



UNIVERSIDAD NACIONAL EXPERIMENTAL  
PARA LAS TELECOMUNICACIONES E INFORMÁTICA (UNETI)  
VICERRECTORADO ACADÉMICO  
PROGRAMA NACIONAL DE FORMACIÓN EN INFORMÁTICA  
U.C: Base de datos I (Informática) Sección 7A

**Caso de Estudio 12: Sistema de Gestión de Mantenimiento de  
Infraestructura para un Ayuntamiento "Ciudad Limpia"**

*(Evaluación 3: Diseño Lógico y Físico de la Base de Datos)*

**TUTOR:**

Yuly Delgado

**AUTOR**

Jostin Perdigón

Cl. V-31.412.625

**Caracas, Marzo de 2025**

## INTRODUCCIÓN

Este informe describe el diseño lógico y físico de la base de datos para el "Sistema de Gestión de Mantenimiento de Infraestructura para un Ayuntamiento 'Ciudad Limpia'". Se detalla la estructura de la base de datos, las relaciones entre las tablas, las optimizaciones para mejorar el rendimiento y las estrategias de respaldo y recuperación.

## Fase 1: Diseño de Tablas y Relaciones

### Modelo Entidad-Relación

Se identificaron las siguientes entidades principales:

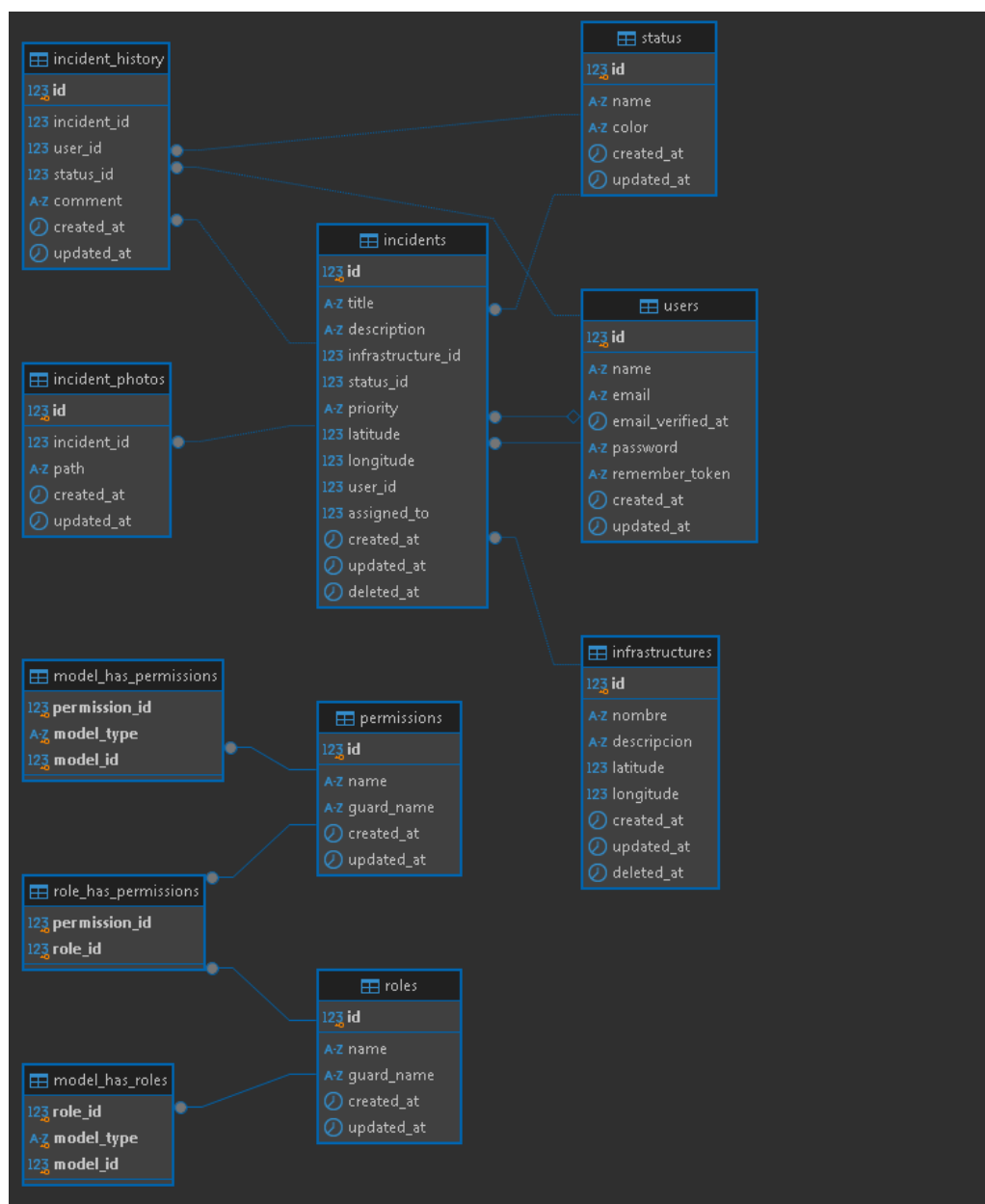
1. **Infraestructura** (ID, tipo, ubicación, descripción, estado, última revisión, historial de mantenimiento)
2. **Incidencia** (ID, tipo, ubicación, descripción, fecha, estado, prioridad, técnico asignado)
3. **OrdenTrabajo** (ID, incidencia asociada, descripción, recursos, fecha inicio, fecha fin, estado)
4. **Personal** (ID, nombre, especialidad, disponibilidad)
5. **Materiales** (ID, nombre, descripción, cantidad, costo)
6. **Presupuesto** (ID, total asignado, gasto actual, gasto por infraestructura)

Relaciones clave:

- Una infraestructura puede tener múltiples incidencias!
- Una incidencia genera una orden de trabajo.
- Una orden de trabajo asigna materiales y personal.

## Diagrama Entidad-Relación (DER)

A continuación se incluirá un diagrama para representar visualmente las entidades y sus relaciones en la base de datos basado en las estructuras extraídas del archivo SQL.



## **REQUISITOS FUNCIONALES:**

Se presentan los requisitos funcionales relevantes del caso::

### **1. Gestión de Infraestructura:**

#### **Atributos**

El sistema debe permitir la gestión de los atributos:

- Tipo de Infraestructura: el usuario podrá seleccionar el tipo de infraestructura tales como: calle, parque, farola alcantarilla, edificio.
- Ubicación: se detalla la información de la ubicación de la incidencia otorgando la dirección o por las coordenadas.
- Descripción: se especifica los datos generales del reporte de la incidencia.
- Estado: se identifica el estado de la incidencia este tendrá 3 opciones: bueno, regular o malo.
- Fecha: se detalla la fecha de la última revisión.
- Historial de mantenimiento.

#### **Funcionalidades:**

El sistema debe permitir la gestión de las siguientes funcionalidades:

- Registro y geo localización de la infraestructura.
- Clasificación de la infraestructura.
- Consulta del historial de mantenimiento de cada elemento.

## **2. Gestión de Incidencias:**

### **Atributos**

El sistema debe permitir la gestión de las incidencias detallando:

- Tipo de incidencia: detalla la información de la avería mediante el reporte: avería, desperfecto, limpieza.
- Ubicación: especifica la dirección de la incidencia o avería.
- Descripción: relata específicamente la incidencia.
- Fecha y Hora de la incidencia: detalla más datos de la novedad.
- Estado: especifica el estado de la incidencia: pendiente, en curso, resuelta.
- Prioridad
- Técnico asignado

### **Funcionalidades:**

- Registro de incidencias: reporte por parte de los ciudadanos a través de App móvil o web, y del personal de la institución.
- Asignación de incidencias: asignación de funciones al personal de mantenimiento.
- Seguimiento del estado de la incidencia
- Notificaciones y alertas

## **3. Gestión de Órdenes de Trabajo**

El sistema debe permitir la gestión de las órdenes de trabajo

**Atributos:**

- Incidencias asociadas
- Descripción del trabajo a realizar
- Recursos necesarios: describe los materiales y el personal requerido.
- Fecha de inicio
- Fecha de culminación
- Estado: especifica el estado del reporte: pendiente, en curso, completada.

**Funcionalidades**

- Generación automática de órdenes de trabajo a partir de las incidencias.
- Asignación de recursos.
- Seguimiento del progreso de las órdenes de trabajo.

**4. Gestión de Recursos ( personal y materiales)**

El sistema debe permitir el registro y control de los recursos estos incluyen: el personal y los materiales.

**Atributos Personal**

- Nombre
- Apellido
- Especialidad

- Disponibilidad.

### **Atributos Materiales**

- Nombre
- Descripción
- Cantidad
- Disponible
- Costo

### **Funcionalidades**

- Gestión de inventario de materiales
- Asignación de recursos a las órdenes de trabajo

## **5. Gestión de Mantenimiento Preventivo**

El sistema debe gestionar las siguientes funcionalidades:

- Planificación de las tareas de mantenimiento preventivo según un calendario.
- Generación automática de órdenes de trabajo para el mantenimiento preventivo.

## **6. Gestión de Presupuesto**

El sistema debe gestionar la gestión del mantenimiento

## **Funcionalidades**

- Seguimiento del presupuesto destinado al mantenimiento
- Control de gastos por tipo de infraestructura y zona
- Generación de informes presupuestarios

## **7. Informes y estadísticas**

El sistema debe gestionar los siguientes informes o reportes:

- Informes de incidencias por: Tipo, zona y prioridad
- Informes de rendimiento de los equipos de mantenimiento
- Informes de costos de mantenimiento
- Informes de estado de la infraestructura

## **8. Funcionalidades Avanzadas**

El sistema debe gestionar las siguientes funcionalidades:

- Integración con sistemas de geolocalización GIS: para la visualización de la infraestructura y las incidencias en un mapa.
- App móvil para los ciudadanos: para reportar incidencias y consultar el estado de las mismas
- Optimización de rutas para los equipos de mantenimiento: para mejorar la eficiencia de los desplazamientos
- Análisis predictivo del estado de la infraestructura: para anticipar las posibles averías y planificar el mantenimiento preventivo de forma más efectiva.



## Requisitos No Funcionales

- **Usabilidad:** Interfaz intuitiva y fácil de usar para el personal del ayuntamiento y para los ciudadanos.
- **Rendimiento:** El sistema debe ser rápido y eficiente, incluso con gran volumen de datos e incidencias.
- **Seguridad:** Protección de la información y acceso restringido a usuarios no autorizados
- **Escalabilidad:** debe poder adaptarse al crecimiento de la ciudad y al aumento de la infraestructura.
- **Disponibilidad:** debe estar disponible la mayor parte del tiempo, especialmente para la gestión de emergencias.

## Fase 2: Inserción, Consulta y Eliminación de Datos

Se definieron procedimientos para:

- Registrar nuevas incidencias.
- Asignar incidencias a técnicos.
- Consultar el estado de las incidencias.
- Generar informes de mantenimiento.

Ejemplo de consulta SQL extraída del archivo:

```
1 INSERT INTO public.incidents (id, title, description, infrastructure_id, status_id, priority, latitude,  
2 updated_at, deleted_at)  
3 VALUES  
4 (1, 'Daño en iluminación', 'Farolas sin funcionar en la zona norte', 1, 1, 'Alta',  
5 4.65800000, -74.09350000, 3, 2, '2025-02-21 21:10:08', '2025-02-21 21:10:08', NULL);
```

## Fase 3: Diseño Lógico y Físico de la Base de Datos

### Tarea 3.1: Diseño Lógico

Se aplicó normalización hasta la tercera forma normal (3NF) para evitar las redundancias y mejorar la integridad de los datos.

Se consideró la escalabilidad del sistema permitiendo la extensión de nuevos tipos de “infraestructura” y roles de usuario

### Diagrama Lógico

Se incluirá un diagrama detallado que represente las tablas, atributos y claves foráneas en la base de datos basado en la estructura del archivo SQL. A continuación se detalla en la siguiente imagen:

**NOTA:** Se Utilizó DBeaver para el diagrama entidad relación y mejor vistas, se desarrolló tablas no relacionadas, sin embargo en esta caso se colocaron las más importantes del caso.

### Tarea 3.2: Diseño Físico

- **Tipos de almacenamiento:** Se usó PostgreSQL con JSONB para ciertos campos (por ejemplo el historial de mantenimiento) y particionamiento en tablas de incidencias por año.

## Índices:

1. Indexado en ID y claves foráneas para mejorar los “joins”.
2. Índice en la columna "estado" de incidencias para consultas rápidas.

Aquí un ejemplo de índice extraído del archivo SQL:

```
CREATE INDEX jobs_queue_index ON public.jobs USING btree (queue);
```

- **Particionamiento:**

1. Particionamiento por fecha en la tabla de incidencias.
2. Uso de tablas particionadas por tipo de infraestructura.

## Escalabilidad:

Al crear un monitoreo que pueda crecer y adaptarse a medida que se expande la infraestructura (capacidad de manejar un mayor volumen de datos, más usuarios y sistemas adicionales sin degradar el rendimiento) La escalabilidad va a asegurar que la solución siga siendo efectiva a medida que cambian las necesidades.

Para el registro y control de "Ciudad Limpia", esto implica:

**Escalabilidad Vertical:** Aumentar la capacidad de los servidores existentes (más CPU, RAM, almacenamiento) para manejar una mayor carga. Esto es útil en las primeras etapas, pero tiene límites físicos.

**Escalabilidad Horizontal:** Añadir más servidores o nodos al sistema para distribuir la carga. Esto es más flexible y adecuado para un crecimiento a largo plazo.

### **Rendimiento a Largo Plazo**

El rendimiento a largo plazo se refiere a la capacidad del sistema para mantener un alto nivel de eficiencia y disponibilidad a medida que crece y evoluciona. Para el sistema "Ciudad Limpia", esto implica:

- Optimizar las consultas a la base de datos para reducir los tiempos de respuesta.
- Implementar índices adecuados y técnicas de caching (Redis, Memcached) para acelerar el acceso a datos frecuentemente solicitados.
- Realizar mantenimiento regular del sistema, incluyendo actualizaciones de software, parches de seguridad y optimizaciones de rendimiento.
- Planificar actualizaciones sin tiempo de inactividad (zero-downtime deployments) para garantizar la disponibilidad continua.

### **Tarea 3.3: Tuneado de Consultas**

- Se utilizó el "EXPLAIN ANALYZE" para evaluar planes de ejecución.
- Aplicación de consultas optimizadas con "COVERING INDEXES".
- Reducción de JOINS innecesarios con subconsultas eficientes.

## Ejemplos de Consultas SQL

Se incluirán ejemplos de consultas optimizadas para inserciones, actualizaciones y búsquedas eficientes en la base de datos.

### Tarea 3.4: Backup y Recuperación

- Estrategia de backup incremental diario y full semanal.
- Uso de **pg\_basebackup** para copias de seguridad automáticas (en este por ejemplo se extrajo el archivo en .SQL formato PLAIN para ser más legible)
- Recuperación con PITR (Point-In-Time Recovery) en caso de fallo.

Importante: Siempre que se intenté recuperar la base de datos se le hace respaldo desde Pgadmin4

## CASOS DE USO

### Descripción de Casos de Uso

Cada caso de uso detalla:

- El actor involucrado.
- Flujo normal de eventos.
- Alternativas y excepciones.

Aquí un ejemplo de cómo se desarrolló:

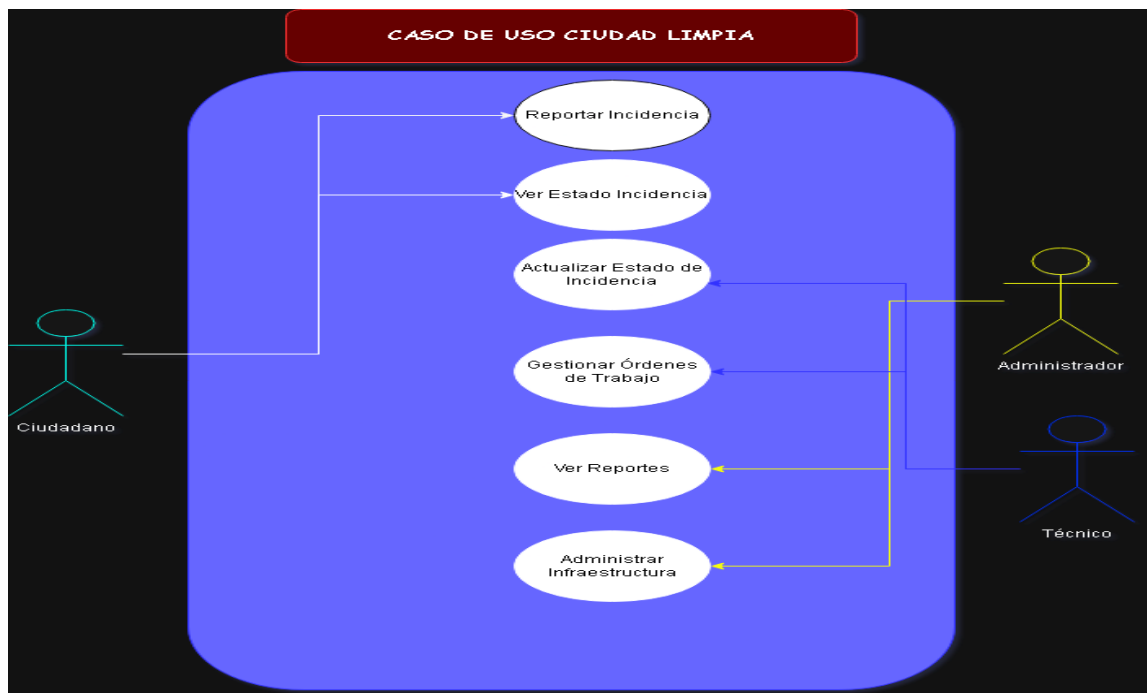
- **Caso de Uso: Reportar Incidencia**

**Actor:** Ciudadano

**Flujo:**

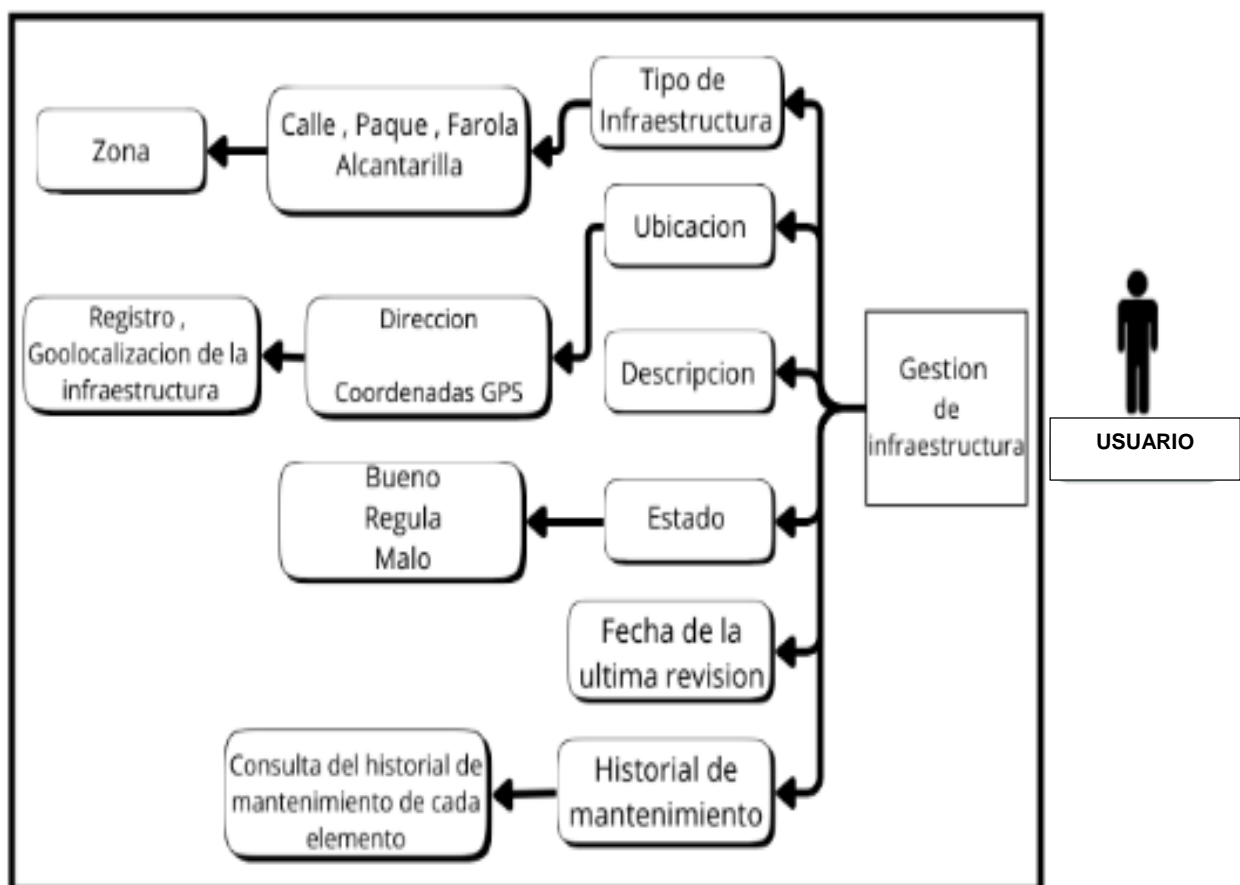
1. El usuario accede a la plataforma.
2. Ingresa los detalles de la incidencia.
3. Envía el formulario y recibe confirmación.

Esto para que el usuario (ciudadano) envíe los detalles de la problemática (incidencia) y el administrador y técnico puedan resolverlo.



## Caso de Uso Gestión de Infraestructura

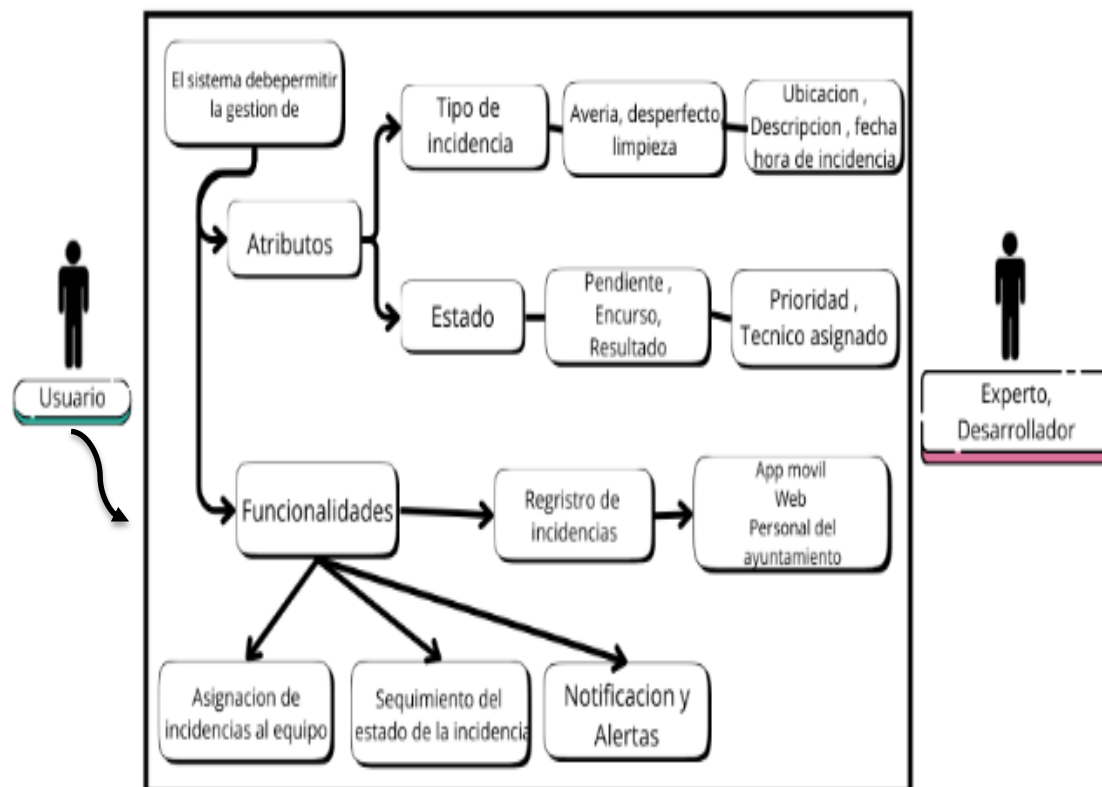
El usuario al acceder al sistema podrá gestionar los atributos según el tipo de infraestructura, ubicación según las coordenadas GPS o dirección, a su vez describirá el reporte de la problemática (incidencia), el estado, fecha e historial.



## Caso de uso gestión de incidencias

El experto proporcionará acceso al usuario mediante la gestión del sistema, que permitirá reportar la incidencia según el tipo, visualizar el estado, la prioridad del reporte y el técnico asignado.

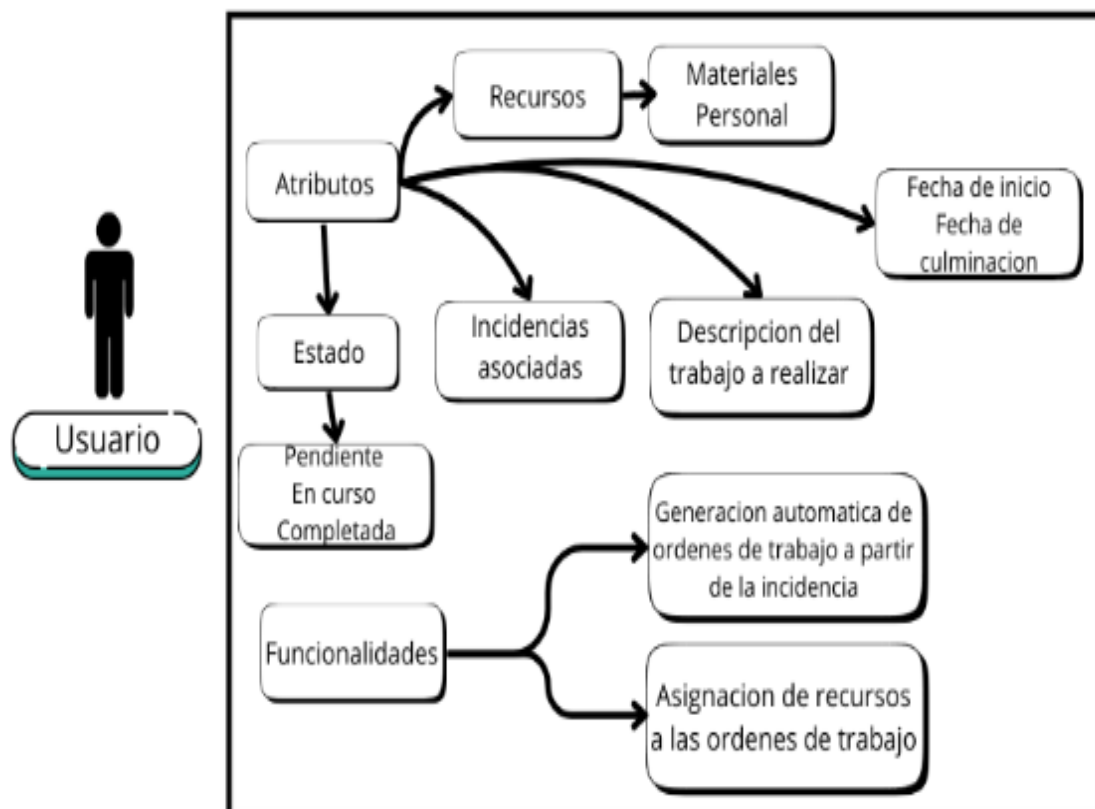
También podrá acceder a las funcionalidades donde reportara el caso por los medios de APP móvil, web o directamente al personal del ayuntamiento.





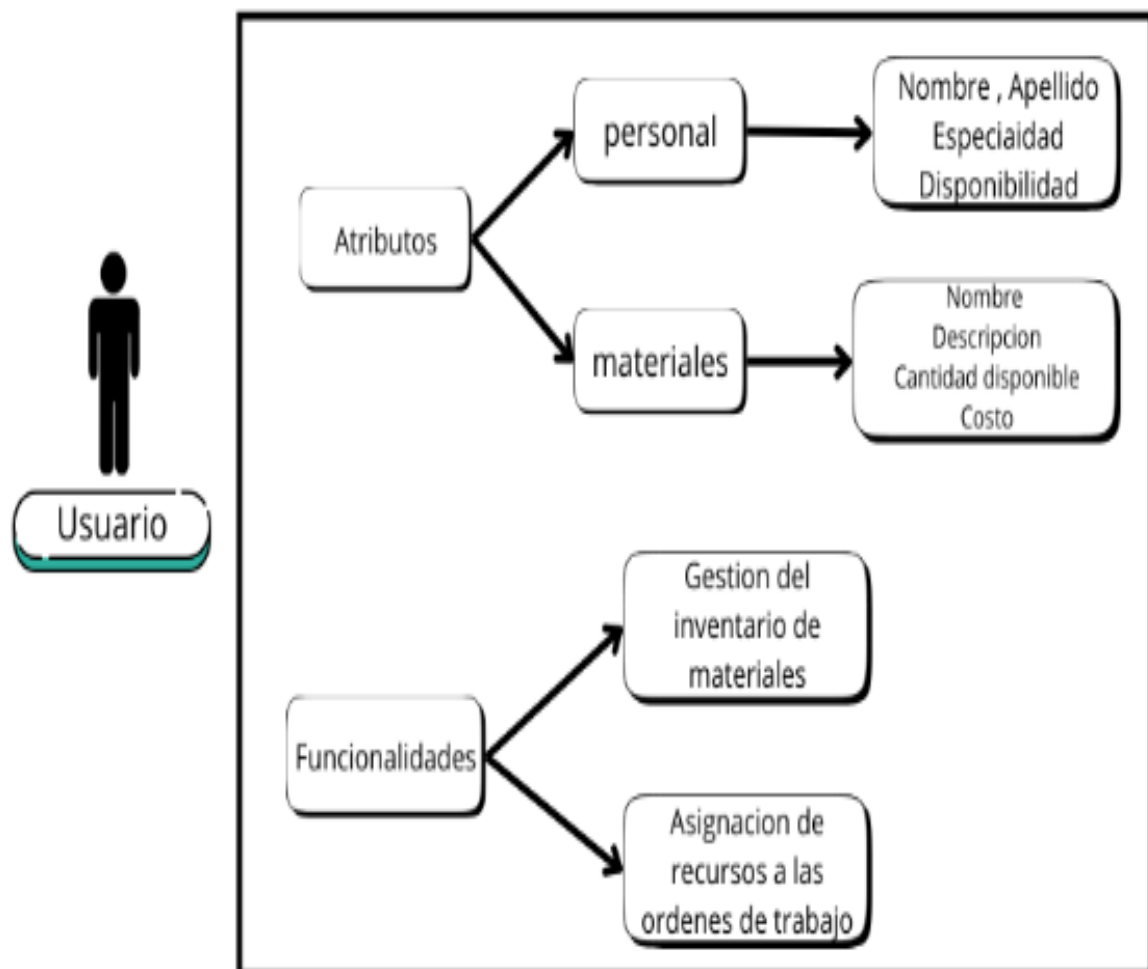
## Caso de uso ordenes de trabajo

El sistema le permitirá al usuario acceder a los atributos disponibles donde encontrara los recursos, las incidencias asociadas y el estado de las mismas.



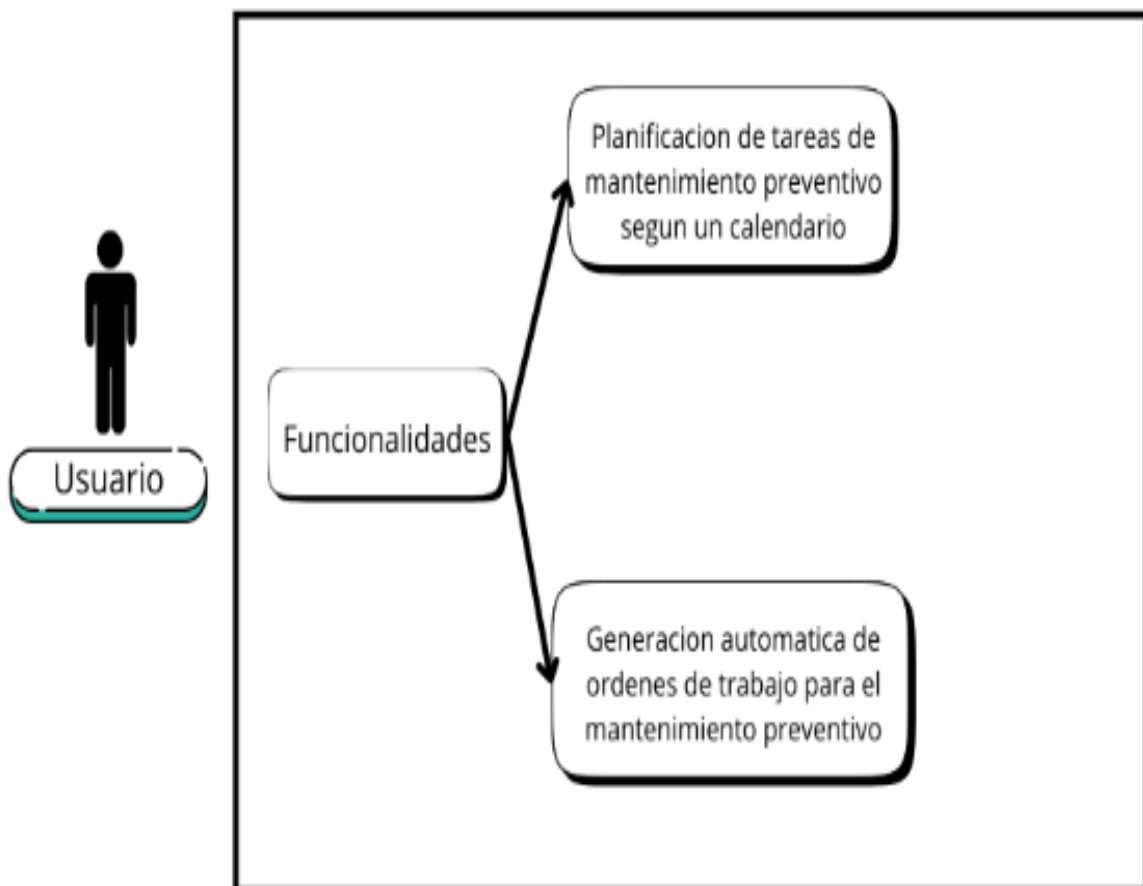
### Caso de uso gestión de recursos (personal y de materiales)

En esta sección el sistema le permite al usuario acceder a los atributos que se refieren al personal y recursos materiales disponibles, costo, especialidad, cantidad y descripción a fin de la asignación del personal y de recursos según las ordenes de trabajo.



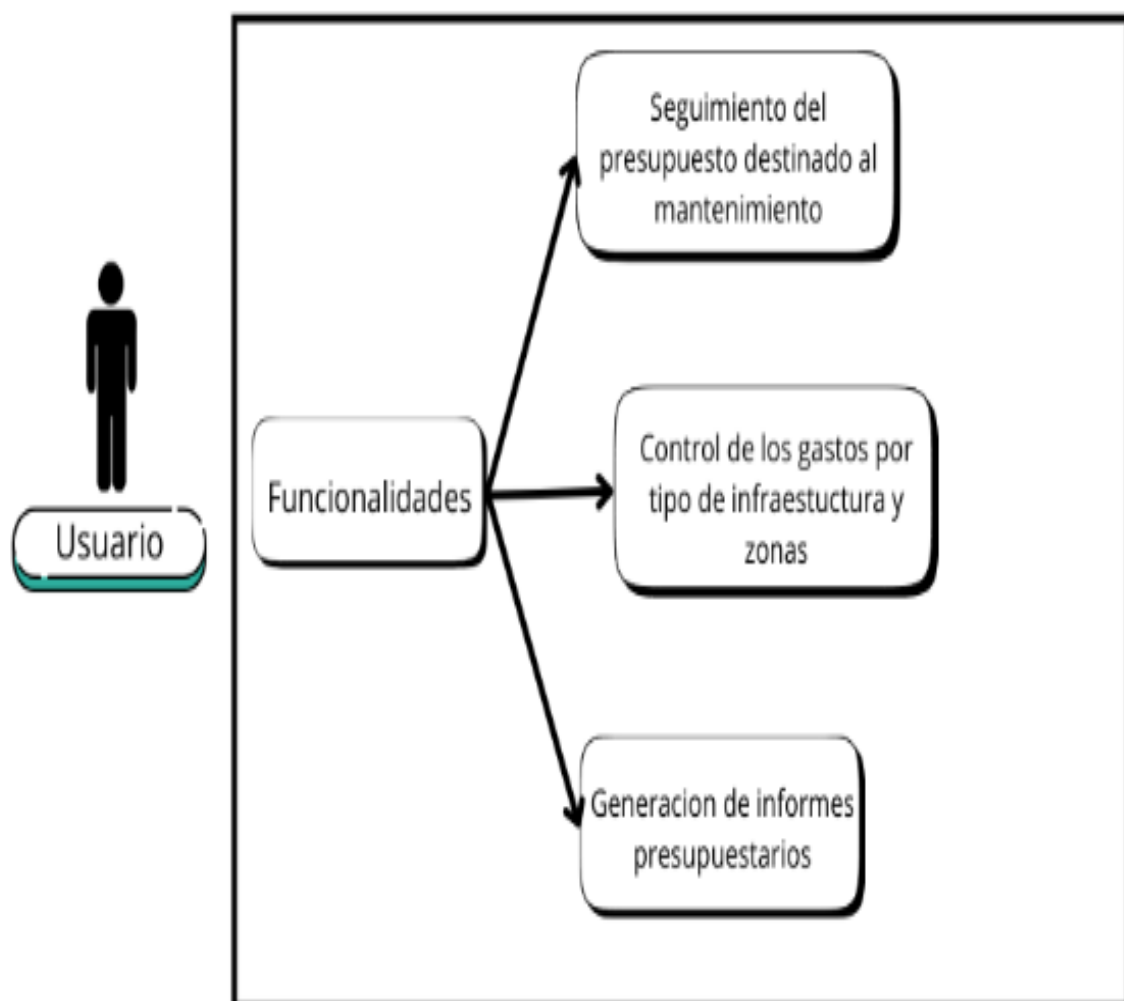
### Caso de uso gestión de mantenimiento preventivo

El sistema permite al sistema la planificación de tareas de mantenimiento preventivo según el calendario, generar las órdenes de trabajo.



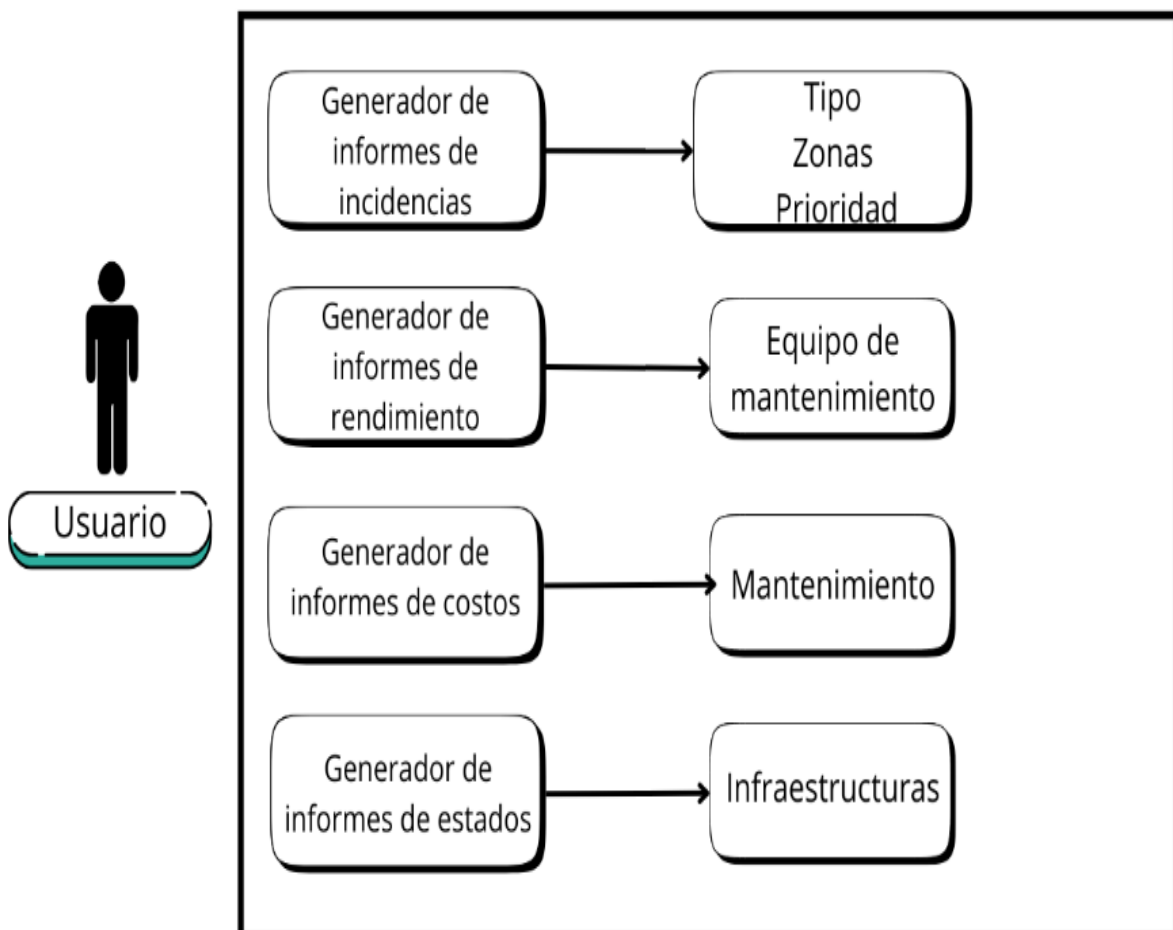
## Caso de uso gestión de presupuesto

El sistema permite la gestión de seguimiento de presupuesto destinado al mantenimiento, a su vez permite el control de gastos según su tipo de infraestructura y zona, generando informes presupuestarios.



## Caso de uso informes y estadísticas

El sistema permitirá al usuario generar informes de incidencias por: tipo, zonas y prioridad, informes de rendimiento de los equipos de mantenimiento, informes de costos de mantenimiento e informes de estado de la infraestructura.



## Seguridad y Privilegios

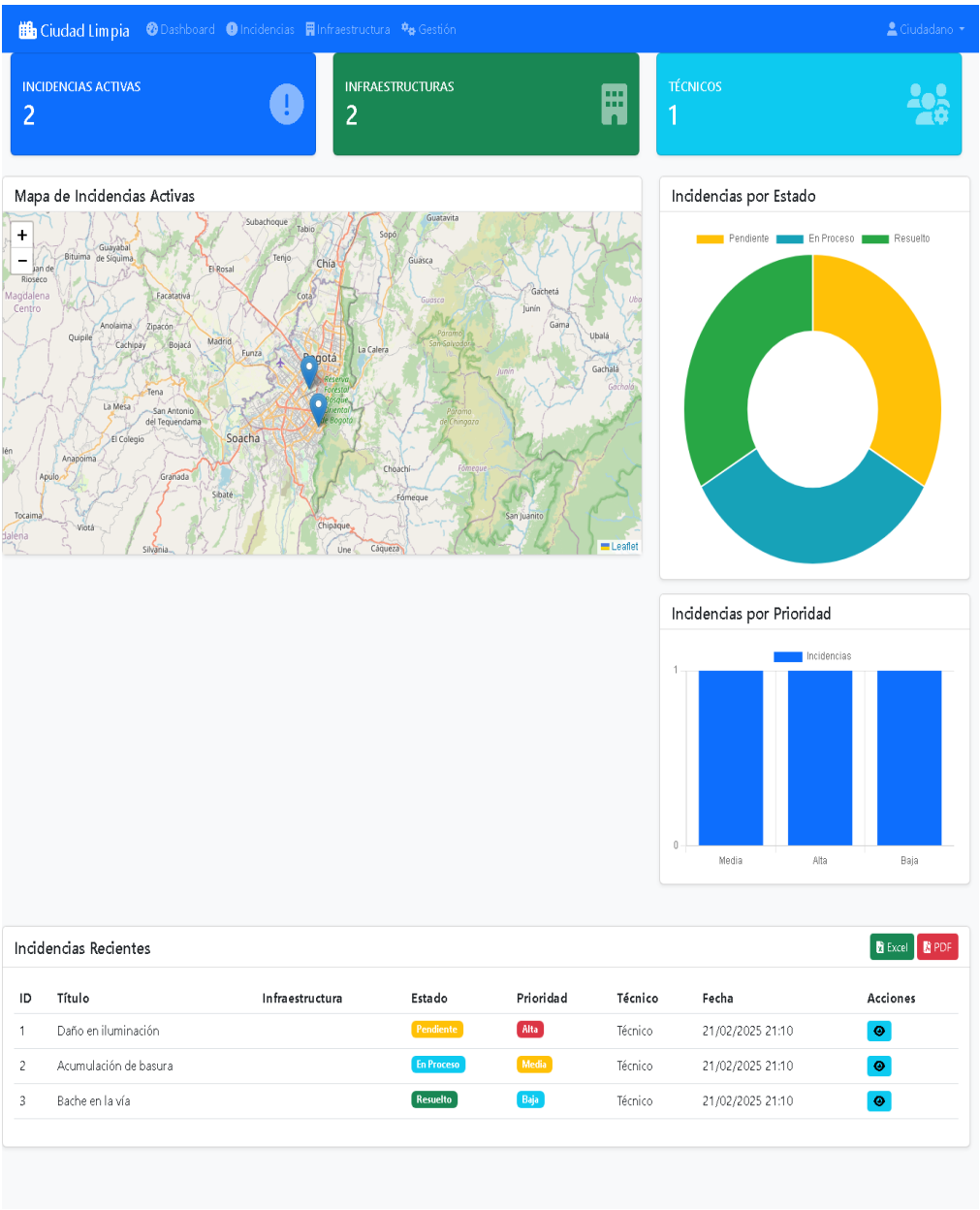
- Se han establecido los privilegios de acceso para el usuario ciudad\_limpia\_user mediante comandos "GRANT".
- Se emplean claves foráneas (importantísimo) y restricciones de integridad para garantizar la consistencia de los datos.

Ejemplo extraído del archivo SQL en Pgadmin4 al hacer Query:

```
1 GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA public TO ciudad_limpia_user;
```


ANEXOS

Ejemplo de cómo se ve la app (demostrativa) con datos y gráfico de las incidencias captadas.



## Acceso:

El sistema permitirá el registro de usuario, a fin de acceder al sistema y procesar las solicitudes correspondientes.



### Ciudad Limpia

Gestión de Infraestructura Urbana

Correo Electrónico

@

Contraseña

☐ Recordarme [¿Olvidaste tu contraseña?](#)

**INICIAR SESIÓN**

[¿No tienes cuenta? Regístrate aquí](#)



## Registro:

El sistema permitirá al usuario el registro a la plataforma mediante la gestión de nuevo usuario: con nombre, dirección de email, y creación de contraseña

A registration form with a light gray background and rounded corners. It contains four text input fields stacked vertically, each with a label above it: 'Name', 'Email', 'Password', and 'Confirm Password'. At the bottom right, there is a dark blue button with the text 'REGISTER' in white. To the left of the button is a link that says 'Already registered?'.

Name

Email

Password

Confirm Password

[Already registered?](#) REGISTER

## Evaluación y Conclusiones

- **Comprensión del diseño de bases de datos:** Se aplicaron conceptos avanzados como normalización, índices y optimización de consultas.
- **Análisis Crítico:** Se realizaron ajustes en base a los requerimientos específicos del sistema.
- **Toma de Decisiones:** Se justifican todas las decisiones técnicas adoptadas.
- **Optimización:** Se implementaron técnicas para mejorar el rendimiento.
- **Seguridad:** Se establecieron políticas de acceso seguras con usuario y contraseñas! (demostración de la plataforma web)

**URL:** <http://ciudadlimpia.onrender.com/>