

Entregable API Rest Siniestros

Descripción

Proyecto API Rest, desarrollado en .Net 8, el cual presenta 2 endpoints, uno de creación y otro de consulta de siniestros viales.

- Arquitectura implementada: Clean Architecture
- Principios: SOLID, DDD, DRY, KISS, YAGNI
- Patrones: Mediator, CQRS

Explicación decisiones arquitectónicas

El proyecto se desarrolló usando un estilo arquitectónico como Clean Architecture, con una arquitectura basada en microservicios, la cual gestiona el dominio de "Siniestros", encargándose así de la creación y consulta de los mismos.

Se crearon 4 proyectos:

SiniestrosViales.Api: Punto de entrada de las peticiones http

SiniestrosViales.Application: Encapsula la lógica del negocio

SiniestrosViales.Domain: Contiene la definición del dominio

SiniestrosViales.Infrastructure: En esta capa se da la persistencia de los datos

Se concibe como un microservicio, porque permite tener la responsabilidad única de gestionar el siniestro, con una base de datos independiente y una arquitectura escalable. Se usó clean architecture, porque se enfoca en el dominio y lo aísla de capas como la infraestructura; permitiendo cambiarla sin afectar el mismo dominio, permitiendo escalar el proyecto a demanda.

Para la persistencia de los datos se usó Entity Framework Core, un ORM que es capaz de dar gestión completa de los datos, a través de sus diferentes funciones; facilitando la persistencia y dando robustez a la implementación.

Se usó CodeFirst, es decir se desarrolló el modelo de siniestros y después se creó la tabla en base de datos por medio de migraciones.

El proyecto implementa principios SOLID, como SRP; en donde a través de CQRS delegamos responsabilidades únicas de lectura y escritura al dominio de Siniestros. Por otro lado, hacemos uso del patrón repository y DIP, en donde centralizamos la persistencia de los datos a través de repositorios y a su vez, no dependemos en el proyecto de implementaciones, si no de abstracciones; esto lo vemos en el uso de interfaces y no de las implementaciones.

Se implementó un Middleware, para capturar todos los posibles errores y dar una respuesta general al usuario. De la misma manera, implementamos una clase que nos permitiera dar una respuesta con el mismo formato, abarcando si la petición es exitosa, un posible mensaje, y la posible data de respuesta.

Finalmente, implementamos un health check, para validar el estado de la base de datos; este lo encontramos en la ruta: <http://localhost:5007/health>

Modelo de Dominio: El dominio está centrado en la entidad Siniestro, que representa un evento vial y encapsula las reglas de negocio como la validación del número de vehículos y víctimas.

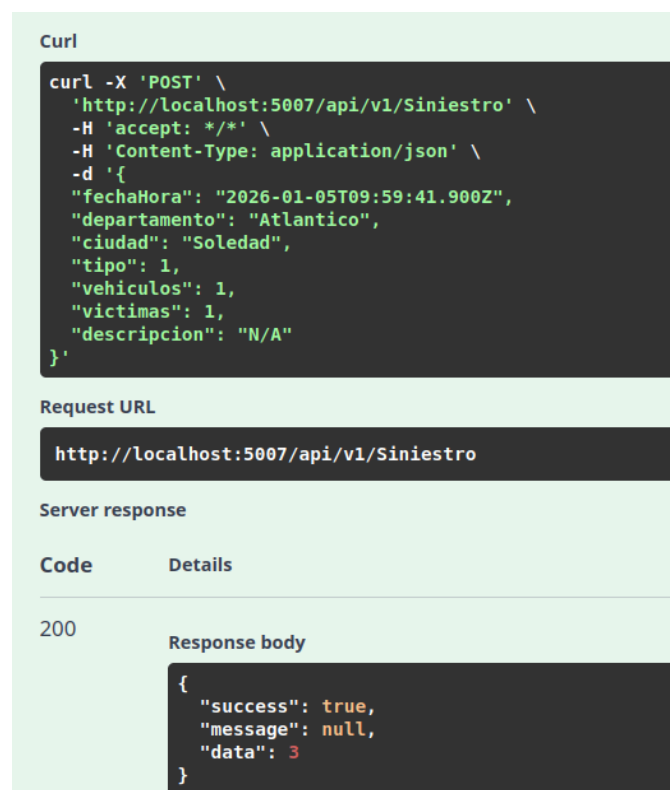
Pruebas Funcionales

Para la ejecución de pruebas se usó postman. Contamos con 2 endpoint como vemos a continuación:



Creación de Sinistro (POST)

Se ejecuta la creación de un siniestro con la siguiente información, el cual nos retorna el código del siniestro creado; que para este caso fue 3:



Consulta de Siniestros (GET)

Ejecutamos la consulta de siniestros.

CASO: Sin filtros, la cual por defecto nos trae los primeros 10 registros:

Curl

```
curl -X 'GET' \
  'http://localhost:5007/api/v1/Siniestro' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5007/api/v1/Siniestro
```

Server response

Code	Details
200	<div>Response body</div> <pre>"success": true, "message": null, "data": { "items": [{ "id": 1, "fechaHora": "2026-01-21T07:14:25.254", "departamento": "BOLIVAR", "ciudad": "Cartagena", "tipo": "Choque", "vehiculosInvolucrados": 2, "numeroVictimas": 3, "descripcion": "Choque en semaforo" }, { "id": 2, "fechaHora": "2026-01-11T08:14:25.254", "departamento": "CUNDINAMARCA", "ciudad": "Bogota", "tipo": "Choque", "vehiculosInvolucrados": 2, "numeroVictimas": 3, "descripcion": "Choque en semaforo" }, { "id": 3, "fechaHora": "2026-01-05T09:59:41.9".</pre>

CASO: Filtrando por departamento:

Curl

```
curl -X 'GET' \
'http://localhost:5007/api/v1/Siniestro?Departamento=bolivar' \
-H 'accept: */*'
```

Request URL

```
http://localhost:5007/api/v1/Siniestro?Departamento=bolivar
```

Server response

Code	Details
200	<p>Response body</p> <pre>{ "success": true, "message": null, "data": { "items": [{ "id": 1, "fechaHora": "2026-01-21T07:14:25.254", "departamento": "BOLIVAR", "ciudad": "Cartagena", "tipo": "Choque", "vehiculosInvolucrados": 2, "numeroVictimas": 3, "descripcion": "Choque en semaforo" }], "totalItems": 1, "page": 1, "pageSize": 10 } }</pre>

CASO: Filtrando por fechas:

Curl

```
curl -X 'GET' \
'http://localhost:5007/api/v1/Siniestro?FechaInicio=2026-01-10&FechaFin=2026-01-28' \
-H 'accept: */*'
```

Request URL

```
http://localhost:5007/api/v1/Siniestro?FechaInicio=2026-01-10&FechaFin=2026-01-28
```

Server response

Code	Details
200	<p>Response body</p> <pre>{ "success": true, "message": null, "data": { "items": [{ "id": 1, "fechaHora": "2026-01-21T07:14:25.254", "departamento": "BOLIVAR", "ciudad": "Cartagena", "tipo": "Choque", "vehiculosInvolucrados": 2, "numeroVictimas": 3, "descripcion": "Choque en semaforo" }, { "id": 2, "fechaHora": "2026-01-11T08:14:25.254", "departamento": "CUNDINAMARCA", "ciudad": "Bogota", "tipo": "Choque", "vehiculosInvolucrados": 2, "numeroVictimas": 3, "descripcion": "Choque en semaforo" }], "totalItems": 2, "page": 1, "pageSize": 10 } }</pre>