Universidad Nacional Autónoma de México
Facultad de Ciencias
Ciencias de la Computación
Semestre 2025-2
Ingeniería de Software

Documentación SISREP: Releases v1.1.0

Bernal Esquivel Mariana Morayma	320298787
Comas Castañeda Mauricio Santiago	320215988
Jacome Delgado Alejandro	320011704
Jímenez Sánchez Emma Alicia	320046155
Peña Villegas Diego Eduardo	321260682
Robledo Ramírez Isaac	320140655

El Consejo del Backend

Karla Socorro García Alcántara	Profesor
Dulce Julieta Mora Hernández	Ayudante
Luis Gerardo Bernabé Gómez	Ayud. Lab.
Brenda Ayala Flores	Ayudante
Diana Laura Luciano Paniagua	Ayudante

26 de marzo de 2025

1. Objetivo del proyecto

Crear un sistema de software cuyos objetivos sean los siguientes:

- Automatizar la recolección de datos relacionados con los accidentes o obstáculos viales (fecha, hora, lugar, tipo de accidente, etc.) y manejar dicha información para presentarla a manera de mapa para cualquier usuario.
- El sistema tendrá roles de usuario con y sin sesión y administradores.
- Los usuarios no registrados, solo podrán ver el mapa y los incidentes, y claro registrarse.
- Los usuarios registrados pueden también ver el mapa, y además integrar al sistema nuevos accidentes, reportar hoyos, baches o cualquier dificultad que se presente en el camino, deben ser capaces de registrarse con un usuario, correo y contraseña propios y iniciar sesión.
- Los administradores de zona sumado a lo mencionado en el rol anterior, podrán modificar el status de los incidentes, ya sea que estén reportado, en revisión y resuelto.

2. Alcance

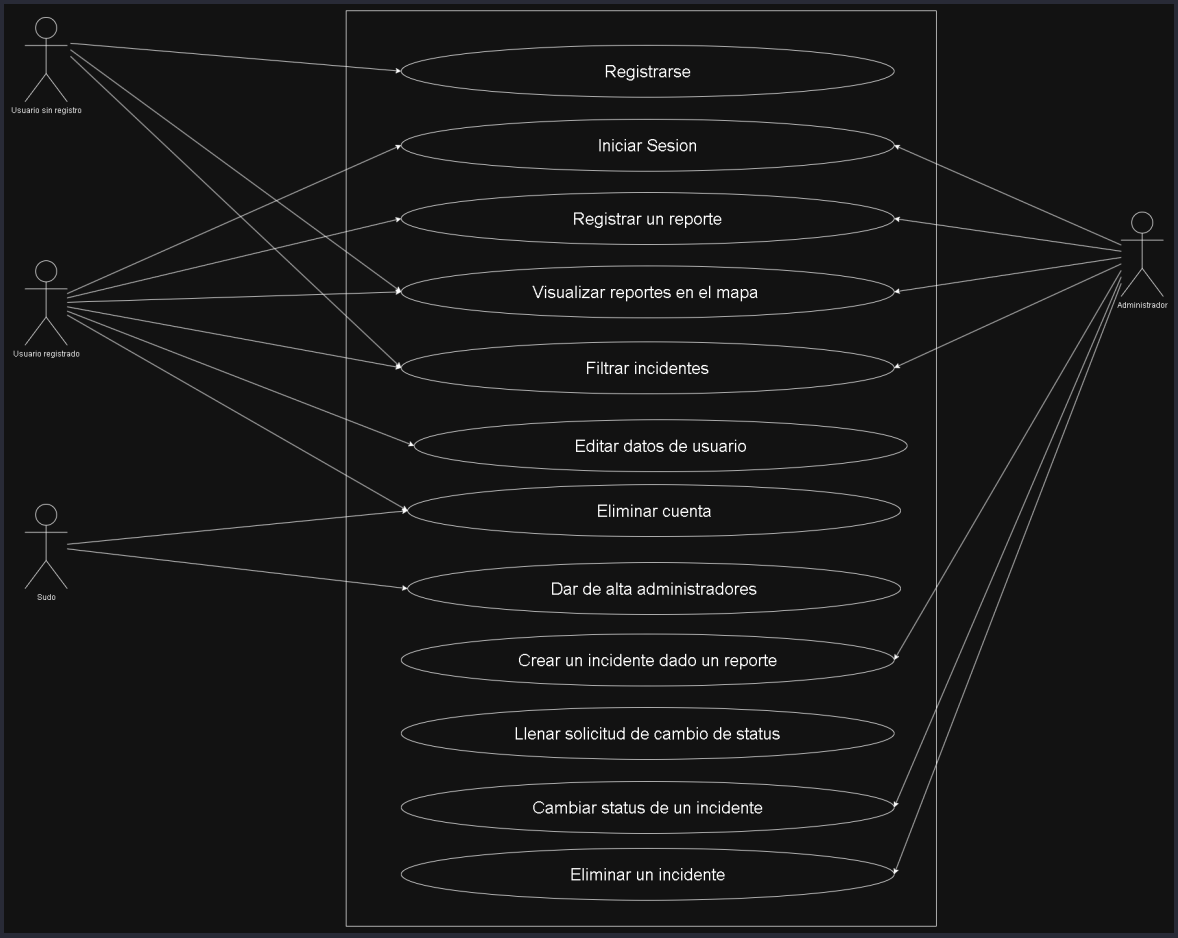
El Sistema de Reporte de Incidentes Urbanos le dará a los ciudadanos una forma fácil y rápida de reportar incidentes que posiblemente aún no sean detectados por las autoridades locales, lo que permitirá que las autoridades correspondientes se hagan cargo de atender y solucionar los reportes de forma mas rápida.

3. Planteamiento de necesidades

Se implementó un sistema que cumpliera con las funciones de dar de alta un reporte de accidente vial, donde los usuarios registrados van a poder dar de alta en el sistema mediante un reporte, registrando los datos necesarios del accidente o del problema vial, donde los usuarios registrados pueden comentar el status del reporte. Es necesario contar con usuarios registrados con un correo, usuario y contraseña para tener acceso al sistema. Para el desarrollo del sistema es necesario contar con las funcionalidades esenciales descritas en el documento.

4. Identificación de las funcionalidades del producto de software

Para simplificar este punto, se puede apreciar el Diagrama General de Casos de Uso, donde podemos visualizar las funcionalidades.



5. Glosario de términos

Término	Definición
SISREP	Sistema de Reporte de Incidentes Urbanos.
IDE	Interfaz de Desarrollo de Entorno.
FrontEnd	Parte del proyecto donde los usuarios interactuan.
Back-end	La lógica del proyecto, no visible para el usuario.
Base de Datos	Estructura para almacenar datos.
Dependencias	Relaciones entre componentes para funcionar correctamente.
API	Aplication Programming Interface. Interfaz que permite la comunicación entre distintos sistemas o componentes.
Endpoint	Ubicación digital donde una API recibe solicitudes de recursos en su servidor.

6. Requerimientos del sistema

▷ Requerimientos funcionales

RF1: Autenticación de Usuarios

- La aplicación permite el acceso de usuarios mediante credenciales únicas: usuario y contraseña.
- Hay autentificación para acceder como usuario (ciudadano) o administrador.

RF2: Gestión de Usuarios

- Los usuarios podrán crear, editar y eliminar su perfil.
- Los administradores podrán eliminar las cuentas de los usuarios y ver sus datos. A su vez estos tendrán su propia cuenta.

RF3: Visualización de Incidentes

- Los incidentes tendrán 3 estados: reportado, en revisión y resueltos
- Los incidentes se verán en el mapa interactivo según su ubicación, estos tendrán un icono correspondiente a su tipo.
- Los usuarios pueden hacer clic en el ícono de un incidente y ver detalles sobre este.

RF4: Gestión de Incidentes

- El sistema debe permitir a los usuarios registrar un incidente, para esto se pedirá la siguiente información obligatoriamente:
  - Tipo del incidente, esta será una selección de los tipos predeterminados: baches, luminarias descompuestas, obstáculos en la vía pública, accidentes automovilísticos, otro.
  - Descripción del incidente.
  - Fotos de prueba del incidente.
  - Ubicación, esta se podrá registrar manualmente (de forma escrita o seleccionando el punto en el mapa), o utilizando la ubicación en tiempo actual del usuario.
- Usuarios que mande pruebas de dicho incidente esté resuelto, el administrador podrá decidir, con ese o más reportes si cambiar el estado del incidente como resuelto.
- Los incidentes se borrarán después de cierto tiempo cuando ya estén en el estado resuelto.
- Los administradores deben recibir notificaciones de nuevos incidentes registrados.

RF5: Filtro de incidentes

- Los usuarios podrán filtrar los incidentes en el mapa por su estado (Reportado, en revisión, resuelto) o por su tipo, fecha o estado.

- Requerimientos no funcionales

- **Seguridad:**
  Definición de protocolo http, acceso restringido a la base de datos, protección de la rama main, los usuarios que reporten incidentes deben estar obligatoriamente registrados en la aplicación, solo administrador puede eliminar y cambiar el estado de cualquier incidente.
- **Usabilidad:**
  Regulación de contenido, interfaz intuitiva y amigable con el usuario.
- **Rendimineto:**
  El sistema deberá soportar al menos 70 usuarios.
  Los datos de incidentes son actualizados en tiempo real.
  El procesado y subida de incidentes tarda 5 segundos en promedio.
  Filtrar incidentes por tipo o tiempo demora aproximadamente de 10 segundos.
  Encontrar incidentes cercanos tarda alrededor de de 20 segundos.
- **Disponibilidad:**
  El sistema deberá estar disponible el 99 % del tiempo.

7. Ambiente de implementación

Release	Repositorio	Link
v1.1.0	Front-end	Repositorio-frontend
v1.1.0	Back-end	Repositorio-backend

Concepto	Herramienta	Versión
Lenguaje backend principal	Kotlin	1.9.25
Lenguaje backend auxiliar	Java	JDK-21
Diagramador UML	Drawio	26.1.0
Biblioteca para la aplicación web	React	19.0.0
Entorno de ejecución frontend	Node.js	22.14.0
Gestion de dependencias frontend	npm	11.2.0
IDE lógico del negocio	SpringBoot	3.4.4
IDE interfaz de usuario	IntelliJ	2024.3.5
Manejador de base de datos	PostgreSQL	17.4.0
Herramienta de Bases de Datos (recomendable)	DBeaver	25.0.1
Herramienta para construcción	Bootstrap	5.3.3
Control versiones	Git	2.49.0

8. Ejecución

Para nuestra base de datos, podemos usar un contenedor de Docker.

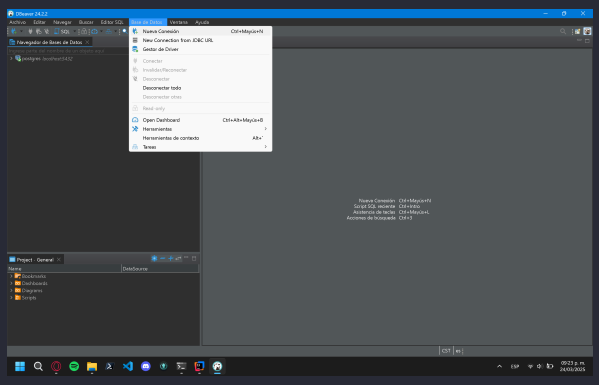
Primero hay que instalar nuestras herramientas, para ello, haremos el ejemplo en distribuciones basadas en debian (los comentarios serán `# Comentario`, no escribirlos):

```
$ sudo apt update
$ sudo apt install open-jdk-21-jdk # Instalamos jdk
$ java --version # Comprobamos la instalación
$ sudo apt install maven # Instalamos maven
```

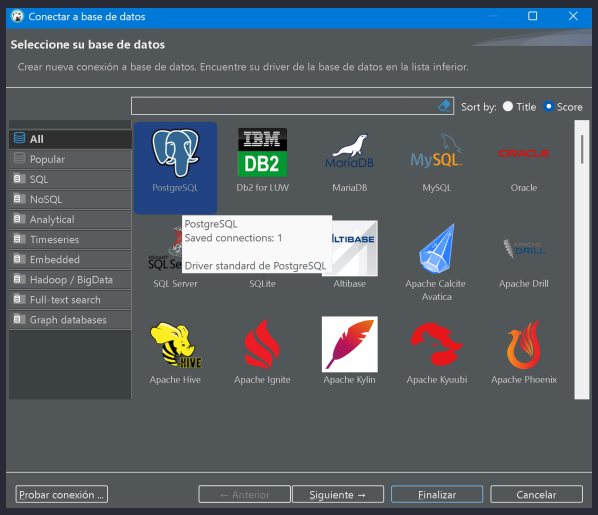
```
$ mvn --version # Comprobamos la instalación
$ sudo apt install apt-transport-https ca-certificates curl software-properties-common # Requisitos para apt con HHTP
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add - # Clave de GPC
$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable" # Agregamos el repositorio de docker para apt
$ sudo apt update
$ sudo apt install docker-ce # Instalamos docker
$ sudo systemctl status docker # Comprobamos el servicio
$ sudo docker run hello-world # Comprobamos la instalción
$ sudo apt update
$ docker pull postgres
$ docker run -d --name Sisrep -e POSTGRES_USER=<usuario> POSTGRES_PASSWORD=<contraseña> -p 5432:5432 postgres # Llenar usuario y contraseña a preferencia, se usará después. NO OLVIDAR.
$ docker ps -a # Averiguamos el ID del contenedor
$ docker start <ID> # Empezamos el container
$ docker stop <ID> # Cuando querramos detenerla
```

Si se quiere usar DBeaver, descargar en: dbeaver.io/download y seguir las instrucciones de descarga.

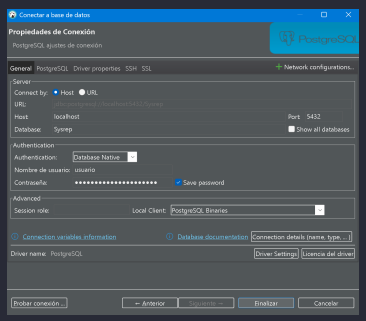
Ya instalada, al abrirse, en la barra de herramientas (aparecerá en la parte superior `Archivo Editar Navegar ...`) y cliqueamos en `Base De Datos`, seguido de Nueva conexión.



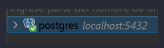
Buscamos PostgreSQL (el logo es un elefante azul).



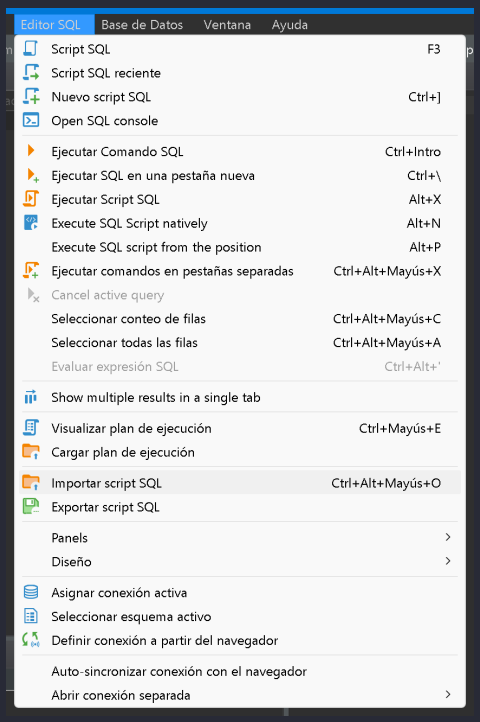
Ahora cambiamos el campo en Database por `Sisrep`, el campo en Nombre de Usuario por el usuario antes escrito al igual que la contraseña. No es necesario modificar nada más. Cliquear en finalizar



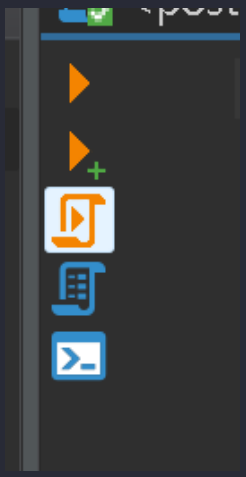
Finalmente, verificar que la conexión se haya establecido de manera correcta viendo el mismo logo de PostgreSQL arriba a la izquierda, junto con una palomita (verde).



A éste en la misma barra de herramientas (a la izquierda de Base de Datos) buscaremos EditorSQL, y seleccionamos Importar Script SQL, colocaremos el `DDL.sql` que encontramos en el directorio `database/` (o podemos seleccioanr Nuevo Script SQL y copiar y pegar el contenido)



En la parte izquierda, y cliqueamos en el icono de un rollo naranja con un triangulo para ya tener una estructura en la Base de Datos.



Prosiguiendo con la instalación:

```
$ sudo apt update
$ sudo apt install nodejs npm -y # Instalamos Nodejs y NPM
$ node -v # Verificamos su instalación
$ npm -v # Verificamos instalación
$ npm install -g typescript
$ npm install vite --save-dev
$ npm install bootstrap
$ npm install react-router-dom
$ npm install sweetalert2
$ npm install react-bootstrap-icons
```

Hay que asegurarse que la Base de datos se encuentra activa y corriendo.

Después de ello, verificar en `src/main/aplication.properties` que tanto el puerto, y nombre de la base de datos, como usuario y contraseña correspondan a las ingresadas.

Ahora, se ejecutará el proyecto:

Ya solo queda crear una carpeta para alojar ambos repositorios.

Creamos la carpeta y dentro de ella ejecutamos

```
$ git clone https://github.com/ingenieria-software-7009-2025-2/frontendcouncil-
    frontend.git
$ cd frontendcouncil-frontend
$ npm install
$ npm run dev
$ cd ..
git clone https://github.com/ingenieria-software-7009-2025-2/backendcouncil-api.git
$ cd backendcouncil-api
$ mvn clean compile
$ mvn install
```

Podemos evitar las últimas dos líneas y usar IntelliJ, y solo pulsar Run.

Lo cual debería correr sin problemas.

Finalmente queda abrir el navegador y usar `http://localhost:8080/`.

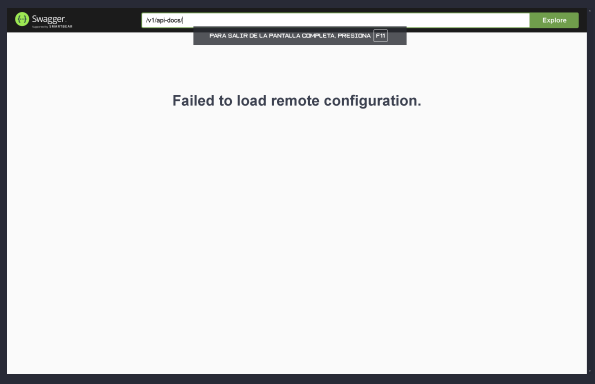
Para acceder a la documentación de la API con OpenAPI ir a la sección siguiente.

9. Definición de endpoints

Para documentar los endpoints, utilizamos `OPEN-API v3.0.0`, que al correr el proyecto, podemos checar visitando en el navegador de preferencia

```
http://localhost:8080/docs-api.html
```

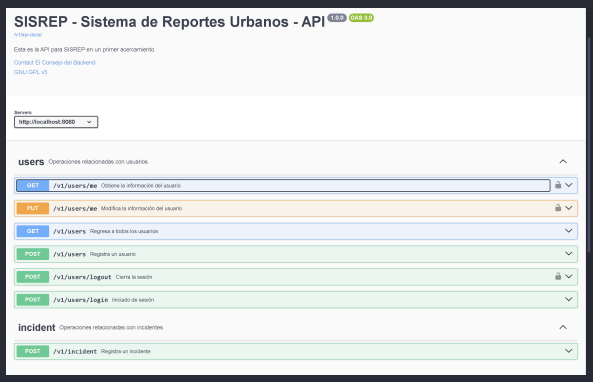
Lo que nos redirigirá a la siguiente página:



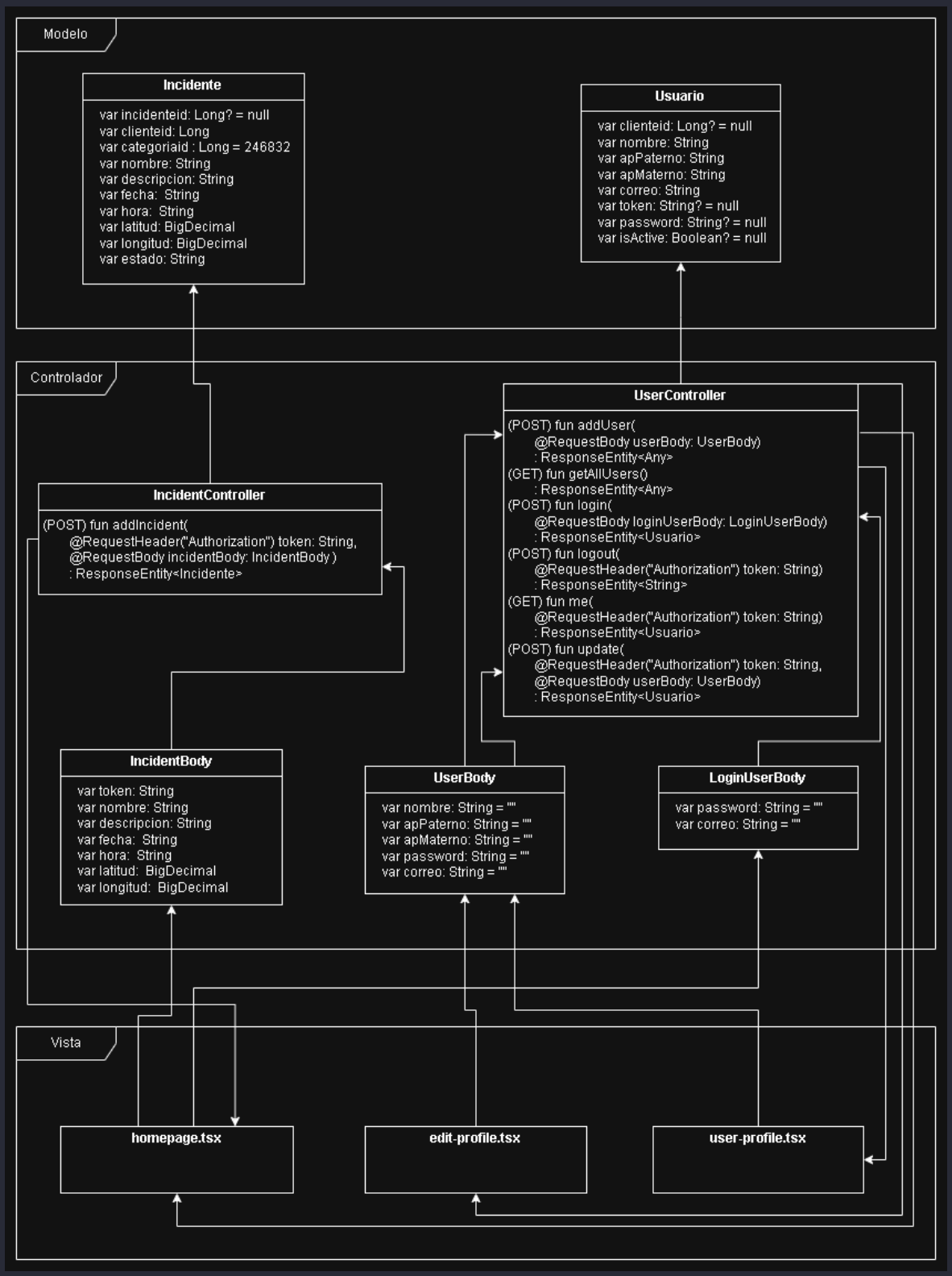
Donde en la barra de búsqueda ubicada en la parte superior, colocaremos:

```
/v1/api-docs/
```

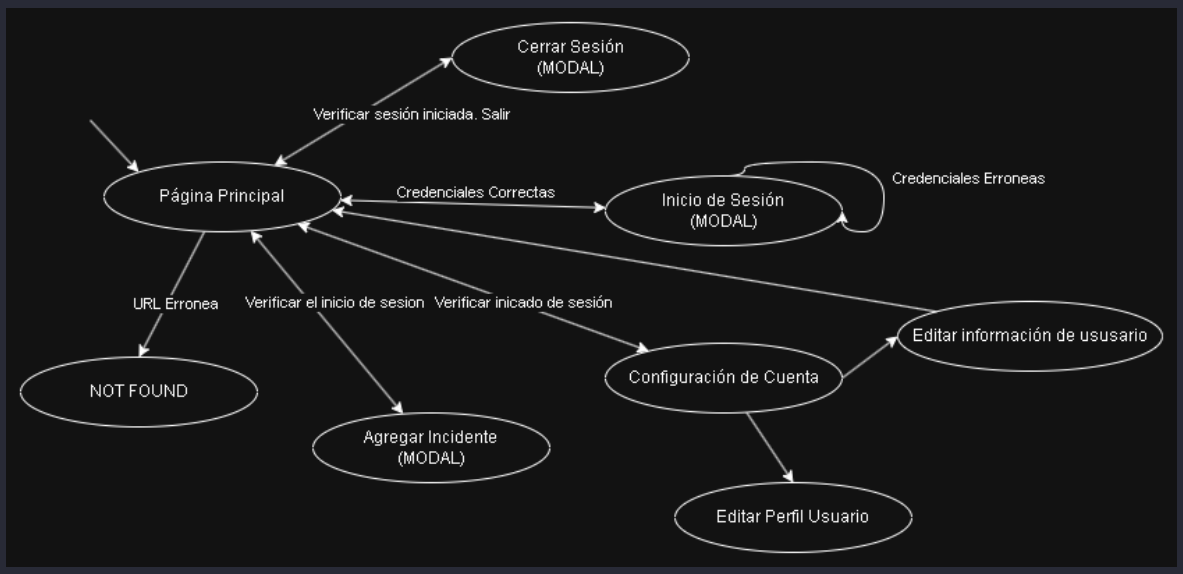
Lo cual nos llevará a la documentación de los endpoints hasta le momento.



10. Diagrama de clases del sistema

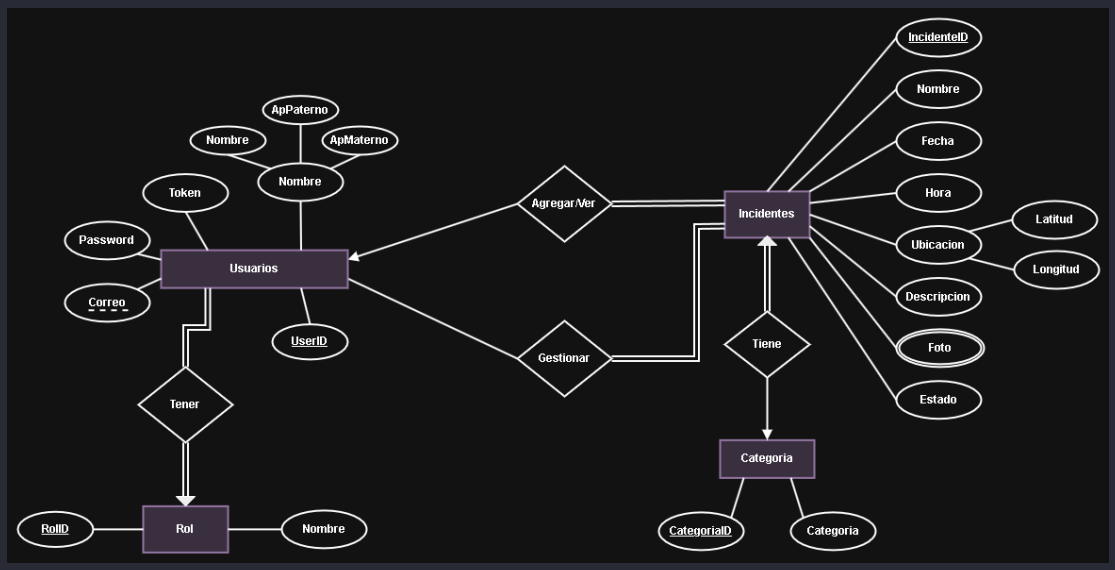


11. Diagrama de Navegación del sistema

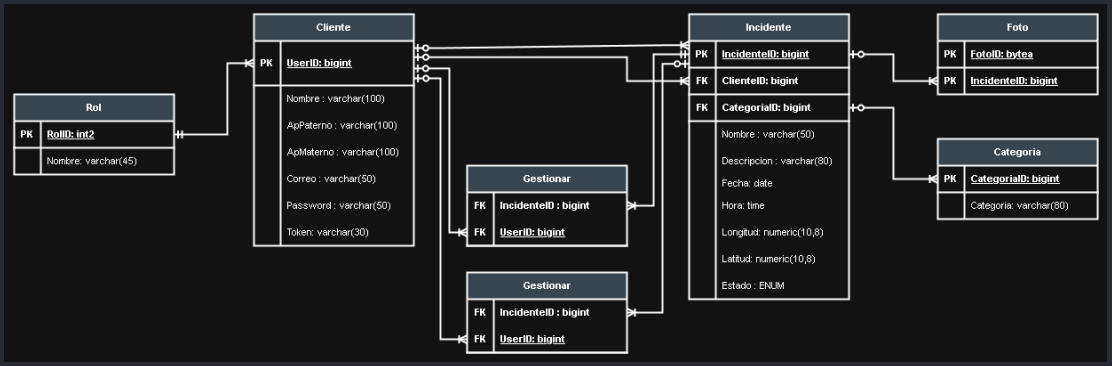


12. Diagramas para el diseño de la Base de Datos usada en el sistema

- ## Diagrama del Modelo Entidad-Relación



- ## Diagrama del Modelo Relacional



13. Integración y pruebas del sistema

La integración de los casos de uso se llevó a cabo utilizando el versionador de código Github. En esta plataforma permite el uso de la versión web para poder almacenar código en la nube, como la opción de mostrarlo localmente en un servidor. En este caso, se estuvo usando las dos versiones ya que se trabajo de manera colaborativa, lo que permitio que el proyecto pudiera ser revisado por el equipo durante el desarrollo.

Durante la etapa de pruebas, se utilizaron los estándares previamente establecidos por la institución, los cuales garantizan el correcto funcionamiento de las plataformas existentes.

Los casos de prueba fueron diseñados para poder verificar las especificaciones funcionales se implementaron correctamente, lo que serían pruebas funcionales. No obstante, se evaluaron otras propiedades esenciales como usabilidad, desempeño, fiabilidad, entre otras.

Las pruebas realizadas se centraron en los siguientes aspectos:

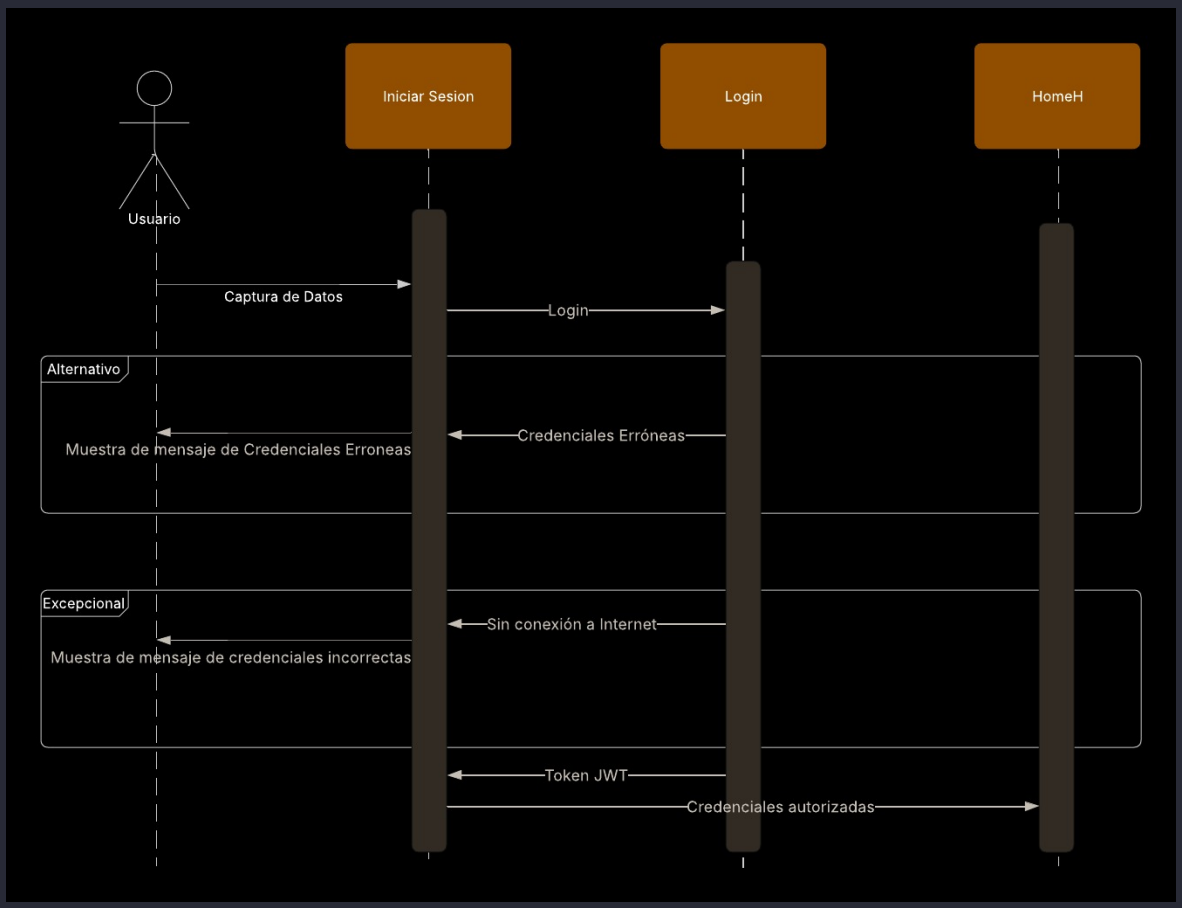
- Funcionalidad: Validar que el sistema cumpla con los requisitos especificados.
- Usabilidad: Evaluar la experiencia de usuario y la facilidad de uso del sistema.
- Eficiencia: Comprobar el rendimiento del sistema bajo diferentes condiciones.
- Seguridad: Garantizar la integridad y confidencialidad de la información.

14. Casos de Uso Normales

A partir del diagrama general de casos de uso, se realiza una explicación amplia con prototipos de interfaz y casos de prueba de cada uno de ellos. Con el objetivo de mostrar la información de una manera digerible.

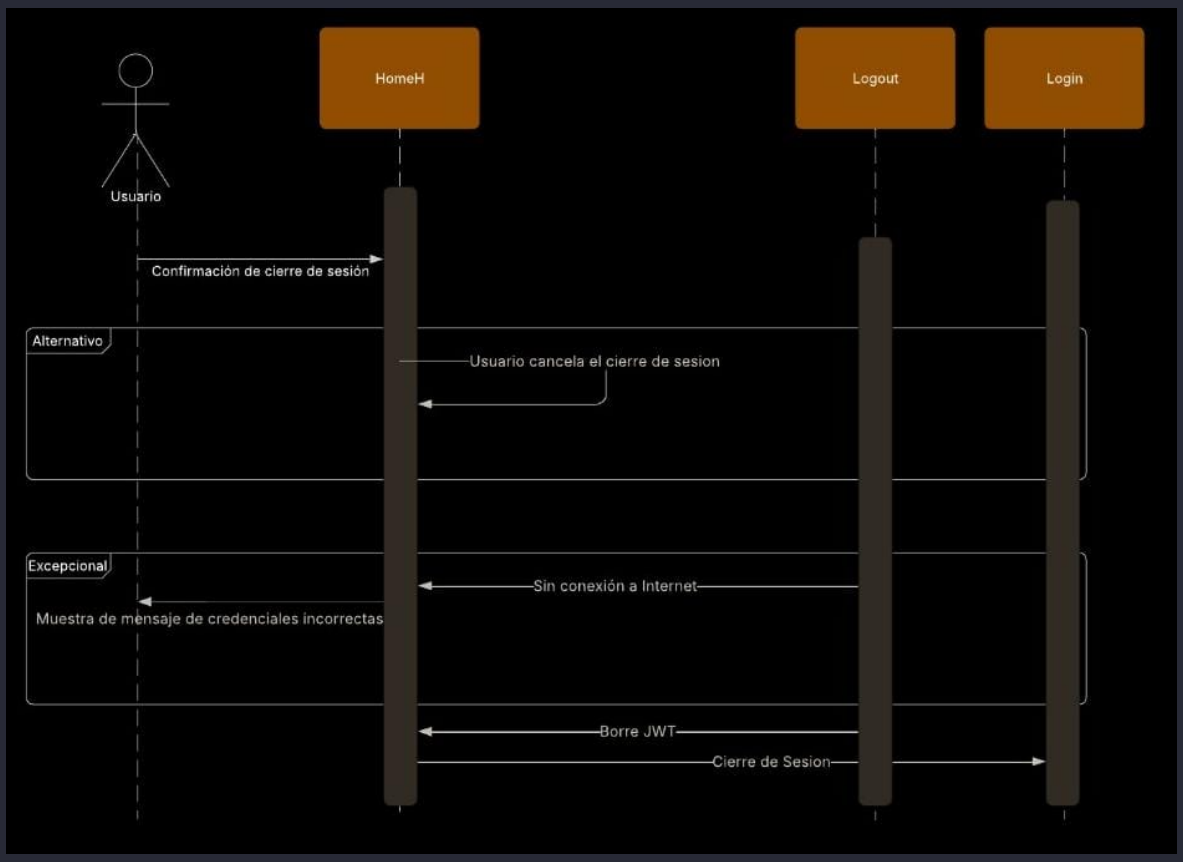
CU1: Inicio de Sesión

- Descripción: Permite darle acceso al juez a la búsqueda de los documentales que se encuentran en competencia.
- Precondiciones: Contar con un usuario y contraseña e ingresarlos a los campos correspondientes.
- El sistema le permite al usuario tener acceso a la búsqueda de cualquier documental.



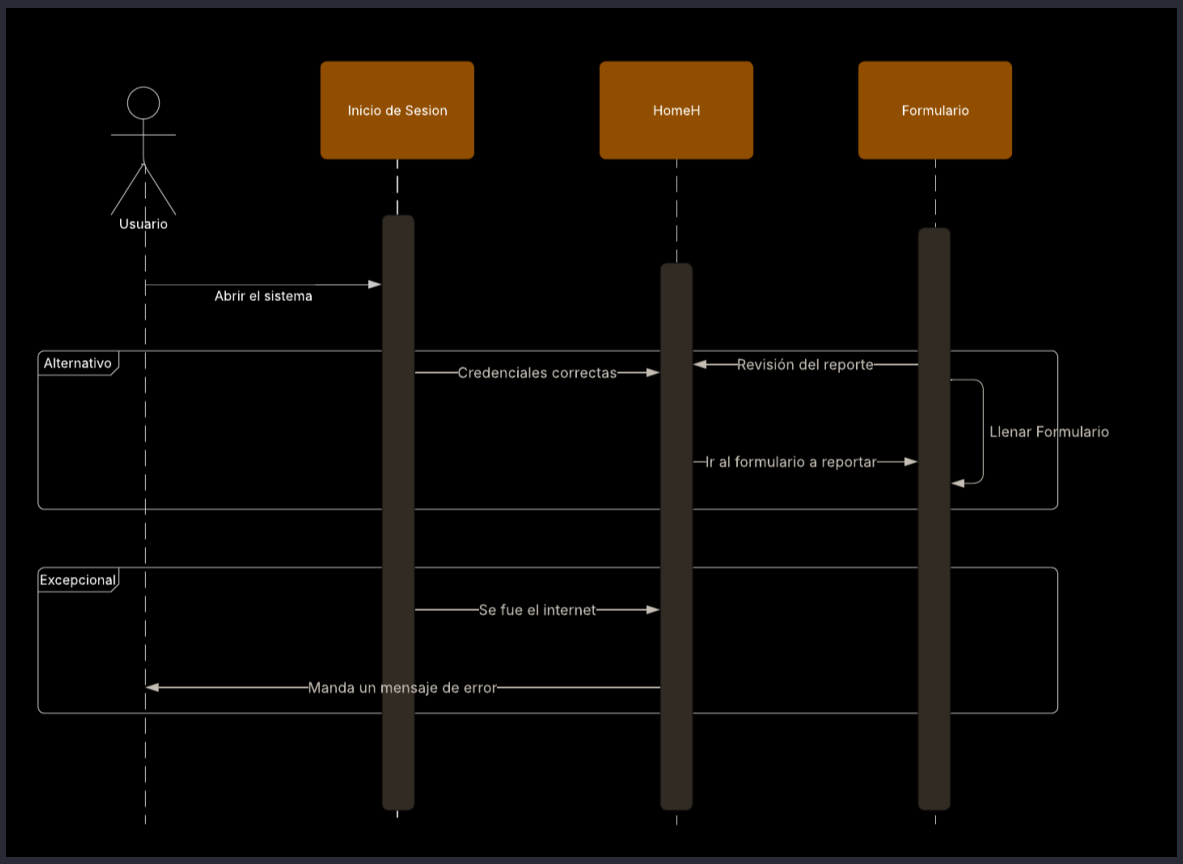
CU2: Cerrar Sesión

- Descripción: Permite terminar y caducar las credenciales con las que tiene acceso al sistema.
- Preecondiciones: Tener una sesión abierta.
- Poscondiciones: El sistema ha cerrado sesión y el usuario no tiene acceso a las funcionalidades del sistema.



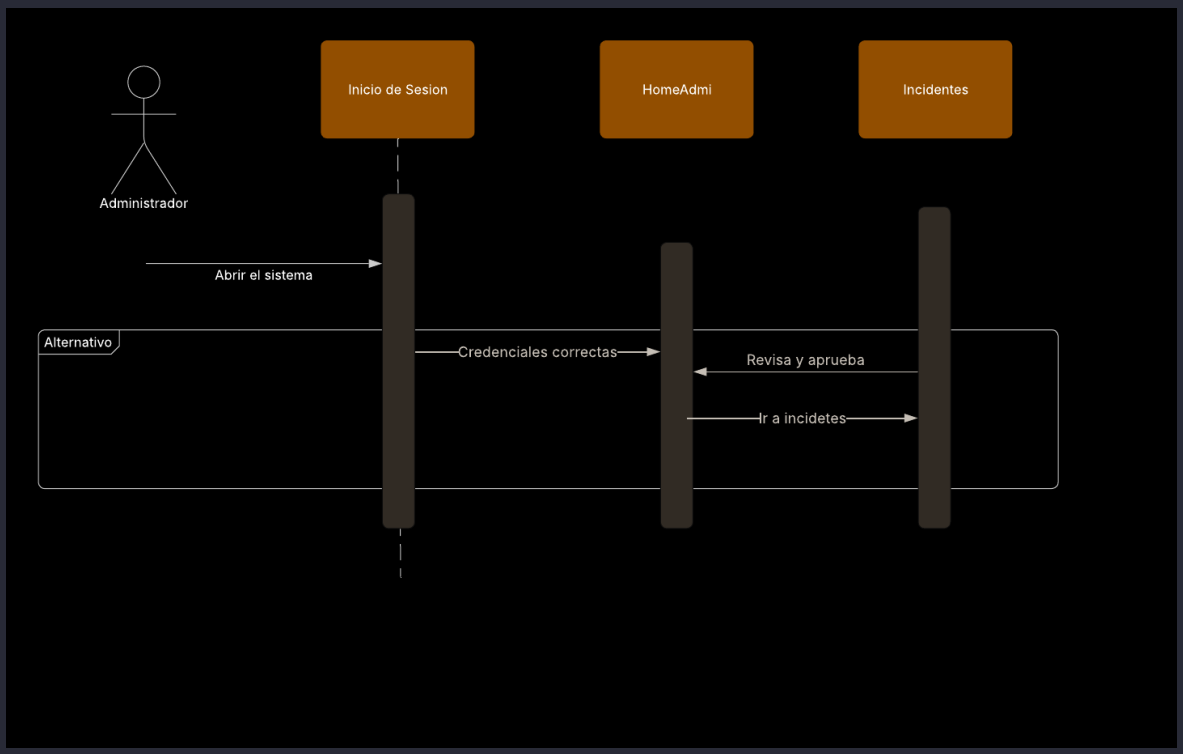
CU3: Reporte de Accidentes:

- Descripción: Llenar la información requerida sobre el accidente donde se deberá de indicar correctamente la dirección del accidente para poder agregarlo correctamente al sistema.
- Precondiciones: Entrar al sistema con un usuario y contraseña. Y llenar correctamente el formulario del reporte.
- Poscondiciones: El incidente deberá de aparecer en el mapa.



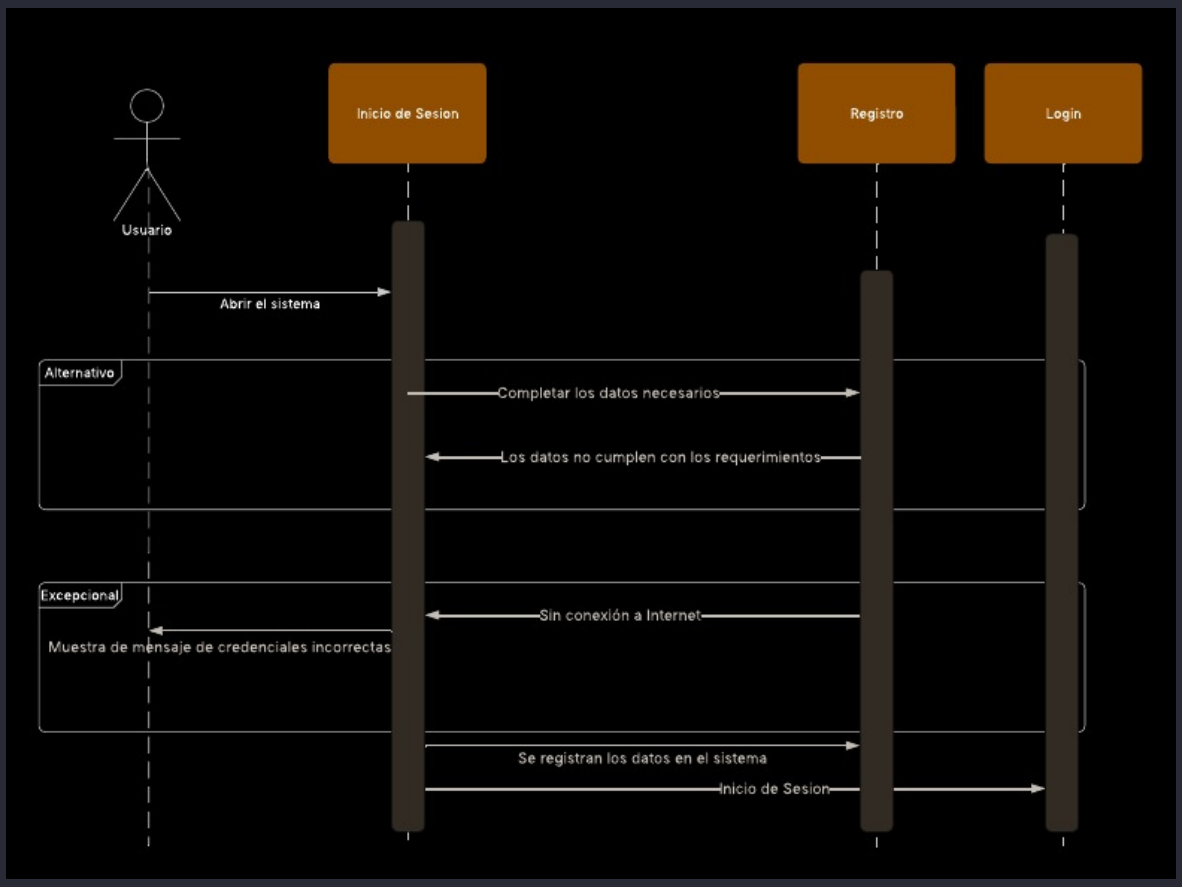
CU4: Gestión y Seguimiento de Reportes:

- Descripción: Poder entrar al sistema como administrador para poder acceder a la sección correspondiente de los reportes, donde se puede filtrar los reportes por grado de importancia y se pueda actualizar el estatus del incidente.
- Precondiciones: Tener un cuenta como administrador y estar en la sección de incidentes.
- Poscondiciones: El estatus del incidente queda actualizado correctamente en el sistema y se envía una notificación a las partes interesadas si el estatus del incidente cambia a “Resuelto” o “En proceso”.



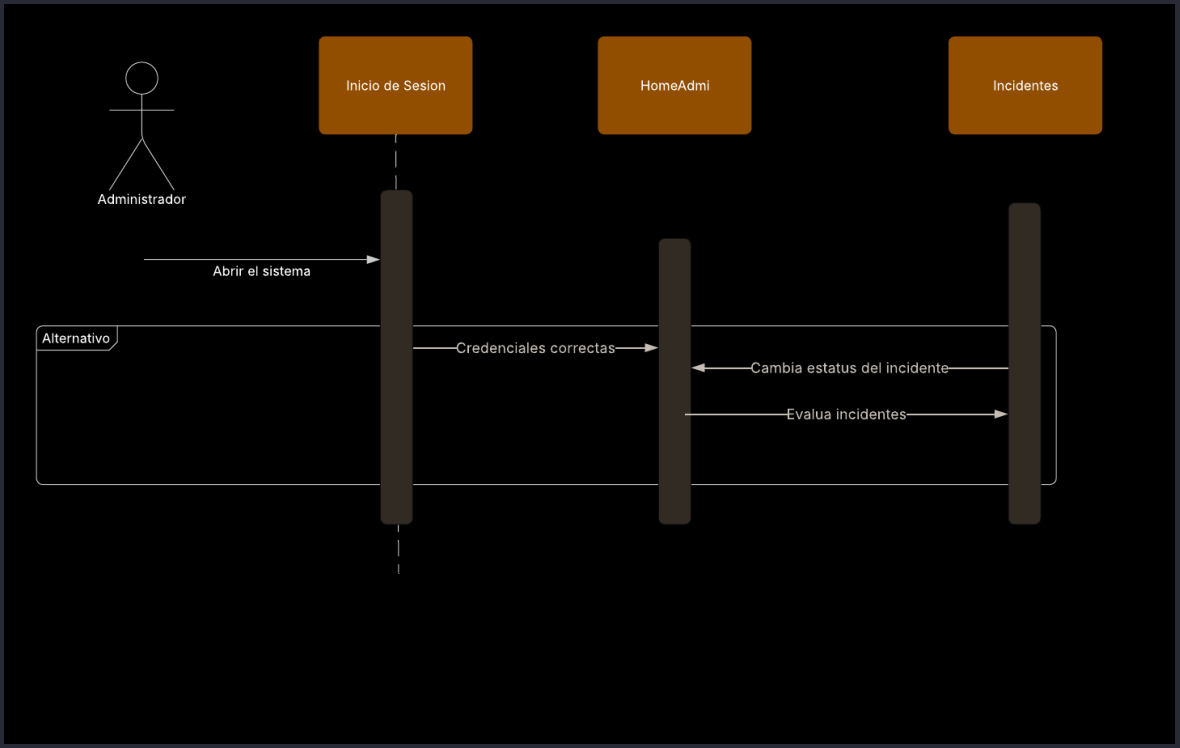
CU5: Registro de cuenta

- Descripción: Los usuarios nuevos al sistema deberán de registrarse con un correo, usuario y contraseña para poder autentificar los y dar autorización que entren al sistema.
- Preecondiciones: Poder registrarse en el sistema con un correo, usuario y contraseña.
- Poscondiciones: Poder entrar al sistema.



CU6: Cambio de estatus de incidente por un administrador

- Descripción: El sistema notifica a usuarios cercanos, o los mismos usuarios reportan el cambio de status y un administrador decide si cambiarlo o no. O bien, el administrador puede cambiar el status de los incidentes.
- Preecondiciones: Un accidente ya existente, una cuenta administrador que cambie el status.
- Poscondiciones: Cambio de status del incidente.



15. Casos de Uso Alternativos

- **CU1Alternativo: Datos Incorrectos**
  - Descripción: Este escenario ocurre cuando el usuario introduce datos incorrectos.
  - Acciones para llevar acabo:
    - Validar los datos ingresados en cada campo.
    - Mostrar un mensaje de error para cuando los datos ingresados son incorrectos.
- **CU2Alternativo: Errores de Validación**
  - Descripción: Ocurre cuando los datos ingresados no cumplen con las reglas de validación.
  - Acciones para llevar acabo:
    - Comprobar la integridad y formato de los datos ingresados.
    - Mostrar mensajes específicos por cada error de validación detectado.

16. Casos de Uso Extraordinarios

- **CUExt1: Desconexión del servidor**
  - Descripción: El servidor se desconecta o deja de estar disponible durante un periodo de tiempo determinado cuando el usuario lo estaba ocupando.
  - Acciones a llevar acabo:

- Detectar la pérdida de conexión y notificar al usuario con un mensaje de error.
- Intentar la reconexión automática en intervalos regulares.
- Tener planes de contigencias en servidores de respaldo o en contenedores.

- **CUExt2:Intentos repetidos de sesion:**
  - Descripción: Se presenta cuando un usuario intenta en repetidas ocasiones entrar al sistema con datos inválidos o incorrectos que no son admisibles.
  - Acciones a llevar acabo:
    - Avisar al usuario el número de intentos fallidos para entrar al sistema.
    - Bloquear temporalmente la cuenta y avisar al usuario el motivo del bloqueo de la cuenta y por cuanto tiempo estará bloqueada.
    - Enviar instrucciones de como se puede desbloquear la cuenta al usuario.
    - Registrar el evento del bloqueo de la cuenta con fines de seguridad.
- **CUExt3:Fallo de equipo o sistema no identificado**
  - Descripción: Se presenta cuando el sistema falla por un error inesperado.
  - Acciones a llevar acabo: Contactar con los programadores, suspender el servicio lo antes posible para evitar problemas mayores, y actuar según el plan de contingencia (como reiniciar), o solucionar el sistema con un hotfix hasta que se encuentre una solución que contemple de la mejor manera la problemática, contactar a los usuarios, para que si alguno tuvo un problema de pérdida de datos, o un cambio de rol, etc, se pueda solucionar.