UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

TALLER DE BASE DE DATOS DIURNO 2-2025

Enunciado 1

Ayudante: Pablo Macuada **Fernando Solis**	**Profesor:** Matías Calderón

Enunciado Laboratorio 1

Entrega: 20 octubre de 2025

Proyecto:

Visualizador de Datos de Cambio Climático

Objetivo: Desarrollar una plataforma que permita a usuarios y científicos visualizar en un mapa interactivo una variedad de datos relacionados con el cambio climático. La aplicación debe ser capaz de filtrar, superponer y analizar tendencias de datos georreferenciados para facilitar la comprensión y el estudio de los fenómenos climáticos.

Tecnologías y Herramientas Requeridas

- **Base de Datos:** PostgreSQL
- **Backend:** Spring Boot con Java
- **Frontend:** Vue.js 3
- **Comunicación:** Axios para las llamadas HTTP
- **Seguridad:** JSON Web Tokens (JWT) para la autenticación y autorización
- **Control de Versiones:** Git y un repositorio en GitHub

Requisitos Específicos

1. Requisitos de la Base de Datos (PostgreSQL)

Deberás diseñar un **esquema de base de datos** normalizado que incluya, al menos, las siguientes tablas:

- **usuarios**: Información de los usuarios del sistema (nombre, email, contraseña hasheada).
- **datasets**: Metadatos de los conjuntos de datos disponibles (nombre, descripción, fuente, fecha de actualización).
- **puntos_medicion**: Ubicaciones geográficas de los puntos de medición (latitud/longitud) con metadatos como tipo de sensor.
- **mediciones**: Datos de medición brutos (valor de la medición, fecha y hora, ID del punto de medición, ID del conjunto de datos).

No se permite el uso de JPA/Hibernate. La comunicación entre la aplicación y la base de datos debe ser exclusivamente a través de **sentencias SQL nativas**.

Deberás implementar los siguientes elementos en la base de datos:

- **Triggers:** Para automatizar procesos como la validación de la inserción de nuevos datos o la actualización de metadatos.
- **Procedimientos Almacenados:** Para encapsular la lógica de negocio compleja, como el cálculo de anomalías o la interpolación de datos.
- **Vistas Materializadas:** Para pre-calcular resúmenes y tendencias de datos que se consultan con frecuencia, mejorando el rendimiento de la aplicación.
- **Índices:** Para optimizar las consultas más comunes, especialmente las búsquedas por ubicación y fecha.

2. Requisitos del Backend (Spring Boot)

- Crear una **API RESTful** que exponga los endpoints necesarios para la gestión de usuarios, datasets y mediciones.
- Implementar un sistema de autenticación de usuarios utilizando **JWT**.
- Desarrollar un endpoint de **login** que devuelva un token válido para las subsiguientes peticiones.
- Proteger las rutas de la API, permitiendo el acceso solo a usuarios autenticados.

- Conectar a la base de datos PostgreSQL y ejecutar consultas utilizando sentencias SQL puras.

3. Requisitos del Frontend (Vue.js)

- Crear una **interfaz de usuario** intuitiva y atractiva.
- Implementar una página de **login** para que los usuarios puedan autenticarse.
- Una vez autenticados, los usuarios deben poder ver un **mapa interactivo** que muestre los puntos de medición y las visualizaciones de datos. (Para esta entrega solo es necesario tener en cuenta en el diseño el espacio adecuado para el mapa)
- La interfaz debe permitir a los usuarios seleccionar y superponer diferentes datasets (p. ej., temperatura, nivel del mar, concentración de CO2).
- Utilizar **Axios** para todas las peticiones HTTP al backend.

Funcionalidades Clave del Sistema

- **Login de Usuario:** Autenticación de usuarios a través de un token JWT.
- **Visualización de Datos:** Los usuarios pueden seleccionar y visualizar diferentes conjuntos de datos climáticos en el mapa.
- **Filtros Dinámicos:** La interfaz debe permitir filtrar los datos por fecha, tipo de medición y región geográfica.
- **Análisis de Tendencias:** Los usuarios pueden seleccionar una región o punto en el mapa y ver gráficos de series temporales de las mediciones. (En esta entrega solo es necesario planificar con la idea del posterior uso del mapa)
- **Reportes de Datos:** Los administradores deben poder generar reportes y estadísticas sobre la información disponible.

Documentación y Entrega

Todo el proyecto debe ser subido a un **repositorio de GitHub**. La entrega debe incluir:

- **Documentación de la Base de Datos:** Un documento que describa el esquema, las relaciones entre tablas y el propósito de cada trigger, procedimiento almacenado, vista materializada e índice.
- **Script de Creación y Carga de Datos:** Un archivo .sql que permita recrear la base de datos completa, incluyendo las tablas, índices, triggers y un conjunto de datos de prueba para realizar las pruebas.
- **Código Fuente:** El código completo del backend (Spring Boot) y el frontend (Vue.js), subido al repositorio de GitHub.
- **README.md:** Un archivo README.md en el repositorio principal que contenga instrucciones claras sobre cómo clonar, configurar y ejecutar la aplicación.

Consultas SQL a desarrollar

1. **Cálculo de Anomalía de Temperatura:** Escribe una consulta SQL que, para cada punto de medición, calcule la temperatura promedio del último año y la compare con el promedio histórico de ese mismo punto. Muestra el ID del punto de medición y la diferencia (`anomalía`) en grados.
2. **Identificación de Puntos con Mayor Variación:** Utilizando funciones de ventana, identifica los 10 puntos de medición con la mayor desviación estándar en los valores de temperatura de los últimos 5 años. Muestra el ID del punto y el valor de la desviación estándar.
3. **Análisis de Correlación Espacial:** Crea una consulta que encuentre todos los puntos de medición de CO2 que están a menos de 50 km de un punto de medición de temperatura específico (dado por sus coordenadas). La consulta debe devolver el ID de ambos puntos y la distancia entre ellos.
4. **Detección de Eventos Extremos:** Escribe una consulta que identifique todos los días en el último año donde la temperatura máxima registrada en cualquier punto de medición superó un umbral de 35°C. Muestra la fecha y el valor máximo de temperatura de ese día.

5. **Simulación de Interpolación de Datos:** Escribe un procedimiento almacenado llamado `interpolar_datos_semanales` que reciba un ID de dataset. El procedimiento debe calcular el promedio semanal de las mediciones y almacenarlo en una tabla de resumen, llenando los días sin datos con el promedio semanal.
6. **Agregación de Datos para Visualización:** Implementa un procedimiento almacenado que, dado un ID de dataset y un rango de fechas, devuelva una serie temporal agregada de las mediciones. Los datos deben agruparse por semana o por mes, dependiendo de la duración del rango de fechas.
7. **Listado de Medidas sin Georreferenciación:** Crea una consulta que muestre todos los puntos de medición que no tienen una ubicación geográfica válida (por ejemplo, `NULL` o `(0,0)`). La consulta debe incluir el ID del punto y la fecha de su última medición.
8. **Análisis de Tendencia Histórica:** Crea una vista materializada llamada `tendencia_mensual` que muestre el valor promedio de cada tipo de medición por mes, desde el inicio del registro de datos. Esta vista debe ser refrescada de forma concurrente para asegurar que los informes de tendencia histórica se carguen rápidamente.