**Documentación Técnica: Modelo y Esquema de Base de Datos - Urban Flow**

**Autor:** Nicolás Páez Lancheros

**Fecha:** 07 de septiembre de 2025

**Versión:** 1.0 **Proyecto:** Plataforma de Monitoreo para Urban Flow

**1. Introducción**

Este documento detalla el diseño conceptual (modelo) y la implementación física (esquema) de la base de datos para el proyecto Urban Flow. La arquitectura de la base de datos se ha desarrollado en **PostgreSQL** para soportar de manera eficiente los dos pilares funcionales de la plataforma:

1. La gestión de usuarios y control de acceso a la aplicación.
2. La ingesta y almacenamiento masivo de datos operativos provenientes de los sensores IoT instalados en las cabinas del sistema de metrocable.

El diseño presentado a continuación es el resultado del análisis de los requerimientos técnicos y funcionales descritos en el documento del proyecto.

**2. Modelo de Entidad-Relación (ERD)**

El siguiente diagrama representa el modelo conceptual de la base de datos. Muestra las entidades principales (tablas), sus atributos clave (columnas) y las relaciones lógicas que las conectan.

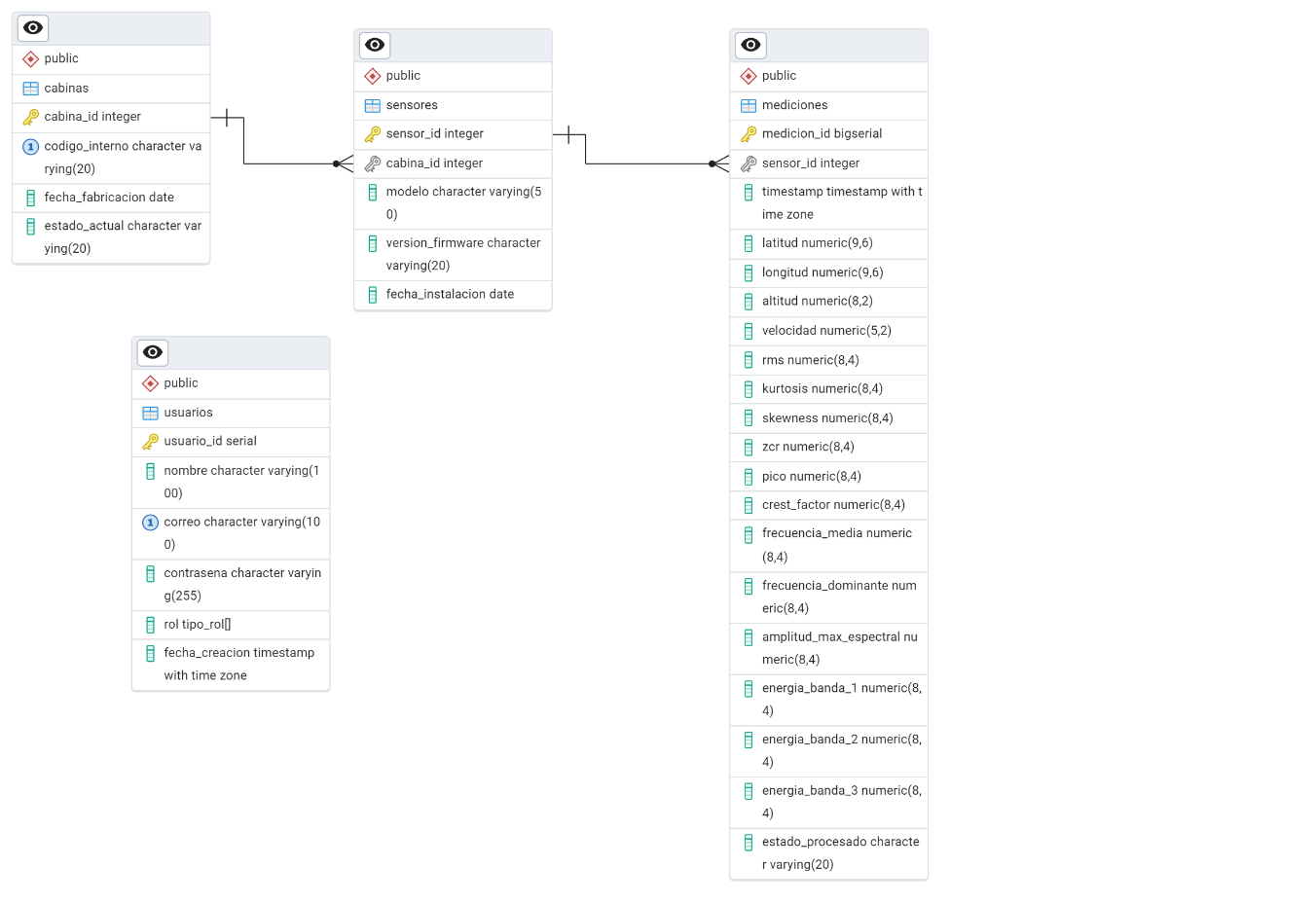
**Diagrama

El contenido generado por IA puede ser incorrecto.**

**2.1. Descripción de Entidades y Relaciones**

El modelo se compone de cuatro entidades principales:

* **Usuarios**: Entidad de aplicación que almacena la información de los usuarios registrados. Es una tabla independiente, ya que su propósito es gestionar la autenticación y los roles de acceso a la plataforma, sin tener relación directa con la operación física del sistema.
* **Cabinas**: Tabla maestra que funciona como inventario de todas las cabinas que operan en el sistema. Contiene información estática y de estado de cada vehículo.
* **Sensores**: Almacena los datos de cada sensor físico. Se ha diseñado con una **relación uno a uno** con la tabla cabinas mediante una llave foránea (cabina\_id) con restricción UNIQUE. Esto garantiza que a cada cabina le corresponde un único sensor.
* **Mediciones**: Es la tabla transaccional principal, diseñada para almacenar el gran volumen de datos que cada sensor transmite por segundo. Mantiene una **relación uno a muchos** con la tabla sensores, ya que un sensor generará un flujo constante y masivo de mediciones a lo largo del tiempo.



**3. Esquema Físico (Script SQL para PostgreSQL)**

El siguiente script contiene el código Data Definition Language (DDL) para la creación de toda la estructura de la base de datos en PostgreSQL. El script incluye la creación de tipos personalizados, tablas, restricciones de integridad (llaves primarias, foráneas, unicidad) y la configuración de índices para optimizar el rendimiento de las consultas.

*-- PASO 1: Creación de Tipos de Datos Personalizados (ENUMS)*

*-- Se crea un tipo ENUM para la columna 'rol' en la tabla 'usuarios'.*

*-- Esto garantiza que solo los valores 'administrador' y 'usuario' sean aceptados,*

*-- mejorando la integridad de los datos.*

CREATE TYPE tipo\_rol AS ENUM ('administrador', 'usuario');

*-- =============================================================================*

*-- PASO 2: Creación de las Tablas*

*-- Tabla: usuarios*

*-- Propósito: Almacena la información de los usuarios registrados en la plataforma.*

*-- Su función es gestionar el acceso (autenticación y autorización).*

CREATE TABLE usuarios (

    usuario\_id SERIAL PRIMARY KEY, *-- ID único autoincremental para cada usuario.*

    nombre VARCHAR(100) NOT NULL, *-- Nombre del usuario.*

    correo VARCHAR(100) UNIQUE NOT NULL, *-- Correo electrónico para el login (debe ser único).*

    contrasena VARCHAR(255) NOT NULL, *-- Se almacena el hash de la contraseña, no el texto plano.*

    rol tipo\_rol DEFAULT 'usuario' NOT NULL, *-- Rol del usuario, con 'usuario' como valor por defecto.*

    fecha\_creacion TIMESTAMP WITH TIME ZONE DEFAULT CURRENT\_TIMESTAMP *-- Fecha y hora de registro.*

);

*-- Tabla: cabinas*

*-- Propósito: Tabla maestra que funciona como inventario de todas las cabinas del sistema.*

CREATE TABLE cabinas (

    cabina\_id INT PRIMARY KEY, *-- ID único de la cabina, proveído por el sistema.*

    codigo\_interno VARCHAR(20) UNIQUE NOT NULL, *-- Código de operación único.*

    fecha\_fabricacion DATE, *-- Fecha de fabricación de la cabina.*

    estado\_actual VARCHAR(20) NOT NULL CHECK (estado\_actual IN ('operativo', 'inusual', 'alerta', 'fuera de servicio')) *-- Estado actual de la cabina.*

);

*-- Tabla: sensores*

*-- Propósito: Almacena la información de cada sensor IoT y lo asocia a una única cabina.*

CREATE TABLE sensores (

    sensor\_id INT PRIMARY KEY, *-- ID único del sensor.*

    cabina\_id INT UNIQUE NOT NULL, *-- Llave foránea a 'cabinas'. Se usa UNIQUE para forzar la relación 1 a 1.*

    modelo VARCHAR(50), *-- Modelo del sensor.*

    version\_firmware VARCHAR(20), *-- Versión del firmware instalado en el sensor.*

    fecha\_instalacion DATE, *-- Fecha en que el sensor fue instalado en la cabina.*

    CONSTRAINT fk\_cabina

        FOREIGN KEY(cabina\_id)

        REFERENCES cabinas(cabina\_id)

        ON DELETE CASCADE *-- Si se elimina una cabina, su sensor asociado también se elimina.*

);

*-- Tabla: mediciones*

*-- Propósito: Tabla transaccional que almacena cada lectura de datos enviada por los sensores.*

*-- Se espera que esta tabla crezca masivamente.*

CREATE TABLE mediciones (

    medicion\_id BIGSERIAL PRIMARY KEY, *-- ID único auto incremental (BIGSERIAL para soportar un gran volumen de datos).*

    sensor\_id INT NOT NULL, *-- Llave foránea que indica qué sensor envió la medición.*

    "timestamp" TIMESTAMP WITH TIME ZONE NOT NULL, *-- Fecha y hora exacta de la medición.*

    latitud DECIMAL(9,6),

    longitud DECIMAL(9,6),

    altitud DECIMAL(8,2),

    velocidad DECIMAL(5,2),

    rms DECIMAL(8,4),

    kurtosis DECIMAL(8,4),

    skewness DECIMAL(8,4),

    zcr DECIMAL(8,4),

    pico DECIMAL(8,4),

    crest\_factor DECIMAL(8,4),

    frecuencia\_media DECIMAL(8,4),

    frecuencia\_dominante DECIMAL(8,4),

    amplitud\_max\_espectral DECIMAL(8,4),

    energia\_banda\_1 DECIMAL(8,4),

    energia\_banda\_2 DECIMAL(8,4),

    energia\_banda\_3 DECIMAL(8,4),

    estado\_procesado VARCHAR(20) CHECK (estado\_procesado IN ('operativo', 'inusual', 'alerta')), *-- Estado resultante del análisis de IA.*

    CONSTRAINT fk\_sensor

        FOREIGN KEY(sensor\_id)

        REFERENCES sensores(sensor\_id)

        ON DELETE CASCADE *-- Si se elimina un sensor, todo su historial de mediciones se borra.*

);

**4. Conclusión**

El modelo y esquema de base de datos presentados cumplen con todos los requisitos funcionales y técnicos definidos para el primer sprint del proyecto. La estructura es robusta, relacionalmente íntegra y está optimizada para la escalabilidad, sentando una base sólida para el desarrollo de los siguientes módulos de la aplicación.