Análisis del Proyecto Integrador

Identificación del endpoint:

- **Endpoint REST consumido:**
    - **GET /api/actors**
      El servicio REST correspondiente se encuentra implementado en el archivo actorRoutes.js, donde la lógica de consulta se ejecuta a través de las funciones definidas en el controlador actorController.js
- **Justificación de la elección del endpoint:**
    - Dado que el sistema requiere manejar datos relacionados con los actores de películas, el endpoint GET /api/actors se convierte en un punto clave para obtener información relevante sobre los actores que luego será utilizada en el análisis de estadísticas, facilitando operaciones como el cálculo de edad promedio o el actor con más roles.
    - Los datos obtenidos se pueden transformar y ampliar en el servicio SOAP para ofrecer operaciones más especializadas, como obtener estadísticas sobre los actores.
- **Descripción técnica del endpoint:**

Característica	Descripción
Método	GET
URL	/api/actors
Formato de respuesta	JSON: Un arreglo de objetos con propiedades actor_id, first_name y last_name
Tecnología de backend	Express con MySQL (consultado en actorController.js y configurado en db.js)

2. Diseño del Microservicio

Objetivo del microservicio:
Construir un microservicio basado en SOAP que ofrezca operaciones para consultar la información de actores y obtener estadísticas relacionadas, a partir de datos extraídos de la API REST existente.

Elección y justificación de la tecnología:

- **Tecnología elegida:** SOAP, definida mediante un contrato (WSDL).
- **Justificación:**

- El estándar SOAP permite definir un contrato preciso (en el archivo actor.wsdl), lo cual da claridad sobre la estructura y formato de la información.
- La integración a través de SOAP es robusta y adecuada para escenarios donde se requiere un contrato formal para el intercambio de datos.
- La implementación con Node.js y el paquete soap facilita la asociación entre las operaciones definidas (como getAllActors y getActorStats) y el consumo del endpoint REST.

Diagrama del flujo de integración:

Cliente SOAP

Microservicio SOAP (ActorService)

Llama a la operación getAllActors o getActorStats

Internamente realiza una petición GET a /api/actors (API REST)

Recibe los datos JSON (lista de actores)

└> Procesa y transforma los datos (por ejemplo, calcula estadísticas)

Envía la respuesta estructurada de acuerdo a actor.wsdl

3. Implementación Técnica

Estructura del Proyecto:

```
├── src/
│   ├── config/
│   │   └── db.js         # Configuración de conexión a MySQL
│   ├── controllers/
│   │   └── actorController.js  # Funciones de consulta para la tabla 'actor'
│   ├── routes/
│   │   └── actorRoutes.js   # Endpoint REST para actores (opcional)
│   ├── soap/
│   │   ├── soapService.js  # Implementación del servicio SOAP y asociación de operaciones
│   │   └── actor.wsdl     # Contrato WSDL del servicio SOAP
│   └── app.js            # Inicializa el servidor Express (API REST)
```

├── .env # Variables de entorno

├── package.json # Dependencias y scripts del proyecto

└── README.md # Documentación del proyecto

Detalles de la implementación:

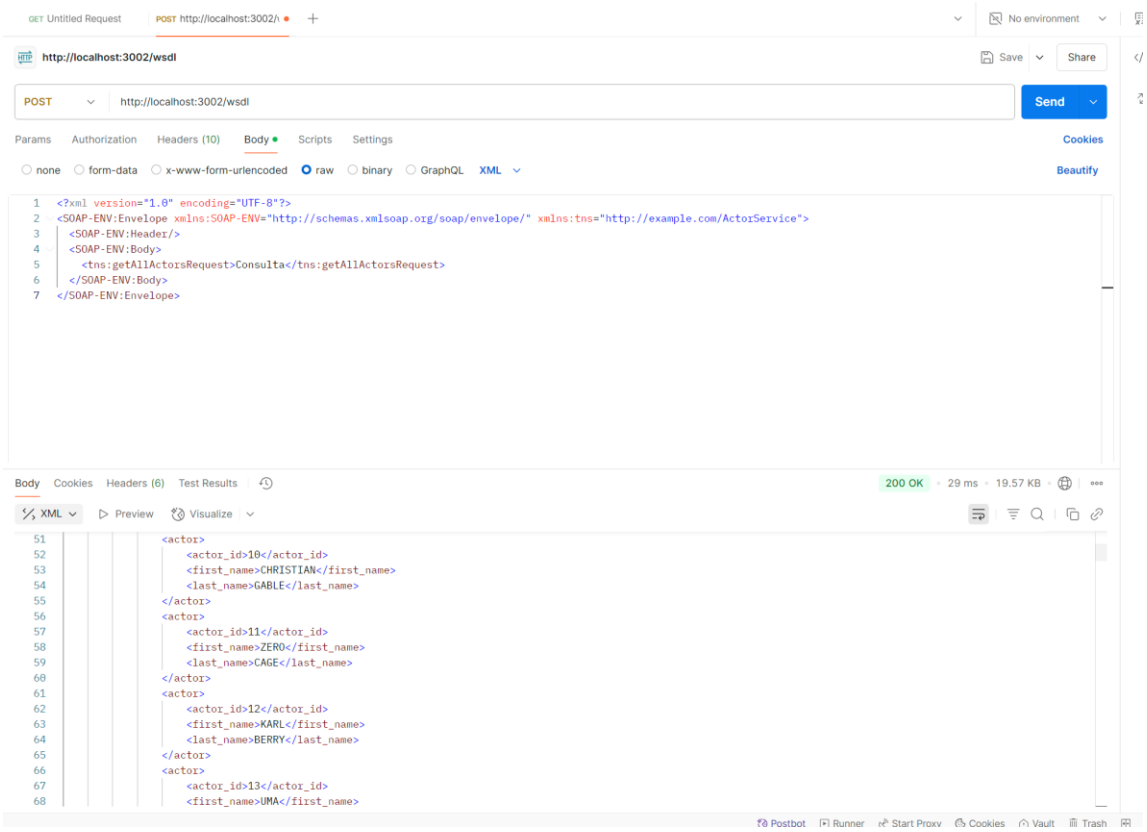
- **API REST:**
  El archivo app.js inicializa un servidor Express, incorpora las rutas definidas en actorRoutes.js y, a su vez, expone el endpoint /api/actors para la consulta de actores mediante actorController.js. La conexión a la base de datos se configura en db.js.
- **Servicio SOAP:**
  En soapService.js se define el objeto de servicio con dos operaciones:
- **getAllActors:** Realiza una petición GET a /api/actors usando axios y retorna la lista de actores en el formato esperado por el contrato.
- **getActorStats:** Además de obtener la lista de actores, calcula estadísticas (total de actores, valor mínimo, máximo y promedio del campo actor_id) y las entrega en la estructura definida en actor.wsdl.

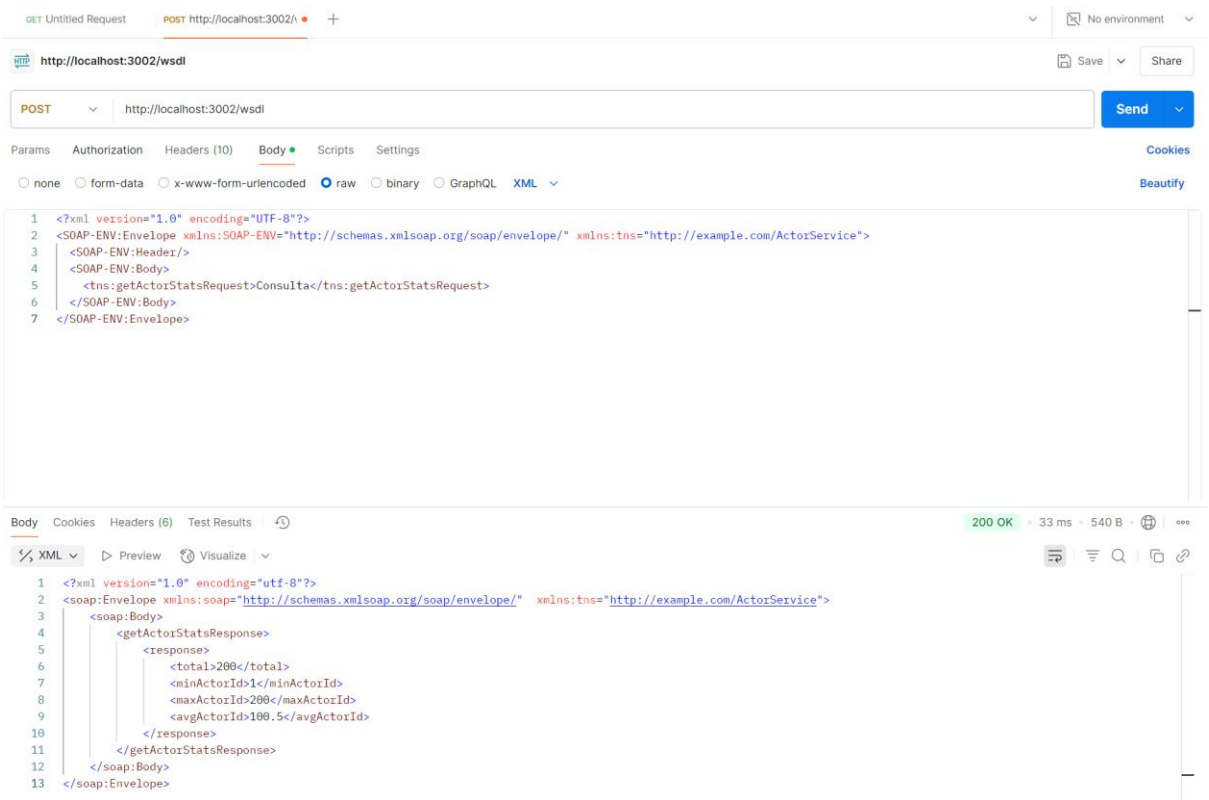
El servicio SOAP se asocia al endpoint /wsdl y utiliza el archivo actor.wsdl para definir el contrato.

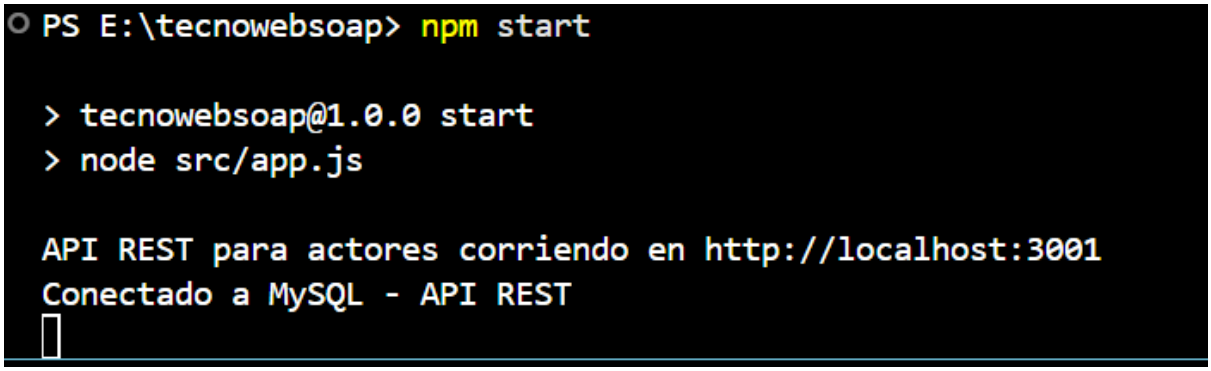
4. Pruebas y Documentación

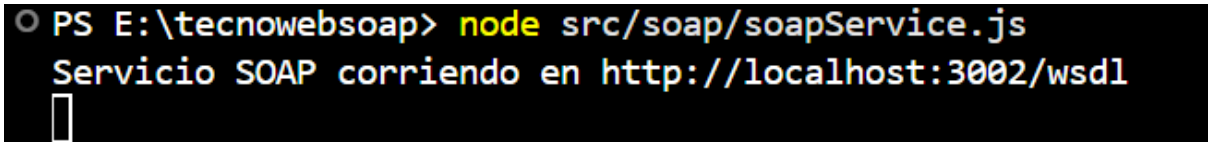
Evidencias funcionales:

- Capturas del POSTMAN


GET Untitled Request
POST http://localhost:3002/\
No environment
http://localhost:3002/wsdl
Save
Share
POST
http://localhost:3002/wsdl
Send
Params
Authorization
Headers (10)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
XML
Beautify
1 <?xml version="1.0" encoding="UTF-8"?>
2 <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tns="http://example.com/ActorService">
3 <SOAP-ENV:Header/>
4 <SOAP-ENV:Body>
5 <tns:getAllActorsRequest>Consulta</tns:getAllActorsRequest>
6 </SOAP-ENV:Body>
7 </SOAP-ENV:Envelope>
Body
Cookies
Headers (6)
Test Results
200 OK
29 ms
19.57 KB
XML
Preview
Visualize
51 <actor>
52 <actor_id>10</actor_id>
53 <first_name>CHRISTIAN</first_name>
54 <last_name>GABLE</last_name>
55 </actor>
56 <actor>
57 <actor_id>11</actor_id>
58 <first_name>ZERO</first_name>
59 <last_name>CAGE</last_name>
60 </actor>
61 <actor>
62 <actor_id>12</actor_id>
63 <first_name>KARL</first_name>
64 <last_name>BERRY</last_name>
65 </actor>
66 <actor>
67 <actor_id>13</actor_id>
68 <first_name>UMA</first_name>
Postbot
Runner
Start Proxy
Cookies
Vault
Trash


GET Untitled Request
POST http://localhost:3002/\
No environment
http://localhost:3002/wsdl
Save
Share
POST
http://localhost:3002/wsdl
Send
Params
Authorization
Headers (10)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
XML
Beautify
1 <?xml version="1.0" encoding="UTF-8"?>
2 <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tns="http://example.com/ActorService">
3 <SOAP-ENV:Header/>
4 <SOAP-ENV:Body>
5 <tns:getActorStatsRequest>Consulta</tns:getActorStatsRequest>
6 </SOAP-ENV:Body>
7 </SOAP-ENV:Envelope>
Body
Cookies
Headers (6)
Test Results
200 OK
33 ms
540 B
XML
Preview
Visualize
1 <?xml version="1.0" encoding="utf-8"?>
2 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tns="http://example.com/ActorService">
3 <soap:Body>
4 <getActorStatsResponse>
5 <response>
6 <total>200</total>
7 <minActorId>1</minActorId>
8 <maxActorId>200</maxActorId>
9 <avgActorId>100.5</avgActorId>
10 </response>
11 </getActorStatsResponse>
12 </soap:Body>
13 </soap:Envelope>





Pruebas manuales que se realizaron:

- Probar el endpoint REST /api/actors mediante Postman para confirmar que la consulta a MySQL retorna los datos.
- Usar un cliente SOAP para invocar getAllActors y verificar que se obtienen todos los actores.
- Invocar getActorStats para asegurarse de que los cálculos de estadísticas son correctos.