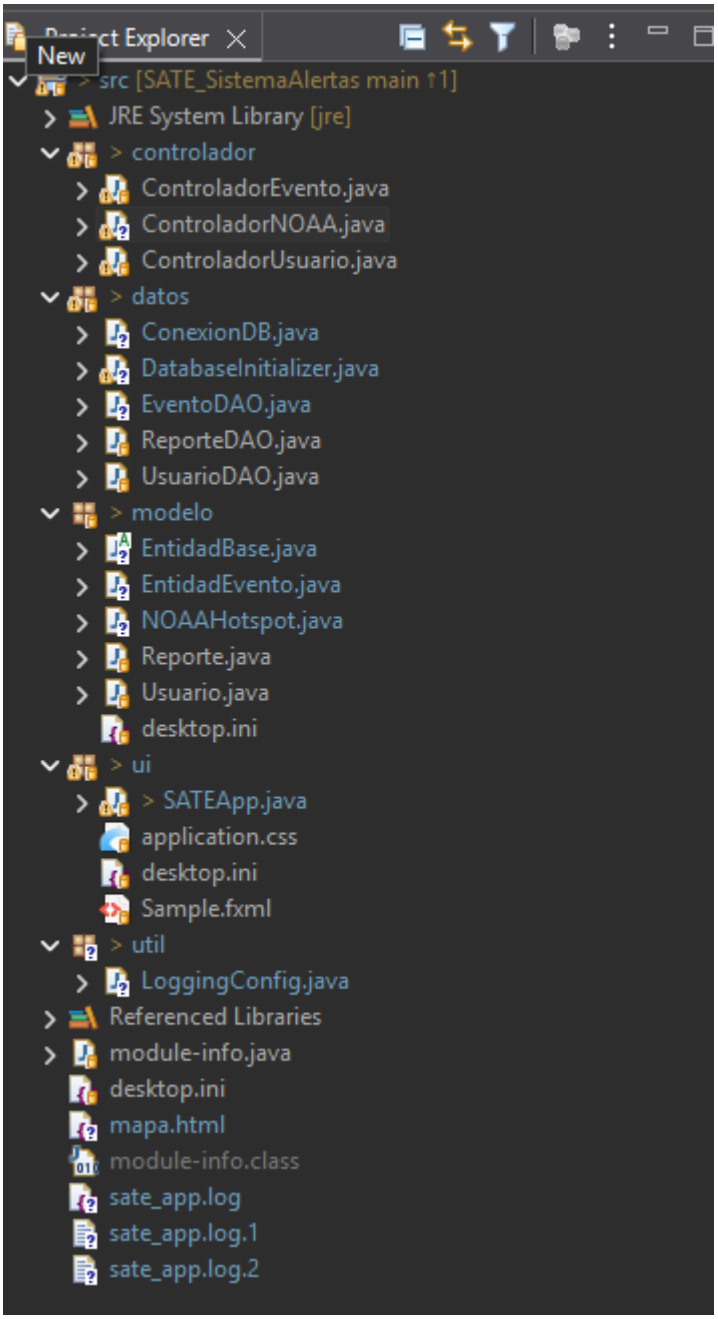
SATE sigue una arquitectura MVC (Modelo-Vista-Controlador) que separa la lógica de negocio, la interfaz de usuario y el acceso a datos. Esta separación facilita el mantenimiento y la escalabilidad del sistema.

- **Modelo**: Contiene las entidades Reporte y Usuario para los datos del sistema ingresados manualmente y NOAAHotspot para los focos de calor de NOAA.
- **Vista**: Implementada con JavaFX, incluye pantallas para login, registro, dashboard, reporte manual,

validación de reportes y gestión de usuarios.

- **Controlador**: Maneja la lógica de negocio y la comunicación entre la vista y los datos.
- **Capa de Datos**: DAOs (Data Access Objects) para acceder a la base de datos y a los servicios externos (FIRMS[9]).

Estructura de Capas (Paquetes)



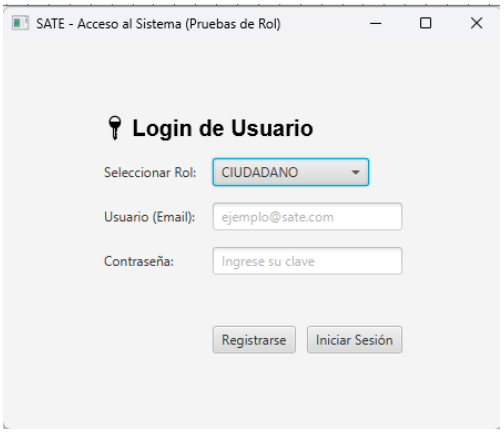
Flujo de Acceso (Login)

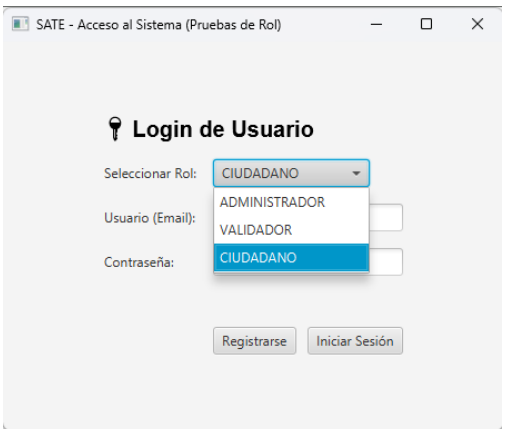
1. El usuario interactúa con la vista *LoginScreenFX* (Capa de Presentación).
2. LoginScreenFX llama al método *iniciarSesion*(email, password) en el ControladorUsuario (Capa de

[9] Fire Information for Resource Management System

Negocio).

3. *ControladorUsuario* consulta al UsuarioDAO (Capa de Datos) para verificar las credenciales.
4. Si las credenciales son válidas, ControladorUsuario devuelve el objeto Usuario.
5. *LoginScreenFX* lanza la pantalla principal (MainDashboardFX), ajustando las funcionalidades visibles según el rol del usuario (ADMINISTRADOR, VALIDADOR, CIUDADANO).





Conexión a la Base de Datos

La conexión a la base de datos es fundamental para la persistencia de los reportes y la información de usuarios. El sistema utiliza una base de datos MySQL para almacenar:

- **Reportes**: Información sobre eventos reportados manualmente por los usuarios (ciudadanos, validadores, administradores). Cada reporte incluye tipo, descripción, coordenadas (en formato WKT), fecha y estado (pendiente, aprobado, rechazado).
- **Usuarios**: Datos de los usuarios del sistema, con roles diferenciados (ciudadano, validador, administrador).

La importancia de la base de datos radica en:

- **Persistencia**: Los datos no se pierden al cerrar la aplicación.
- **Gestión de Estados**: Permite el flujo de trabajo de los reportes (desde que se crean hasta que son aprobados o rechazados).
- **Seguridad**: Almacena credenciales de usuarios y roles para control de acceso.
- **Histórico**: Mantiene un registro de todos los eventos para análisis posteriores.

Conexión a FIRMS (NASA)

FIRMS (Fire Information for Resource Management System) es un servicio de la NASA que proporciona datos de focos de calor en tiempo casi real (NRT), detectados por satélites como MODIS y VIIRS. La integración con FIRMS permite:

- **Datos en Tiempo Real**: Obtener focos de calor activos en las últimas 24-48 horas.
- **Cobertura Global**: Acceso a datos de todo el mundo, con especial foco en Sudamérica y Argentina.
- **Automatización**: Los focos de calor se cargan automáticamente sin intervención manual, lo que agiliza la detección temprana de incendios.

La importancia de esta conexión es:

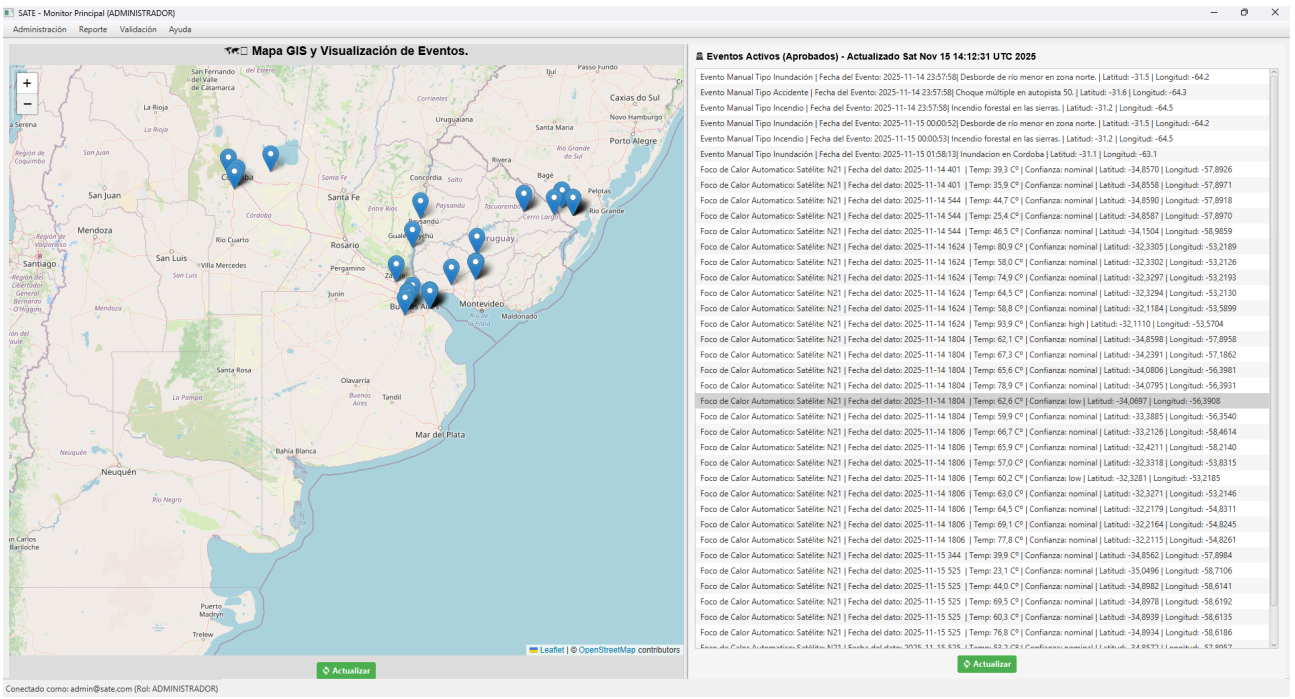
- **Complementariedad**: Los reportes manuales (del sistema SATE) y los automáticos (FIRMS) se combinan para ofrecer una visión más completa de la situación.
- **Rapidez en la detección**: Los satélites pueden detectar focos de calor en zonas remotas o de difícil acceso, permitiendo una respuesta más rápida.
- **Validación Cruzada**: Los validadores pueden contrastar los reportes manuales con los datos de FIRMS para tomar decisiones más informadas.

En el caso de SATE utilizamos **VIIRS_NOAA21_NRT** (VIIRS NOAA-21 Near Real-Time). Debe obtenerse una API KEY que se ingresa en el módulo controlador para NOAA (*ControladorNOAA.java*):

```
private static final String API_KEY = "0520efcxxxxxxxxxxxxxxxxxxxxx";
```

2. Funcionalidades Clave y Detalle por Rol

El sistema SATE ofrece un conjunto de funcionalidades que varían según el rol asignado al usuario.

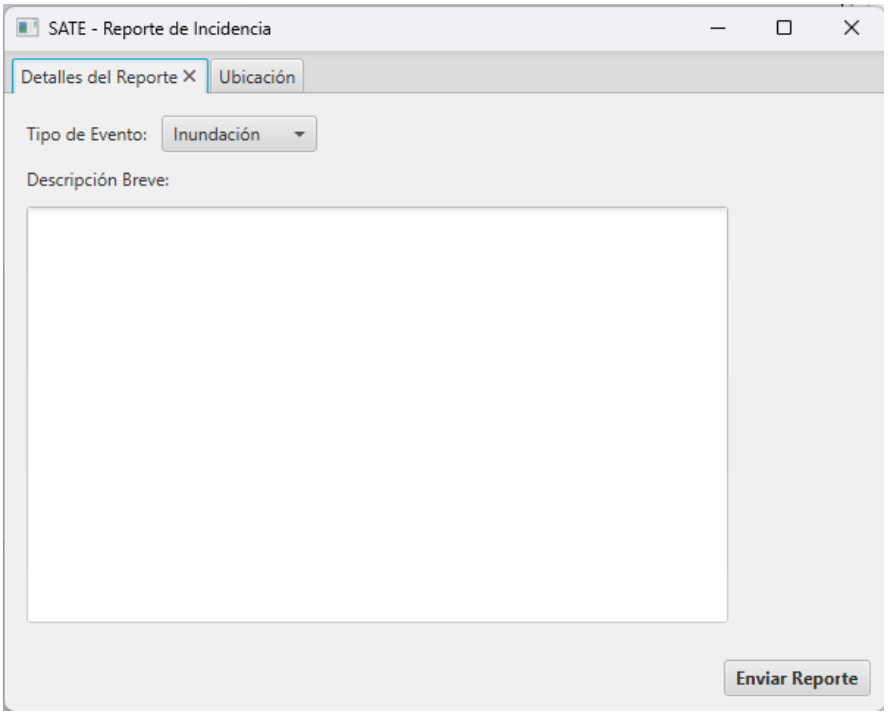


A. Funcionalidades de Autenticación y Registro

Módulo	Descripción	Vistas Implicadas
Login	Permite el acceso seguro al sistema. Dirige al usuario al Dashboard principal según su nivel de permiso (rol).	LoginScreenFX
Registro	Permite dar de alta nuevos usuarios, inicialmente con el rol por defecto de CIUDADANO.	RegisterScreenFX

B. Funcionalidades de Reporte

Funcionalidad	Descripción	Rol(es) Autorizado(s)
Reporte Manual	Proporciona un formulario para que los usuarios (ciudadanos o personal de emergencias) ingresen manualmente un nuevo evento (Incendio, Inundación, etc.), incluyendo una descripción y las coordenadas geográficas (WKT). Estos reportes entran en la cola de PENDIENTES.	**Todos (ADMINISTRADOR, VALIDADOR, CIUDADANO)**
Reporte Automático	Estos reportes se ingresan automáticamente tomando datos para toda América del Sur sobre focos de calor (posibles incendios) de la plataforma FIRMS (NASA). Estos incluyen una descripción (incluido el nivel de confianza del evento) y las coordenadas geográficas (WKT). Estos reportes estan aprobados automaticamente y no se almacenan en la BBDD.	**Todos (ADMINISTRADOR, VALIDADOR, CIUDADANO)**

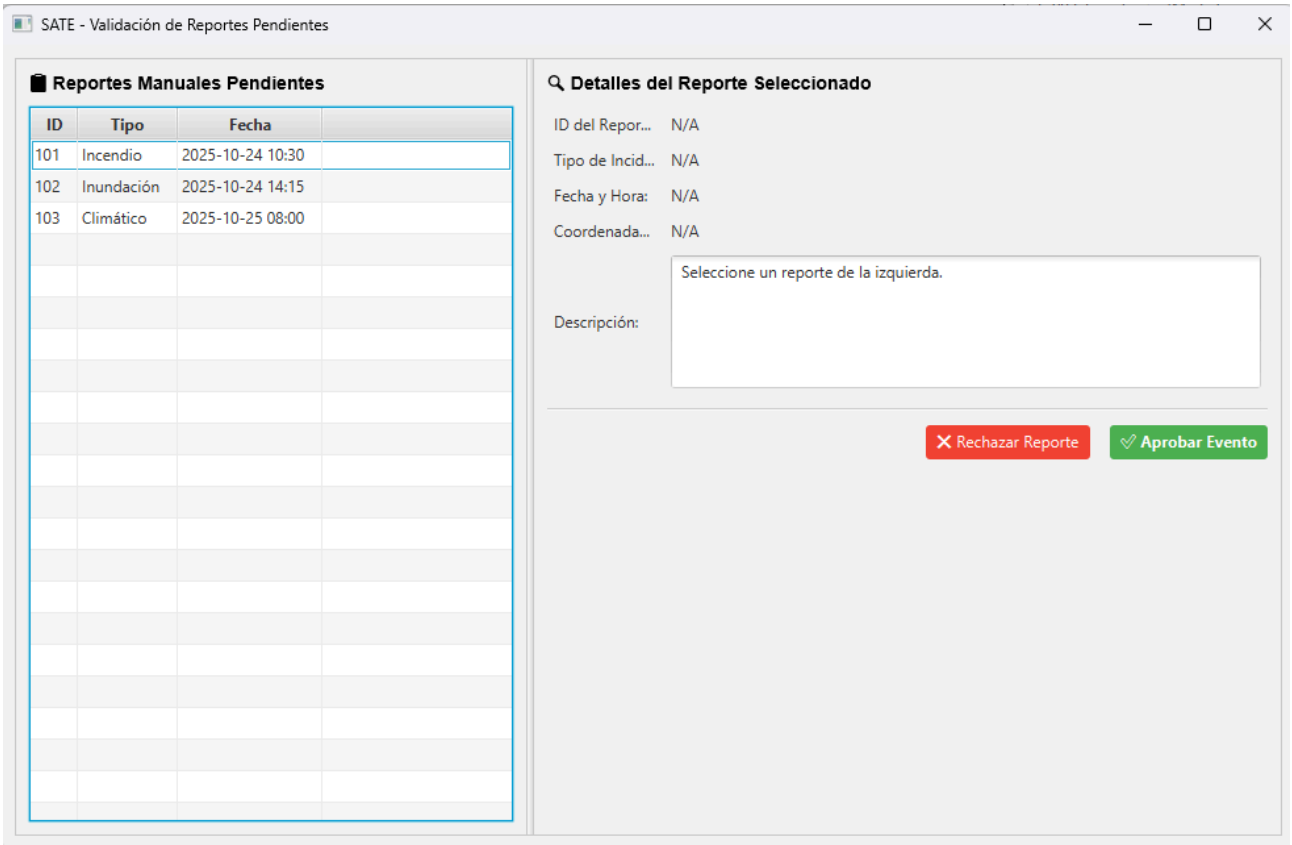


C. Funcionalidades de Validación (Negocio Crítico)

Esta es la funcionalidad central para transformar una alerta bruta en un evento oficial del sistema.

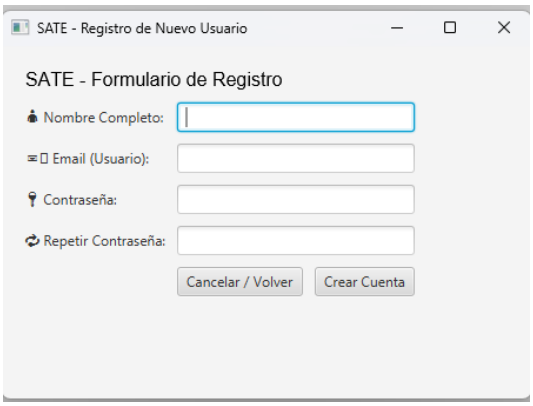
Funcionalidad	Descripción	Rol(es) Autorizado(s)
Revisar Pendientes	Muestra una tabla con todos los Reportes en estado PENDIENTE almacenados por el ReporteDAO.	ADMINISTRADOR, VALIDADOR

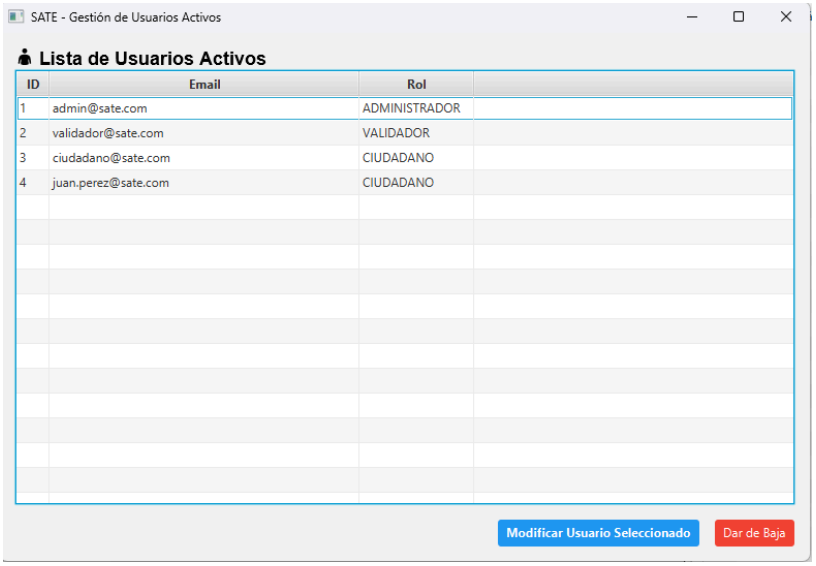
Aprobar Evento	Tras seleccionar un reporte, el usuario lo aprueba. La lógica del ControladorEvento actualiza el estado a APROBADO y lo elimina de la cola de pendientes. (Simula la creación de un Evento Oficial).	ADMINISTRADOR, VALIDADOR
Rechazar Reporte	Descarta el reporte por duplicado o inconsistencia. El ControladorEvento lo marca como RECHAZADO y lo retira de la cola de pendientes.	ADMINISTRADOR, VALIDADOR



D. Funcionalidades de Administración

Funcionalidad	Descripción	Rol(es) Autorizado(s)
Gestión de Usuarios	Muestra una tabla con todos los usuarios activos del sistema, incluyendo su ID, Email y Rol. Permite simular las acciones de Modificación y Baja de usuarios.	ADMINISTRADOR, VALIDADOR





3. Funcionamiento de la Interfaz de Usuario (UI)

La clase principal, MainDashboardFX, centraliza la navegación y aplica las restricciones de acceso.
Estructura del Dashboard (MainDashboardFX)

1. **Menú de Navegación (Top):** Construye la barra de menú (MenuBar). La visibilidad de las opciones (Administración, Validación) está condicionada por el rol con el que se inicia la sesión.
2. **Panel GIS (Centro - 75%):** Un panel simulado que representa la integración futura de un componente de visualización geográfica (GIS) y mapeo de eventos. Botón de actualización de las referencias en el mapa.
3. **Reportes Activos (Derecha - 25%):** Una barra lateral que simula la lista de eventos que ya han sido validados y están activos. Botón de actualización de las referencias en el mapa.
4. **Barra de Estado (Bottom):** Muestra el rol y el estado de conexión actual del usuario.

Detalle de Implementación JavaFX

- **Modelo para TableView:** Las entidades **Reporte** y **UserSimulado** utilizan SimpleStringProperty o getters compatibles con JavaFX Property Value Factory. Esto es esencial para que la TableView pueda reflejar automáticamente los datos de las entidades de forma limpia y eficiente.
- **Encapsulamiento de Vistas:** Cada pantalla (Login, Registro, Validación, Reporte Manual) se encapsula en su propia clase Application (o Stage), promoviendo la modularidad y haciendo la UI

más fácil de mantener y probar.

Bibliografía

- (N.d.). Eos.org. Retrieved September 13, 2025, from https://eos.org/editors-vox/seeing-surface-water-from-space
- (N.d.-b). Mappinggis.com. Retrieved September 14, 2025, from https://mappinggis.com/2024/05/openlayers-vs-leaflet-mejor/
- Pressman, R. S. (2005). Software Engineering: A Practitioner's Approach. McGraw Hill Higher Education
- 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. (2024). Modulo 1-4 - Lectura 1-4. POO
- Siglo XXI. (2024). Modulo 1-4 - Lectura 1-4. Estructuras de Datos
- Siglo XXI. (2025). Modulo 1-2 - Lectura 1-2. Seminario de Practica Profesional
- Bravo, L. D., García, U., Hernández, M. M., & Ruiz, M. V. (2013). La entrevista, recurso flexible y dinámico. *Investigación en educación médica*, *2*(7), 162–167. https://doi.org/10.1016/s2007-5057(13)72706-6
- Nasa-firms. (n.d.). Nasa.gov. Retrieved November 15, 2025, from https://firms.modaps.eosdis.nasa.gov/api/

Repositorio

Se puede acceder al codigo fuente, imagenes y otra documentación a través de la siguiente URL que conduce al repositorio *SATE_SistemaAlerta*:

- https://github.com/emilianobaum/SATE_SistemaAlertas.git