



**Documento de Diagnóstico y Plan de Acción  
Implementación de un Sistema de Alerta Temprana (SAT) de Deslizamientos**

**Octubre de 2025**

**Felipe Ruiz Zea**

**Ciencia de Datos e Ingeniería (Prueba Técnica para el puesto de analista de datos  
en mantenimiento)**

Contenido

<b>1. Diagnóstico y Alcance del Proyecto</b>	<b>3</b>
1.1. Contexto de la Problemática	3
1.2. Objetivo General	4
1.3. Objetivos Específicos	4
1.4. Alcance y Supuestos Técnicos	4
1.5. Limitaciones Identificadas	4
1.6. Resultado Esperado	5
<b>2. Fuentes y Estructura de Datos</b>	<b>5</b>
2.1. Panorama General de las Fuentes de Información	5
2.2. Modelo de Datos Unificado	6
2.3. Pipeline ETL Serverless: Especificación Técnica	6
2.4. Supuestos y Dependencias Técnicas	7
<b>3. Análisis Exploratorio y Definición del Modelo de Riesgo</b>	<b>7</b>
3.1. Objetivo del Análisis	7
3.2. Procesamiento de Datos Geoespaciales a Gran Escala	7
3.3. Cálculo del Riesgo Estático Multifactorial	9
3.4. Definición del Modelo de Alerta Dinámica: Matriz de Riesgo	9
3.5. Validación a través del Prototipo Técnico	10
<b>4. Propuesta de Arquitectura Técnica Escalable (AWS)</b>	<b>11</b>
4.1. Principios de Diseño	11
4.2. Arquitectura de Referencia Cloud-Native (AWS)	12
4.2.1. Frontend (Capa de Visualización para el CSM)	12
4.2.2. Backend (Capa de Ingesta y Lógica)	12
4.2.3. Base de Datos (Capa de Persistencia y "Puente")	13
4.3. Flujo de Datos en Producción	13
<b>5. Plan de Acción e Implementación (Fase 1)</b>	<b>13</b>
5.1. Objetivo de la Fase	13
5.2. Cronograma de Implementación Detallado	13
5.3. Entregables Finales de la Fase 1	14
5.4. Requisitos y Supuestos para la Implementación	14
<b>6. Criterios de Éxito y Próximos Pasos</b>	<b>14</b>
6.1. Criterios de Éxito (Fase 1)	14

TABLE 2: ESTRUCTURA DE LA TABLA MAESTRA TORRES .....6

FIGURE 1-1:MAPA SIMULADO CON LAS TORRES TOLEDO SAMORÉ Y SAMORÉ – BANADÍA .....3

1. Diagnóstico y Alcance del Proyecto

1.1. Contexto de la Problemática

El departamento de Arauca, Colombia, presenta condiciones geográficas y climáticas que lo hacen altamente susceptible a fenómenos de movimientos en masa y deslizamientos de terreno, especialmente durante los períodos de lluvias intensas. Estas situaciones afectan de manera directa la infraestructura de transmisión de energía eléctrica, generando riesgos sobre torres, líneas y estaciones de ISA INTERCOLOMBIA.

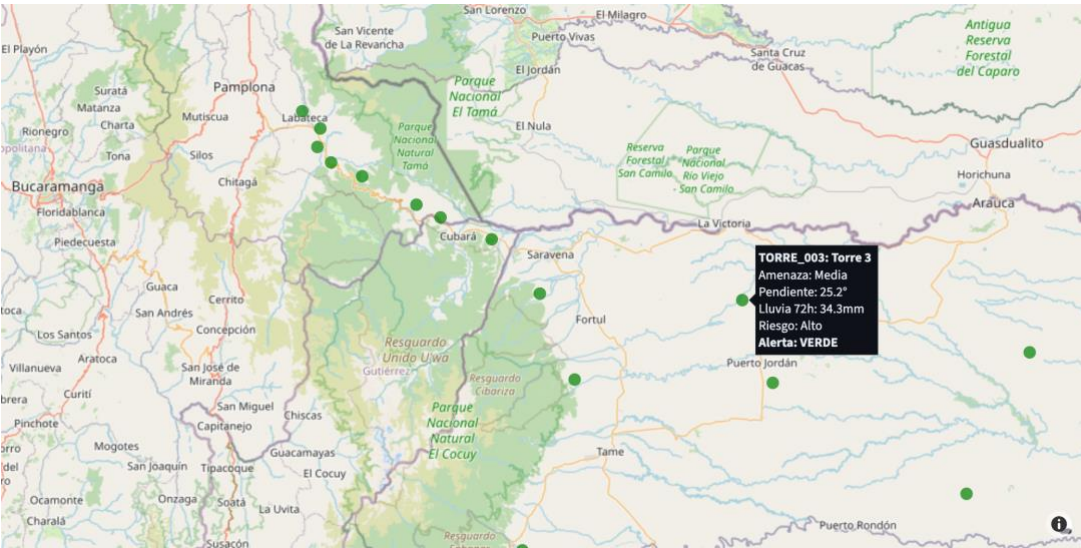


Figure 1-1:Mapa Simulado con las torres Toledo Samoré y Samoré – Banadía

En este contexto, se propone desarrollar un Sistema de Alertas Tempranas (SAT) orientado a identificar zonas críticas con alta probabilidad de deslizamientos. El objetivo es apoyar la toma de decisiones operativas durante eventos climáticos extremos, proporcionando señales oportunas sobre la vulnerabilidad del terreno y la exposición de activos críticos del sistema de transmisión.

La iniciativa se enmarca en los objetivos de gestión de riesgo climático y resiliencia operativa, alineados con la estrategia de sostenibilidad y continuidad del servicio energético.

## **1.2. Objetivo General**

Diseñar una arquitectura técnica de referencia para un sistema de alertas tempranas de deslizamientos, utilizando un enfoque de ciencia de datos, herramientas geoespaciales y tecnologías en la nube (AWS, asumida como plataforma base).

## **1.3. Objetivos Específicos**

- Analizar fuentes de datos relevantes (meteorológicas, topográficas, geológicas y de infraestructura eléctrica).
- Evaluar la disponibilidad y accesibilidad de los datos abiertos del Servicio Geológico Colombiano (SGC) y del IDEAM, identificando brechas y requerimientos.
- Simular un conjunto mínimo de datos representativos para validar el flujo de procesamiento y análisis espacial.
- Prototipar una visualización exploratoria en Streamlit que permita demostrar la integración y potencial del modelo.
- Proponer una arquitectura de nube escalable que soporte ingesta continua, modelado predictivo y visualización operacional.
- Definir un plan de acción a corto plazo (en días) que establezca los pasos iniciales para pasar del diagnóstico al desarrollo técnico.

## **1.4. Alcance y Supuestos Técnicos**

El presente documento corresponde a la fase diagnóstica y exploratoria del proyecto. Durante esta fase:

- Se asume el uso de AWS como entorno de nube principal para el despliegue de los componentes del sistema (ETL, modelado, almacenamiento y visualización).
- La arquitectura propuesta es agnóstica, y puede replicarse en GCP o Azure sin cambios estructurales significativos.
- La información geoespacial proveniente del Servicio Geológico Colombiano (SGC) y del IDEAM se considera de acceso restringido o no completamente pública; por tanto, se utilizaron datos simulados con las mismas estructuras y categorías oficiales para validar los flujos de integración.
- El prototipo Streamlit desarrollado tiene un carácter demostrativo, y no pretende ser el producto final, sino un vehículo para validar la factibilidad técnica del flujo de datos y visualización geoespacial.

## **1.5. Limitaciones Identificadas**

- Disponibilidad y granularidad de datos: Los conjuntos del SGC y del IDEAM no ofrecen actualizaciones en tiempo real, limitando la capacidad predictiva inmediata.
- Heterogeneidad de fuentes: Diferencias en formatos (GeoTIFF, CSV, Shapefile, API REST) y sistemas de coordenadas requieren procesos ETL de normalización.

- Ausencia de etiquetado histórico validado: Se carece de registros consistentes de eventos de deslizamiento georreferenciados para entrenar modelos supervisados.
- Capacidad de integración operacional: Se debe garantizar interoperabilidad con los sistemas internos de ISA (por ejemplo, SCADA o bases de datos de mantenimiento).

## 1.6. Resultado Esperado

La entrega de este diagnóstico tiene como propósito dejar definido un modelo técnico conceptual y funcional, que establezca los componentes mínimos para la implementación de un Sistema de Alertas Tempranas (SAT) con capacidad de:

- Recibir, procesar y analizar datos geoespaciales y meteorológicos.
- Identificar áreas críticas de amenaza.
- Emitir alertas configurables a partir de umbrales definidos.
- Servir como base para la implementación futura en producción en la nube (AWS).

## 2. Fuentes y Estructura de Datos

### 2.1. Panorama General de las Fuentes de Información

Para el desarrollo del sistema de alertas tempranas de deslizamientos en Arauca, se requiere la integración de múltiples fuentes de datos heterogéneas, que combinan información geológica, topográfica, meteorológica e infraestructural.

Estas fuentes tienen distintos niveles de disponibilidad, frecuencia de actualización y formatos de entrega, lo que implica la necesidad de una arquitectura flexible de ingesta y almacenamiento en la nube.

A continuación, se presentan las principales fuentes identificadas, diferenciando su estado de disponibilidad real, su formato original, y las suposiciones o simulaciones utilizadas en esta fase.

*Table 1: Fuentes de Datos para el Sistema de Alerta Temprana*

Fuente	Tipo	Formato / Acceso	Frecuencia	Disponibilidad real	Uso en prototipo
Servicio Geológico Colombiano (SGC) – Zonificación de Amenaza	Estático	GeoTIFF / Shapefile	Anual	Parcial (sin API pública)	Simulada (categorías oficiales)
IDEAM – Pronóstico Meteorológico	Dinámico	API REST (JSON)	Diario / Horario	Limitada (requiere autorización)	Simulada (valores estimados)
Infraestructura de ISA INTERCOLOMBIA	Estático	CSV / Base de datos corporativa	Variable	Restringida	Simulada (15 torres de prueba)
Historial de Deslizamientos	Histórico	CSV / Excel	Eventual	Escasa / No pública	Simulada con registros sintéticos
Modelo Digital de Elevación (MDE – NASA / SRTM)	Estático	GeoTIFF	Fijo	Pública (copias en AWS Open Data)	Usada directamente

Precipitaciones acumuladas (NASA POWER / ERA5)	Dinámico	API REST	3–6 horas	Pública	Usada directamente
--	----------	----------	-----------	---------	--------------------

### 2.2. Modelo de Datos Unificado

La estrategia central de la arquitectura de datos es la creación de un Modelo de Datos Unificado. En lugar de procesar datos de múltiples fuentes en tiempo real durante cada consulta, un proceso de backend se encargará de consolidar toda la información relevante en una única tabla maestra en la base de datos. Esta tabla, que llamaremos torres, será el corazón del sistema, donde cada registro representa una torre enriquecida con todas sus variables de riesgo.

Table 2: Estructura de la Tabla Maestra torres

Fuente de Datos	Formato	Tecnología de Integración	Frecuencia
Infraestructura ISA	CSV / Base de Datos	Carga inicial con <i>AWS Data Pipeline</i> o script en Python (pandas + SQLAlchemy).	Estático
Mapa de Amenaza (SGC)	GeoTIFF / Shapefile	Script de preprocesamiento con <i>Rasterio</i> y <i>GeoPandas</i> .	Estático
Modelo Digital de Elevación (MDE)	GeoTIFF	Preprocesamiento con <i>Rasterio</i> para cálculo de pendiente.	Estático
Precipitación (API IDEAM / NASA)	JSON	<i>AWS Lambda</i> con librería <i>requests</i> y almacenamiento en S3.	Cada hora

### 2.3. Pipeline ETL Serverless: Especificación Técnica

El núcleo del sistema es un pipeline ETL (Extract-Transform-Load) automatizado y serverless para minimizar la gestión y el costo.

- Extract (Extracción)
  - Tecnología: Función AWS Lambda (runtime Python 3.12).
  - Orquestación: Amazon EventBridge Scheduler configurado para invocar la Lambda cada hora.
  - Proceso: La Lambda ejecuta una llamada GET a la API de precipitación (ej. Open-Meteo) para las coordenadas de cada torre.
- Transform (Transformación)
  - Tecnología: La misma función AWS Lambda.
  - Librerías Clave: Pandas para calcular la suma de la ventana rodante de 72 horas; Numpy para operaciones numéricas.
  - Proceso: Calcula lluvia\_acum\_72h\_mm. Aplica la lógica de la Matriz de Riesgo (una serie de if/elif/else) para determinar el nivel\_alerta.
- Load (Carga)

- Tecnología: La misma función AWS Lambda.
- Librería Clave: psycopg2-binary para la conexión con la base de datos PostgreSQL.
- Proceso: La Lambda se conecta a Amazon RDS y ejecuta una sentencia UPDATE sobre la tabla activos\_monitoreados para actualizar los campos lluvia\_acum\_72h\_mm, nivel\_alerta y ultima\_actualizacion para cada torre.

## **2.4. Supuestos y Dependencias Técnicas**

**Credenciales:** Se requiere un Secret en AWS Secrets Manager para almacenar de forma segura las credenciales de la base de datos y de la API meteorológica, que será consumido por la función Lambda.

**Historial de Eventos:** El principal gap técnico es la ausencia de una base de datos de eventos históricos. Se recomienda crear una tabla eventos\_historicos y un mecanismo simple (ej. una app interna o un formulario) para que los equipos de campo puedan registrar futuros deslizamientos. Este dato es indispensable para validar y mejorar el modelo a futuro.

# **3. Análisis Exploratorio y Definición del Modelo de Riesgo**

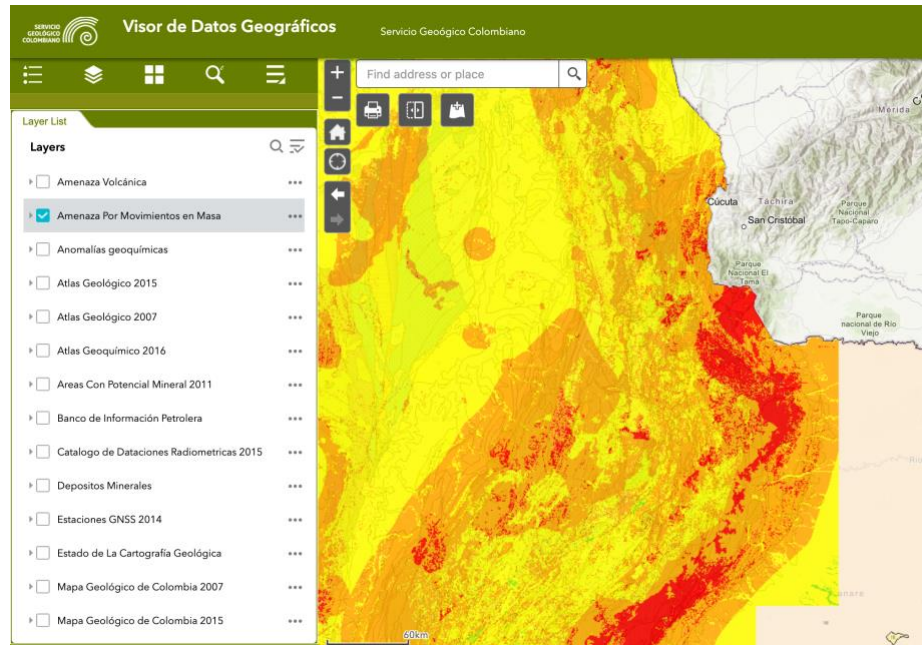
## **3.1. Objetivo del Análisis**

El objetivo de esta fase fue traducir datos geográficos y meteorológicos crudos en una lógica de alerta operacional. Las metas específicas fueron:

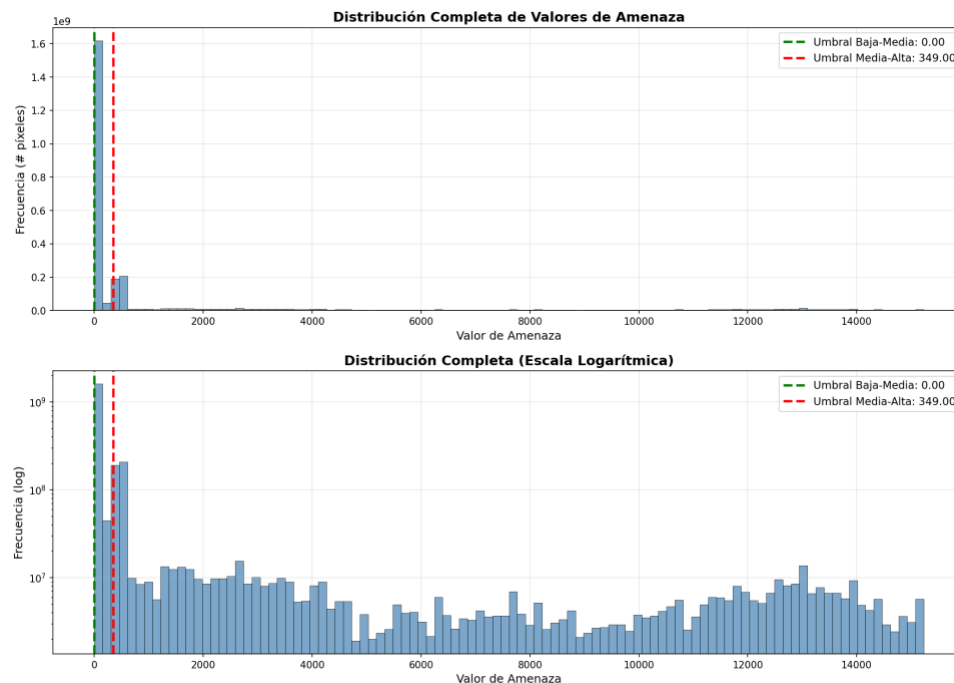
- Procesar el Mapa Nacional de Amenaza (AmeMM\_100k.tif) a escala completa para entender su distribución estadística real.
- Cuantificar múltiples factores de riesgo estático para cada torre.
- Definir una Matriz de Alerta dinámica que combine el riesgo estático con el disparador de precipitación.
- Validar la lógica y la visualización a través de un prototipo técnico funcional.

## **3.2. Procesamiento de Datos Geoespaciales a Gran Escala**

El insumo principal, el Mapa Nacional de Amenaza (AmeMM\_100k.tif), es un archivo GeoTIFF de gran tamaño (varios GB). Cargar este archivo completo en memoria para su análisis es ineficiente y no escalable.



Por ello, se implementó una estrategia de procesamiento por bloques (chunking) utilizando Python con la librería Rasterio. Este método consiste en leer y procesar el archivo en fragmentos (Windows de 2048x2048 píxeles), permitiendo calcular estadísticas precisas del dataset completo sin exceder los límites de la memoria RAM. Este enfoque es fundamental para asegurar que la metodología es aplicable a datasets de escala nacional.





Este proceso permitió analizar la totalidad de los píxeles válidos del mapa para obtener una comprensión profunda de la distribución de los valores de amenaza.

### 3.3. Cálculo del Riesgo Estático Multifactorial





Con un entendimiento completo de la distribución de los datos del SGC, se procedió a construir un modelo de riesgo estático multifactorial para determinar la vulnerabilidad inherente de cada torre, combinando dos factores clave.

- **Amenaza Geológica (SGC):** Se utilizó el Mapa Nacional de Amenaza, que clasifica el territorio en cinco niveles categóricos: Muy Baja, Baja, Media, Alta, y Muy Alta. A cada torre se le asignó la categoría correspondiente a su ubicación geoespacial.
- **Factor Topográfico (Pendiente):** Utilizando un Modelo Digital de Elevación (MDE), se calculó la pendiente exacta en grados para la ubicación de cada torre. Una mayor pendiente incrementa sustancialmente el riesgo de inestabilidad del terreno.

Estos dos factores se combinaron en una Clasificación\_Riesgo unificada (ej. Bajo, Medio, Alto) que representa el riesgo base de la torre, antes de considerar la lluvia.

### 3.4. Definición del Modelo de Alerta Dinámica: Matriz de Riesgo

El núcleo del sistema es la Matriz de Riesgo dinámica, que genera la alerta operacional. Esta lógica, implementada en la función calcular\_nivel\_alerta del prototipo, cruza el nivel de Amenaza SGC de la torre con el disparador de precipitación acumulada en 72 horas. El modelo se basa en un archivo de configuración (umbrales\_lluvia.csv) que define umbrales de lluvia específicos para cada uno de los cinco niveles de amenaza, como se muestra en el siguiente fragmento de la lógica de decisión:

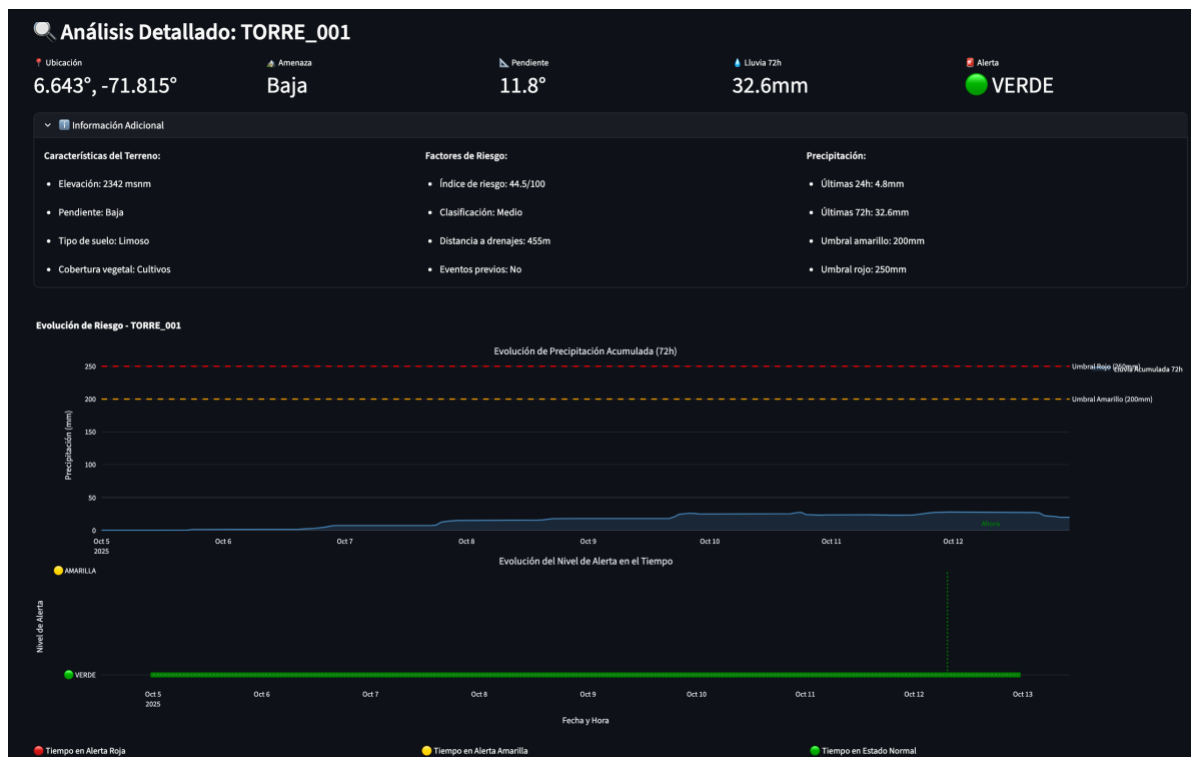
Condición	Nivel de Amenaza (amenaza)	Lluvia 72h (lluvia_72h)	Resultado	Símbolo	Código Numérico
Si no existe coincidencia del valor de amenaza en umbrales_df	—	—	VERDE		1
Si lluvia_72h >= Umbral_Rojo_mm	Coincide en umbrales_df	Mayor o igual al umbral rojo	ROJA		3
Si lluvia_72h >= Umbral_Amarillo_mm	Coincide en umbrales_df	Mayor o igual al umbral amarillo pero menor al rojo	AMARILLA		2
Si lluvia_72h < Umbral_Amarillo_mm	Coincide en umbrales_df	Menor al umbral amarillo	VERDE		1

Nivel de Amenaza SGC	Alerta  AMARILLA	Alerta  ROJA
Muy Baja	> 250 mm	> 300 mm
Baja	> 200 mm	> 250 mm
Media	> 150 mm	> 200 mm
Alta	> 100 mm	> 120 mm
Muy Alta	> 80 mm	> 100 mm

Este enfoque granular es significativamente más preciso, ya que reconoce que una torre en una zona de amenaza "Muy Alta" puede entrar en estado crítico con una fracción de la lluvia que necesitaría una torre en una zona de amenaza "Baja". Esta es la lógica exacta que se implementará en la función AWS Lambda.

### 3.5. Validación a través del Prototipo Técnico

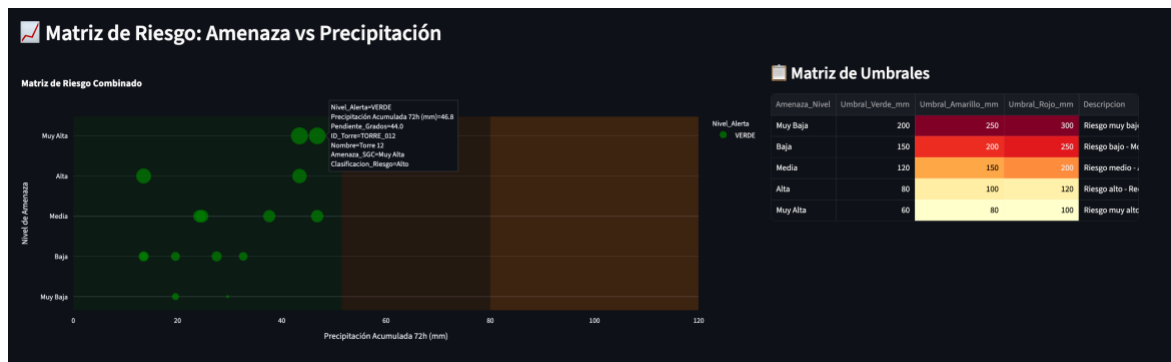
El prototipo en Streamlit fue la herramienta esencial para validar este modelo multifactorial de extremo a extremo. Su función no fue ser el producto final, sino mitigar el riesgo del proyecto al confirmar que:



Es factible integrar múltiples capas de datos (coordenadas, amenaza, pendiente, lluvia) en un único sistema.



La lógica de la Matriz de Riesgo de 5 niveles puede ser implementada y genera alertas coherentes con datos simulados.



La visualización de resultados es efectiva: El mapa geoespacial y los gráficos detallados por torre fueron validados como herramientas intuitivas y de alto valor para comunicar la conciencia situacional a los operadores del CSM.

## 4. Propuesta de Arquitectura Técnica Escalable (AWS)

El prototipo técnico validó la lógica del modelo, pero su arquitectura monolítica (donde la interfaz de usuario está acoplada al procesamiento de datos) no es viable para un entorno de producción debido a la alta latencia y la falta de escalabilidad. Para la solución operacional, se propone una **arquitectura de microservicios desacoplada y nativa de la nube**, diseñada para un rendimiento óptimo y una alta fiabilidad.

### 4.1. Principios de Diseño

- **Desacoplamiento:** La ingesta y procesamiento de datos (backend) deben ser completamente independientes de la capa de visualización (frontend). El operador nunca debe esperar por una llamada a una API.

- **Escalabilidad:** La arquitectura debe soportar un incremento de 10x a 100x en el número de activos monitoreados sin necesidad de rediseño.
- **Fiabilidad:** El sistema debe ser tolerante a fallos transitorios (ej. APIs externas no disponibles) y operar de forma autónoma.
- **Costo-Eficiencia:** Priorizar servicios *serverless* para minimizar costos fijos y la carga de gestión operacional.

## 4.2. Arquitectura de Referencia Cloud-Native (AWS)

La solución se desplegará íntegramente en Amazon Web Services, utilizando servicios gestionados para cada capa lógica.

### 4.2.1. Fronted (Capa de Visualización para el CSM)

Esta capa es la interfaz de usuario final que consumirán los operadores del CSM. Su único propósito es traducir los datos procesados por el backend en una visualización gráfica, interactiva y fácil de interpretar, cumpliendo con el requisito principal del proyecto.

- Tecnología de Interfaz (Streamlit): Se utilizará para construir el dashboard visual, incluyendo el mapa de alertas, los gráficos y las tablas. Es la herramienta que crea la experiencia de usuario.
- Tecnología de Empaquetado (Docker): La aplicación Streamlit se empaquetará en un contenedor Docker. Esto asegura que la aplicación se ejecute de manera consistente y segura.
- Servicio de Despliegue (AWS Fargate): Este servicio ejecutará el contenedor Docker 24/7, exponiéndolo a través de una URL web segura. Fargate garantiza que el dashboard esté siempre disponible, sea escalable y no requiera gestión de servidores, cumpliendo el requisito de una solución robusta.

### 4.2.2. Backend (Capa de Ingesta y Lógica)

- Tecnología: Función AWS Lambda con runtime de Python 3.12.
  - Orquestación: Amazon EventBridge Scheduler. Se configurará una regla para invocar la función Lambda de forma periódica (cron(0 \* \* \* ? \*)), es decir, al inicio de cada hora.
  - Función: Esta función contendrá la lógica de negocio principal: consultar la API de precipitación para cada torre, calcular los acumulados y aplicar la Matriz de Riesgo para determinar el nivel de alerta.
3. Base de Datos (Capa de Persistencia y "Puente")

#### 4.2.3. Base de Datos (Capa de Persistencia y "Puente")

- Tecnología: Amazon RDS (Relational Database Service) con el motor PostgreSQL y la extensión geoespacial PostGIS.
- Función Clave: Actúa como el punto de desacoplamiento.
- El backend (Lambda) escribe en ella el estado actualizado de cada torre cada hora.
- El frontend (Streamlit) solo lee de ella, ejecutando una consulta SQL simple y rápida.

#### 4.3. Flujo de Datos en Producción

El flujo de trabajo en la arquitectura propuesta es el siguiente:

1. Cada Hora: EventBridge invoca la función Lambda.
2. La Lambda lee las coordenadas de las torres desde RDS.
3. La Lambda ejecuta las llamadas a la API de precipitación, calcula lluvia\_acum\_72h\_mm y nivel\_alerta.
4. La Lambda se conecta a RDS y ejecuta una sentencia UPDATE para actualizar los registros de cada torre con los nuevos datos.
5. En cualquier momento: Un operador del CSM accede a la URL del dashboard.
6. AWS Fargate sirve la aplicación Streamlit.
7. Streamlit se conecta a RDS, ejecuta un SELECT sobre la tabla activos\_monitoreados y renderiza el dashboard con los datos ya procesados en menos de un segundo.

### 5. Plan de Acción e Implementación (Fase 1)

#### 5.1. Objetivo de la Fase

El objetivo de esta fase es desplegar la arquitectura descrita en la Sección 4 para tener un **Prototipo Operacional Mínimo Viable (MVP)** completamente funcional en un plazo de **5 días hábiles**. Este plan está diseñado para una ejecución ágil por un equipo con experiencia en AWS y Python.

#### 5.2. Cronograma de Implementación Detallado

El plan se divide en tareas diarias, con entregables específicos para validar el progreso.

Día	Actividad Clave	Tecnologías / Servicios	Entregable
1	Configuración de Infraestructura Base	AWS RDS (PostgreSQL + PostGIS), VPC, Security Groups, IAM Roles	Instancia de base de datos aprovisionada y accesible, con el esquema de tablas creado. Roles de seguridad definidos.
2	Carga de Datos Estáticos y Containerización	Python (psycopg2), Docker, AWS ECR	1. Datos de torres, amenaza (raster clasificado) y pendiente cargados en RDS. 2. Imagen Docker del dashboard de Streamlit construida y subida a ECR.
3	Despliegue del Frontend	AWS Fargate, Application Load Balancer (ALB)	URL pública y funcional del dashboard, mostrando los datos estáticos leídos desde RDS.
4	Despliegue del Backend Serverless	AWS Lambda, Amazon EventBridge Scheduler	Función Lambda desplegada y configurada para ejecutarse cada hora, actualizando exitosamente los datos de precipitación en RDS.
5	Integración, Seguridad y Entrega	AWS Cognito, Pruebas End-to-End	1. Sistema 100% funcional e integrado. 2. Acceso al dashboard restringido por autenticación. 3. Sesión de entrega y handover al CSM.

### 5.3. Entregables Finales de la Fase 1

Al concluir el quinto día, el equipo del CSM tendrá acceso a:

- Un Dashboard Operacional Funcional: Una URL segura para acceder al sistema de monitoreo en tiempo real.
- Infraestructura como Código (IaC) - Opcional: Plantillas de Terraform o AWS CloudFormation que definen toda la infraestructura, permitiendo su replicación o modificación de manera controlada.
- Documentación Técnica y de Usuario: Un resumen de la arquitectura desplegada y una guía rápida para los operadores del CSM.

### 5.4. Requisitos y Supuestos para la Implementación

- Permisos de AWS: Se requiere acceso a la cuenta de AWS de ISA con permisos suficientes para crear los recursos mencionados (RDS, Fargate, Lambda, IAM).
- Equipo Técnico: Se asume la disponibilidad de al menos un ingeniero con experiencia en DevOps/Cloud y Python.
- Datos de Entrada: Se debe contar con el archivo de coordenadas de las torres y el raster clasificado de amenaza al inicio del Día 1.

## 6. Criterios de Éxito y Próximos Pasos

### 6.1. Criterios de Éxito (Fase 1)

El éxito de esta fase de implementación rápida se medirá a través de criterios operacionales y técnicos cuantificables:

1. Entrega a Tiempo: El sistema completo está desplegado y 100% funcional en la infraestructura de AWS en el plazo de 5 días hábiles.
2. Rendimiento del Frontend: El tiempo de carga inicial del dashboard de Streamlit y las interacciones del usuario deben ser inferiores a 2 segundos.
3. Fiabilidad del Backend: La tasa de ejecución exitosa de la función AWS Lambda debe ser superior al 99.9%. Los datos en el dashboard deben tener una antigüedad máxima de 1 hora.
4. Adopción por el Usuario: El equipo del Centro de Supervisión y Maniobra (CSM) debe validar formalmente que la herramienta es intuitiva, fiable y que aporta un valor tangible a su conciencia situacional durante la operación.

## 6.2. Riesgos Potenciales y Plan de Mitigación

Se han identificado los siguientes riesgos técnicos para la Fase 1, junto con sus planes de mitigación:

Riesgo	Impacto	Plan de Mitigación
<b>Fiabilidad de la API de Precipitación</b>	Medio	La función AWS Lambda se implementará con una política de reintentos con exponential backoff. Se configurará una alarma en Amazon CloudWatch para notificar al equipo técnico en caso de fallos persistentes de la API externa.
<b>Precisión de los Umbrales Iniciales</b>	Bajo	Los umbrales de la Matriz de Riesgo se almacenarán como parámetros configurables. El sistema está diseñado para que estos puedan ser ajustados fácilmente a medida que se recopilen datos de eventos reales y se obtenga retroalimentación de campo.
<b>Calidad del Mapa de Amenaza (SGC)</b>	Bajo	El riesgo de una baja resolución en el mapa SGC se mitiga al incorporar la pendiente del terreno como una segunda variable de riesgo estático. Esto proporciona una capa adicional de granularidad y validación.

## 6.3. Visión a Futuro y Próximos Pasos (Fase 2)

La arquitectura de la Fase 1 está diseñada explícitamente para ser el cimiento de la Fase 2: Evolución a un Modelo Predictivo. El sistema inicial es determinístico: responde a la pregunta "¿Se cumplen las condiciones de riesgo?". El siguiente paso estratégico es evolucionar hacia un sistema que utilice machine learning para responder a una pregunta mucho más potente: "¿Cuál es la probabilidad de que ocurra un evento en esta ubicación?".

Esta transición convierte el SAT de un sistema de monitoreo a una herramienta de pronóstico inteligente.

#### **6.3.1. Hoja de Ruta Tecnológica Detallada:**

**Consolidación del Historial de Eventos:** El activo más valioso que se generará es el dataset de verdad terreno (ground truth dataset). La prioridad será alimentar la tabla eventos\_historicos en Amazon RDS con datos de deslizamientos reales (fecha, coordenada, magnitud). Este historial es indispensable, ya que es la única forma de validar objetivamente la precisión del modelo y, más importante, de entrenarlo para que reconozca patrones que llevaron a fallas reales en el pasado.

**Ingeniería de Características (Feature Engineering):** Para que el modelo "aprenda" mejor, necesitamos darle más contexto. Se integrarán nuevas fuentes de datos para crear características (features) que capturen la física del fenómeno de manera más directa que solo la lluvia:

**Humedad del Suelo Satelital:** Utilizando datos de misiones como SMAP de la NASA (disponibles en AWS Open Data), pasaremos de inferir la saturación del suelo a partir de la lluvia a tener una medida directa. Esto es un salto cualitativo en la precisión del modelo.

**Índice de Vegetación (NDVI):** Calculado a partir de imágenes de Sentinel-2. Una vegetación sana y densa (alto NDVI) suele estar correlacionada con una mayor estabilidad del suelo. Un cambio brusco en el NDVI podría ser un precursor de inestabilidad.

#### **6.3.2. Entrenamiento del Modelo de Machine Learning**

**Tecnología:** Se utilizará el ecosistema de Amazon SageMaker para gestionar todo el ciclo de vida del modelo (entrenamiento, evaluación y despliegue). Esto elimina la necesidad de gestionar infraestructura de cómputo para el entrenamiento.

**Algoritmo:** Se entrenará un modelo de clasificación basado en árboles de decisión potenciados por gradiente, como XGBoost o LightGBM. Estos algoritmos son el estándar de la industria para datos tabulares (nuestro caso de uso) porque son extremadamente eficientes y capaces de capturar interacciones complejas y no lineales entre las variables (por ejemplo, cómo 100mm de lluvia afectan de manera diferente a una pendiente de 20° vs una de 40°).



### **6.3.3. Despliegue del Modelo como un Microservicio**

El modelo entrenado se desplegará como un endpoint en Amazon SageMaker, que es esencialmente una API privada y escalable para hacer predicciones. La arquitectura del sistema evolucionará de la siguiente manera:

La función AWS Lambda será modificada. En lugar de ejecutar la lógica de la Matriz de Riesgo, su nueva tarea será:

1. Recopilar todas las características en tiempo real (lluvia, humedad del suelo, etc.).
2. Enviar estos datos como una solicitud al endpoint de SageMaker.
3. Recibir una respuesta con la probabilidad de deslizamiento (ej. {"probabilidad": 0.85}).

Este puntaje de riesgo probabilístico (0-100%) es mucho más granular y accionable para el CSM que una simple alerta roja, permitiendo una priorización mucho más fina de los recursos.