STOCK EXCHANGE MONITORING SYSTEM

Ingeniería de Software II

Presentado por:

Ledy Mayerly Astudillo Calderon

Harold Andres Molano Rosero

Santiago Nieto Guaca

Janier Yulder Gomez Galindez

Universidad del Cauca

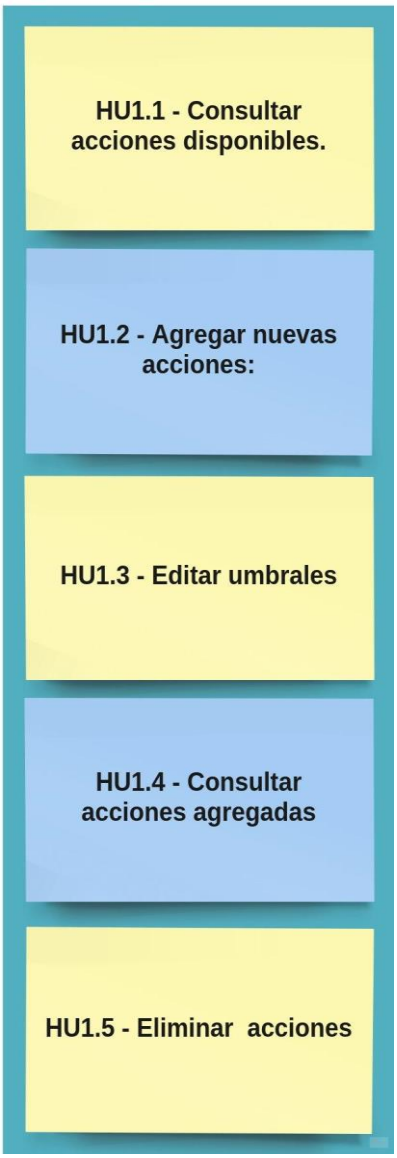
Facultad de Ingeniería Electrónica y Telecomunicaciones

Programa de Ingeniería de Sistemas

Popayán, Cauca

2023

Product Backlog



Cualidades del sistema - Escalabilidad y Modificabilidad

1. **Escalabilidad:**

- **Escenario de Calidad:** Garantizar que el sistema maneje 3,000 usuarios simultáneos.
- **Tácticas:**
    - **Escalado de la distribución de almacenamiento:** Escalando el sistema de almacenamiento (Sharding permite dividir los datos en

múltiples shards o particiones lógicas) se puede manejar más volumen de datos y operaciones sin afectar al cliente.

- **Retención de rendimiento:** Es importante que el rendimiento del servidor no se degrade conforme aumentan los clientes y las peticiones. La eficiencia de recursos y concurrencia ayudan a mantener el rendimiento estable. Esto garantiza una buena experiencia para el cliente independiente de la carga.

  Dado que este proyecto se ha desarrollado con fines educativos y utilizando arreglos para almacenar datos, no se ha implementado directamente la técnica de Sharding para el escalado de la distribución de almacenamiento. En lugar de ello, la aplicación utiliza una estructura de datos simple basada en arreglos y listas.

  Si se desea aplicar tácticas específicas para abordar el escenario de calidad de manejar 3,000 usuarios simultáneos, sería necesario reestructurar la aplicación para implementar técnicas como Sharding. Esto implicaría cambios significativos en la arquitectura de almacenamiento y en la lógica de acceso a datos, así como la configuración de un entorno de implementación que admita la distribución de datos.

- **Patrón de diseño Cliente - Servidor:**

  La separación clara entre el cliente y el servidor, utilizando el patrón de diseño Cliente-Servidor, contribuye a la escalabilidad. Los clientes pueden realizar solicitudes de manera eficiente al servidor, y el servidor, por su parte, puede manejar múltiples conexiones simultáneas.

2. **Modificabilidad:**
   - **Escenario de Calidad:** Para facilitar cambios en la base de datos y el desarrollo de nuevas funcionalidades, se han adoptado las siguientes tácticas:
   - **Tácticas:**
     - **Desacoplamiento de componentes:** La arquitectura ha sido diseñada con un enfoque en el desacoplamiento de componentes. La lógica de presentación, aplicación y acceso a datos se encuentra separada, permitiendo cambios

independientes en cada capa. Esto facilita la incorporación de nuevas funcionalidades o modificaciones en componentes específicos sin afectar al resto del sistema.

- **Uso de Interfaces:** Se ha optado por el uso de interfaces para desacoplar componentes y permitir un intercambio más fácil. Las interfaces proporcionan contratos claros entre las capas y componentes, permitiendo la sustitución de implementaciones sin afectar la interacción con otras partes del sistema.

- **Patrón de diseño de capas:**
  Se utiliza el patrón de diseño de capas tanto a nivel de arquitectura como en el código fuente. Este enfoque facilita la modificación de componentes específicos sin afectar otras partes del sistema. La separación lógica en capas permite un desarrollo más modular y mantenible.

Perspectivas de la Arquitectura:

- **Componentes del Sistema:**

  Patrón Arquitectónico: Capas.

  Descripción:

  La arquitectura del sistema sigue el patrón de capas, organizando de manera clara y definida las distintas capas lógicas, tales como Presentación, Aplicación y Datos. Esta estructura facilita la separación de responsabilidades y la gestión modular del sistema.

- **Despliegue de Componentes:**

  Patrón Arquitectónico: Cliente-Servidor.

  Descripción:

  La arquitectura adopta el patrón Cliente-Servidor para separar eficientemente la lógica del cliente y del servidor. Esta separación permite una gestión más efectiva de las solicitudes, mejorando la escalabilidad y la mantenibilidad del sistema. El cliente y el servidor interactúan de manera clara y definida, contribuyendo a una arquitectura más robusta.

- **Módulos del Código Fuente:**

  -Patrón Arquitectónico: Capas.

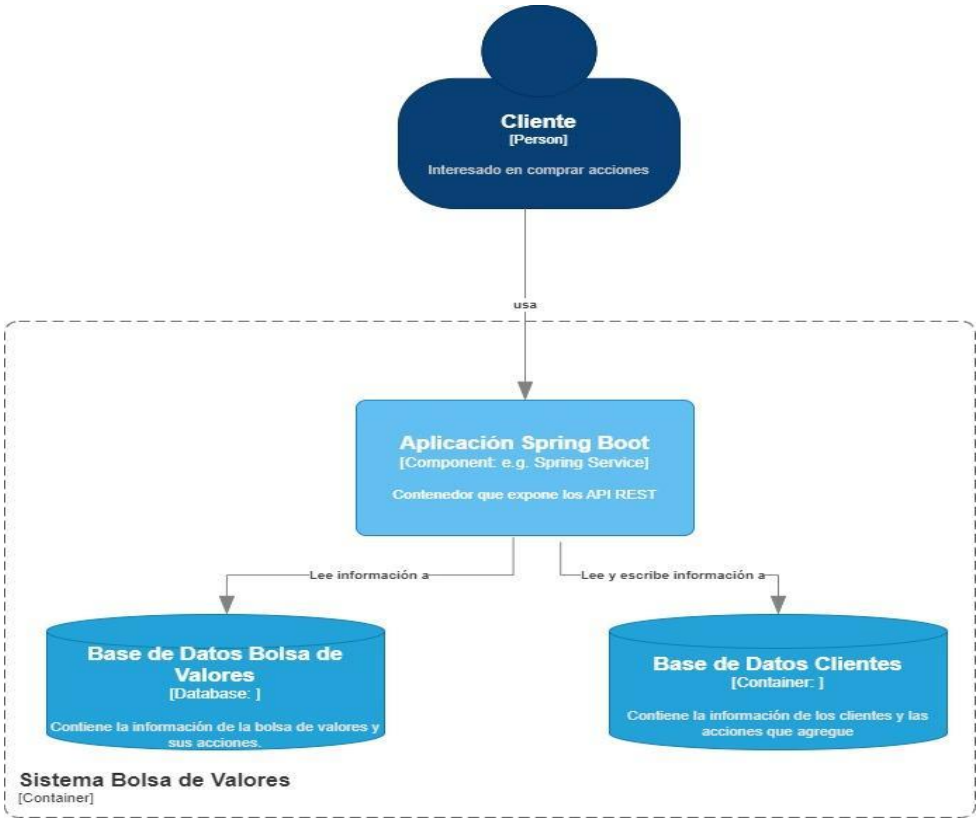
Descripción:

A nivel del código fuente, se implementa el patrón de capas para organizar el sistema en paquetes o módulos según las capas lógicas predefinidas. Esta estructura modular facilita la comprensión del código y permite cambios independientes en cada capa, mejorando así la mantenibilidad y la flexibilidad del sistema.

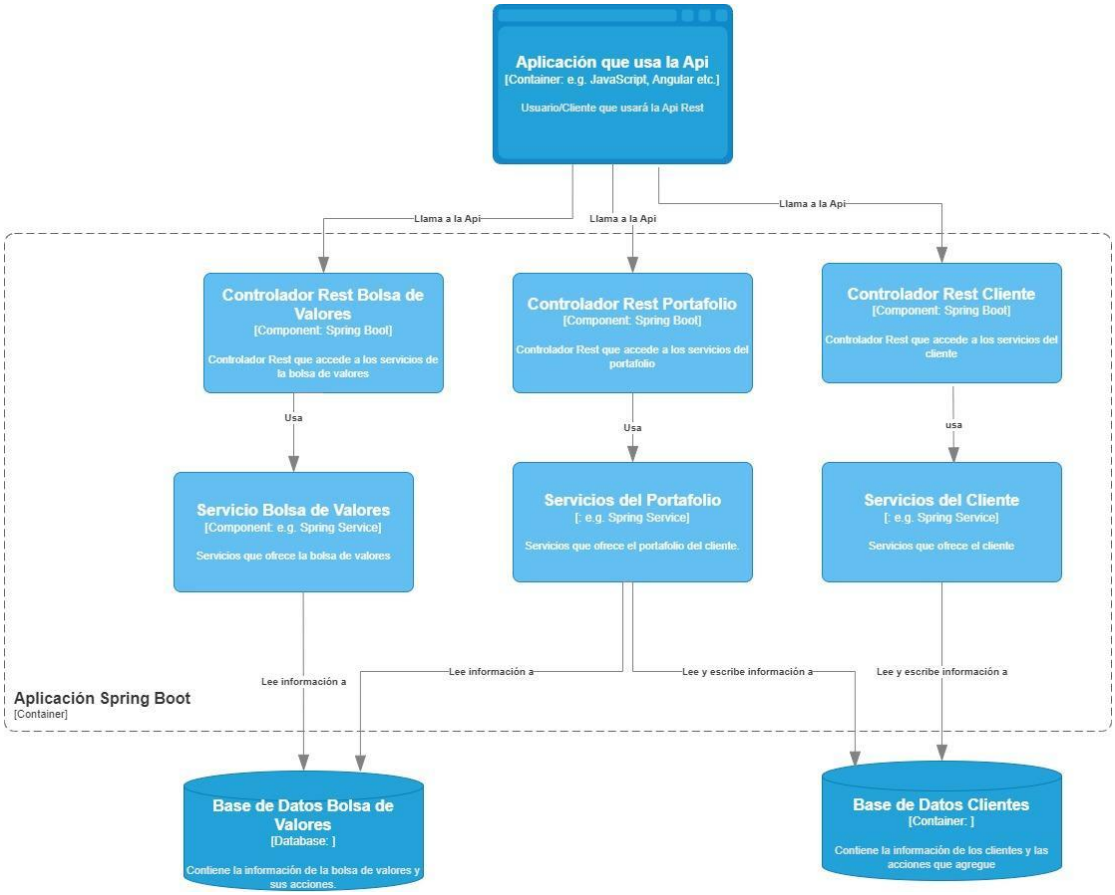
Modelo C4

- **Diagrama Contexto:**

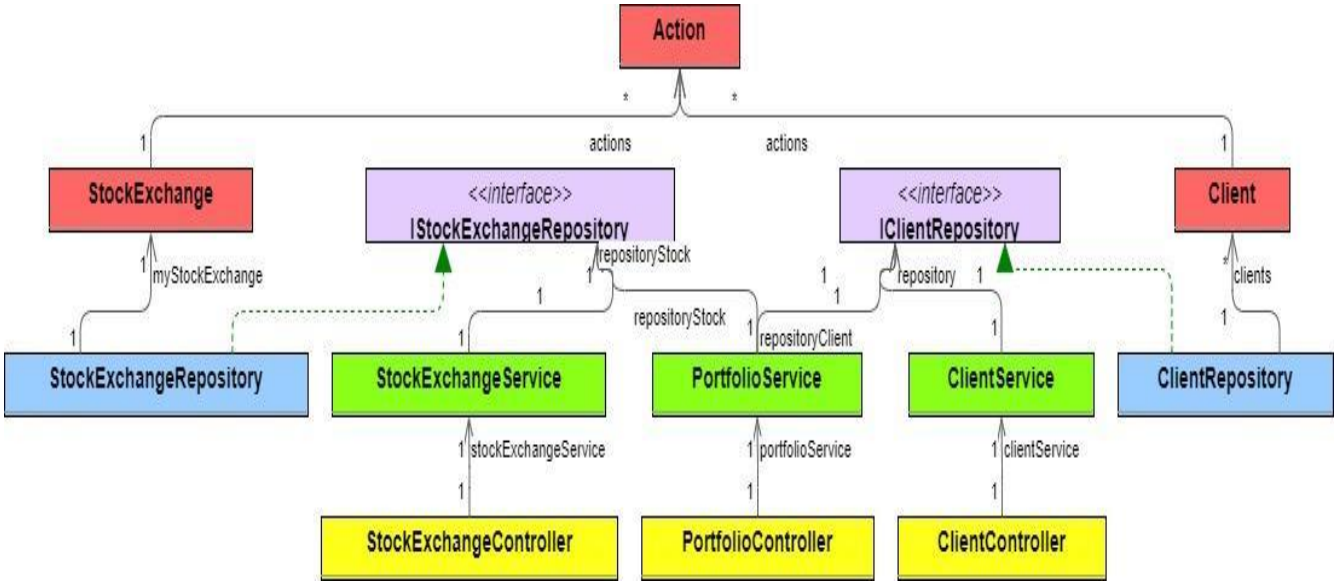
- **Diagrama Contenedores:**



- **Diagrama Componentes:**

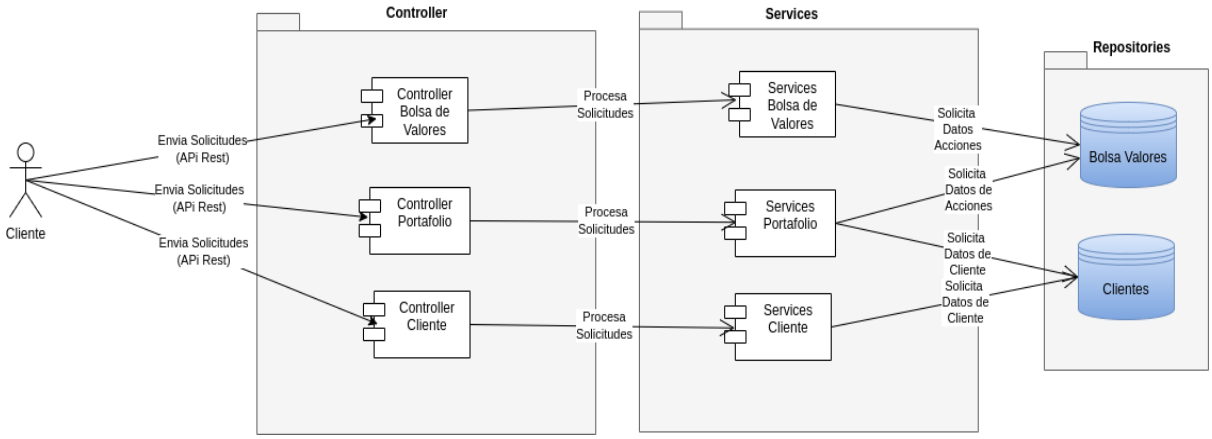


- **Diagrama Código:**

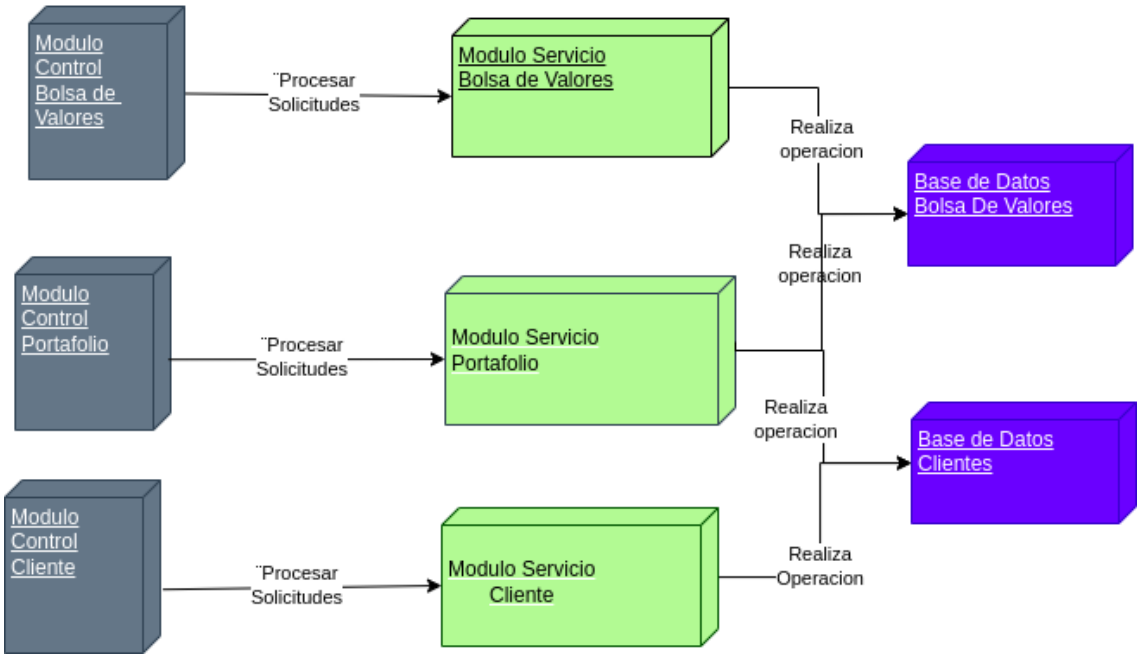


Diagramas UML:

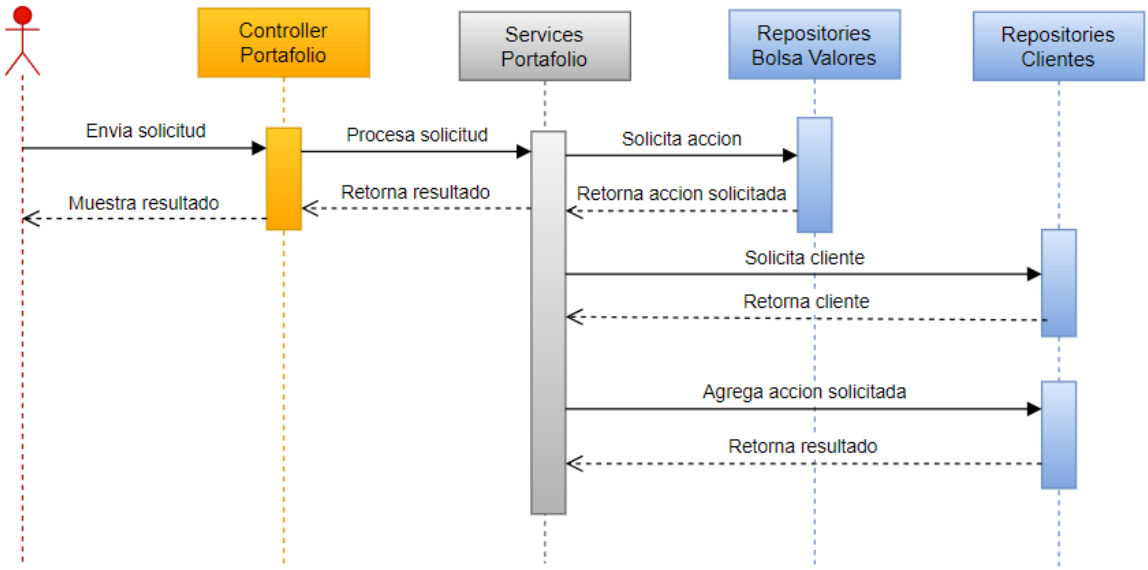
- **Diagrama de componentes y conectores**



- **Diagrama de módulos**

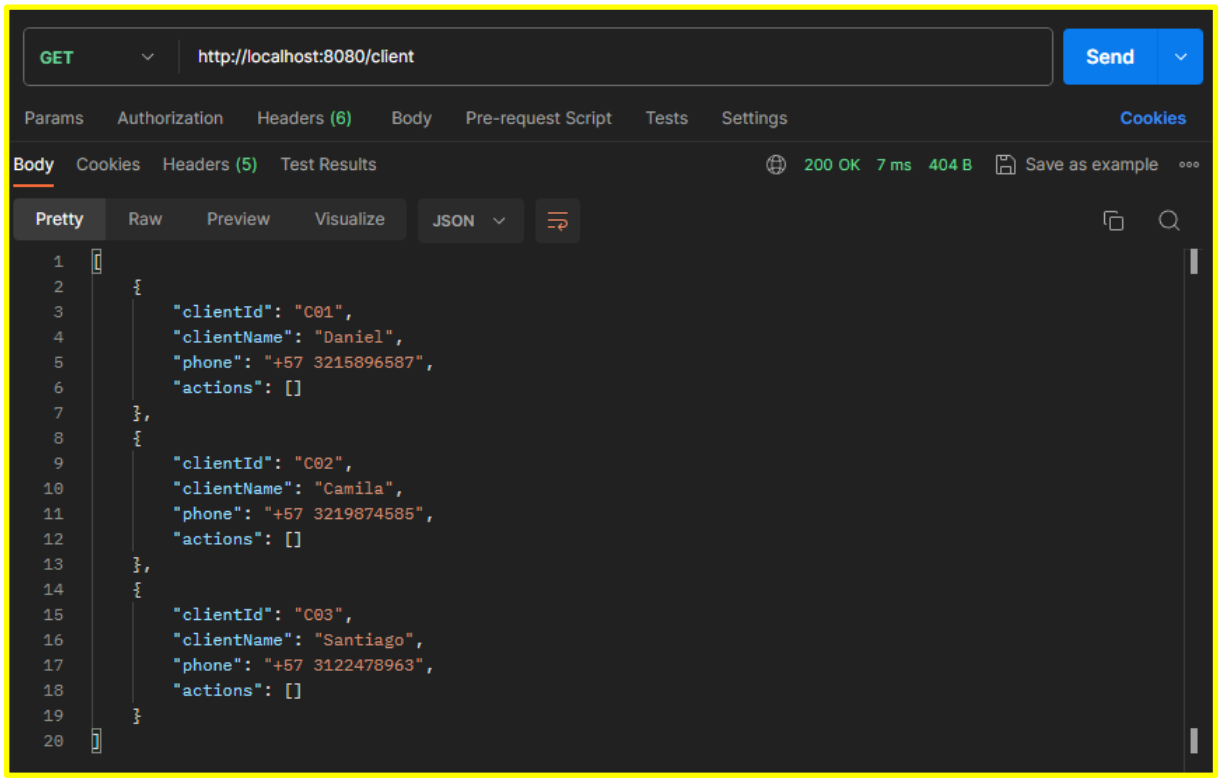


- **Diagrama de secuencia de agregar acciones**

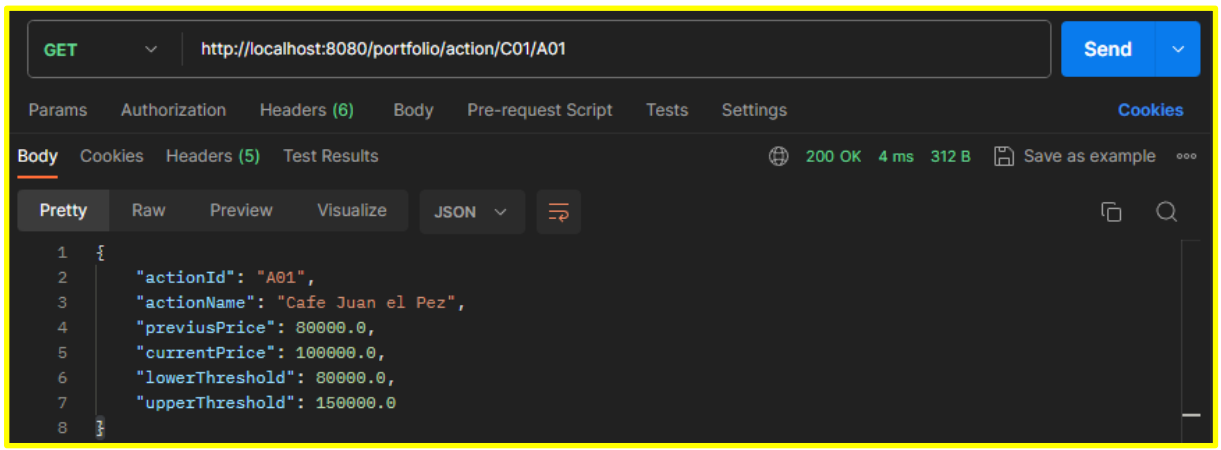


Evidencia consultas en Postman:

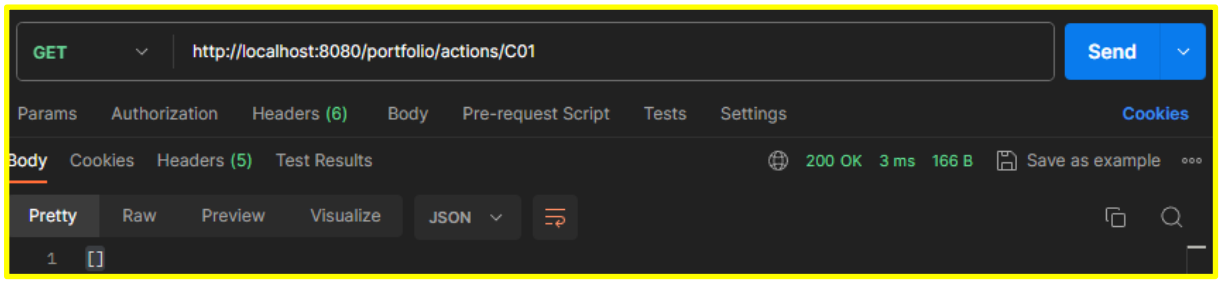
- **get clients:**



- **get client:**

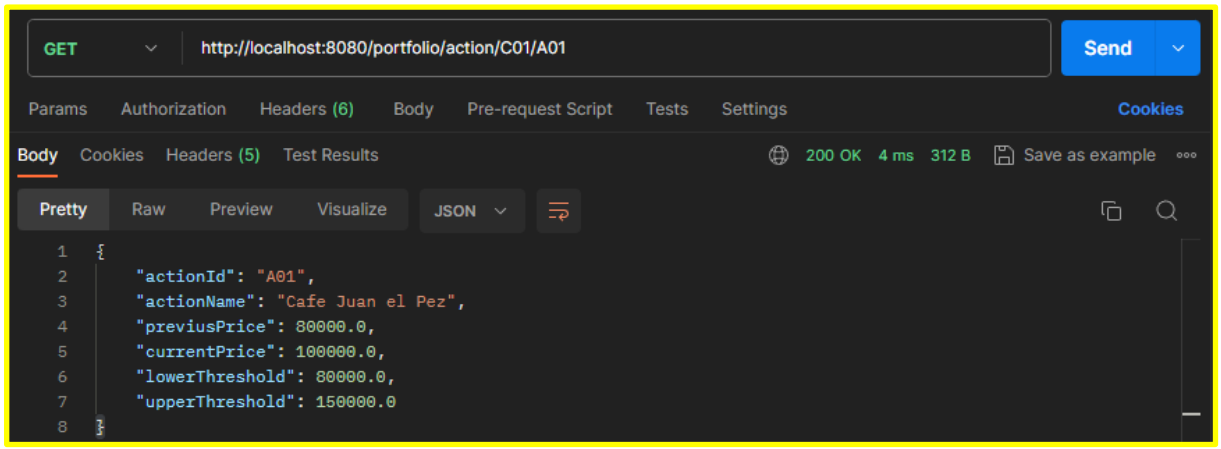


- **get client actions:**

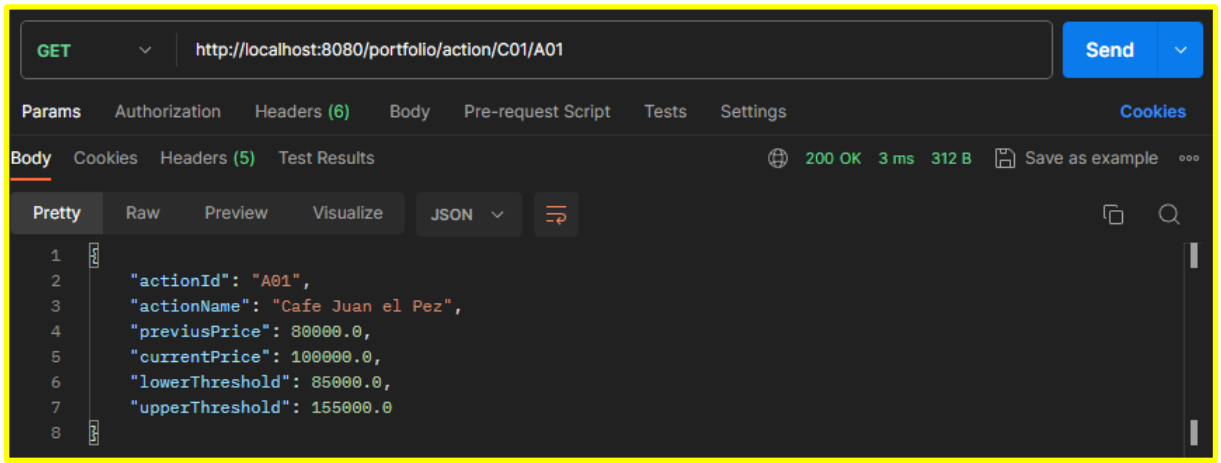


- **post client action:**

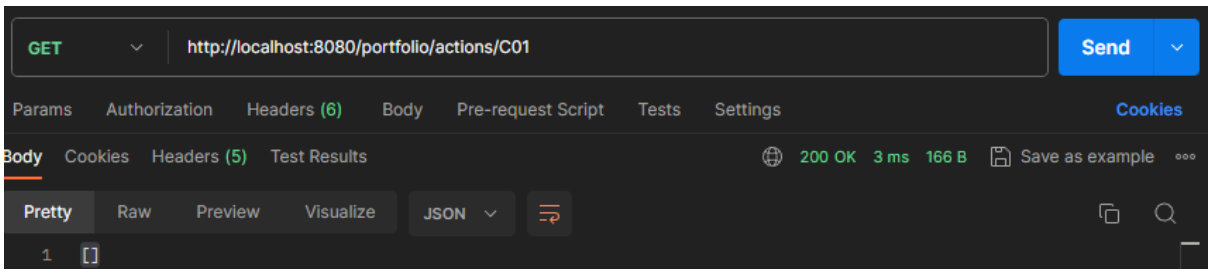
- **get client action:**



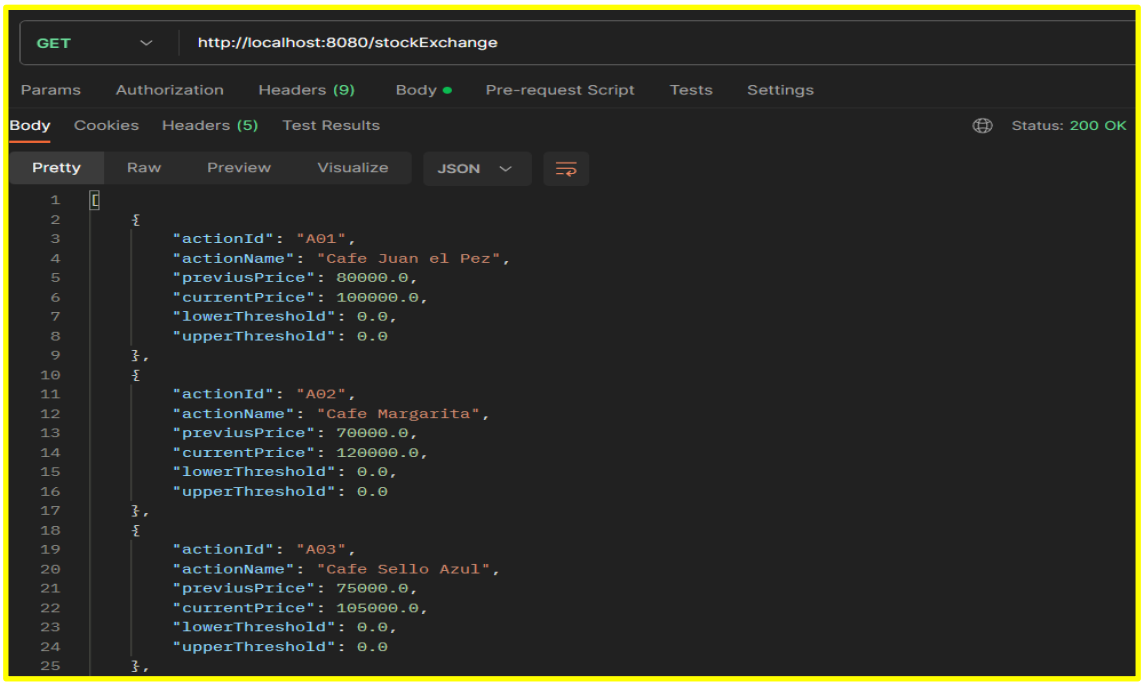
- **put client action:**



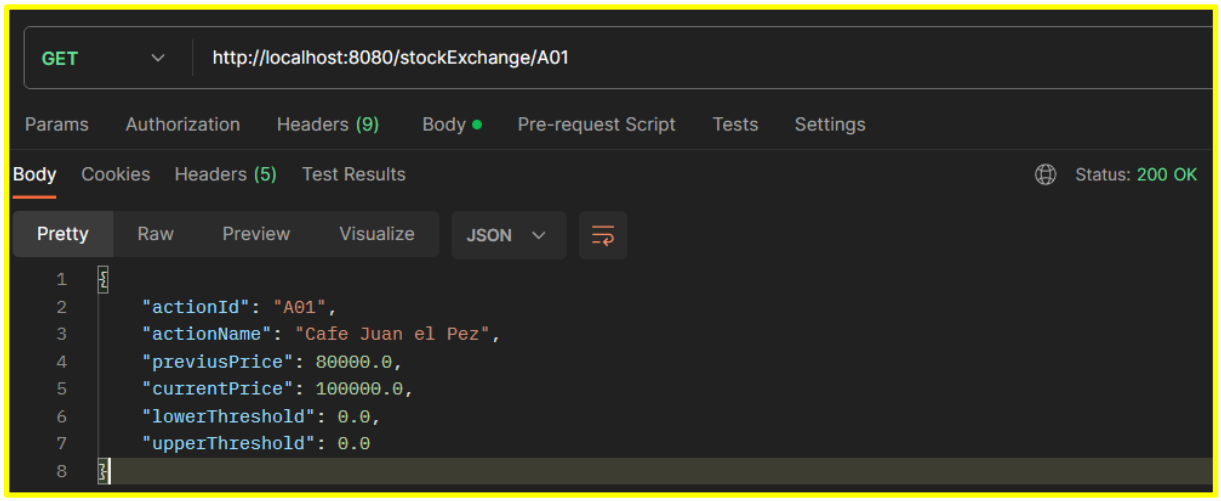
- **del client action:**



- **get global actions:**



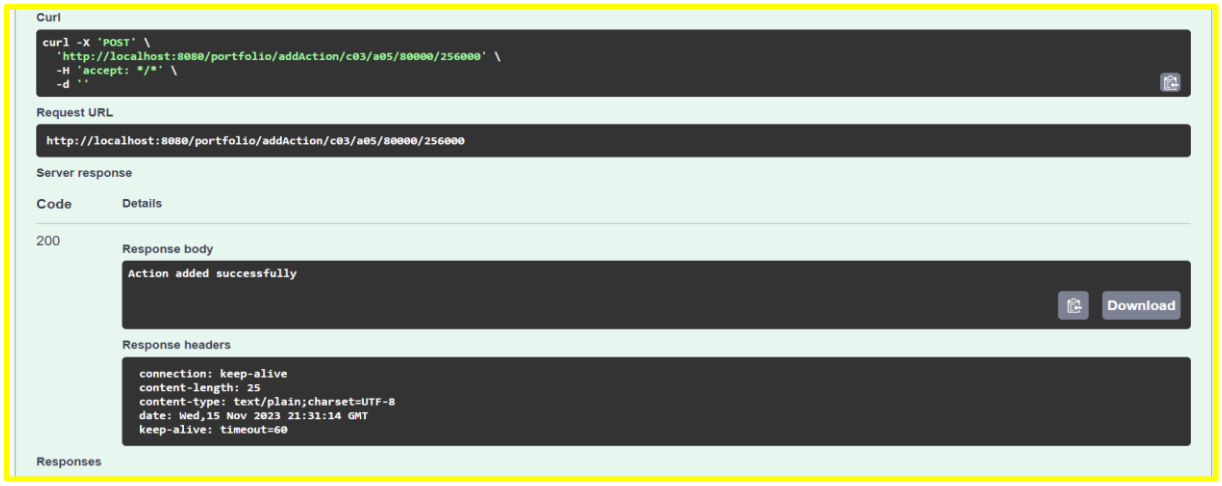
- **get global action:**



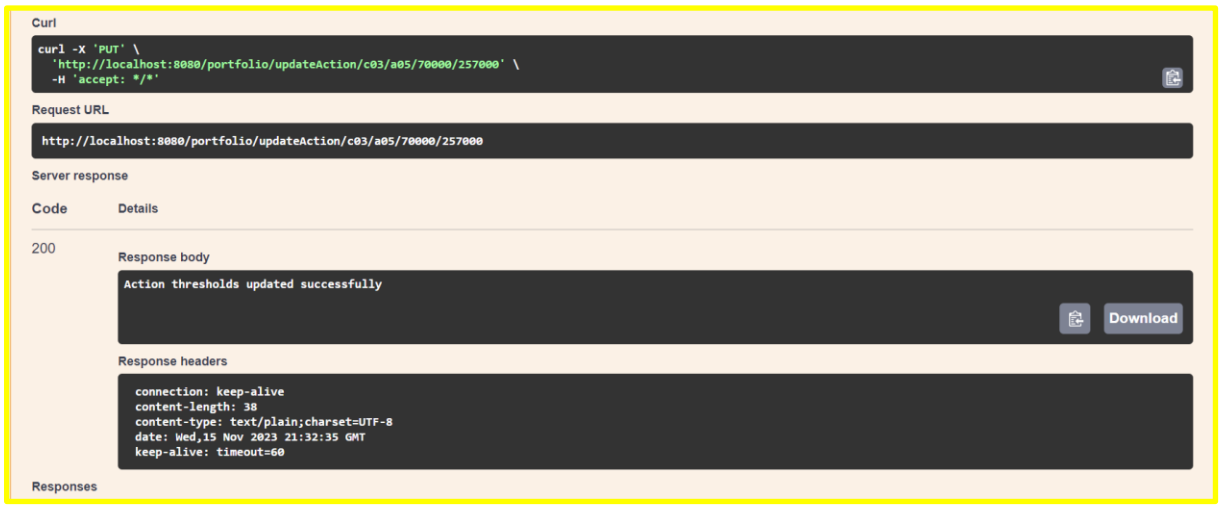
Enlace Swagger

http://localhost:8080/swagger-ui/index.html

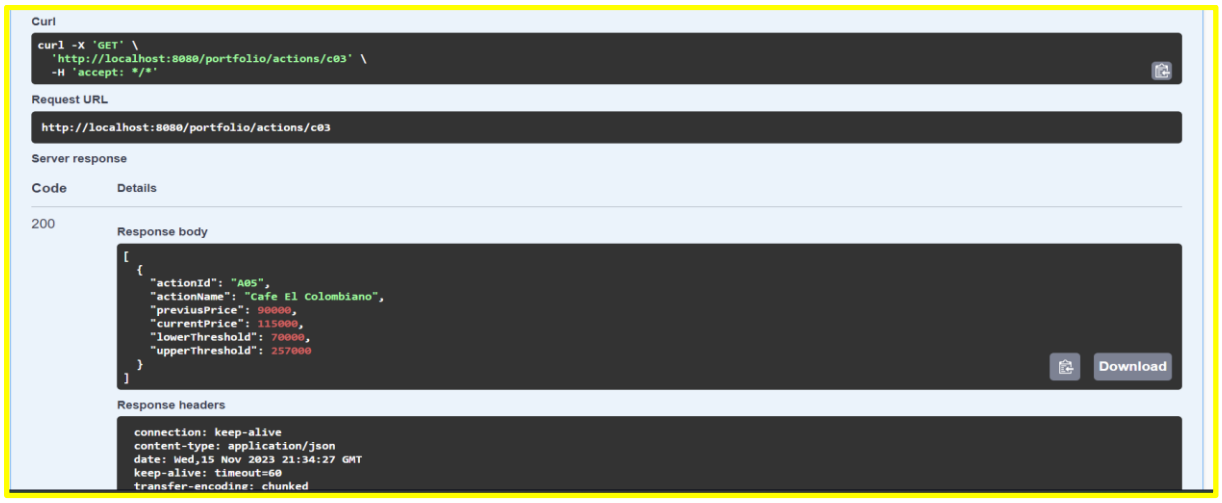
POST:



PUT:



GET: Obtiene la lista de acciones asociadas a un cliente.



GET:Busca una acción asociada a un cliente por los IDs del cliente y la acción.

DELETE:

GET:Obtiene la lista completa de acciones en el repositorio de la bolsa de valores.

GET:Busca una acción por su ID.

Curl

```
curl -X 'GET' \
  'http://localhost:8080/stockExchange/a05' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:8080/stockExchange/a05
```

Server response

Code	Details
200	Response body

```
{
  "actionId": "A05",
  "actionName": "Cafe El Colombiano",
  "previusPrice": 90000,
  "currentPrice": 115000,
  "lowerThreshold": 0,
  "upperThreshold": 0
}
```

Download

Response headers

```
connection: keep-alive
content-type: application/json
date: Wed,15 Nov 2023 21:47:20 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Responses

GET:Obtiene la lista de todos los clientes en el repositorio.

Curl

```
curl -X 'GET' \
  'http://localhost:8080/client' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:8080/client
```

Server response

Code	Details
200	Response body

```
[
  {
    "clientId": "C01",
    "clientName": "Daniel",
    "phone": "+57 3215896587",
    "actions": []
  },
  {
    "clientId": "C02",
    "clientName": "Camila",
    "phone": "+57 3219874585",
    "actions": []
  },
  {
    "clientId": "C03",
    "clientName": "Santiago",
    "phone": "+57 3122478963",
    "actions": [
      {
        "actionId": "A05",
```

GET:Busca un cliente por su ID.

Curl

```
curl -X 'GET' \
  'http://localhost:8080/client/c03' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:8080/client/c03
```

Server response

Code	Details
200	Response body

```
{
  "clientId": "C03",
  "clientName": "Santiago",
  "phone": "+57 3122478963",
  "actions": [
    {
      "actionId": "A05",
      "actionName": "Cafe El Colombiano",
      "previusPrice": 90000,
      "currentPrice": 115000,
      "lowerThreshold": 70000,
      "upperThreshold": 257000
    }
  ]
}
```

Download

Response headers