UNIVERSIDAD PRIVADA DE TACNA

FACULTAD DE INGENIERÍA

Escuela Profesional de Ingeniería de Sistemas

"Sistema Web y Móvil de gestión de incidencias vía PHP y Flutter para la mejora de las infraestructuras públicas del distrito Gregorio Albarracín"

Curso: *Construcción de Software I*

Docente: *Ing. Flor Rodríguez, Alberto Jonathan*

Integrantes:

Hurtado Ortiz, Leandro ***(2015052384)***
Castañeda Centurion, Jorge Enrique ***(2021069822)***

Tacna – Perú
2025

Sistema Web y Móvil de gestión de incidencias vía PHP y Flutter para la mejora de las infraestructuras públicas del distrito Gregorio Albarracín

Documento de Arquitectura de Software

Versión *1.0*

CONTROL DE VERSIONES					
Versión	Hecha por	Revisada por	Aprobada por	Fecha	Motivo
1.0	JL	JL	JL	14/04/2025	Versión Original

ÍNDICE

1. INTRODUCCIÓN

1.1. Propósito

El propósito de este documento es describir la arquitectura del sistema "Sistema Web y Móvil de gestión de incidencias" utilizando el modelo arquitectónico 4+1. Este sistema busca mejorar la detección y gestión de incidencias en infraestructuras públicas del distrito Gregorio Albarracín mediante la participación ciudadana.

El modelo 4+1 permite representar la arquitectura del sistema desde diferentes puntos de vista: lógica, desarrollo, procesos, despliegue y casos de uso. Esta documentación facilitará la comprensión del sistema por parte de todos los involucrados en el desarrollo, implementación y mantenimiento.

1.2. Alcance

El sistema contempla una aplicación móvil para ciudadanos y empleados desarrollada en Flutter, una plataforma web en PHP para administradores, y una API backend común conectada a una base de datos PostgreSQL. Permitirá reportar incidencias con ubicación e imagen, asignarlas a empleados, actualizarlas, monitorear su estado y generar reportes. Está dirigido al distrito Gregorio Albarracín y enfocado en mejorar la gestión municipal mediante tecnología y participación ciudadana.

1.3. Definición, siglas y abreviaturas

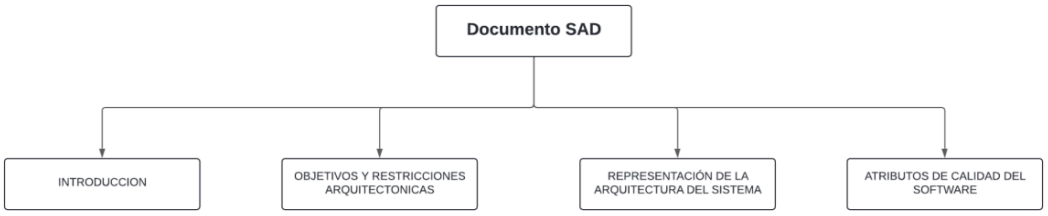
- **API (Application Programming Interface):**
  Conjunto de reglas y protocolos que permiten la comunicación entre diferentes componentes de software, como las aplicaciones móviles y el servidor.

- **JWT (JSON Web Token):**
  Método seguro de autenticación basado en tokens, utilizado para gestionar sesiones de usuario de forma cifrada.

- **Flutter:**
  Framework de desarrollo multiplataforma que permite crear aplicaciones móviles tanto para Android como para iOS desde una única base de código.

- **PHP:**
  Lenguaje de programación del lado del servidor, utilizado en este proyecto para construir el backend y la plataforma web del administrador.

1.4. Organización del documento



2. OBJETIVOS Y RESTRICCIONES ARQUITECTÓNICAS

2.1. Requerimientos Funcionales

Código	Requerimiento	Descripción
RF-01	Reporte de incidencias	Permitir a los ciudadanos registrar incidentes a través de la aplicación, proporcionando una descripción, imágenes y geolocalización.
RF-02	Autenticación de Usuarios	Permitir a los empleados registrarse e iniciar sesión en la aplicación.
RF-03	Monitoreo y Actualización de Incidencias	Permitir a los empleados visualizar una lista de las incidencias asignadas y actualizar su estado.
RF-04	Autenticación de Administradores	Permitir a los administradores registrarse e iniciar sesión en la plataforma.
RF-05	Gestión del Dashboard	Permitir a los administradores asignar incidencias a empleados y supervisar su estado a través de un dashboard.
RF-06	Generación de Reportes de Incidencias	Permitir a los administradores generar reportes consolidados sobre las incidencias registradas.

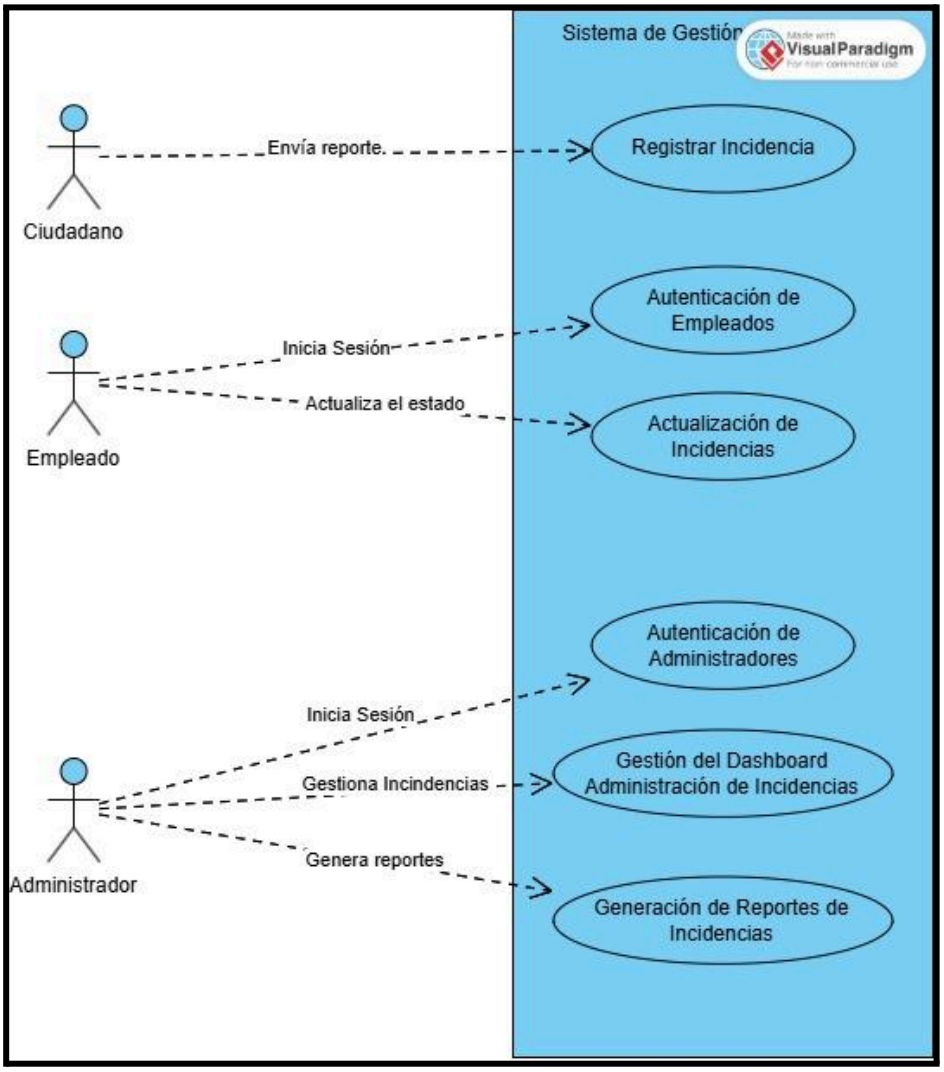
2.2. Requerimientos No Funcionales - Atributos de Calidad

Código	Requerimiento	Descripción
RNF-01	Seguridad	El sistema debe encriptar las contraseñas para las sesiones con el uso de token JWT.
RNF-02	Rendimiento	El sistema debe tener un buen rendimiento gracias a operaciones relacionadas con la base de datos usando buenas prácticas como autoload y namespaces, además de minimizar el uso de funciones globales y conseguir un código más limpio, evitando en todo momento el uso de las consultas SQL en posibles loops del código general.
RNF-03	Usabilidad	El sistema debe tener interfaces sencillas de usar tanto para administradores como para usuarios generales.

3. REPRESENTACIÓN DE LA ARQUITECTURA DEL SISTEMA

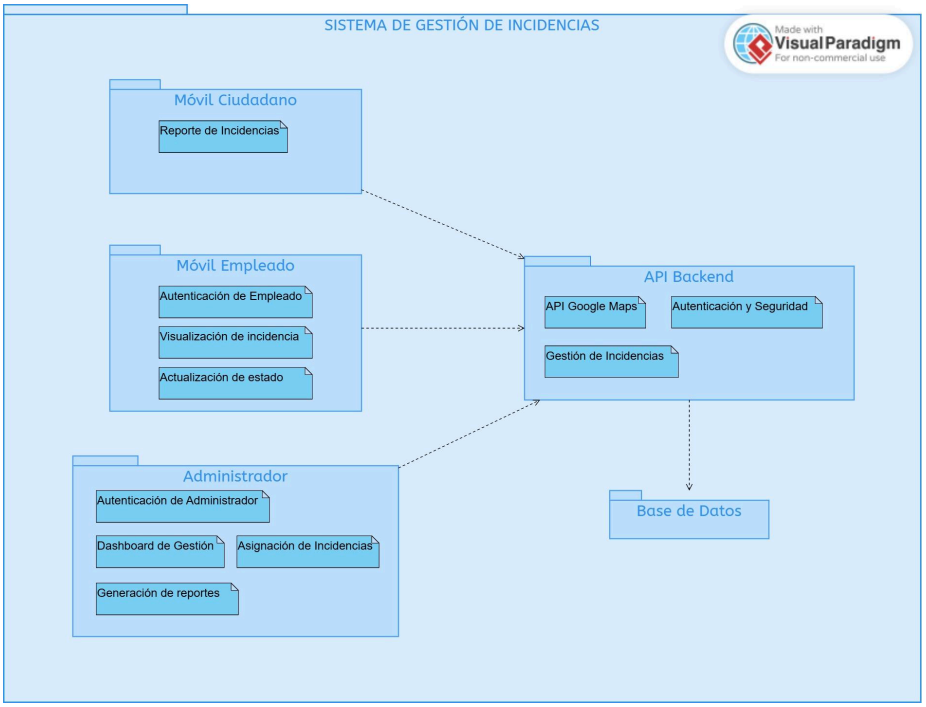
3.1. Vista de Caso de Uso

3.1.1. Diagramas de Casos de Uso



3.2. Vista Lógica

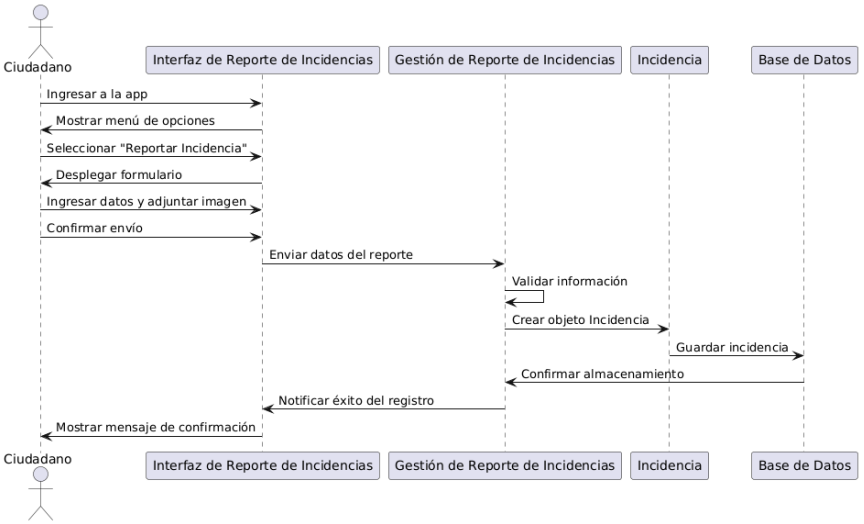
3.2.1. Diagrama de Subsistemas



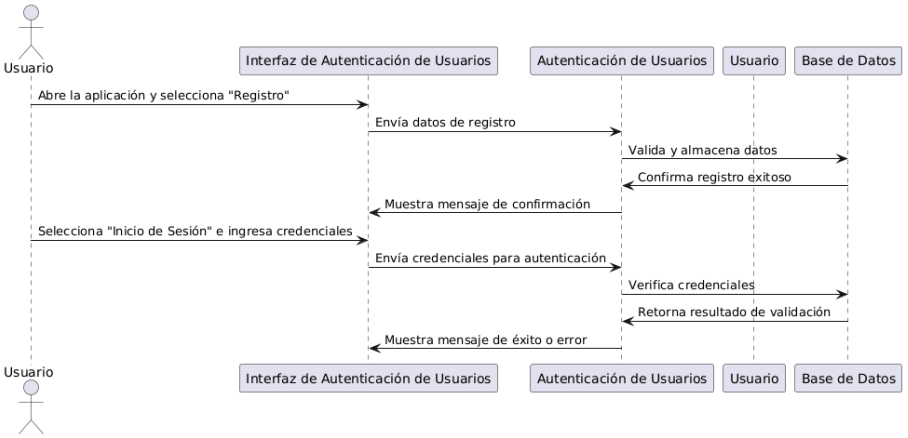
El diagrama muestra la estructura del proyecto con los subpaquetes principales: Móvil Ciudadano, Móvil Empleado, Administrador, API Backend y Base de datos junto a las relaciones correspondientes según los requerimientos.

3.2.2. Diagrama de Secuencia

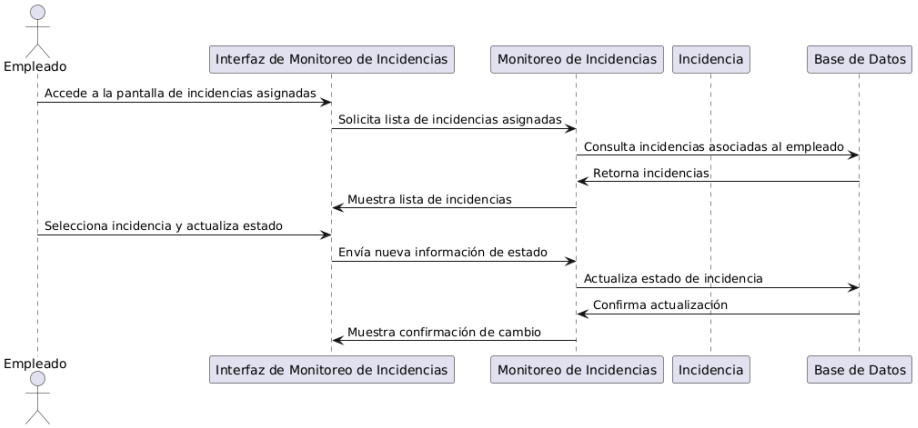
- RF01: Reporte de Incidencias



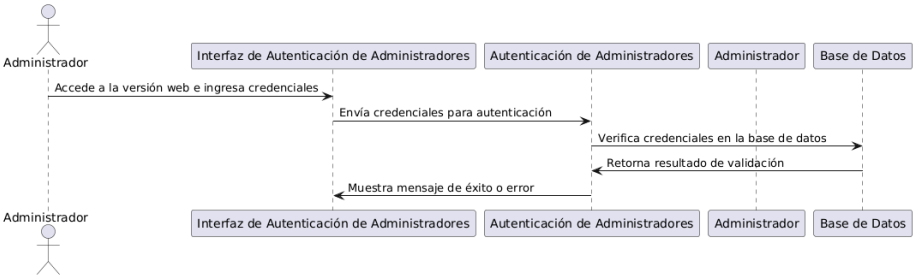
- RF02: Autenticación de Usuarios (Empleados)



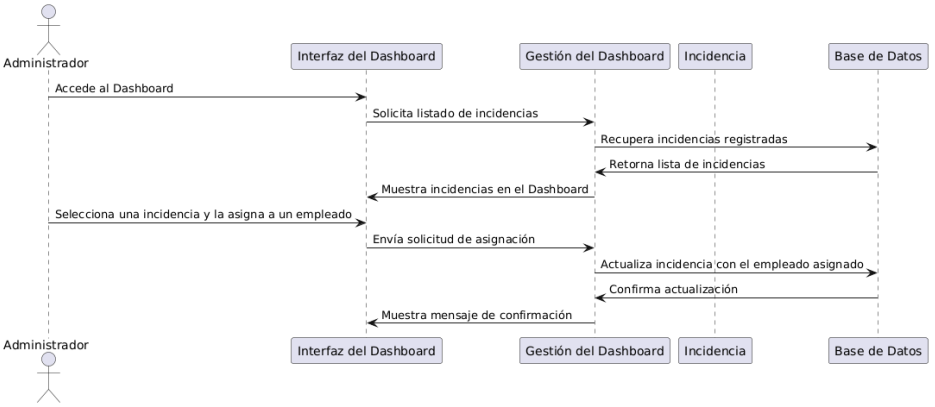
- RF03: Monitoreo y Actualización de de Incidencias



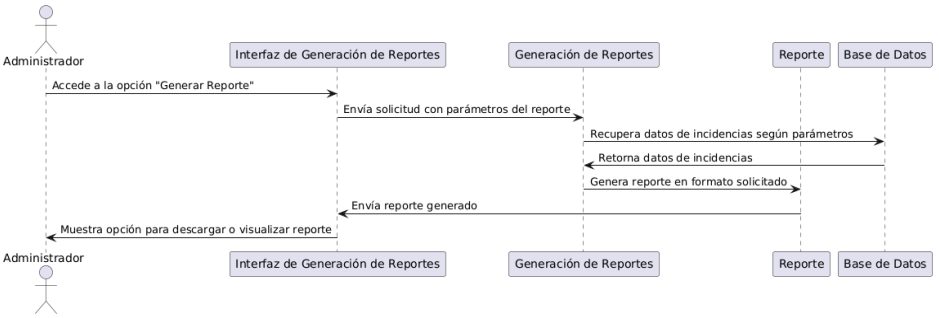
- RF04: Autenticación de Administradores



- RF05: Gestión del Dashboard

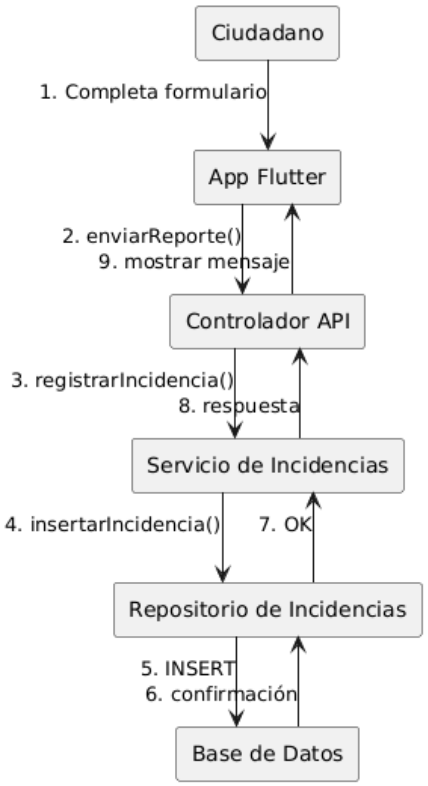


- RF06: Generación de Reportes de Incidencias

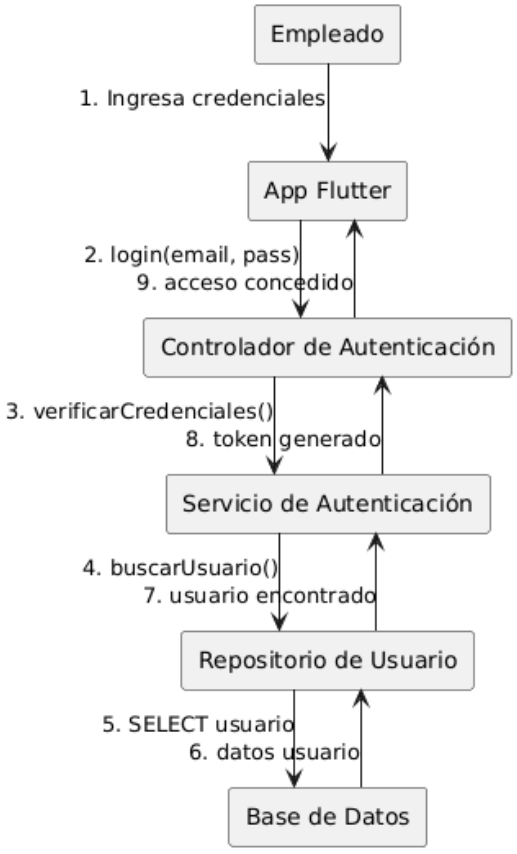


3.2.3. Diagrama de Colaboración

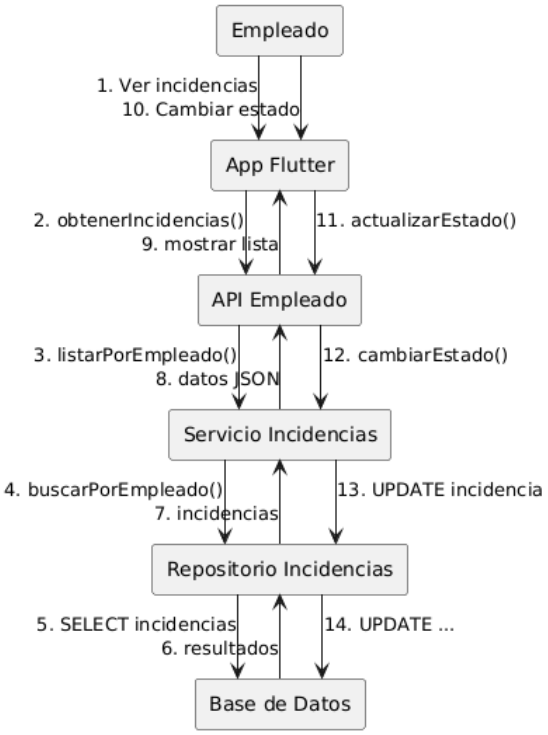
- RF01: Reporte de Incidencias



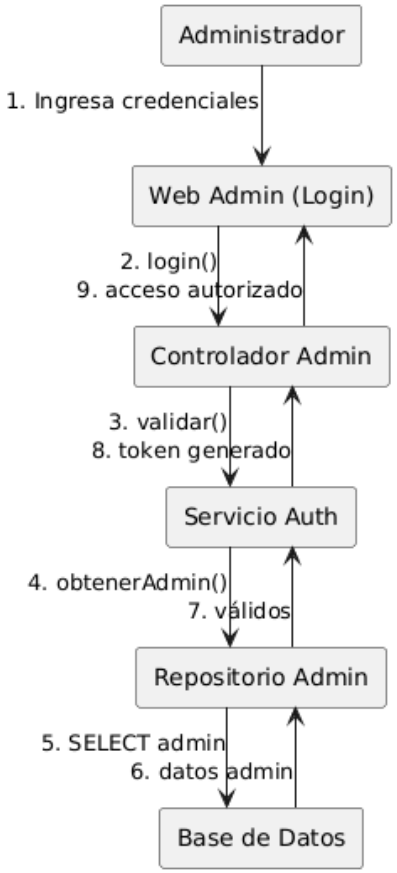
- RF02: Autenticación de Usuarios (Empleados)



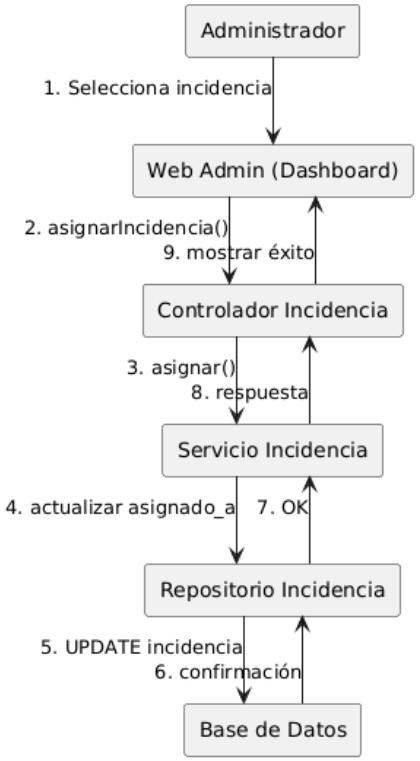
- RF03: Monitoreo y Actualización de de Incidencias



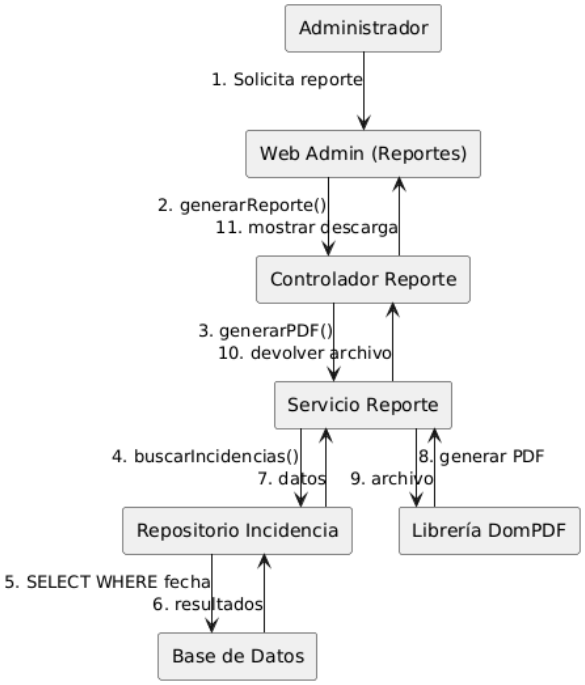
- RF04: Autenticación de Administradores



- RF05: Gestión del Dashboard

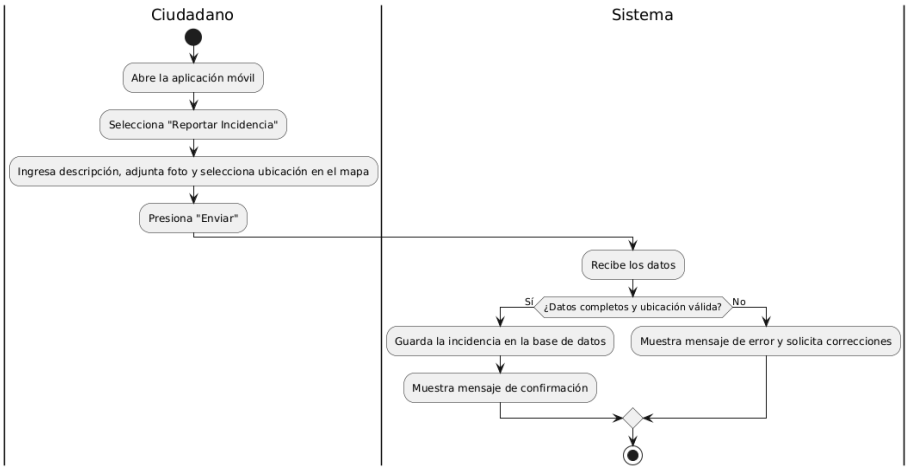


- RF06: Generación de Reportes de Incidencias

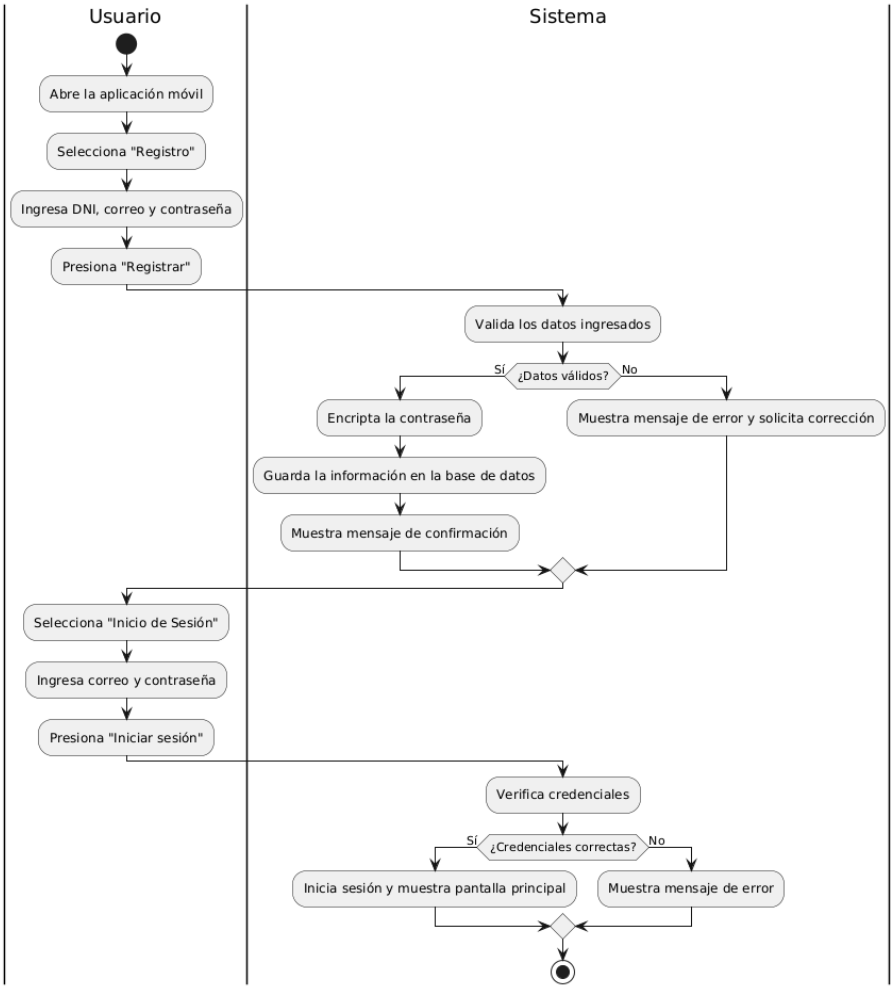


3.2.4. Diagrama de Objetos

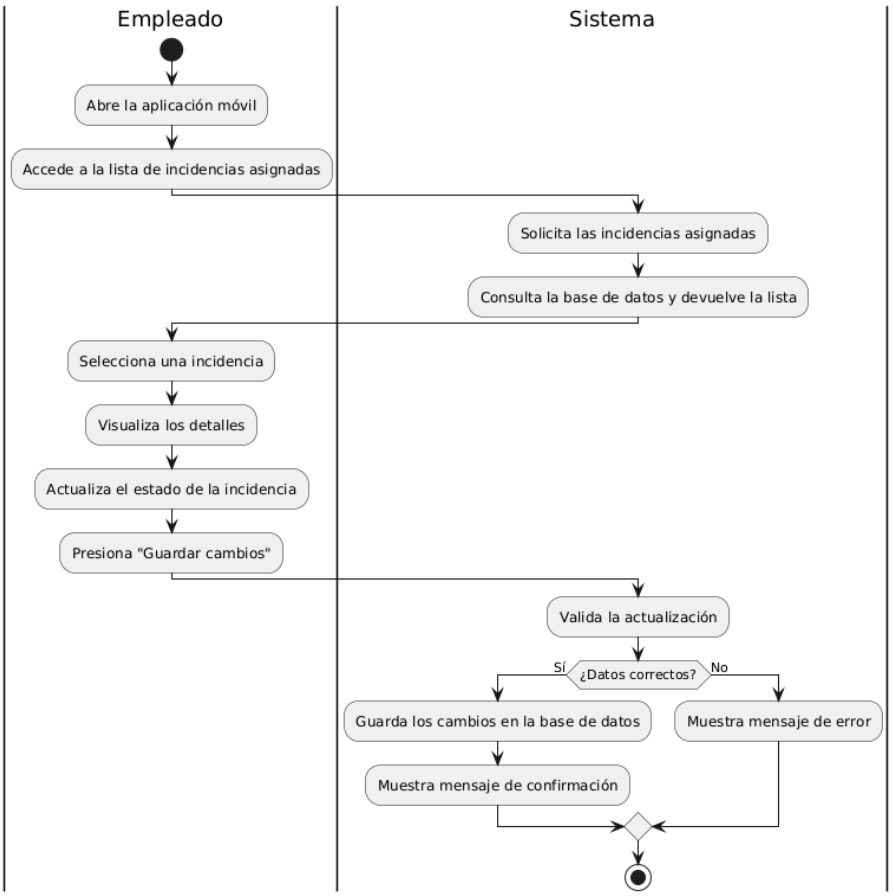
- RF01: Reporte de Incidencias



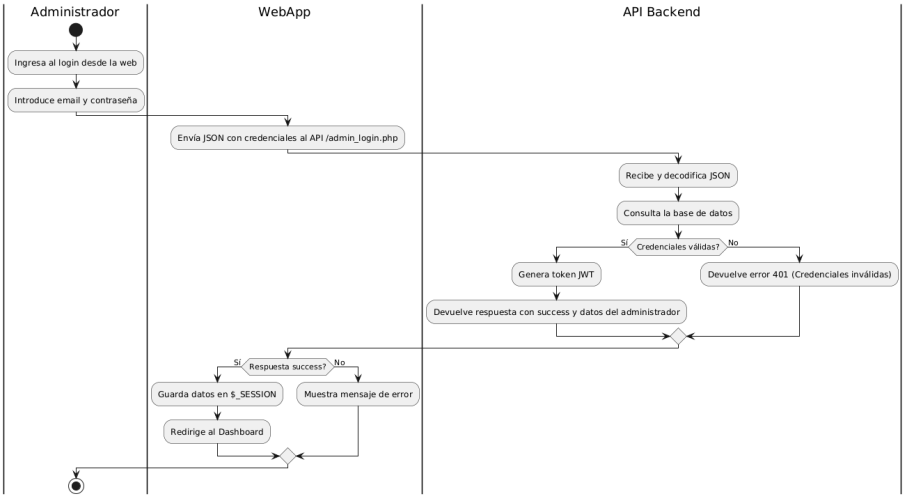
- RF02: Autenticación de Usuarios (Empleados)



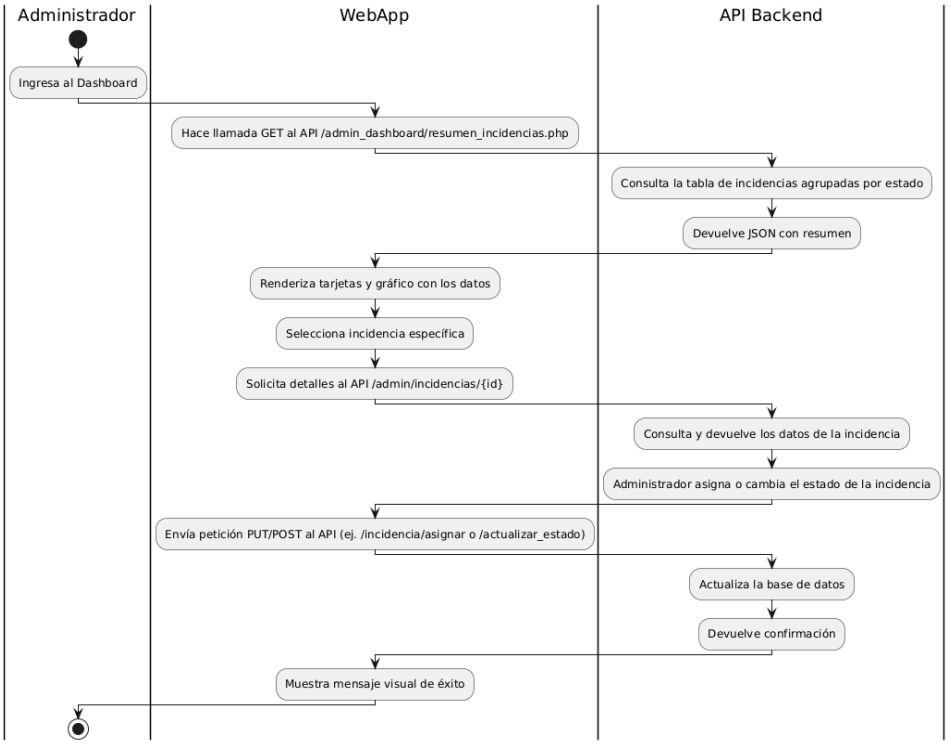
- RF03: Monitoreo y Actualización de de Incidencias



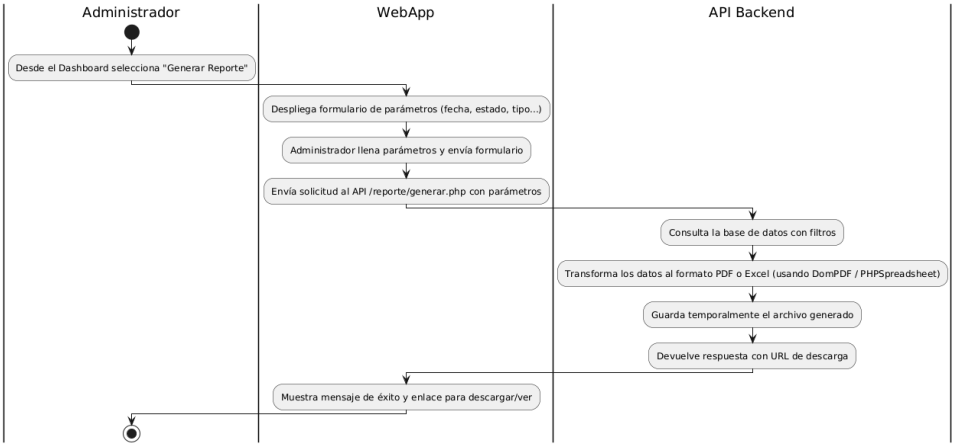
- RF04: Autenticación de Administradores



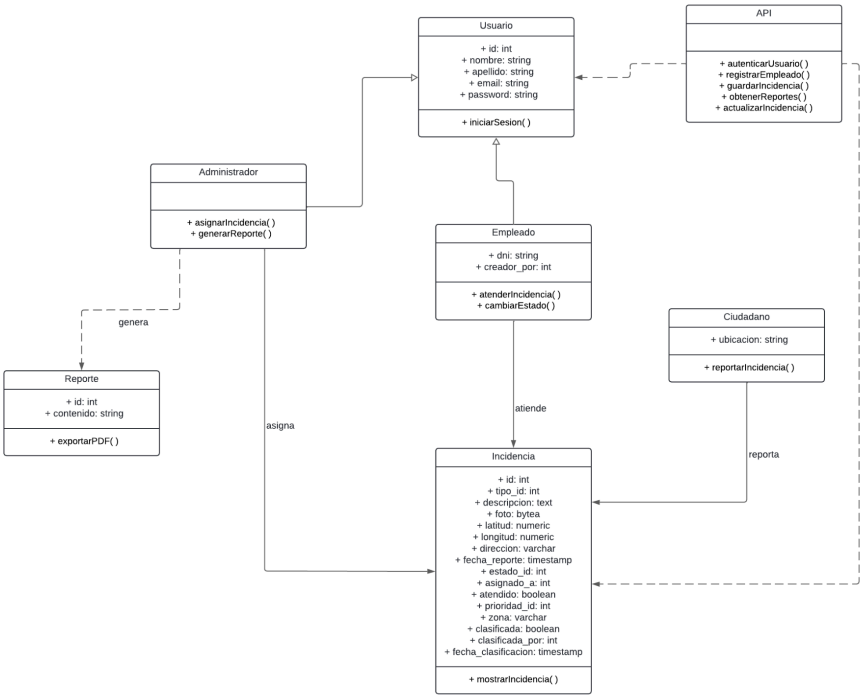
- RF05: Gestión del Dashboard



- RF06: Generación de Reportes de Incidencias

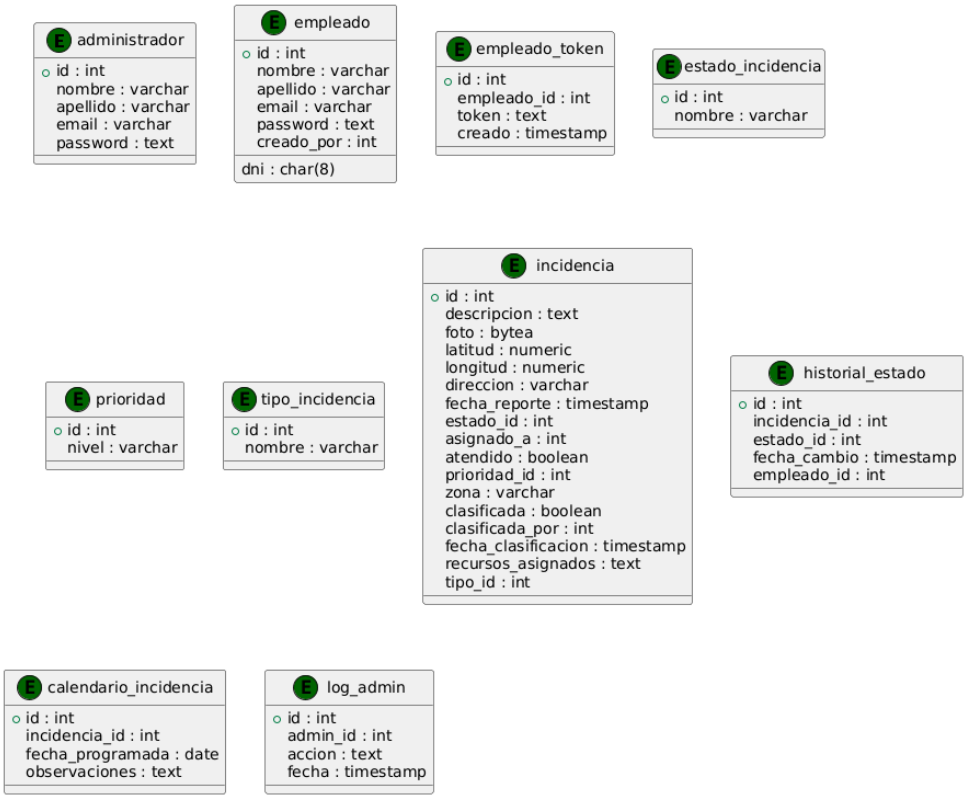


3.2.5. Diagrama de Clases



Este diagrama de clases, correspondiente a la vista lógica del SAD, define las principales entidades del sistema, sus atributos, métodos y relaciones. Representa cómo se estructuran los componentes de negocio y cómo interactúan entre sí para cumplir con los casos de uso definidos.

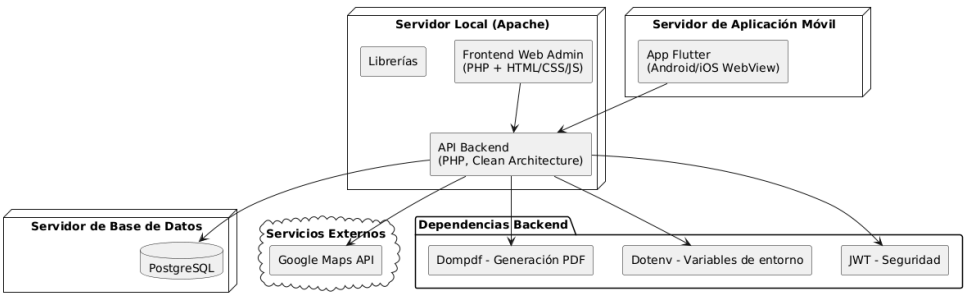
3.2.6. Diagrama de Base de Datos



El siguiente diagrama muestra las principales entidades del sistema y sus atributos. Las relaciones entre entidades están correctamente implementadas mediante claves foráneas en la base de datos, pero no se representan visualmente en esta vista con el fin de mantener la claridad y enfocar la atención en la estructura interna de cada entidad.

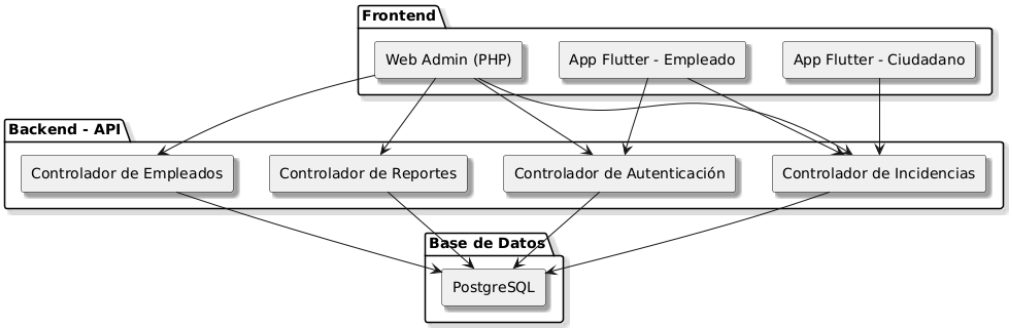
3.3. Vista de Implementación

3.3.1. Diagrama de arquitectura de software



El diagrama muestra la arquitectura del software distribuida en tres servidores: uno local con el frontend web y el backend en PHP, uno móvil con la app Flutter, y uno de base de datos con PostgreSQL. El sistema se apoya en servicios externos como Google Maps y librerías como Dompdf, Dotenv y JWT para funciones clave como autenticación y generación de reportes.

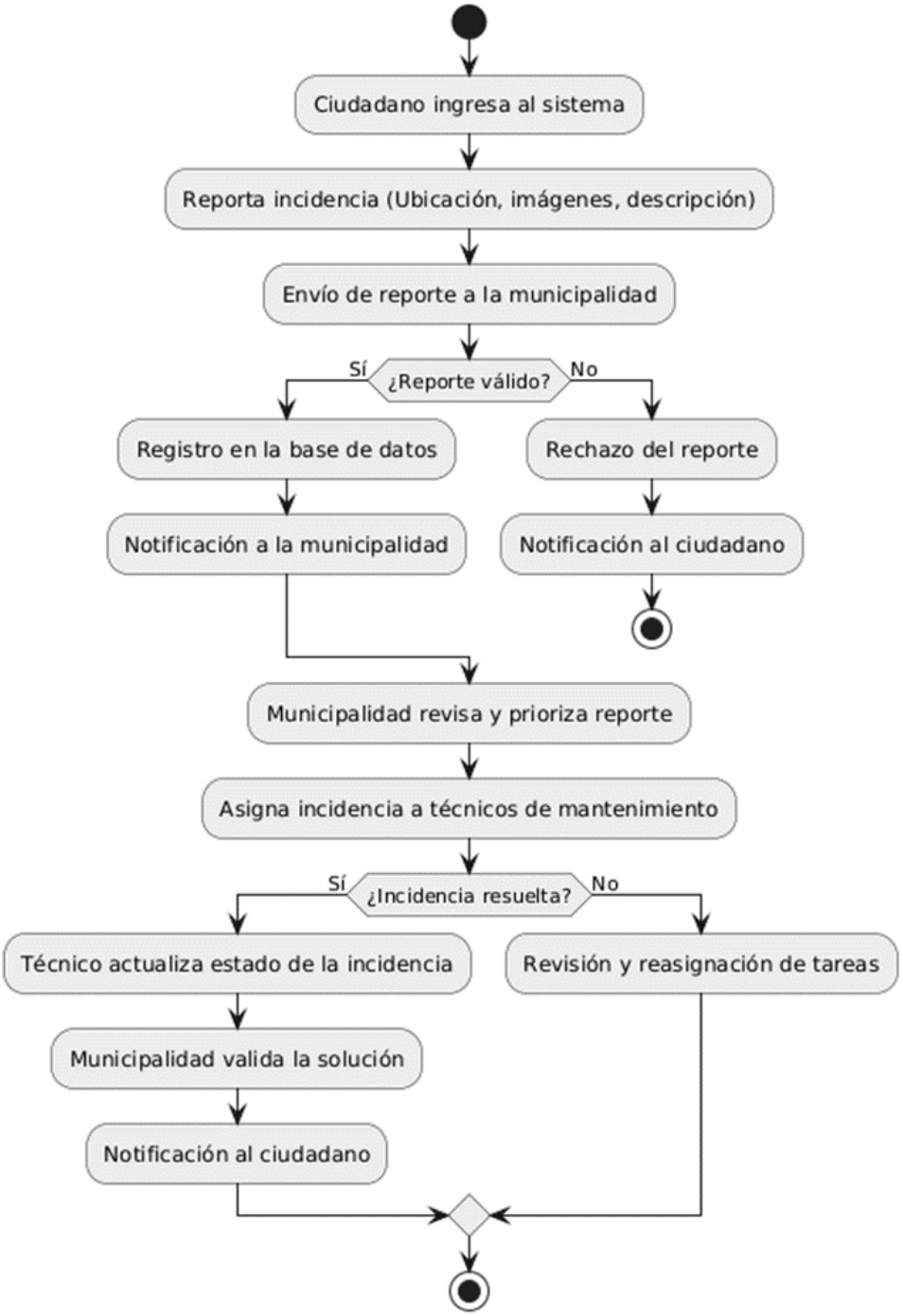
3.3.2. Diagrama de arquitectura del sistema



El siguiente diagrama de componentes muestra la arquitectura del sistema organizada por capas. Las aplicaciones móviles y la plataforma web consumen servicios expuestos por el backend mediante controladores específicos. Cada controlador gestiona una funcionalidad principal del sistema (autenticación, incidencias, reportes, empleados) y se comunica directamente con la base de datos PostgreSQL para almacenar y recuperar información.

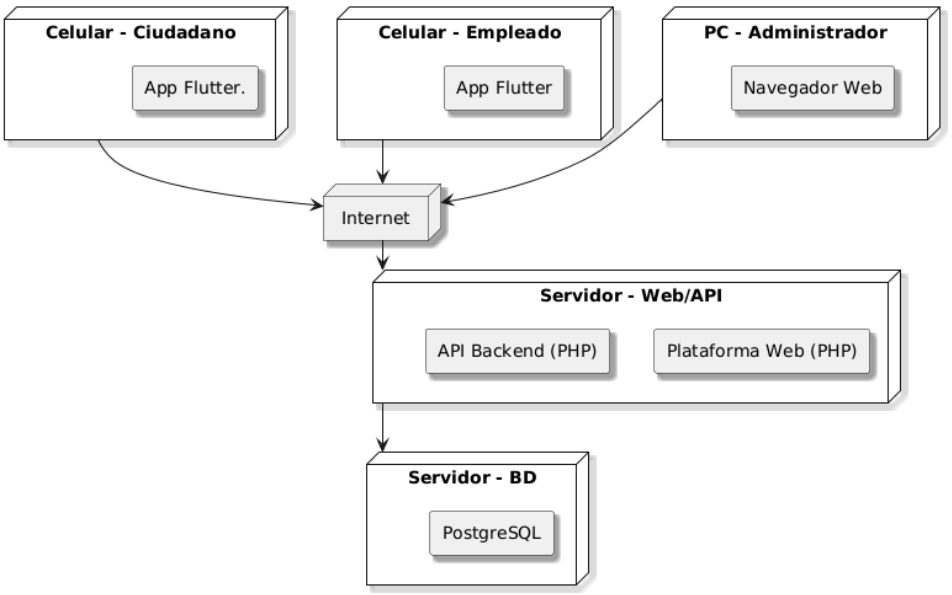
3.4. Vista de Procesos

3.4.1. Diagrama de Procesos del Sistema



3.5. Vista de Despliegue

3.5.1. Diagrama de despliegue



El siguiente diagrama de despliegue muestra cómo ciudadanos, empleados y el administrador acceden al sistema desde sus respectivos dispositivos a través de Internet. La app Flutter es utilizada en celulares por ciudadanos y empleados, mientras que el administrador accede desde un navegador web. Todos se conectan al servidor principal que aloja la plataforma web y la API Backend en PHP, los cuales interactúan con una base de datos PostgreSQL para el almacenamiento y gestión de la información.

4. ATRIBUTOS DE CALIDAD DEL SOFTWARE

4.1. Escenario de Funcionalidad

Escenario: Registro y gestión de incidencias

- Situación: Un ciudadano desea reportar una incidencia desde la app móvil.
- Actor: Ciudadano
- Evento disparador: El ciudadano selecciona “Reportar incidencia”, llena el formulario y lo envía.
- Respuesta esperada: El sistema recibe la información y registra correctamente la incidencia para su posterior gestión por el administrador.

4.2. Escenario de Usabilidad

Escenario: Interacción intuitiva en las interfaces de usuario

- Situación: Un ciudadano usa la app móvil por primera vez.
- Actor: Ciudadano
- Evento disparador: Ingreso a la app e inicio del proceso de reporte.
- Respuesta esperada: La interfaz permite navegar y completar el proceso sin dificultad, gracias a su diseño claro y accesible.

4.3. Escenario de Confiabilidad

Escenario: Acceso estable al sistema durante su uso

- Situación: Un empleado accede al sistema durante su jornada laboral.
- Actor: Empleado
- Evento disparador: El empleado abre la app y consulta sus incidencias.
- Respuesta esperada: El sistema se encuentra disponible, muestra la información esperada y no presenta fallos durante la interacción.

4.4. Escenario de Rendimiento

Escenario: Consulta fluida de incidencias por parte del administrador

- Situación: El administrador accede al dashboard desde la plataforma web.
- Actor: Administrador
- Evento disparador: Visualiza la lista de incidencias reportadas.
- Respuesta esperada: El sistema carga y muestra los datos de manera eficiente, permitiendo su revisión y gestión sin demoras notorias.

4.5. Escenario de Mantenibilidad

Escenario: Actualización o mejora del sistema

- Situación: Se requiere realizar cambios en el backend o añadir nuevas funcionalidades.
- Actor: Equipo de desarrollo
- Evento disparador: Implementación de mejoras o correcciones.
- Respuesta esperada: La estructura modular del sistema facilita las modificaciones sin generar impactos negativos en el resto de la plataforma.

4.6. Otros Escenarios

Escenario de Escalabilidad:

- Situación: Aumenta la cantidad de usuarios o de incidencias registradas.
- Respuesta esperada: El sistema continúa operando correctamente gracias a su diseño modular y uso de tecnologías que permiten una expansión progresiva.