INGENIERIA DE SOFTWARE

Semestre: 2025-2 Equipo: BUMPER

Fecha: 06 de Abril de 2025



Especificación de Requerimientos

Índice

1.	Introducción	1					
2. Benchmark							
3.	Requerimientos	4					
4.	Casos de Uso de Bumper 4.1. Acceso y Registro de Usuario	5 5 6 6 6					
5. Tecnología Usada							
6.	. Diagramas						
7 Diagrama de Race de Datos							

1. Introducción

Proyecto: Bumper

- Aplicación web para el registro y visualización de incidentes urbanos.
- Bumper planea ser una herramienta colaborativa para mejorar comunicación ciudadana.
- Permitirá a ciudadanos, comerciantes y organizaciones vecinales participar activamente en seguimiento de reportes.

Equipo:BUMPER 1 de 38

Requisitos del cliente

Registro de incidentes

- Permitir a los usuarios marcar la ubicación del incidente en un mapa interactivo.
- Adjuntar una o más fotografías que respalden el incidente reportado.
- Registrar una breve descripción del problema.

Actualización de estado del incidente

• Permitir que cualquier usuario (no solo el creador del reporte) actualice el estado del incidente a Resuelto", siempre que adjunte pruebas fotográficas que validen la solución.

Visualización de incidentes

• Mostrar todos los incidentes en un mapa interactivo, categorizados por tipo y estado (Reportado, En proceso, Resuelto).

2. Benchmark

Este paso lo hemos completado para conocer las **soluciones actuales** en el mundo, conocer las fortalezas, debilidades, las formas de trabajo, conocer de sus interfaces y procesos. Con esta información tendremos un panorama de lo que ya existe y podremos hacer un camino de usuario para lograr una experiencia más amigable de nuestra aplicación.

	FixMyStreet	SeeClickFix	Mejora tu	Colab.re (Brasil)
	(Reino Unido)	(EE.UU.)	Ciudad	Colabire (Brasil)
Marcación en	Sí	Sí (Google	Sí (Google	Sí (Leaflet/O-
mapa	(OpenStreetMap)	Maps)	Maps)	penStreetMap)
Adjuntar	Sí (1+ imágenes)	Sí (hasta 5	Sí (fotos y	Sí (3+ imágenes)
fotos		imágenes)	videos)	Si (5+ illiagelies)
Descripción breve	Texto libre	Campos predefinidos + texto	Texto libre	Texto + etiquetas
Actualización	No (solo creado-	Sí (usuarios	Sí (usuarios y	Sí (usuarios
de estado	r/autoridades)	verificados)	autoridades)	registrados)
Visualización de incidentes	Mapa con filtros	Mapa interactivo con capas	Mapa con íconos	Mapa con filtros y heatmaps
API del mapa	OpenLayers +	Google Maps	Google Maps	Leaflet API +
Al I dei mapa	OpenStreetMap	API + Esri	API	OpenStreetMap
Tecnología	Perl, PostgreSQL, OpenLayers	React, Node.js, AWS	MySQL/Firebase	JavaScript, Python, PostgreSQL
Disponibilidad	Reino Unido, Australia	EE.UU., Canadá	España, México	Brasil, Argentina, Uruguay

Cuadro 1: Comparativa de plataformas de gestión de incidentes urbanos

Equipo:BUMPER 2 de 38

Enlaces a las páginas

■ FixMyStreet (Reino Unido)

Mejora tu Ciudad (España)

■ SeeClickFix (EE.UU.)

■ Colab.re (Brasil)

	FixMyStreet (UK)	SeeClickFix (EEUU)	Mejora tu Ciudad (ES)	Colab.re (BR)
Fortalezas	Integración autoridades Open-source PostgreSQL robusto	Interfaz intuitiva Colaboración real-time Escalabilidad AWS	Multimedia completo Flexibilidad descripciones	Visualización avanzada Stack moderno Foco LATAM
Debilidades	Actualizaciones limitadas Alcance reducido	APIs pagas Límite imágenes	Tecnología oculta Cobertura indefinida	Baja visibilidad global Registro obligatorio
Base de datos	PostgreSQL	AWS RDS (MySQL/PG)	MySQL/Firebase	PostgreSQL
Propuesta de valor	Comunicación simple ciudadano- gobierno	Transparencia mediante reportes verificados	Empoderamiento ciudadano con evidencia visual	Soluciones basadas en datos LATAM

Cuadro 2: Comparativa de puntos específicos

Conclusiones

Basado en el análisis de la competencia, se identificaron prácticas clave que se aplicarán en *Bumper*:

- En FixMyStreet y Colab.re, los filtros en el mapa mejoran la navegación; implementaremos filtros por estatus en React con Leaflet.
- SeeClickFix permite actualizar el estatus con usuarios verificados; habilitaremos esta función para el creador del reporte vía Spring Boot.
- Mejora tu Ciudad destaca por subir multimedia; integraremos la carga de 1-3 fotos, almacenadas en Supabase o el servidor.
- Colab.re usa Leaflet y OpenStreetMap con éxito; adoptaremos esta API gratuita para el mapa interactivo.
- Identificamos en las vistas que componentes como barras laterales y dropmenus mejorar la experiencia de los usuarios al realizar reportes.

Estas decisiones aprovechan las fortalezas de la competencia, adaptándolas a nuestra pila tecnológica (Spring Boot, React, PostgreSQL) para una solución eficiente y diferenciada.

Equipo:BUMPER 3 de 38

3. Requerimientos

Requerimientos Funcionales (RF)

- **RF1:** El sistema debe permitir el registro de nuevos usuarios con nombre, apellido, correo y contraseña únicos.
- **RF2:** El sistema debe autenticar usuarios mediante correo y contraseña, y gestionar el estado de sesión con un token.
- **RF3:** El sistema debe permitir a los usuarios reportar incidentes, incluyendo tipo, ubicación, coordenadas y hasta 3 fotos.
- **RF4:** El sistema debe permitir a los usuarios consultar los incidentes reportados por cualquier usuario, visualizándolos en un mapa con íconos diferenciados por tipo.
- **RF5:** El sistema debe permitir a los usuarios consultar, actualizar el estado (pendiente, en proceso, resuelto) y eliminar los incidentes que hayan reportado.
- **RF6:** El sistema debe permitir a los usuarios cerrar sesión de manera segura.
- **RF7:** El sistema debe permitir la descarga de una imagen con los detalles de un incidente para su difusión en redes sociales, incluyendo foto, tipo, estatus, mensaje de concientización, logo y enlace a la web.
- **RF8:** El sistema debe permitir la actualización de la contraseña del usuario. (*Pendiente de implementación*)

Requerimientos No Funcionales (RNF)

- RNF1: El sistema debe garantizar la seguridad de las contraseñas mediante almacenamiento cifrado
- **RNF2:** El sistema debe responder a las solicitudes en menos de 2 segundos bajo carga normal.
- **RNF3:** El sistema debe ser accesible desde navegadores modernos y dispositivos móviles.
- RNF4: El sistema debe validar los datos de entrada para evitar registros o reportes incompletos o inválidos.
- RNF5: El sistema debe mantener la integridad referencial entre usuarios e incidentes.
- RNF6: El sistema debe permitir la escalabilidad para soportar un crecimiento en el número de usuarios y reportes.
- RNF7: El sistema debe generar imágenes de reporte con calidad suficiente para su publicación en redes sociales.

Equipo:BUMPER 4 de 38

4. Casos de Uso de Bumper

4.1. Acceso y Registro de Usuario

Actor principal: Usuario no autenticado

Propósito: Permitir que un nuevo usuario se registre en el sistema para poder reportar incidentes.

Precondición: El usuario no debe estar registrado previamente.

Flujo principal:

• El usuario accede a la página de registro.

- Ingresa nombre, apellido, correo electrónico y contraseña.
- El sistema valida el formato del correo y la unicidad.
- Si los datos son válidos, el sistema crea la cuenta y muestra un mensaje de éxito.
- Si hay errores, el sistema informa el motivo (correo inválido o ya registrado).

Postcondición: El usuario queda registrado y puede iniciar sesión.

4.2. Inicio de Sesión

Actor principal: Usuario registrado

Propósito: Permitir que un usuario autenticado acceda a las funcionalidades avanzadas del sistema.

Precondición: El usuario debe estar registrado.

Flujo principal:

- El usuario accede a la página de inicio de sesión.
- Ingresa su correo electrónico y contraseña.
- El sistema valida las credenciales.
- Si son correctas, el usuario accede a su cuenta.
- Si son incorrectas, el sistema muestra un mensaje de error.

Postcondición: El usuario queda autenticado en el sistema.

4.3. Creación de Reporte de Incidente

Actor principal: Usuario autenticado

Propósito: Permitir que un usuario reporte un nuevo incidente en la ciudad.

Precondición: El usuario debe estar autenticado.

Flujo principal:

- El usuario selecciona la opción "Reportar Incidente".
- Completa el formulario con ubicación, tipo de incidente, descripción y fotos.
- El sistema valida los datos y registra el incidente.
- El sistema muestra un mensaje de confirmación y el incidente aparece en el mapa.

Postcondición: El incidente queda registrado y visible en el mapa.

Equipo:BUMPER 5 de 38

4.4. Consulta y Gestión de Mis Reportes

Actor principal: Usuario autenticado

Propósito: Permitir que el usuario consulte, actualice el estatus o elimine los incidentes que ha reportado.

Precondición: El usuario debe estar autenticado y tener al menos un incidente reportado.

Flujo principal:

■ El usuario accede a la sección "Mis reportes".

- El sistema muestra la lista de incidentes reportados por el usuario.
- El usuario puede seleccionar un incidente para ver detalles.
- El usuario puede cambiar el estatus del incidente (por ejemplo, a "Resuelto").
- El usuario puede eliminar un incidente propio si lo desea.

Postcondición: El usuario gestiona sus reportes; los cambios se reflejan en el sistema.

4.5. Visualización de Incidentes en el Mapa

Actor principal: Usuario autenticado o invitado

Propósito: Permitir que cualquier usuario visualice los incidentes reportados en el mapa.

Precondición: El sistema debe tener incidentes registrados.

Flujo principal:

■ El usuario accede a la página principal.

- El sistema muestra un mapa con íconos representando los incidentes (por tipo: basura, bache, etc.).
- El usuario puede hacer clic en un ícono para ver detalles básicos (fotos, estatus, tipo).
- El usuario puede filtrar los incidentes por tipo o estatus usando la barra lateral.

Postcondición: El usuario visualiza la información pública de los incidentes.

4.6. Cierre de Sesión

Actor principal: Usuario autenticado

Propósito: Permitir que el usuario cierre su sesión de manera segura.

Precondición: El usuario debe estar autenticado.

Flujo principal:

■ El usuario selecciona la opción "Cerrar sesión".

■ El sistema invalida la sesión y muestra la pantalla de inicio.

Postcondición: El usuario vuelve al estado de invitado.

Equipo:BUMPER 6 de 38

4.7. Cambio de Contraseña

Actor principal: Usuario autenticado

Propósito: Permitir que un usuario autenticado cambie su contraseña desde su perfil.

Precondición: El usuario debe estar autenticado.

Flujo principal:

■ El usuario accede a su perfil

• Selecciona la opción *Cambiar contraseña*.

- Ingresa su contraseña actual, la nueva contraseña y la confirmación de la nueva contraseña.
- El sistema valida que la contraseña actual sea correcta y que las nuevas contraseñas coincidan.
- El sistema actualiza la contraseña del usuario.

Postcondición: El usuario puede acceder nuevamente con la nueva contraseña.

4.8. Descarga de Reporte de Incidente como Imagen para Redes Sociales

Actor principal: Usuario autenticado o invitado

Propósito: Permitir que el usuario descargue una imagen con la información clave de un incidente para compartir en redes sociales.

Precondición: El sistema debe tener incidentes registrados.

Flujo principal:

- El usuario accede a la página de detalles de un incidente.
- Selecciona la opción "Descargar imagen para compartir".
- El sistema genera una imagen con:
 - Una foto del incidente (si está disponible).
 - El tipo de incidente y una breve descripción.
 - El estatus actual.
 - Un texto de concientización o llamado a la acción.
 - El logo de la aplicación y un enlace al sitio web.
- El usuario descarga la imagen en formato JPG o PNG.

Postcondición: El usuario tiene una imagen lista para compartir en redes sociales, promoviendo la conciencia ciudadana y el uso de la aplicación. *Nota: Este caso de uso requiere la implementación de una función para generar la imagen con los datos del incidente.*

Equipo:BUMPER 7 de 38

5. Tecnología Usada

Backend

Lenguaje de Programación: Kotlin

- *Razón de uso:* Kotlin es un lenguaje moderno, conciso y seguro que mejora la productividad del desarrollo al reducir el código repetitivo y minimizar errores comunes (como null pointer exceptions). Su interoperabilidad con Java permite aprovechar bibliotecas existentes, mientras que su sintaxis clara facilita el mantenimiento del código.
- Ventaja para Bumper: Ideal para implementar una API RESTful robusta (RF2, RF3, RF5, RF6) que gestione autenticación, cierre de sesión y operaciones sobre reportes, así como la generación de imágenes para compartir incidentes (RF7). Su eficiencia y escalabilidad facilitan la integración de nuevas funciones como actualización de contraseña (RF8) y notificaciones futuras.

■ Framework: Spring Boot

- *Razón de uso:* Spring Boot simplifica la creación de aplicaciones backend con configuraciones automáticas, integración nativa con bases de datos y soporte para seguridad. Su arquitectura basada en microservicios permite manejar solicitudes simultáneas con baja latencia (RNF2).
- Ventaja para Bumper: Facilita la gestión de reportes, autenticación de usuarios y la lógica de negocio para la descarga de imágenes de incidentes (RF7). Su modularidad permite crecer hacia funcionalidades como recuperación de contraseña o notificaciones (RF8, sugerido).

Frontend

React

- *Razón de uso:* React es una biblioteca de JavaScript que permite construir interfaces dinámicas y responsivas (RNF4) mediante componentes reutilizables. Su enfoque en el estado y la renderización eficiente soporta interacciones en tiempo real, como filtros en el mapa y vistas previas de reportes.
- *Ventaja para Bumper*: Perfecto para integrar un mapa interactivo con íconos personalizados por tipo de incidente (RF4, RF5), una barra lateral dinámica y la opción de descargar imágenes de reportes para redes sociales (RF7). Su ecosistema (e.g., react-leaflet) simplifica la integración con mapas y la experiencia móvil (RNF3).

Base de Datos

PostgreSQL en Supabase

- *Razón de uso:* PostgreSQL es un sistema de base de datos relacional robusto y de código abierto, ideal para almacenar datos estructurados como reportes con ubicaciones, estatus y fotos. Supabase añade una capa de facilidad con almacenamiento de archivos (fotos) y autenticación integrada, reduciendo la complejidad inicial.
- *Ventaja para Bumper:* Garantiza la persistencia de los reportes y usuarios, permite consultas rápidas para filtros y visualización en el mapa (RF4, RF5), y soporta la escalabilidad para más

Equipo:BUMPER 8 de 38

usuarios y datos (RNF6). El almacenamiento de imágenes en Supabase facilita la generación de imágenes para compartir (RF7).

API para Mapas

- Leaflet con OpenStreetMap (Opción Principal)
 - *Razón de uso:* Leaflet es una biblioteca ligera y de código abierto para mapas interactivos, combinada con OpenStreetMap, una fuente de datos gratuita y global. Esto permite visualizar y marcar incidentes (RF4, RF5) sin costos asociados, a diferencia de APIs comerciales.
 - *Ventaja para Bumper:* Su integración con React (via react-leaflet) asegura un mapa fluido y personalizable, con íconos diferenciados por tipo de incidente, filtros dinámicos y baja latencia (RNF2). Es ideal para un MVP escalable y económico (RNF6), con soporte para marcadores, popups y visualización de detalles para descarga de imágenes (RF7).

Equipo:BUMPER 9 de 38

6. Diagramas

Acceso y Registro de Usuario

Proceso de Registro de Usuario



Equipo:BUMPER 10 de 38

Inicio de Sesión

Proceso de Inicio de Sesión del Usuario



Equipo:BUMPER 11 de 38

Creación de Reporte de Incidente

Proceso de Reporte de Incidente



Equipo:BUMPER 12 de 38

Consulta y Gestión de Mis Reportes

Gestión de Reportes de Incidentes



Equipo:BUMPER 13 de 38

Visualización de Incidentes en el Mapa

Visualización de Incidentes en el Mapa



Acceso a la Página Principal

El usuario navega a la página principal.

El sistema muestra un mapa con íconos de incidentes.

Mostrar Mapa con Íconos





Hacer Clic en un Ícono

El usuario hace clic en un ícono para ver detalles.

El sistema muestra detalles básicos del incidente.

Mostrar Detalles Básicos





Filtrar Incidentes

El usuario filtra incidentes por tipo o estatus.

El usuario visualiza la información pública de los incidentes.

Visualizar Información Pública



Equipo:BUMPER 14 de 38

Cierre de Sesión

Secuencia de Cierre de Sesión



Equipo:BUMPER 15 de 38

Cambio de Contraseña

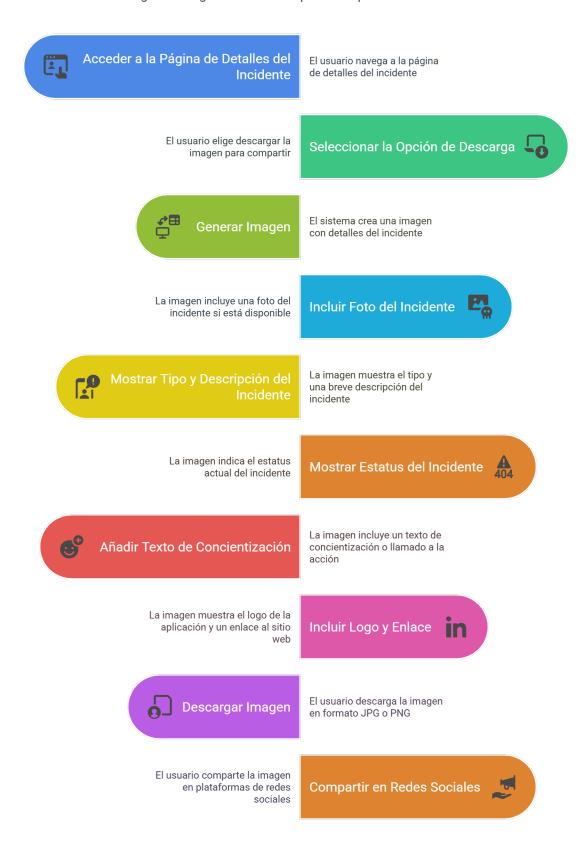
Cambio de Contraseña



Equipo:BUMPER 16 de 38

Descarga de Reporte

Descarga de Imagen de Incidente para Compartir en Redes Sociales



Equipo:BUMPER 17 de 38

Documentación de la API REST

1. Registrar Usuario

Endpoint

POST /v1/users/create

Registra un nuevo usuario en el sistema.

```
Request

URL: http://localhost:8080/v1/users/create
Método: POST
Headers: Content-Type: application/json
Body:

{
    "nombre": "Juan",
    "apellido": "Pérez",
    "correo": "juan@example.com",
    "password": "secreto123"
}
```

Response

Status: 201 Created

Body:

```
"mensaje": "Usuario registrado exitosamente",
"usuario": {
    "id": 1,
    "nombre": "Juan",
    "apellido": "Pérez",
    "correo": "juan@example.com",
    "token": "inactivo",
    "numeroIncidentes": 0,
    "fechaRegistro": "2025-05-04T12:00:00"
}
```

Equipo:BUMPER 18 de 38

```
Status: 400 Bad Request Body (correo inválido):
```

```
{
    "mensaje": "Correo electrónico inválido"
}
```

Response

```
Status: 400 Bad Request Body (validación):
```

```
{
    "mensaje": "Error de validación"
}
```

Response

Status: 500 Internal Server Error

Body:

```
{
    "mensaje": "Error interno al registrar usuario"
}
```

2. Obtener Usuario por Correo

Endpoint

GET /v1/users/correo/{correo}

Obtiene los datos de un usuario por su correo electrónico.

Request

URL: http://localhost:8080/v1/users/correo/juan@example.com

Método: GET

Equipo:BUMPER 19 de 38

```
Status: 200 OK Body:
```

```
"mensaje": "Usuario encontrado",
    "usuario": {
        "id": 1,
        "nombre": "Juan",
        "apellido": "Pérez",
        "correo": "juan@example.com",
        "token": "activo",
        "numeroIncidentes": 0,
        "fechaRegistro": "2025-05-04T12:00:00"
}
```

Response

Status: 404 Not Found

Body:

```
{
    "mensaje": "Usuario no encontrado"
}
```

Response

Status: 500 Internal Server Error

Body:

```
{
    "mensaje": "Error al buscar usuario"
}
```

3. Obtener Usuario por ID

Endpoint

GET /v1/users/{id}

Obtiene los datos de un usuario por su ID.

Equipo:BUMPER 20 de 38

Request

URL: http://localhost:8080/v1/users/1

Método: GET

Response

Status: 200 OK

Body:

```
"mensaje": "Usuario encontrado",
    "usuario": {
        "id": 1,
        "nombre": "Juan",
        "apellido": "Pérez",
        "correo": "juan@example.com",
        "token": "activo",
        "numeroIncidentes": 0,
        "fechaRegistro": "2025-05-04T12:00:00"
}
```

Response

Status: 404 Not Found

Body:

```
{
    "mensaje": "Usuario no encontrado"
}
```

Response

Status: 500 Internal Server Error

Body:

```
{
    "mensaje": "Error al buscar usuario"
}
```

Equipo:BUMPER 21 de 38

4. Login Usuario

Endpoint

POST /v1/users/login Inicia sesión de usuario.

```
Request

URL: http://localhost:8080/v1/users/login
Método: POST
Headers: Content-Type: application/json
Body:

{
    "correo": "juan@example.com",
    "password": "secreto123"
}
```

Response

Status: 200 OK Body:

```
"mensaje": "Login exitoso",
"usuario": {
    "id": 1,
    "nombre": "Juan",
    "apellido": "Pérez",
    "correo": "juan@example.com",
    "token": "activo",
    "numeroIncidentes": 0,
    "fechaRegistro": "2025-05-04T12:00:00"
}
```

Response

Status: 401 Unauthorized

Body:

```
{
    "mensaje": "Credenciales inválidas"
}
```

Equipo:BUMPER 22 de 38

```
Response

Status: 500 Internal Server Error
Body:

[
"mensaje": "Error en el servidor"
]
```

5. Logout Usuario

Endpoint

POST /v1/users/logout Cierra la sesión del usuario.

```
Request
```

```
URL: http://localhost:8080/v1/users/logout
```

Método: POST

Headers: Content-Type: application/json

Body:

```
{
    "correo": "juan@example.com"
}
```

Response

Status: 200 OK

Body:

```
{
    "mensaje": "Sesión cerrada correctamente"
}
```

Response

Status: 404 Not Found

Body:

```
{
    "mensaje": "Usuario no encontrado"
}
```

Equipo:BUMPER 23 de 38

Response **Status:** 500 Internal Server Error **Body:** "mensaje": "Error al cerrar sesión"

6. Actualizar Usuario

Endpoint

PUT /v1/users

Actualiza los datos de un usuario.

```
Request
```

```
URL: http://localhost:8080/v1/users
Método: PUT
Headers: Content-Type: application/json
Body:
     "id": 1,
     "nombre": "Juan",
     "apellido": "Pérez",
     "correo": "juan@example.com",
     "password": "nuevoPassword123",
     "token": "activo",
     "numeroIncidentes": 0
```

24 de 38 Equipo: BUMPER

```
Status: 200 OK Body:
```

```
"mensaje": "Usuario actualizado correctamente",
"usuario": {
    "id": 1,
    "nombre": "Juan",
    "apellido": "Pérez",
    "correo": "juan@example.com",
    "token": "activo",
    "numeroIncidentes": 0,
    "fechaRegistro": "2025-05-04T12:00:00"
}
```

Response

Status: 400 Bad Request Body (correo inválido):

```
{
    "mensaje": "Correo electrónico inválido"
}
```

Response

Status: 400 Bad Request Body (validación):

```
{
    "mensaje": "Error de validación"
}
```

Response

Status: 500 Internal Server Error

Body:

```
{
    "mensaje": "Error al actualizar usuario"
}
```

Equipo:BUMPER 25 de 38

1. Crear Incidente

Endpoint

POST /v1/incidentes/create Registra un nuevo incidente en el sistema.

```
Request

URL: http://localhost:8080/v1/incidentes/create
Método: POST
Headers: Content-Type: application/json
Body:

{
    "usuarioId": 1,
    "tipoIncidente": "BACHES",
    "ubicacion": "Calle 123, Colonia Centro",
    "latitud": 19.4326,
    "longitud": -99.1332,
    "tipoVialidad": "CALLE"
}
```

Response

Status: 201 Created

Body:

```
"mensaje": "Incidente creado exitosamente",
    "incidente": {
        "id": "20250504120000_BAC_CAL",
        "usuarioId": 1,
        "tipoIncidente": "BACHES",
        "ubicacion": "Calle 123, Colonia Centro",
        "latitud": 19.4326,
        "longitud": -99.1332,
        "horaIncidente": "2025-05-04T12:00:00",
        "tipoVialidad": "CALLE",
        "estado": "PENDIENTE",
        "fotos": []
    }
}
```

Equipo:BUMPER 26 de 38

```
Response

Status: 404 Not Found
Body:

{
    "mensaje": "Usuario no encontrado",
    "usuarioId": 1
```

```
Status: 500 Internal Server Error Body:
```

```
"mensaje": "Error al registrar el incidente",
    "error": "Descripción del error"
}
```

2. Obtener todos los incidentes

Endpoint

GET /v1/incidentes/all

Obtiene todos los incidentes registrados.

Request

URL: http://localhost:8080/v1/incidentes/all

Método: GET

Equipo:BUMPER 27 de 38

Status: 200 OK

Body (con incidentes):

```
"mensaje": "Incidentes encontrados",
"total": 2,
"incidentes": [
    {
        "id": "20250504120000_BAC_CAL",
        "usuarioId": 1,
        "tipoIncidente": "BACHES",
        "ubicacion": "Calle 123, Colonia Centro",
        "latitud": 19.4326,
        "longitud": -99.1332,
        "horaIncidente": "2025-05-04T12:00:00",
        "tipoVialidad": "CALLE",
        "estado": "PENDIENTE",
        "fotos": []
    // ...otros incidentes
1
```

Response

```
Status: 200 OK
```

Body (sin incidentes):

```
"mensaje": "No se encontraron incidentes registrados",
   "incidentes": []
}
```

Response

Status: 500 Internal Server Error

Body:

```
{
    "mensaje": "Error al obtener los incidentes"
}
```

Equipo:BUMPER 28 de 38

3. Obtener incidentes por usuario

Endpoint

GET /v1/incidentes/usuario/{usuarioId} Obtiene todos los incidentes de un usuario específico.

Request

URL: http://localhost:8080/v1/incidentes/usuario/1

Método: GET

Response

Status: 200 OK

Body:

```
{
    "mensaje": "Incidentes encontrados para el usuario",
    "usuario": {
        "id": 1,
        "nombre": "Juan Pérez"
    } ,
    "total": 1,
    "incidentes": [
        {
            "id": "20250504120000_BAC_CAL",
            "usuarioId": 1,
            "tipoIncidente": "BACHES",
            "ubicacion": "Calle 123, Colonia Centro",
            "latitud": 19.4326,
            "longitud": -99.1332,
            "horaIncidente": "2025-05-04T12:00:00",
            "tipoVialidad": "CALLE",
            "estado": "PENDIENTE",
            "fotos": [],
            "usuario": {
                "id": 1,
                "nombre": "Juan",
                "apellido": "Pérez"
        }
    ]
```

Equipo:BUMPER 29 de 38

```
Status: 404 Not Found
```

Body:

```
{
    "mensaje": "Usuario no encontrado"
}
```

Response

Status: 500 Internal Server Error

Body:

```
"mensaje": "Error al obtener los incidentes del usuario"
}
```

4. Obtener incidente por ID

Endpoint

GET /v1/incidentes/{id} Obtiene un incidente por su ID.

Request

URL: http://localhost:8080/v1/incidentes/20250504120000_BAC_CAL

Método: GET

Equipo:BUMPER 30 de 38

Response Status: 200 OK **Body:** "mensaje": "Incidente encontrado", "incidente": { "id": "20250504120000_BAC_CAL", "usuarioId": 1, "tipoIncidente": "BACHES", "ubicacion": "Calle 123, Colonia Centro", "latitud": 19.4326, "longitud": -99.1332, "horaIncidente": "2025-05-04T12:00:00", "tipoVialidad": "CALLE", "estado": "PENDIENTE", "fotos": [] }

Response

Status: 404 Not Found

Body:

```
{
    "mensaje": "Incidente no encontrado"
}
```

Response

Status: 500 Internal Server Error

Body:

```
{
    "mensaje": "Error al buscar el incidente"
}
```

5. Actualizar estado de incidente

Endpoint

PUT /v1/incidentes/update-status/{id}

Actualiza el estado de un incidente.

Equipo:BUMPER 31 de 38

```
Request

URL: http://localhost:8080/v1/incidentes/update-status/20250504120000_BAC_CAI
Método: PUT
Headers: Content-Type: application/json
Body:

{
    "estado": "RESUELTO"
}
```

```
Status: 200 OK
Body:

{
    "mensaje": "Estado actualizado correctamente",
    "incidente": {
        "id": "20250504120000_BAC_CAL",
        "usuarioId": 1,
        "tipoIncidente": "BACHES",
        "ubicacion": "Calle 123, Colonia Centro",
        "latitud": 19.4326,
        "longitud": -99.1332,
        "horaIncidente": "2025-05-04T12:00:00",
        "tipoVialidad": "CALLE",
        "estado": "RESUELTO",
        "fotos": []
    }
}
```

```
Status: 403 Forbidden
Body:

{
    "mensaje": "No tienes permiso para modificar este incidente"
}
```

Equipo:BUMPER 32 de 38

```
Status: 404 Not Found
```

Body:

```
{
    "mensaje": "Incidente no encontrado"
}
```

Response

Status: 500 Internal Server Error

Body:

```
"mensaje": "Error al actualizar el estado del incidente"
}
```

6. Buscar incidentes cercanos

Endpoint

GET /v1/incidentes/cercanos

Busca incidentes cercanos a una ubicación geográfica.

Request

URL: http://localhost:8080/v1/incidentes/cercanos?latitud=19.4326&ldngitud=-

Método: GET

Equipo:BUMPER 33 de 38

```
Status: 200 OK Body:
```

```
"mensaje": "Búsqueda completada",
"parametros": {
    "latitud": 19.4326,
    "longitud": -99.1332,
    "radioKm": 5.0
} ,
"total": 1,
"incidentes": [
    {
        "id": "20250504120000_BAC_CAL",
        "usuarioId": 1,
        "tipoIncidente": "BACHES",
        "ubicacion": "Calle 123, Colonia Centro",
        "latitud": 19.4326,
        "longitud": -99.1332,
        "horaIncidente": "2025-05-04T12:00:00",
        "tipoVialidad": "CALLE",
        "estado": "RESUELTO",
        "fotos": []
]
```

Response

```
Status: 500 Internal Server Error
```

Body:

```
{
    "mensaje": "Error al buscar incidentes cercanos"
}
```

7. Eliminar incidente

Endpoint

DELETE /v1/incidentes/{id}

Elimina un incidente creado por un usuario.

Equipo:BUMPER 34 de 38

```
Request
```

URL: http://localhost:8080/v1/incidentes/20250504120000_BAC_CAL?usuarioId=1

Método: DELETE

Response

```
Status: 200 OK
```

Body:

```
{
    "mensaje": "Incidente eliminado correctamente"
}
```

Response

Status: 403 Forbidden

Body:

```
{
    "mensaje": "No tienes permiso para eliminar este incidente"
}
```

Response

Status: 404 Not Found

Body:

```
{
    "mensaje": "Incidente no encontrado"
}
```

Response

Status: 500 Internal Server Error

Body:

```
{
    "mensaje": "Error al eliminar el incidente",
    "error": "Descripción del error"
}
```

Equipo:BUMPER 35 de 38

7. Diagrama de Base de Datos

Estructura General

La base de datos está diseñada para gestionar usuarios y reportes de incidentes urbanos. Consta de dos tablas principales: usuarios y incidentes. Cada usuario puede registrar múltiples incidentes, y cada incidente puede contener hasta tres fotos, almacenadas como URLs en un arreglo.

Tabla usuarios

Esta tabla almacena la información básica de cada usuario registrado en el sistema. Los campos principales son:

- id: Identificador único autoincremental (SERIAL PRIMARY KEY)
- nombre y apellido: Datos personales del usuario (VARCHAR(100) NOT NULL)
- correo: Correo electrónico único, utilizado para autenticación (VARCHAR(100) UNIQUE NOT NULL)
- password: Contraseña cifrada del usuario (VARCHAR(100) NOT NULL)
- token: Indica si el usuario tiene una sesión activa o inactiva (VARCHAR(100) DEFAULT 'inactivo')
- numero_incidentes: Contador automático de incidentes reportados por el usuario (INTEGER DEFAULT 0)
- fecha_registro: Fecha y hora de registro del usuario (TIMESTAMP DEFAULT CURRENT_TIMESTAMP)

Se incluyen índices para optimizar búsquedas por correo (idx_usuarios_correo) y token (idx_usuarios_token).

Tabla incidentes

Esta tabla almacena los reportes de incidentes realizados por los usuarios. Sus campos principales son:

- id: Identificador único del incidente (VARCHAR(50) PRIMARY KEY), generado por el backend
- usuario_id: Referencia al usuario que reportó el incidente (INTEGER NOT NULL REFEREN-CES usuarios(id) ON DELETE CASCADE)
- tipo_incidente: Categoría del incidente (VARCHAR(20) NOT NULL), con valores permitidos: ILUMINACION, BACHES, BASURA, OTRO
- ubicación: Descripción textual de la ubicación (VARCHAR(200) NOT NULL)
- latitud y longitud: Coordenadas geográficas del incidente (DOUBLE PRECISION NOT NULL), con restricciones de rango

Equipo:BUMPER 36 de 38

- hora_incidente: Fecha y hora en que se reportó el incidente (TIMESTAMP NOT NULL DE-FAULT CURRENT_TIMESTAMP)
- tipo_vialidad: Tipo de vialidad donde ocurrió el incidente (VARCHAR(50) NOT NULL), con valores permitidos: CALLE, AVENIDA, CERRADA, OTRO
- estado: Estado del incidente (VARCHAR(20) NOT NULL DEFAULT 'PENDIENTE'), con valores permitidos: PENDIENTE, EN_PROCESO, RESUELTO
- fotos: Arreglo de URLs (TEXT[] DEFAULT "::text[]), con un máximo de 3 fotos

Se incluyen restricciones CHECK para asegurar la validez de los datos (límites de longitud, valores permitidos y cantidad máxima de fotos).

Integridad y Automatización

Para mantener la integridad de los datos y automatizar procesos:

- Trigger de contador de incidentes (actualizar_numero_incidentes): Actualiza automáticamente el campo numero_incidentes en la tabla usuarios al insertar o eliminar incidentes.
- Restricción de fotos: Limita a un máximo de 3 fotos por incidente en el campo fotos.

Índices y Optimización

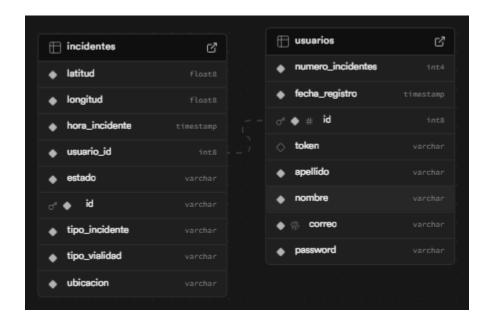
Se han creado índices en campos frecuentemente consultados para mejorar el rendimiento de las consultas:

- idx_usuarios_correo: Índice en el campo correo de la tabla usuarios.
- idx_usuarios_token: Índice en el campo token de la tabla usuarios.
- idx_incidentes_ubicacion: Índice en los campos latitud y longitud de la tabla incidentes.
- idx_incidentes_estado: Índice en el campo estado de la tabla incidentes.
- idx_incidentes_usuario: Índice en el campo usuario_id de la tabla incidentes.
- idx_incidentes_fecha: Índice en el campo hora_incidente de la tabla incidentes, ordenado de forma descendente.
- idx_incidentes_usuario_id: Índice en el campo usuario_id de la tabla incidentes.

Seguridad y Sesiones

La gestión de sesiones se realiza mediante el campo token en la tabla usuarios. Este campo indica si el usuario tiene una sesión activa o inactiva. Se puede ampliar con una columna de expiración para sesiones temporales.

Equipo:BUMPER 37 de 38



Equipo:BUMPER 38 de 38