Examen Final

Ingeniería de Software II

Presentado por:

Ledy Mayerly Astudillo Calderon

Harold Andres Molano Rosero

Santiago Nieto Guaca

Janier Yulder Gomez Galindez

Universidad del Cauca

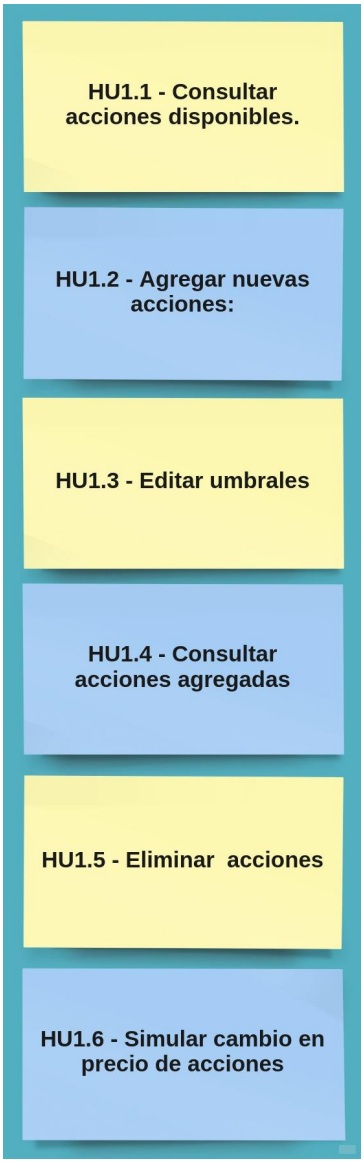
Facultad de Ingeniería Electrónica y Telecomunicaciones

Programa de Ingeniería de Sistemas

Popayán, Cauca

2023

Product Backlog



Cualidades del sistema - Escalabilidad y Modificabilidad

1. **Escalabilidad:**

   Escenario de Calidad - EC01: Garantizar que el sistema maneje 3,000 usuarios simultáneos.

Tácticas:

Arquitectura de Publicador-Suscriptor:

- Implementar una arquitectura de publicador-suscriptor para desacoplar unidades funcionales.
- Facilitar la escalabilidad horizontal al permitir la adición de nuevos suscriptores y publicadores de forma independiente.

Escalabilidad Horizontal en Contenedores Docker:

- Diseñar la arquitectura para permitir el despliegue y escalado horizontal en contenedores Docker.
- Permitir que diferentes componentes del sistema se ejecuten en instancias separadas para distribuir la carga de trabajo.

Sistema de Mensajería RabbitMQ:

- Utilizar RabbitMQ como sistema de mensajería para la comunicación asíncrona entre componentes.
- Distribuir consultas entre diferentes publicadores y suscriptores de manera eficiente.

Tolerancia a Fallos:

- Implementar un sistema robusto de tolerancia a fallos que garantice la disponibilidad del sistema.
- Manejar situaciones de error y recuperación para minimizar el impacto en la experiencia del usuario.

Monitoreo y Escalabilidad Elástica:

- Incorporar herramientas de monitoreo para realizar un seguimiento del rendimiento del sistema.
- Establecer umbrales de carga y configurar la escalabilidad elástica para ajustar dinámicamente los recursos según la demanda.

Particionamiento de Funcionalidades:

- Identificar unidades funcionales independientes y particionar para distribuir la carga de manera efectiva.
- Permitir que distintas funciones del sistema escalen independientemente según sea necesario.

Caché Distribuida:

- Implementar una caché distribuida para almacenar datos temporales y reducir las consultas a la base de datos.
- Mejorar la eficiencia y velocidad de respuesta del sistema.

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.

Escenario de Calidad - EC02: El sistema debe soportar el aumento del número de usuarios y de consultas, tanto a nivel local como nacional, sin comprometer la calidad del servicio ni la seguridad de los datos. Para ello, se debe utilizar una arquitectura de publicador-suscriptor que divida el sistema en unidades funcionales independientes y desacopladas, que puedan ser desplegadas y escaladas de forma horizontal en contenedores Docker. El sistema debe utilizar un sistema de mensajería RabbitMQ que distribuya las consultas entre los diferentes publicadores, y debe implementar un sistema

de tolerancia a fallos que garantice la disponibilidad y la recuperación del sistema en caso de errores.

Tácticas:

Reducción de la Dependencia de Recursos Externos:

- Identificar y reducir la dependencia de servicios externos que puedan ser puntos únicos de fallo.
- Minimizar el impacto en la escalabilidad si un servicio externo experimenta problemas.

Algoritmos de Enrutamiento Dinámico:

- Utilizar algoritmos de enrutamiento dinámico para redirigir el tráfico de manera eficiente.
- Ajustar dinámicamente las rutas en función de la carga y la disponibilidad de los servidores.

Distribución de Contenido Estático:

- Implementar servicios de distribución de contenido (CDN) para distribuir contenido estático a nivel global.
- Mejorar la velocidad de carga de recursos estáticos y reducir la carga en los servidores principales.

Respuesta a Picos de Tráfico:

- Diseñar estrategias para manejar picos de tráfico repentinos.
- Implementar políticas de respaldo y provisionamiento automático para escalar rápidamente en momentos de alta demanda.

Escalabilidad Asimétrica:

- Permitir la escalabilidad asimétrica, donde diferentes partes del sistema pueden escalar de manera independiente según la demanda.
- Ajustar recursos específicamente en áreas críticas para mejorar la eficiencia.

Priorización de Servicios Críticos:

- Identificar y priorizar servicios críticos que deben mantenerse altamente disponibles.
- Asignar recursos adicionales a servicios clave para garantizar su rendimiento constante.

Estrategias de Desescalado:

- Implementar estrategias de desescalado para liberar recursos no utilizados durante períodos de baja demanda.
- Optimizar los costos y la eficiencia del uso de recursos.

2. **Modificabilidad:**

Escenario de Calidad - EC03: Para facilitar los cambios en la base de datos y asegurar una adaptación eficiente, se realizarán las modificaciones necesarias en un tiempo no mayor a una semana, con un esfuerzo estimado de 2 personas-mes. Adicionalmente, se implementará una gráfica que muestre el cambio de acciones, completando esta tarea en un plazo de 1 semana con un esfuerzo de 0.5 personas-mes.

Tácticas:

- **Dividir una responsabilidad:** Al dividir las responsabilidades en el sistema, se facilita la adaptación eficiente al realizar cambios en la base de datos. Al tener componentes o módulos claramente definidos y separados, es más fácil comprender, mantener y modificar cada parte del sistema de manera independiente. Esto contribuye a cumplir con el plazo de una semana para realizar las modificaciones necesarias en la base de datos.
- **Mantener la coherencia semántica:** Es esencial para evitar inconsistencias y errores en el sistema, incluyendo la base de datos. Al aplicar esta táctica, se garantiza que los cambios realizados en la base de datos sean consistentes con el resto del sistema.
- **Utilizar la encapsulación:** Al utilizar la encapsulación, se facilita la adaptación eficiente al realizar cambios en la base de datos. Al encapsular la lógica de la base de datos, se crea una capa de abstracción que permite realizar modificaciones en la estructura o la tecnología subyacente sin afectar directamente a otros componentes del sistema. Esto mejora la flexibilidad y la modificabilidad del sistema en general, lo que ayuda a cumplir con el plazo y el esfuerzo estimado de 2 personas-mes para las modificaciones en la base de datos.
- **Utilizar un intermediario y restringir las vías de comunicación:** Al utilizar un intermediario y restringir las vías de comunicación, se reduce la dependencia directa entre los componentes del sistema, incluyendo la base de datos. Al establecer un punto centralizado de

comunicación, se facilita la introducción de cambios y mejoras en la base de datos sin afectar directamente a todos los componentes. Esto minimiza el impacto de los cambios en diferentes partes del sistema y contribuye a la adaptación eficiente en el plazo estimado de una semana y con un esfuerzo de 0.5 personas-mes para implementar la gráfica de cambio de acciones.

Escenario de Calidad - EC04: El sistema debe permitir la integración de nuevas fuentes de datos para la bolsa de valores, por ejemplo, si se desea incorporar datos de otras bolsas internacionales o de otros sectores económicos. Para ello, se debe diseñar una interfaz de aplicación que defina los métodos y los parámetros necesarios para obtener los datos de la bolsa de valores, y que pueda ser implementada por diferentes adaptadores de infraestructura que se comuniquen con las bases de datos externas. El sistema debe facilitar la adición de nuevos adaptadores sin afectar al resto de los componentes, y debe proveer un mecanismo de configuración que permita seleccionar el adaptador adecuado según la fuente de datos solicitada.

Tácticas:

- **Dividir una responsabilidad:** Esta táctica consiste en dividir la responsabilidad de integrar nuevas fuentes de datos en componentes más pequeños y manejables. En este caso, se puede dividir la responsabilidad de obtener los datos de la bolsa de valores en módulos separados que se encargan de interactuar con las diferentes fuentes de datos.
- **Utilizar la encapsulación:** La encapsulación es una táctica que ayuda a reducir el acoplamiento entre los componentes. En este caso, se puede aplicar la encapsulación para encapsular la lógica de comunicación con las bases de datos externas en un componente específico. Esto permitirá que los adaptadores de infraestructura se comuniquen con ese componente sin necesidad de conocer los detalles internos de cómo se obtienen los datos.

- **Elevar el nivel de abstracción:** Elevar el nivel de abstracción implica utilizar abstracciones más generales y abstractas en lugar de detalles específicos de implementación. En este caso, se puede aplicar esta táctica para definir una interfaz de aplicación general que permita obtener los datos de la bolsa de valores, en lugar de depender de detalles específicos de cada fuente de datos. Esto permitirá que diferentes adaptadores de infraestructura implementen esa interfaz sin afectar al resto del sistema.
- **Utilizar un intermediario y restringir las vías de comunicación:** Esta táctica implica introducir un componente intermediario que se encargue de la comunicación entre los adaptadores de infraestructura y el resto del sistema. En este escenario, se puede aplicar esta táctica para que el intermediario restrinja las vías de comunicación y garantice que los adaptadores solo se comuniquen con los componentes necesarios. Esto permitirá la adición de nuevos adaptadores sin afectar al resto de los componentes.

Patrón de diseño hexagonal:

- Se utiliza el patrón de arquitectura hexagonal tanto a nivel de arquitectura como en el código fuente. Este patrón desacopla el dominio central del negocio de los detalles externos, permitiendo cambios fáciles en cualquiera de los lados sin afectar el otro. Por ejemplo, se puede cambiar la base de datos o la interfaz de usuario sin modificar la lógica central. De esta forma, el patrón hexagonal facilita la modificabilidad al generar un desacoplamiento fuerte entre el dominio y los detalles que pueden variar.

Perspectivas de la Arquitectura:

- **Componentes del Sistema:**

  Patrón Arquitectónico: Hexagonal / Puertos y Adaptadores

  Descripción:

La arquitectura del sistema sigue el patrón hexagonal, organizando el sistema en capas concéntricas. En el centro se ubica el dominio con la lógica de negocio. Alrededor hay una capa de interfaces que desacoplan el núcleo de los detalles externos. Por fuera se encuentran los adaptadores que implementan esas interfaces.

Esta estructura desacopla el dominio central del resto de módulos y componentes externos, permitiendo cambios independientes de un lado o del otro. De esta manera se facilita la modificabilidad y se genera un bajo acoplamiento entre componentes.

- **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.

  Patrón Arquitectónico: Publicador-Suscriptor.

  Descripción:

  Para la integración asíncrona con el sistema externo AgenteFinanciero, se implementa el patrón arquitectónico Publicador-Suscriptor mediante el middleware de mensajería RabbitMQ. El componente que monitorea los precios de las acciones actúa como publicador, publicando mensajes a RabbitMQ cuando se cruzan los umbrales de precio configurados. El sistema AgenteFinanciero actúa como suscriptor, suscrito a la cola de mensajes de RabbitMQ para recibir notificaciones asíncronas cuando ocurren cambios significativos en los precios. Esto permite una comunicación desacoplada entre los dos sistemas.

- **Módulos del Código Fuente:**

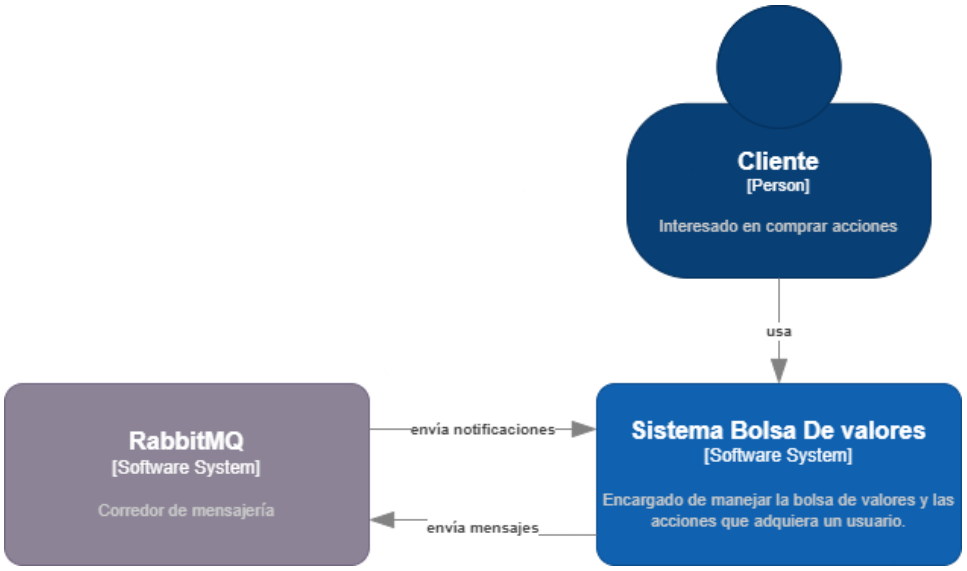
  Patrón Arquitectónico: Hexagonal.

  Descripción:

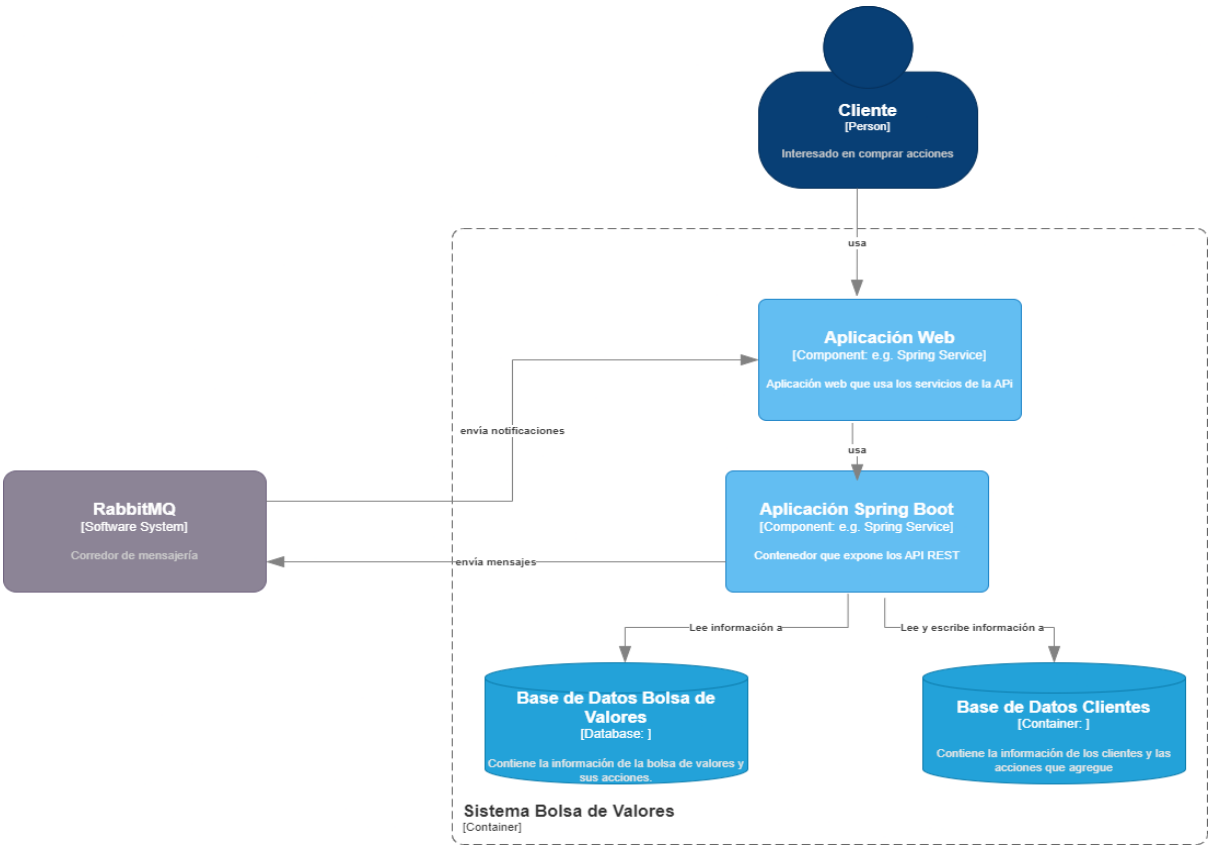
A nivel del código fuente, se implementa el patrón hexagonal o de puertos y adaptadores para organizar el sistema en módulos separados para el dominio, las interfaces y los adaptadores externos. El módulo de dominio contiene las entidades de negocio, la lógica central y los servicios principales. El módulo de interfaces proporciona contratos bien definidos para separar el dominio de los módulos externos. Finalmente, están los módulos adaptadores que implementan dichas interfaces.

Modelo C4

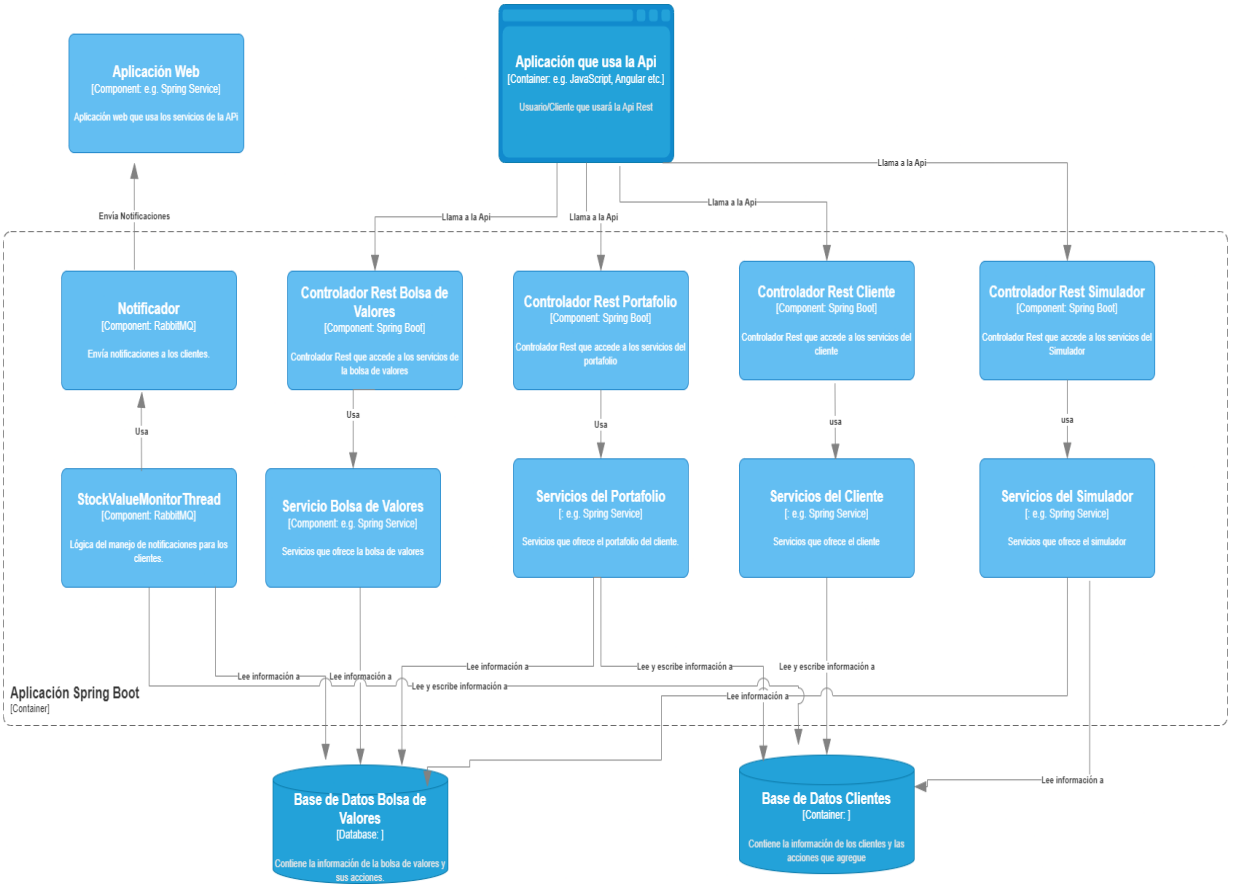
- **Diagrama Contexto:**



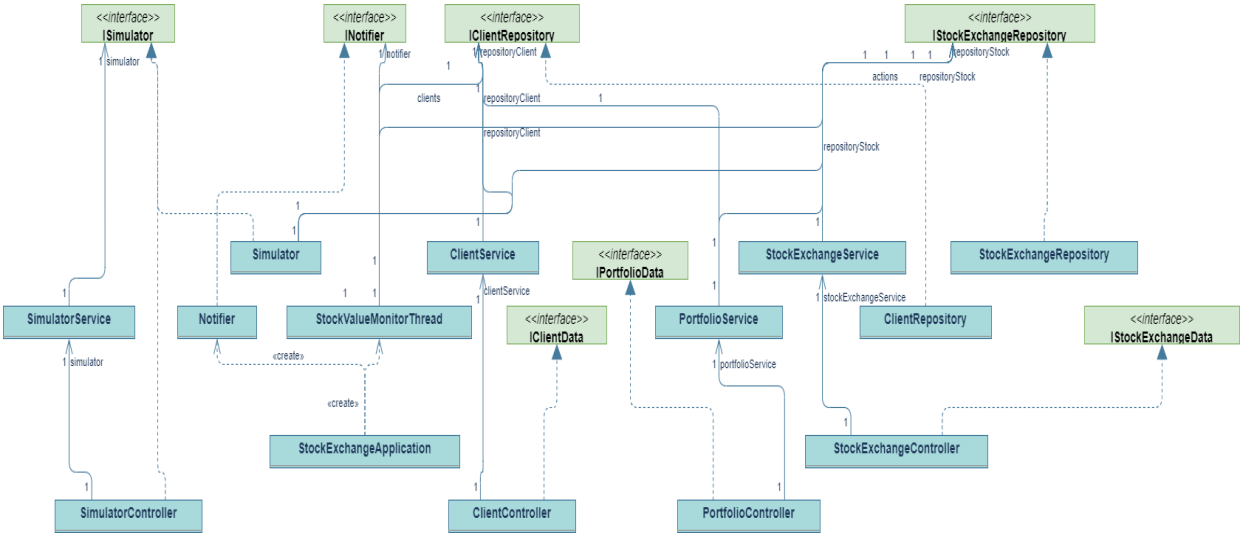
- **Diagrama Contenedores:**



- **Diagrama Componentes:**

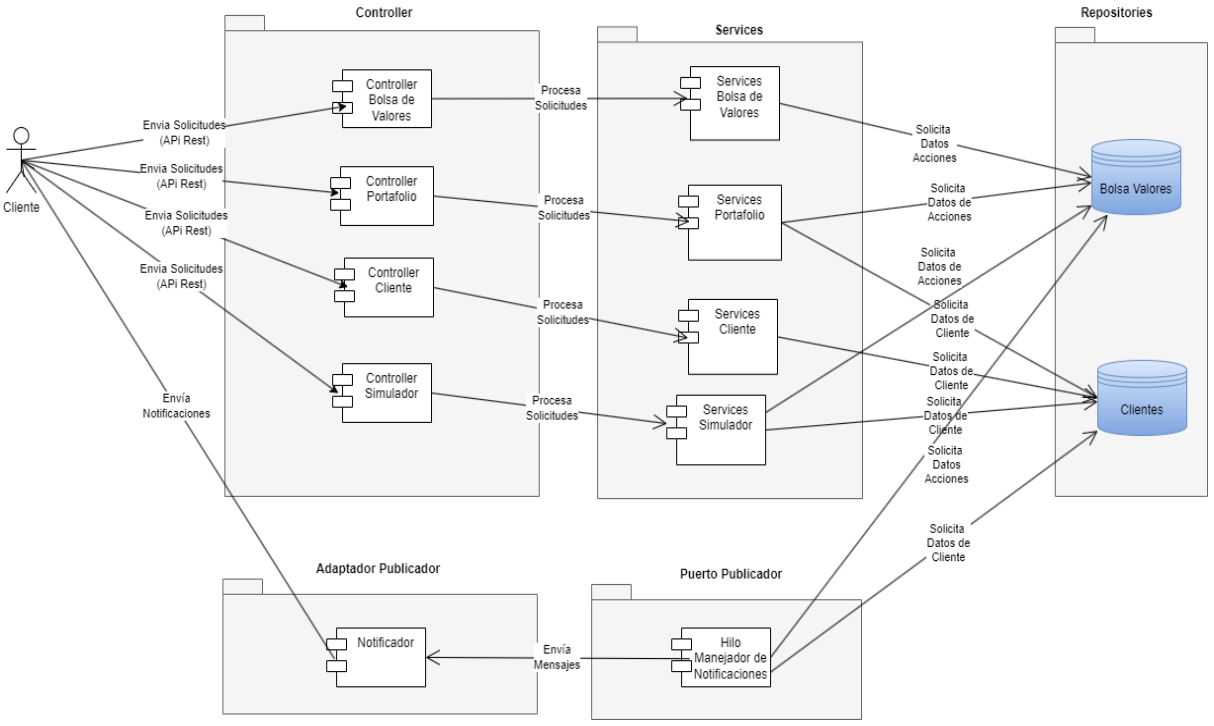


- **Diagrama Código:**

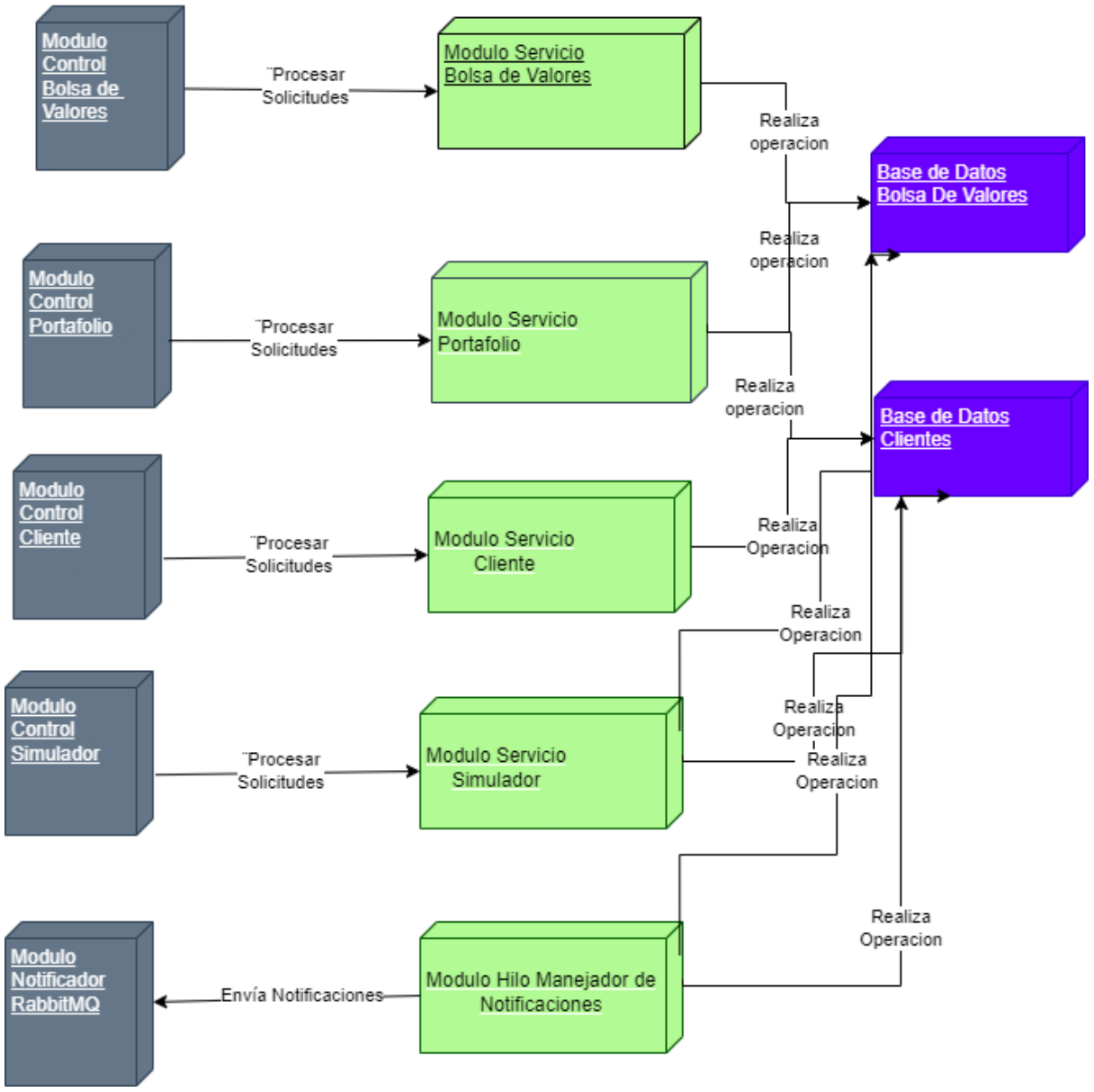


Diagramas UML:

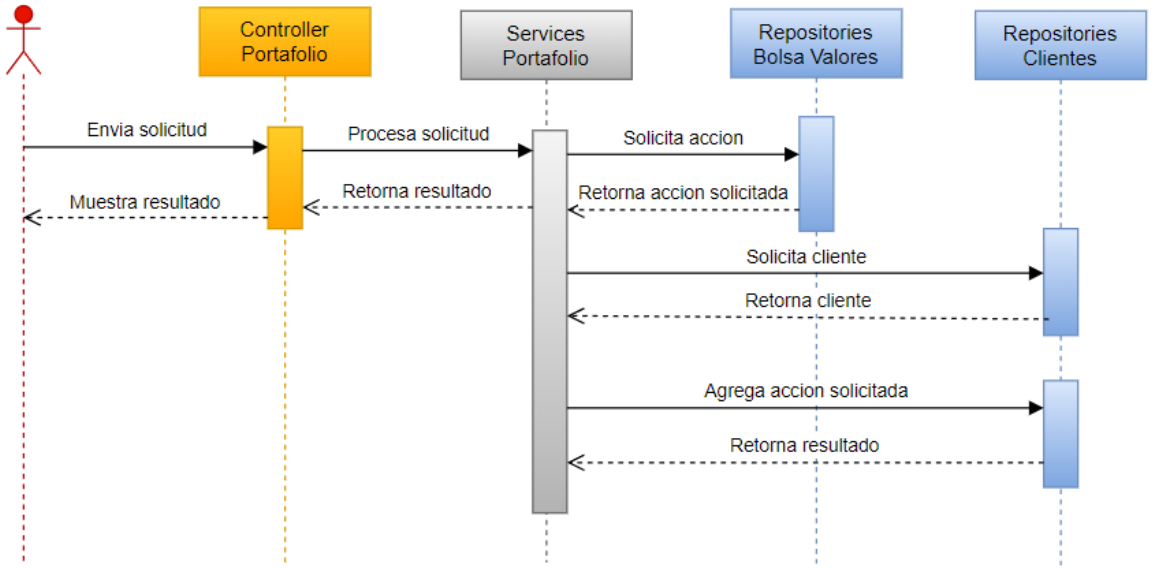
- **Diagrama de componentes y conectores**



- **Diagrama de módulos**

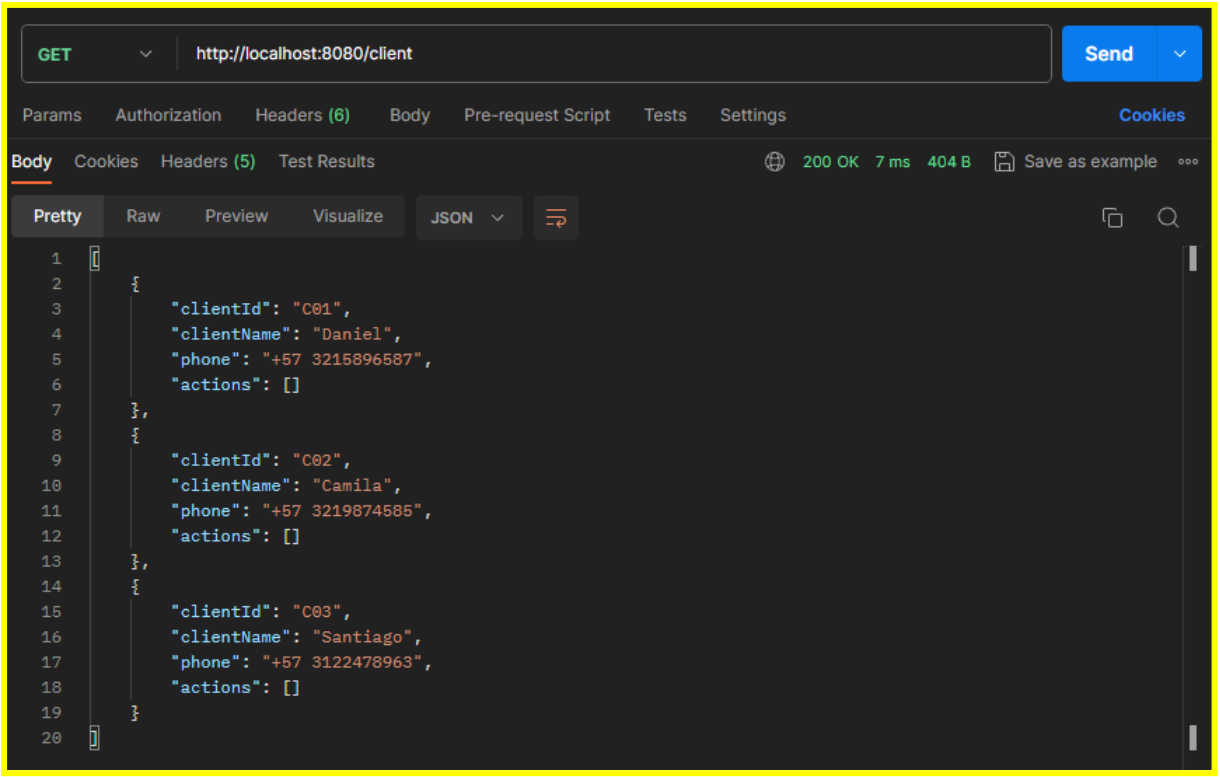


- **Diagrama de secuencia de agregar acciones**

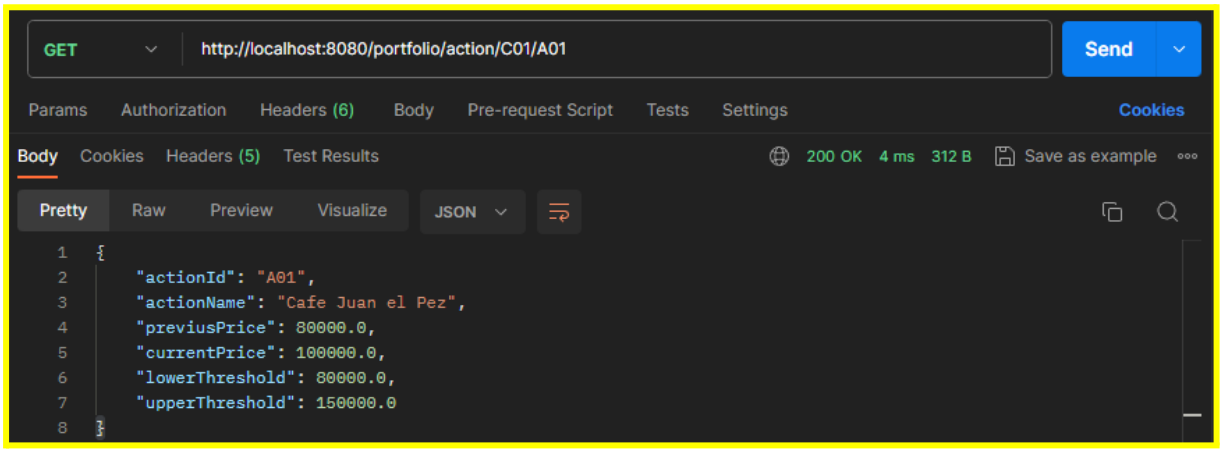


Evidencia consultas en Postman:

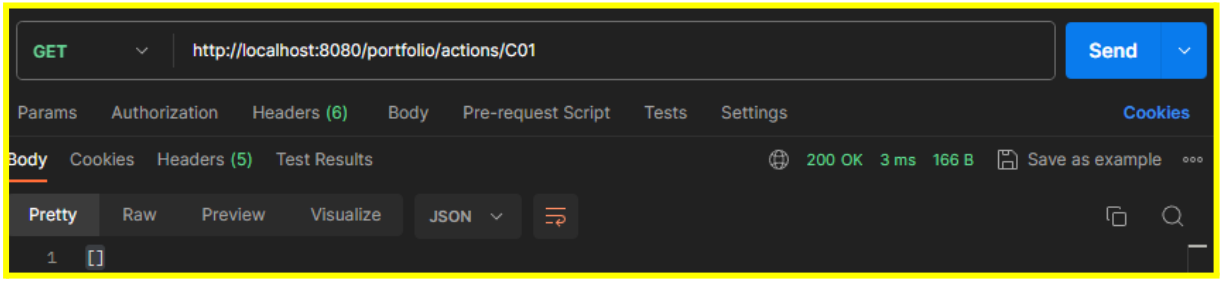
- **get clients:**



- **get client:**

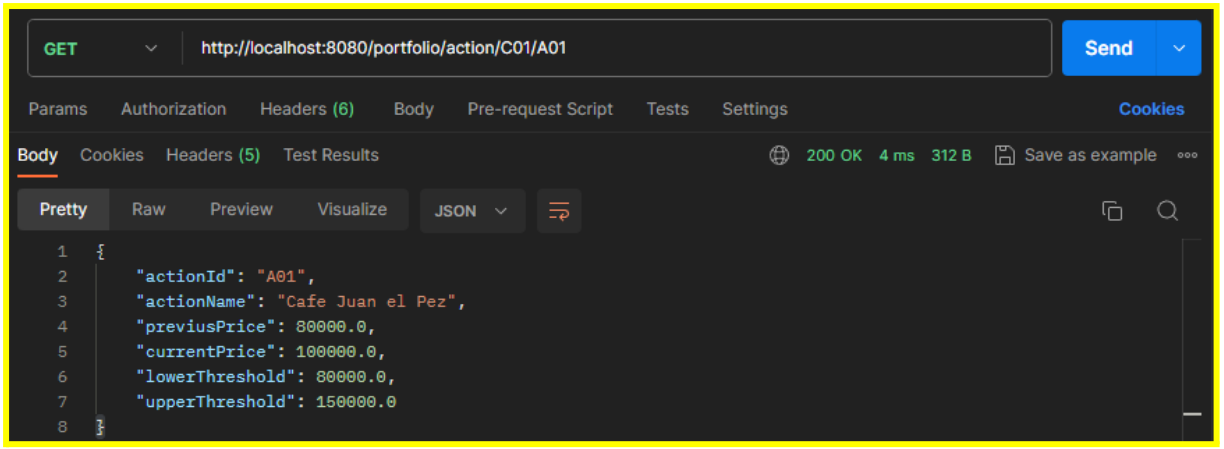


- **get client actions:**

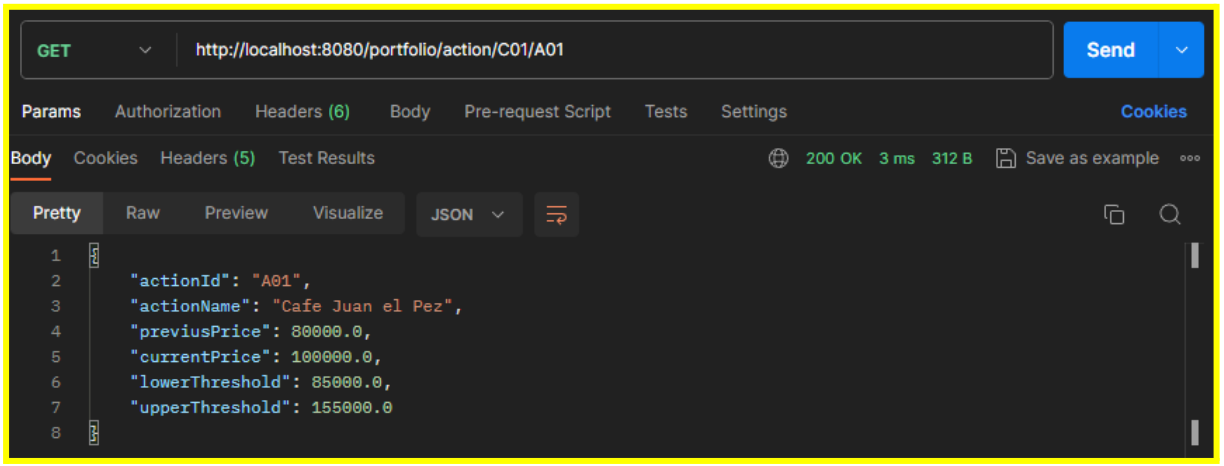


- **post client action:**

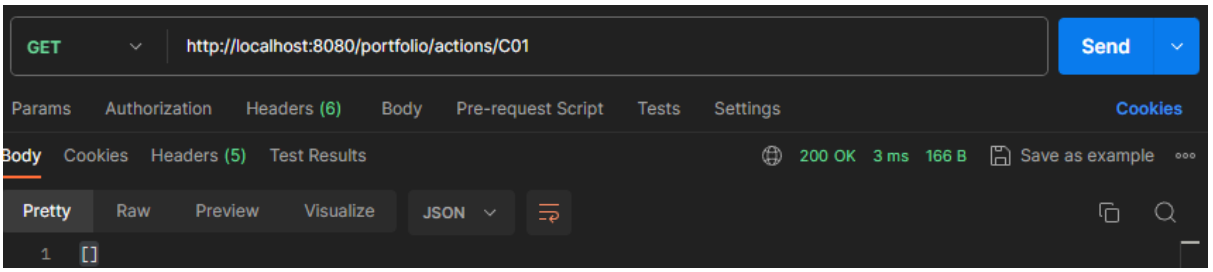
- **get client action:**



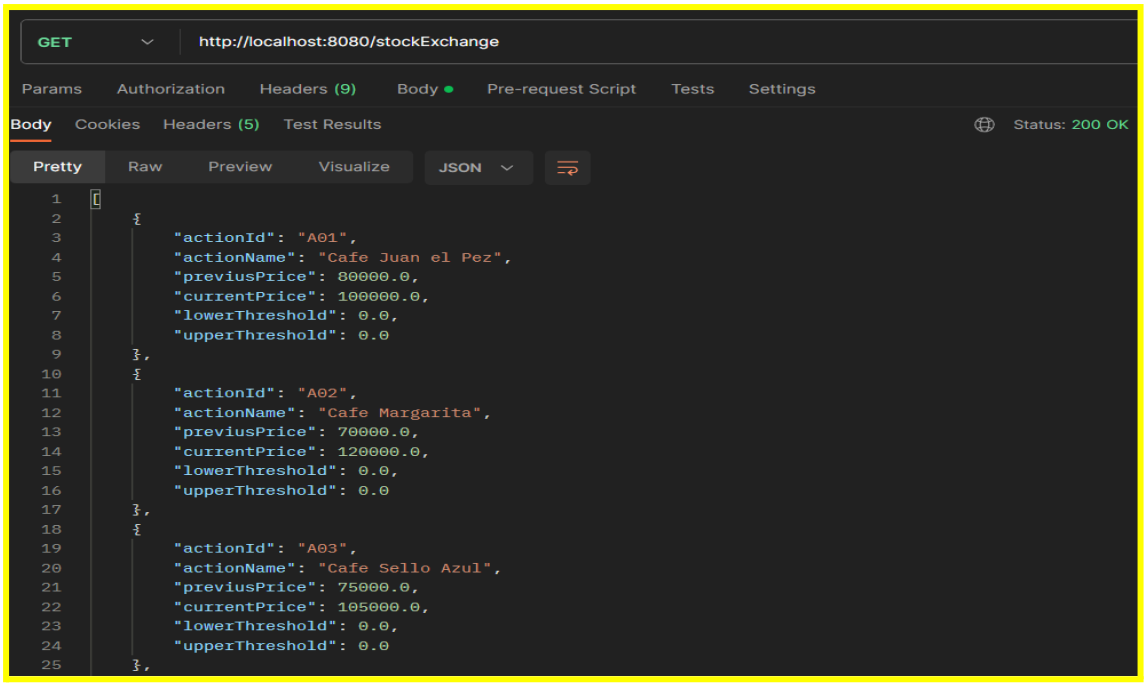
- **put client action:**



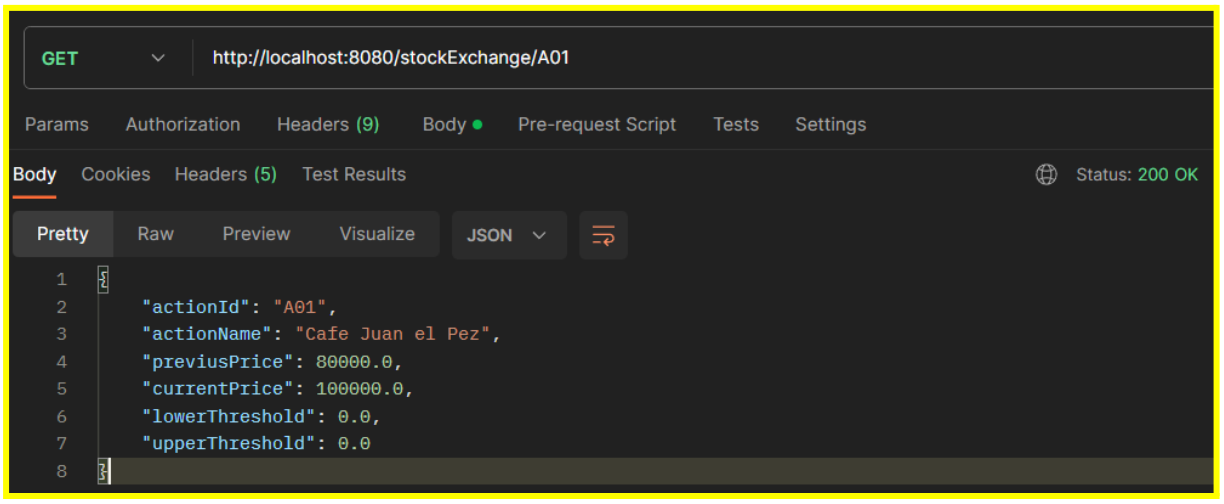
- **del client action:**



- **get global actions:**



- **get global action:**



Enlace Swagger

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

POST:

Curl

```
curl -X 'POST' \
  'http://localhost:8080/portfolio/addAction/c03/a05/80000/256000' \
  -H 'accept: */*' \
  -d ''
```

Request URL

```
http://localhost:8080/portfolio/addAction/c03/a05/80000/256000
```

Server response

Code	Details
200	Response body

```
Action added successfully
```

Download

Response headers

```
connection: keep-alive
content-length: 25
content-type: text/plain;charset=UTF-8
date: Wed,15 Nov 2023 21:31:14 GMT
keep-alive: timeout=60
```

Responses

PUT:

Curl

```
curl -X 'PUT' \
  'http://localhost:8080/portfolio/updateAction/c03/a05/70000/257000' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:8080/portfolio/updateAction/c03/a05/70000/257000
```

Server response

Code	Details
200	Response body

```
Action thresholds updated successfully
```

Download

Response headers

```
connection: keep-alive
content-length: 38
content-type: text/plain;charset=UTF-8
date: Wed,15 Nov 2023 21:32:35 GMT
keep-alive: timeout=60
```

Responses

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

Curl

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

Request URL

```
http://localhost:8080/portfolio/actions/c03
```

Server response

Code	Details
200	Response body

```
[
  {
    "actionId": "A05",
    "actionName": "Cafe El Colombiano",
    "previusPrice": 90000,
    "currentPrice": 115000,
    "lowerThreshold": 70000,
    "upperThreshold": 257000
  }
]
```

Download

Response headers

```
connection: keep-alive
content-type: application/json
date: Wed,15 Nov 2023 21:34:27 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

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

Curl

```
curl -X 'GET' \
  'http://localhost:8080/portfolio/action/c03/a05' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:8080/portfolio/action/c03/a05
```

Server response

Code	Details
200	Response body

```
{
  "actionId": "A05",
  "actionName": "Cafe El Colombiano",
  "previusPrice": 90000,
  "currentPrice": 115000,
  "lowerThreshold": 70000,
  "upperThreshold": 257000
}
```

Download

Response headers

```
connection: keep-alive
content-type: application/json
date: Wed,15 Nov 2023 21:37:29 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Responses

DELETE:

Curl

```
curl -X 'DELETE' \
  'http://localhost:8080/portfolio/deleteAction/c03/a05' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:8080/portfolio/deleteAction/c03/a05
```

Server response

Code	Details
200	Response body

```
Action deleted successfully
```

Download

Response headers

```
connection: keep-alive
content-length: 27
content-type: text/plain;charset=UTF-8
date: Wed,15 Nov 2023 21:42:49 GMT
keep-alive: timeout=60
```

Responses

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

Curl

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

Request URL

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

Server response

Code	Details
200	Response body

```
[
  {
    "actionId": "A01",
    "actionName": "Cafe Juan el Pez",
    "previusPrice": 80000,
    "currentPrice": 100000,
    "lowerThreshold": 0,
    "upperThreshold": 0
  },
  {
    "actionId": "A02",
    "actionName": "Cafe Margarita",
    "previusPrice": 70000,
    "currentPrice": 120000,
    "lowerThreshold": 0,
    "upperThreshold": 0
  },
  {
    "actionId": "A03",
```

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