



ROOTSOFTWORKS

# PROYECTO ANDINA TRADING

*Su aliado en inversiones de alto impacto*

**Cliente:**

Universidad El Bosque

**Presentado por:**

Edison M. Beltran  
Johann F. Toncon

# Índice de Contenidos

**01** Contexto del proyecto

**02** Requerimientos

**03** Procesos de negocio

**04** Patrones de diseño

**05** Atributos de calidad

**06** Métricas

**07** EDT

**08** Cronograma

# índice de Contenidos

- |           |                          |           |                                   |
|-----------|--------------------------|-----------|-----------------------------------|
| <b>09</b> | Metodología SCRUM        | <b>13</b> | Usabilidad                        |
| <b>10</b> | Business Canvas Model    | <b>14</b> | Matriz de riesgos                 |
| <b>11</b> | Arquitectura de software | <b>15</b> | Recomendaciones del arquitecto IT |
| <b>12</b> | Diagramas UML            | <b>16</b> | Plan de calidad                   |

# **CONTEXTO**

## **Proyecto de Andina Trading**

El proyecto se centra en el desarrollo de un sistema de software para gestionar la transacción de acciones en una bolsa de valores con el nombre de Andina Trading. Este software le va a permitir a los comisionistas de bolsa asesorar a los inversionistas en la compra y venta de acciones en tiempo real. Los usuarios del sistema serán inversionistas, comisionistas, y empresas emisoras de acciones en los mercados de Ecuador, Perú, Venezuela y Colombia. La solución incluirá funcionalidades para la gestión de inversionistas, simulación del comportamiento de las acciones y un portal para que los inversionistas gestionen sus inversiones.

# CONTEXTO

## Proyecto de Andina Trading

- **Objetivo general**

- Desarrollar una plataforma de software que permita la gestión integral de las transacciones de acciones en una bolsa de valores, con funcionalidades avanzadas para la simulación, monitoreo y ejecución de operaciones bursátiles.

- **Objetivos específicos**

- Desarrollar un portal seguro de autenticación para inversionistas y comisionistas usando JWT.
- Implementar un módulo de simulación y proyección de precios de acciones.
- Desarrollar una API RESTful que soporte operaciones bursátiles en tiempo real.
- Garantizar la persistencia de datos de las transacciones utilizando una base de datos MySQL.
- Implementar un sistema de reportes para el registro y seguimiento de eventos clave del sistema.

# REQUERIMIENTOS

Con la mira sobre la ISO 29110 respecto a la calidad del ciclo de vida del software (Especificamente en la fase de análisis), identificamos los siguientes requerimientos con sus respectivas estimaciones:

- Requerimientos funcionales:
  - Permitir registrar nuevos inversionistas, y nuevos contratos con un comisionista. **(2 puntos)**
  - Para la gestión de órdenes de compra y venta y seguimiento de inversiones, se requiere de un portal protegido con un login, donde el inversionista pueda:
    - Iniciar sesión **(2 puntos)**
    - Revisar la rentabilidad de los movimientos del comisionista **(1 punto)**.
    - El inversionista podrá indicarle al comisionista órdenes de compra o venta en tiempo real. **(3 puntos)**
    - El inversionista debe poder ver la fluctuación de las acciones donde participa. **(2 puntos)**

# REQUERIMIENTOS

- Generar reportes con el consolidado de los movimientos financieros que se hicieron en un periodo de tiempo y demás cuestiones por inversionista, comisionista y empresa. **(3 puntos)**
- Crear y asignar ciudades y países y agregar una descripción de la situación económica. **(1 punto)**
- Crear reportes que logren permitir una trazabilidad de las acciones de cada usuario en el sistema, para efectos de auditorías. Estos logs se generan a partir de los siguientes eventos **(3 puntos)**:
  - Creación de usuario
  - Eliminación de usuario
  - Operación de compra
  - Operación de venta
  - Cambio de configuración
  - Intento fallido de inicio de sesión
- Crear archivos de backup cada cierto tiempo, en una carpeta específica, y un apartado para importar o cargar un archivo de backup **(3 puntos)**
- Conectarse con la bolsa de valores de cada ciudad por cualquier medio, y poder ver en tiempo real los movimientos de la bolsa seleccionada **(5 puntos)**.
- Un portal pensado para el comisionista donde él podrá realizar las órdenes de compra o venta de acciones en la bolsa seleccionada, de acuerdo a las indicaciones del inversionista:
- Ver las indicaciones del inversionista en tiempo real. **(2 puntos)**
- Realizar órdenes de compra / venta **(3 puntos)**
- El software debe permitir generar consolidados donde se evidencie la información de los inversionistas, valor por acción, empresas accionistas, situación financiera de las empresas y comisiones por comisionista. **(3 puntos)**

# REQUERIMIENTOS

- Requerimientos no funcionales:
  - Colores a utilizar: Fondo blanco (#f2f7f1), Botones verdes (#2d6a4f), Menús Blancos (#FFFFFF)
  - Compatible con Windows 10 o superior
  - Lenguaje de programación Java 8 o superior
  - No debe consumir más de 1GB de RAM
  - No debe pesar más de 1 GB



# PROCESOS DE NEGOCIO

## Contabilidad y finanzas

- Registro de transacciones
- Valoración de carteras
- Generación de reportes financieros

## Gestión de recursos humanos

- Registro de inversionistas y comisionistas
- Gestión de contratos
- Servicio al cliente

## Ventas / Compras

- Recepción de órdenes
- Ejecución de órdenes
- Gestión del riesgo



# PROCESOS DE NEGOCIO

## Auditoría

- Auditoría interna
- Monitoreo al cumplimiento normativo

## Sistemas de información

- Gestión de la seguridad
- Integraciones con las bolsas de valores

# PATRONES DE DISEÑO

Para evitar repetir código y matarnos la cabeza con problemas que ya otros han resuelto, implementamos los siguientes patrones de diseño

## MVC

En Spring Boot, la estructura estándar facilita el uso de MVC para la separación de lógica de negocio (servicios), acceso a datos (repositorios) y presentación (controladores y vistas).

## DAO

Se usa para gestionar la interacción con la base de datos, proporcionando una interfaz clara para las operaciones CRUD (repositorios) y separando la lógica de acceso a datos del resto de la aplicación.

## DTO

En una aplicación con múltiples capas, los DTO son útiles para enviar datos entre la capa de servicio y la capa de presentación, evitando exponer directamente las entidades.

# PATRONES DE DISEÑO

Para evitar repetir código y matarnos la cabeza con problemas que ya otros han resuelto, implementamos los siguientes patrones de diseño

## Factory Method

Este patrón es útil cuando se necesita crear objetos con lógica específica, evitando instancias directas. Aunque se usa principalmente la inyección de dependencias para controlar la creación de instancias, puede aplicarse en situaciones donde se requieren instancias con configuraciones específicas, como generación de PDFs.

## Facade

La capa de servicio actúa como una fachada al centralizar y simplificar la lógica de negocio, evitando que los controladores accedan directamente a repositorios o lógica compleja. Esta capa coordina las operaciones necesarias en el sistema de manera centralizada.

## Command

Este patrón nos es útil a la hora de gestionar las transacciones, ya que nos enseña a encapsular las solicitudes como objetos, para tener la posibilidad de deshacerlas, ejecutarlas después o almacenarlas.

# ATRIBUTOS DE CALIDAD

- **Seguridad:** Uso de Spring Security con JWT para garantizar que solo usuarios autenticados puedan acceder a recursos privados. Además, la autorización basada en roles para diferenciar entre usuarios regulares y posibles administradores.
- **Escalabilidad:** La arquitectura del sistema permitirá un crecimiento modular, soportando la adición de nuevos países y mercados.
- **Usabilidad:** La interfaz de usuario estará diseñada para ser intuitiva, permitiendo a los usuarios interactuar fácilmente con la plataforma en dispositivos móviles y de escritorio.



# MÉTRICAS

Basadas en puntos de función

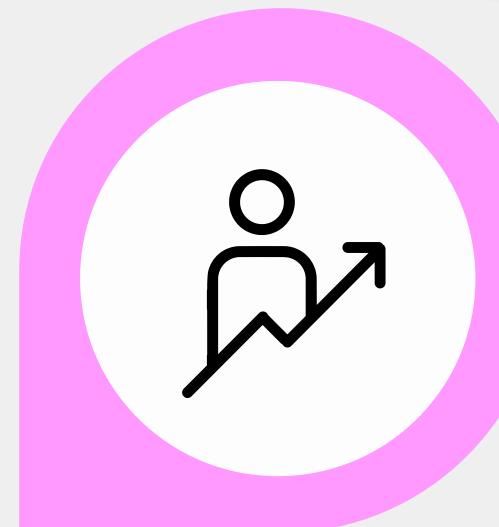
## Entradas Externas (EE)

- Registro de inversionistas
- Generación de contratos
- Emisión de órdenes de compra/venta



## Salidas Externas (SE)

- Generación de reportes
- Confirmación de órdenes



## Consultas Externas (CE)

- Estado de inversionista
- Consulta de transacciones en tiempo real

## Archivos Lógicos Internos (ALI)

- Base de datos de usuarios
- Base de datos de transacciones



## Archivos de Interfaz Externos (AIE)

- Interfaz con las bolsas de valores

# COMPLEJIDAD DEL SOFTWARE

Ponderación según los puntos de función

Componente	Cantidad	Complejidad	Factor de Ponderación	Puntos de Función
Entradas Externas (EE)	3	Media	4	12
Salidas Externas (SE)	2	Media	5	10
Consultas Externas (CE)	2	Baja	3	6
Archivos Lógicos Internos (ALI)	2	Alta	10	20
Archivos de Interfaz Externos (AIE)	1	Alta	7	7

**Total: 55 Puntos de Función**

# FACTORES DE AJUSTE DE VALOR (FAV)

Estos factores dependen de preguntas sobre el sistema que se califican de 0 a 5 según su relevancia

- ¿Requiere el sistema copias de seguridad y recuperación fiables? -> 4
- ¿Requiere comunicación de datos? -> 5
- ¿Existen funciones de procesamiento distribuido? -> 4
- ¿Es crítico el rendimiento? -> 5
- ¿Se ejecutará en un entorno operativo existente y fuertemente utilizado? -> 4
- ¿Requiere entrada de datos interactiva? -> 3
- ¿Requiere que las transacciones de entrada se lleven a cabo sobre múltiples pantallas? -> 3
- ¿Se actualizan los archivos maestros de forma interactiva? -> 4
- ¿Son complejas las entradas, salidas o peticiones? -> 4
- ¿Es complejo el procesamiento interno? -> 5
- ¿Se ha diseñado el código para ser reutilizable? -> 3
- ¿Se ha diseñado el sistema para soportar múltiples instalaciones? -> 2
- ¿Se ha diseñado la aplicación para facilitar los cambios? -> 3
- ¿Es fácil de usar para el usuario final? -> 4

Total: 53 Puntos

# CÁLCULO DE LOS PUNTOS DE FUNCIÓN

Fórmula:  $PF = \text{Conteo total} * [0,65 + 0,01 * \sum F_i]$

Desarrollo:

$$PF = 55 * [0,65 + (0,01 * 53)]$$

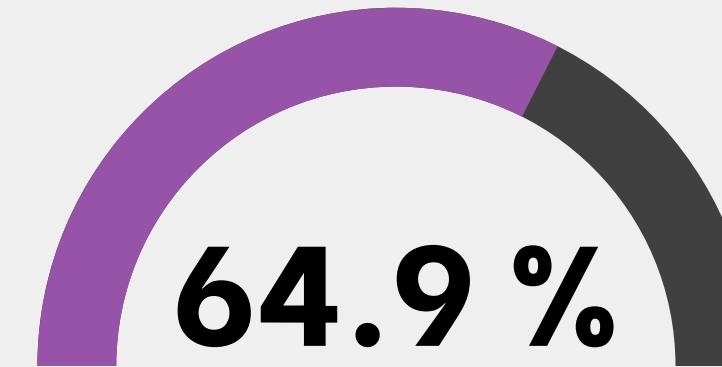
$$PF = 55 * [0,65 + 0,53]$$

$$PF = 55 * 1,18$$

$$PF = \mathbf{64.9}$$

# COMPLEJIDAD DEL SOFTWARE

Calculada luego de todo este trabajo con los puntos de función



## RESULTADO FINAL

El cálculo de 64.9 Puntos de Función (PF) para el sistema Andina Trading refleja la complejidad funcional del sistema (el puntaje indica que es moderadamente complejo), teniendo en cuenta las entradas, salidas, consultas, archivos internos y externos, así como los factores de ajuste que incluyen la criticidad del rendimiento, la necesidad de copias de seguridad, y la complejidad del procesamiento.

Este valor puede ser utilizado para estimar el esfuerzo, tiempo y costo del desarrollo del software, así como para comparar la complejidad de este sistema con otros proyectos similares.

# EDT



# Cronograma

	SEP	OCT	NOV
KAN-18 Levantamiento de requerimientos		■	
KAN-20 Acta de constituci...		■	
KAN-19 Cronograma		■	
KAN-23 Documento de arquitectura del softwa...		■	
KAN-24 Documento procesos de negocio		■	
KAN-25 Diagrama casos de u...		■	
KAN-26 Diagrama de despliegue		■	
KAN-29 Documento patrones de diseño		■	
KAN-27 Diagrama de component...		■	
KAN-28 Diagrama de clases		■	
KAN-30 Documento usabilidad		■	
KAN-36 WBS		■	
KAN-39 Diagramas de flujo		■	
KAN-31 Métricas del softwa...		■	
KAN-32 Análisis biopsicosoci...		■	
KAN-33 Recomendaciones Arquitecto...		■	

# SCRUM ROLES



## Product Owner

Nuestro PO debe ser una persona con amplio conocimiento en mercados bursátiles, que entienda las necesidades de los inversionistas y comisionistas



## Scrum Master

El SM debe ser alguien con experiencia en metodologías ágiles, comunicación efectiva y manejo de equipos. Será un facilitador del equipo.



## Equipo de desarrollo

Debe estar compuesto por ingenieros de software, diseñadores y especialistas en pruebas que tengan habilidades técnicas sólidas y capacidad de trabajo en equipo.

# ARTEFACTOS SCRUM

Elementos que proveen transparencia y oportunidades de inspección y adaptación en el proyecto.

## **Product Backlog**

El Product Backlog es una lista priorizada de todas las funcionalidades, mejoras y correcciones que se deben implementar en el sistema. Este backlog es dinámico y se actualiza constantemente a medida que el proyecto avanza.

## **Sprint Backlog**

Contiene los elementos del Product Backlog seleccionados para ser trabajados durante un sprint. También incluye las tareas técnicas que el equipo debe realizar para completar dichos elementos.

## **Incremento**

El Incremento es el producto funcional y completo que se entrega al final de cada sprint. En Andina Trading, los incrementos pueden incluir módulos completamente desarrollados o mejoras que agreguen valor al sistema.



# CEREMONIAS SCRUM

## Sprint

Se propone que los sprints para "Andina Trading" duren 3 semanas, lo que permitirá un equilibrio entre desarrollo y revisión de funcionalidad.

## Sprint planning

Al inicio de cada sprint, el equipo, junto con el Product Owner, selecciona los elementos del Product Backlog que se trabajarán durante el sprint.

## Daily SCRUM

Reunión diaria de 15 minutos donde cada miembro del equipo responde a las siguientes preguntas:

- ¿Qué hice ayer?
- ¿Qué haré hoy?
- ¿Qué impedimentos tengo?



# CEREMONIAS SCRUM

## Sprint Review

Al final de cada sprint, el equipo presenta el incremento de software al Product Owner y otros interesados para recibir retroalimentación. Se evalúa si se han cumplido los objetivos del sprint y obtener sugerencias para mejorar en el siguiente.

## Sprint Retrospective

Reunión que se realiza al final de cada sprint para reflexionar sobre el trabajo realizado y proponer mejoras en los procesos y la colaboración del equipo. Para mejorar continuamente la forma de trabajar y detectar áreas de oportunidad.

# BUSINESS MODEL CANVAS



# ARQUITECTURA DEL SOFTWARE

- Ya que el proyecto "Andina Trading" tiene que manejar múltiples módulos y funcionalidades interconectadas (como el portal de inversionistas, la gestión de transacciones en tiempo real y la generación de reportes), se ha decidido utilizar una arquitectura basada en microservicios, esta arquitectura permite que cada funcionalidad o módulo sea independiente, facilitando su desarrollo, escalabilidad, despliegue y mantenimiento.



# DISEÑO DE LA ARQUITECTURA

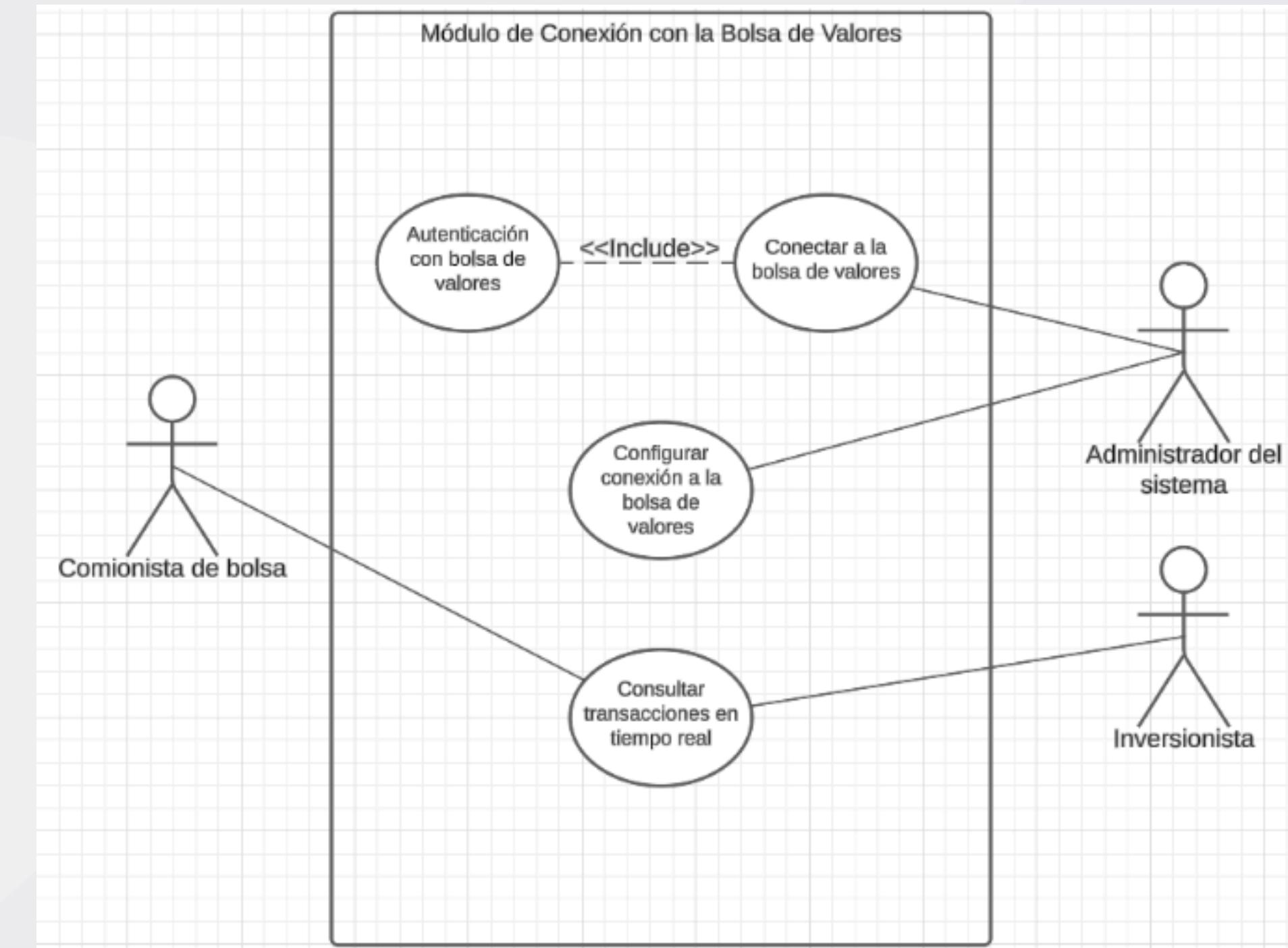
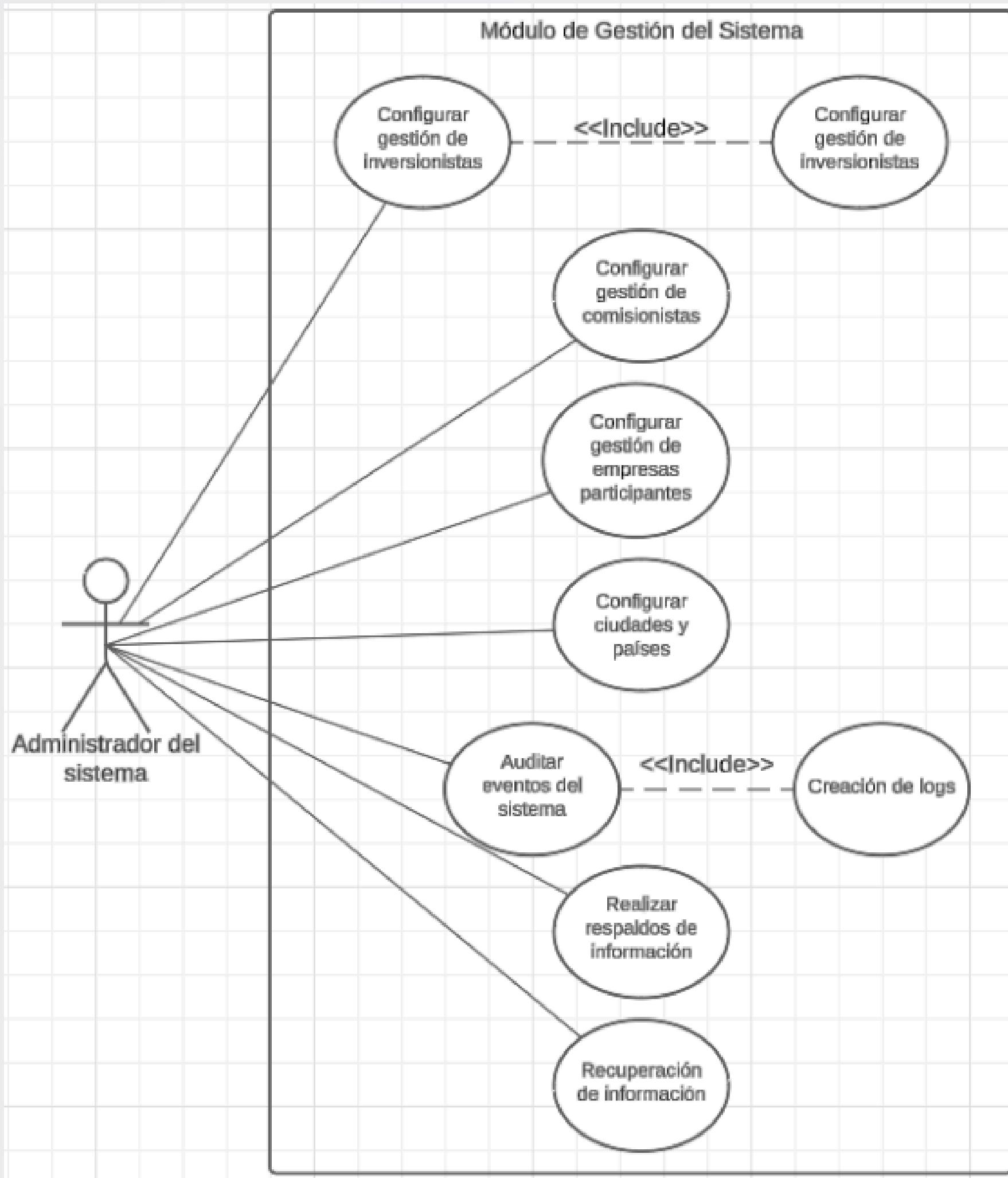


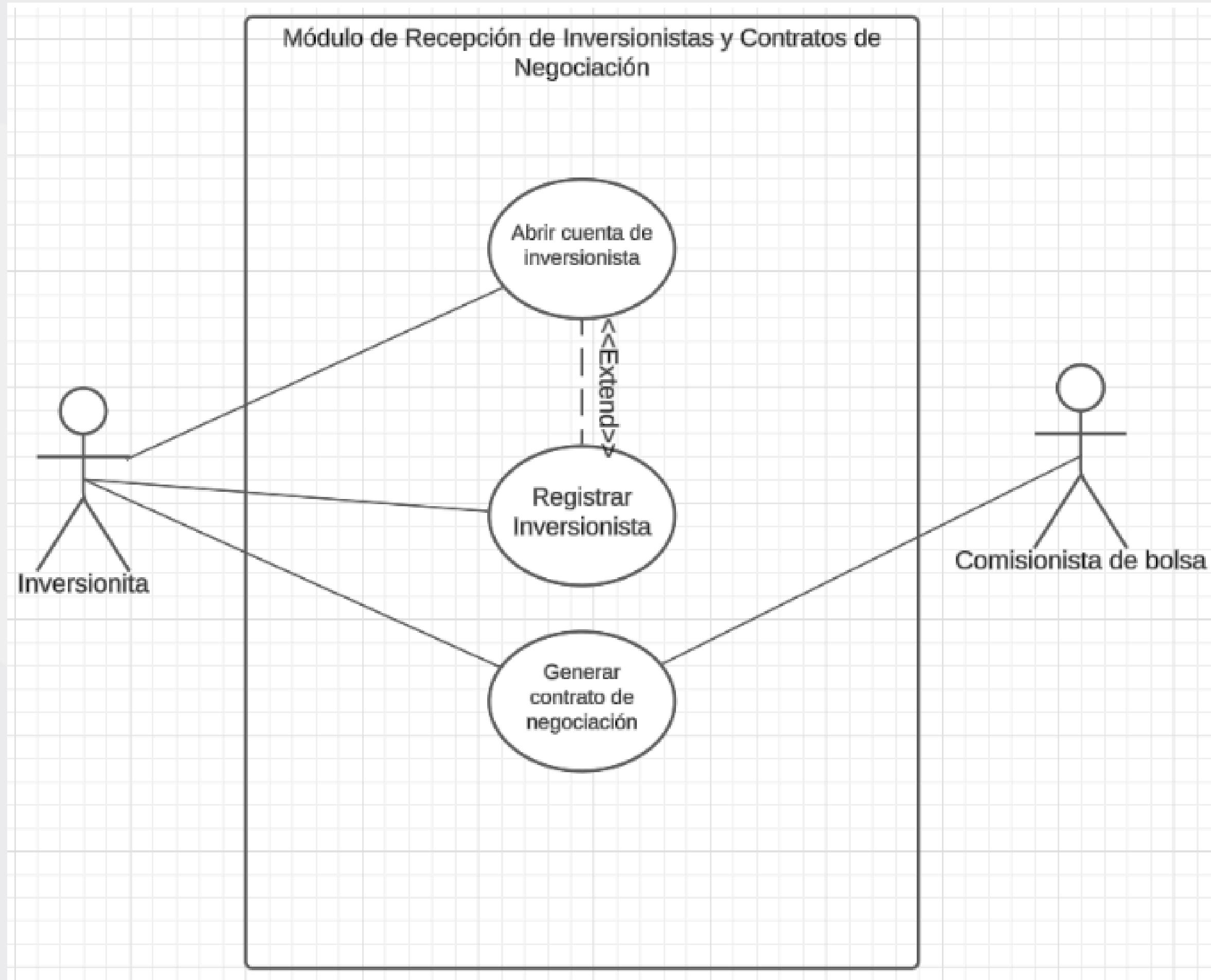
- **Módulo de Gestión de Inversionistas y Comisionistas:** Maneja la información y configuración de los actores principales del sistema.
- **Módulo de Conexión con la Bolsa de Valores:** Se encarga de la conexión en tiempo real con las diferentes bolsas y mercados financieros.
- **Portal del Inversionista:** Proporciona la interfaz para que los inversionistas gestionen sus órdenes y vean el rendimiento de sus acciones.
- **Módulo de Transacciones:** Gestiona las órdenes de compra/venta y las ejecuta en el mercado.
- **Módulo de Reportes:** Genera reportes sobre las actividades financieras y comisiones.
- **API Gateway:** Actúa como punto de entrada unificado para todos los microservicios, permitiendo un control centralizado de las peticiones y el acceso a cada módulo.
- **Servicio de Autenticación y Autorización:** Controla el acceso seguro de usuarios y roles al sistema.

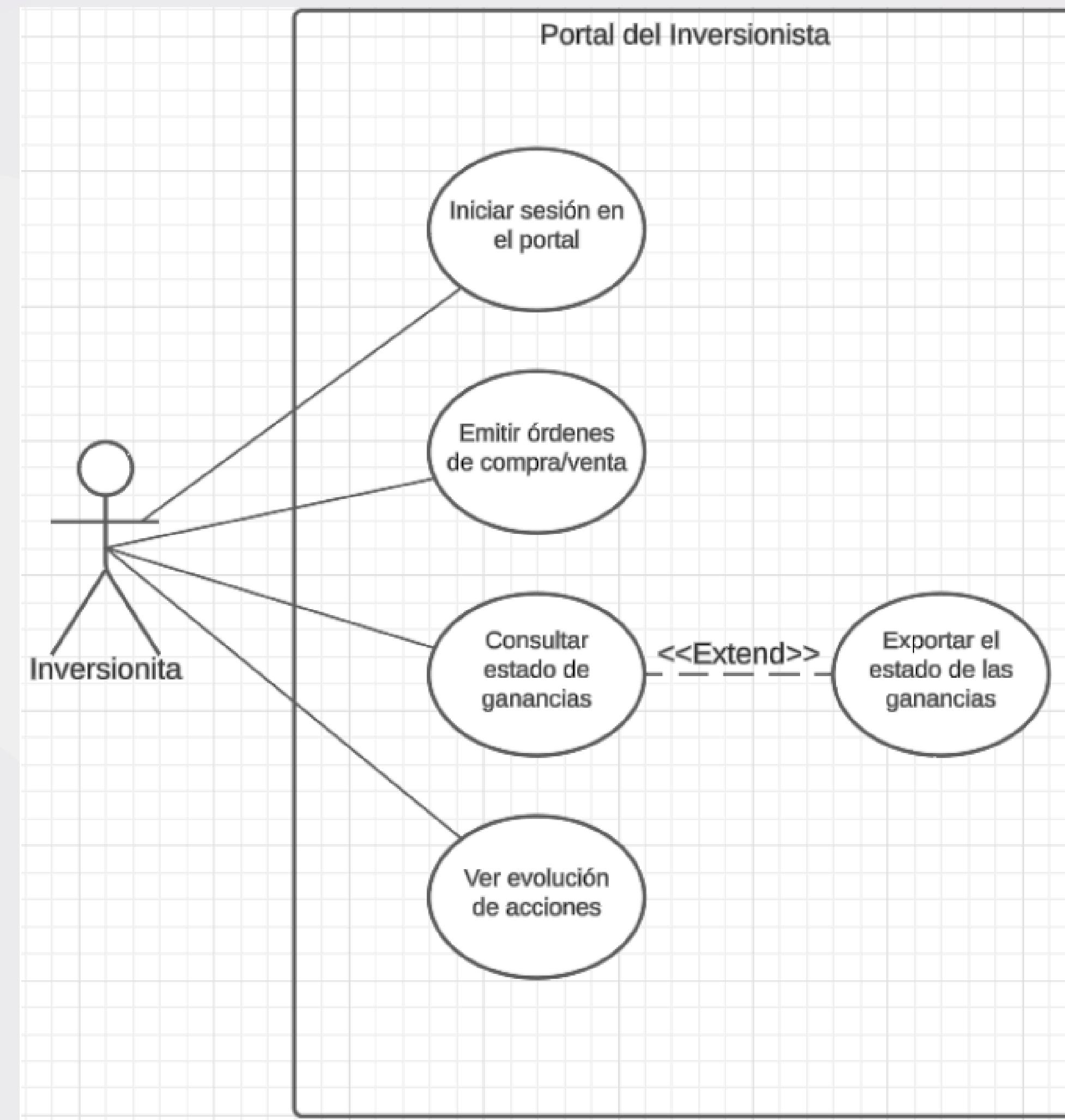
# **DIAGRAMAS UML**

## **Casos de uso**

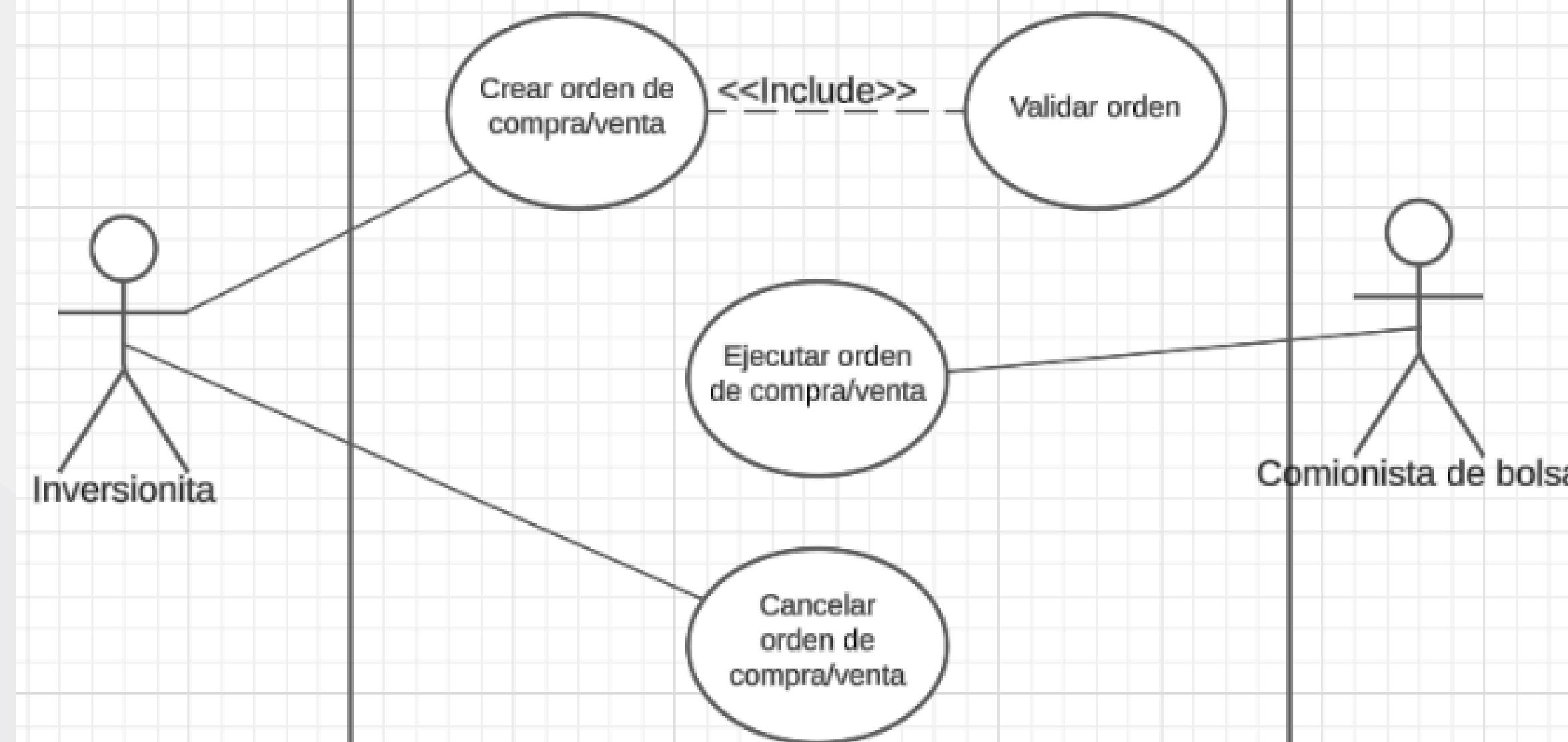
Para poder mostrar la interacción de los diferentes usuarios (administrador del sistema, inversionista y comisionista de bolsa) con el sistema por medio de 7 diagramas de casos de uso que representan cada uno de los módulos.



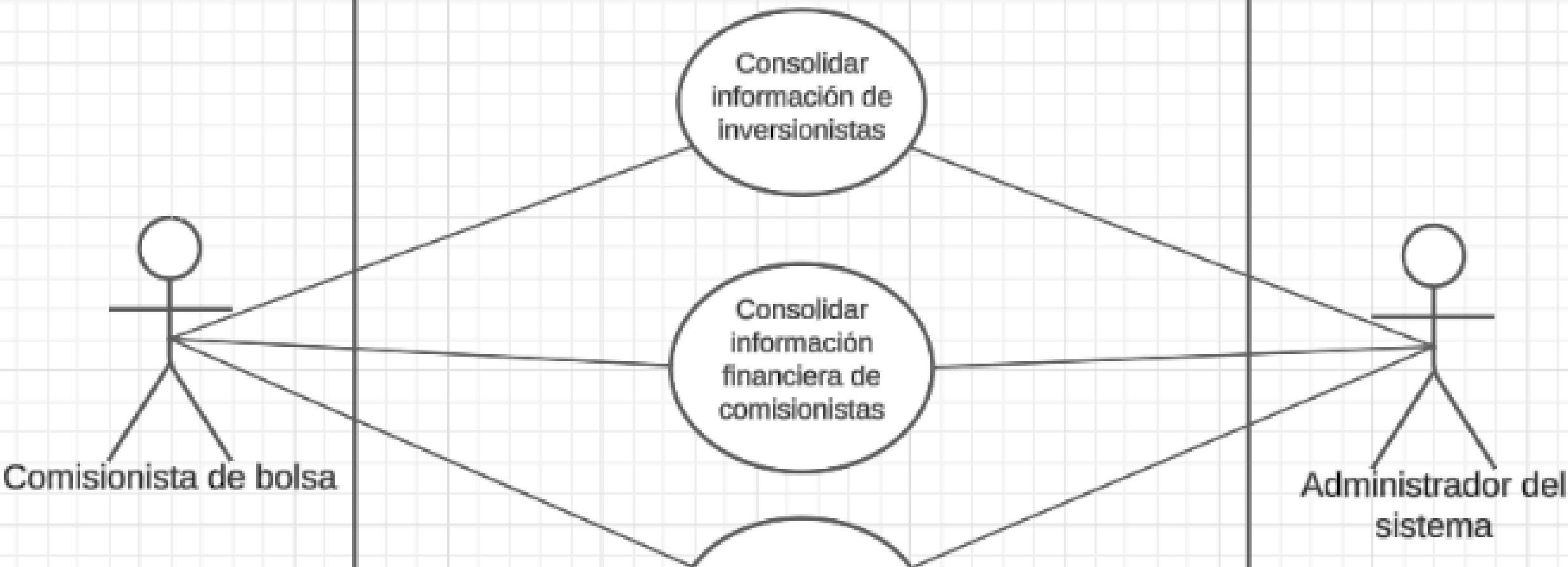


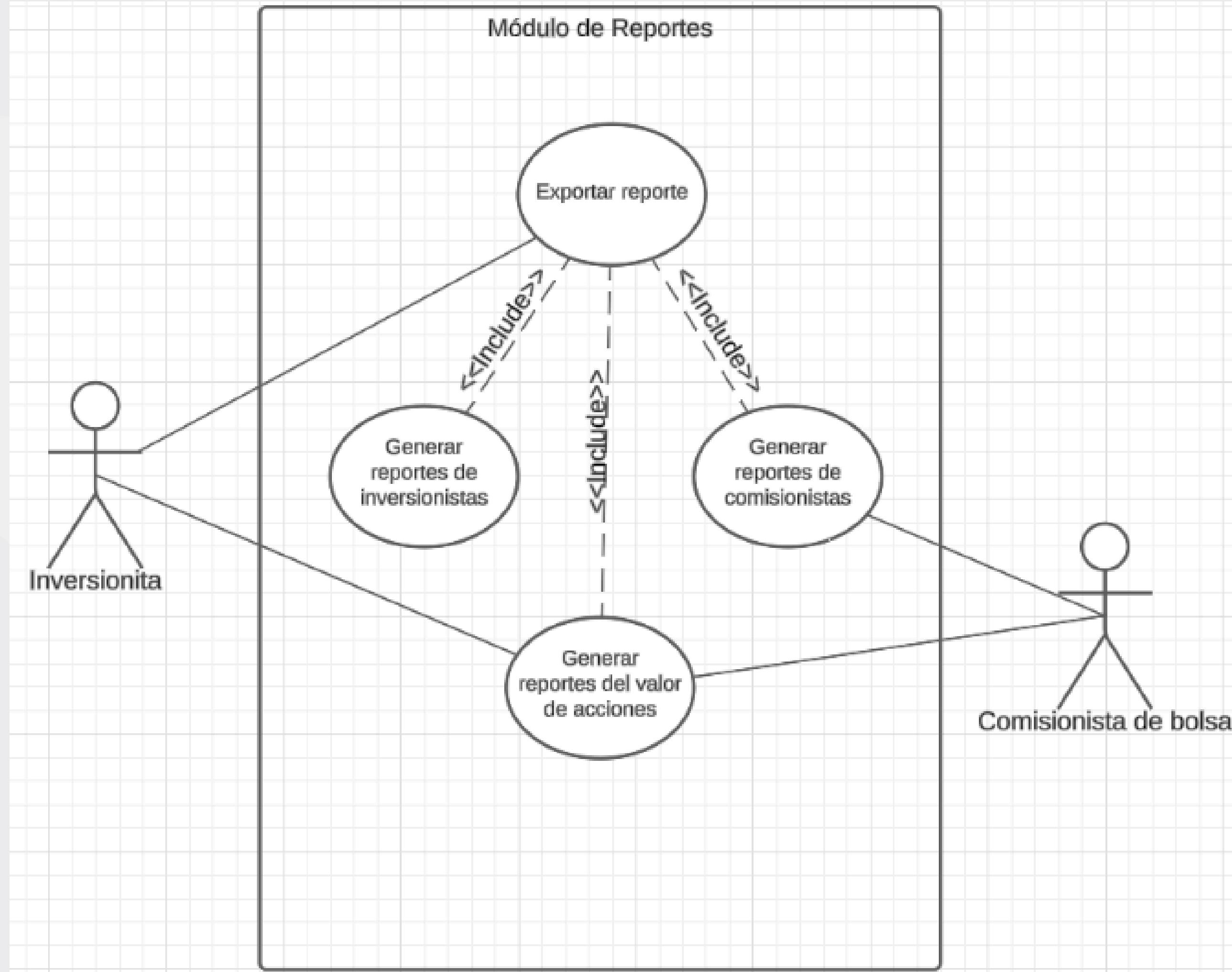


Módulo de Generación de Órdenes de Compra/Venta en la  
Bolsa de Valores



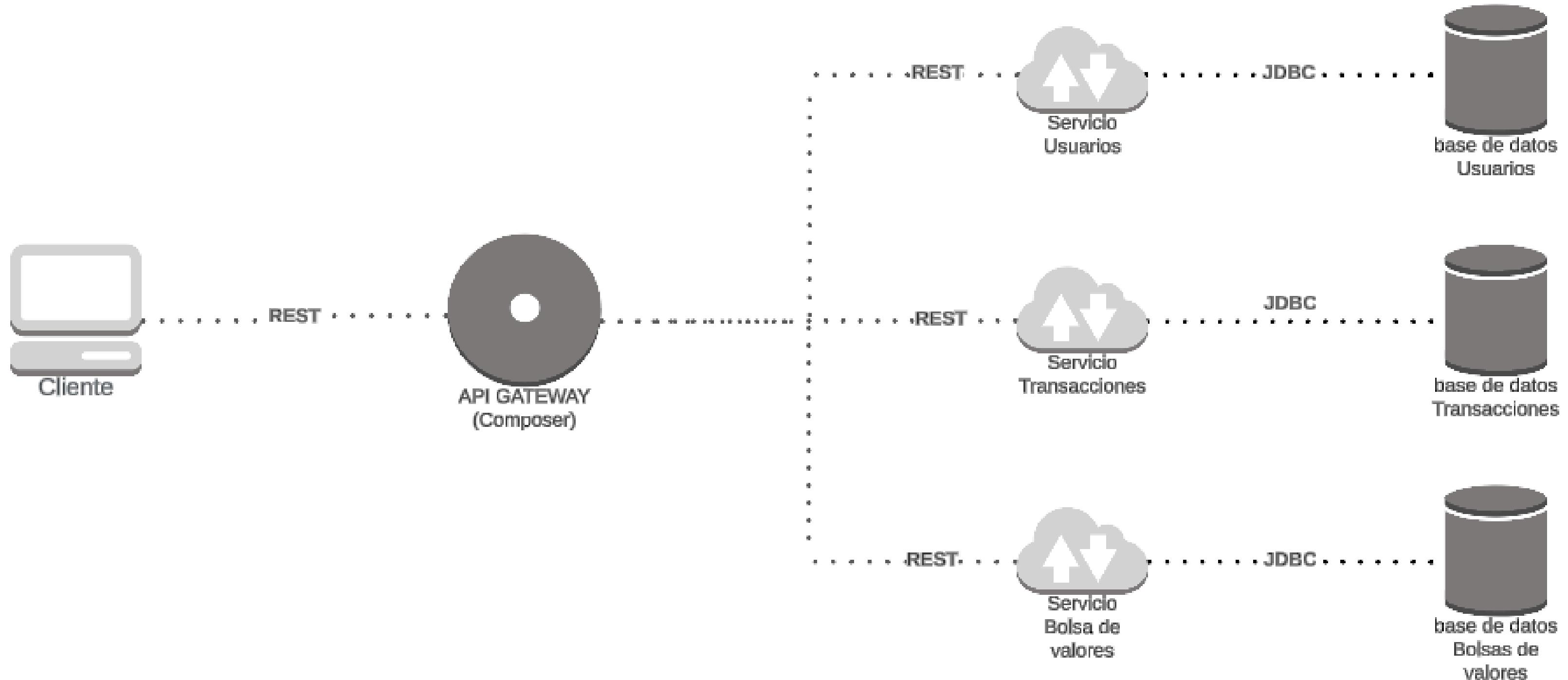
## Módulo de Consolidación de Información





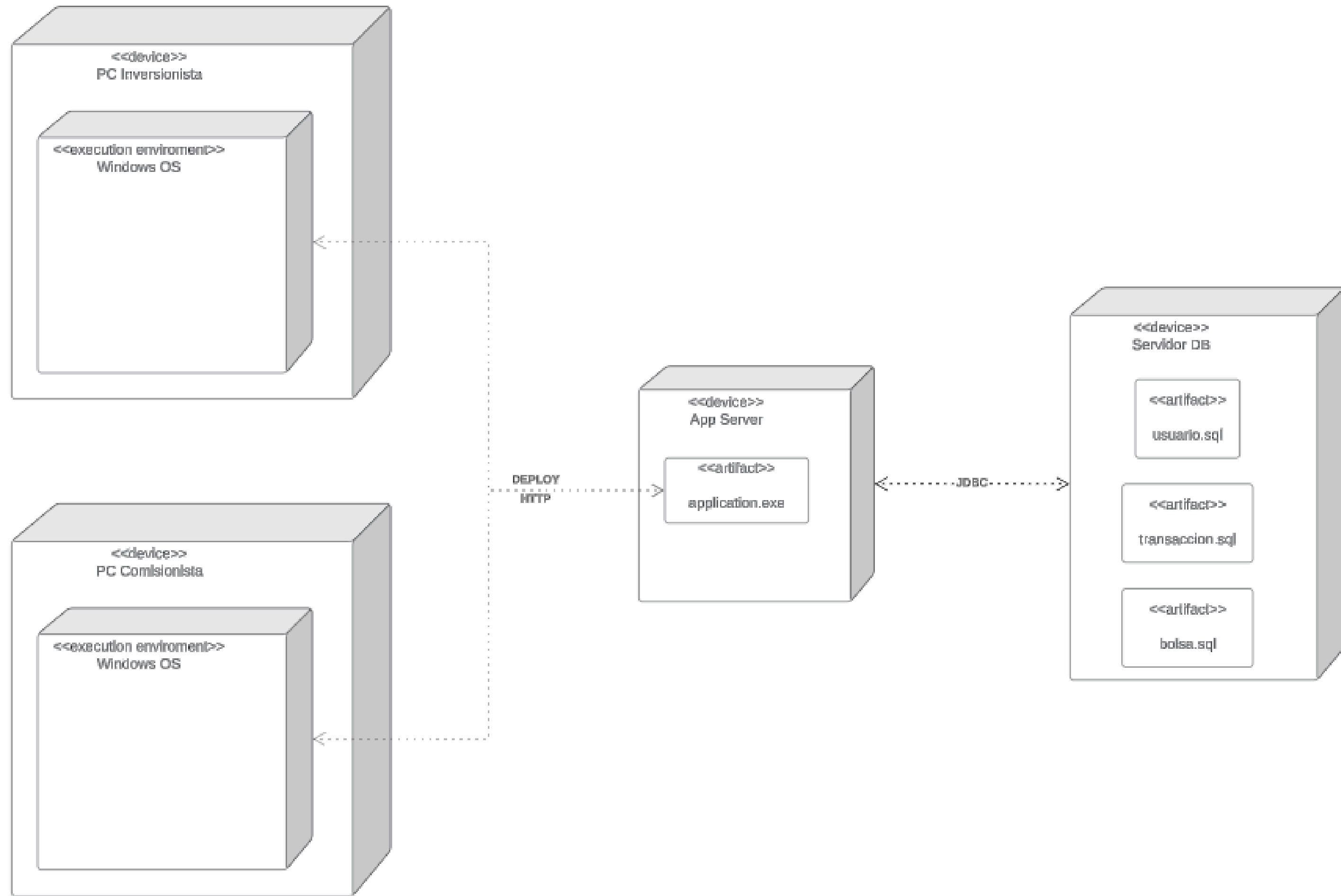
# **DIAGRAMAS UML**

**Componentes**



# **DIAGRAMAS UML**

**Despliegue**



# **DIAGRAMAS UML**

**Diagrama de clases**

**(Ver documento del proyecto)**

# **DIAGRAMAS UML**

**Diagramas de flujo**

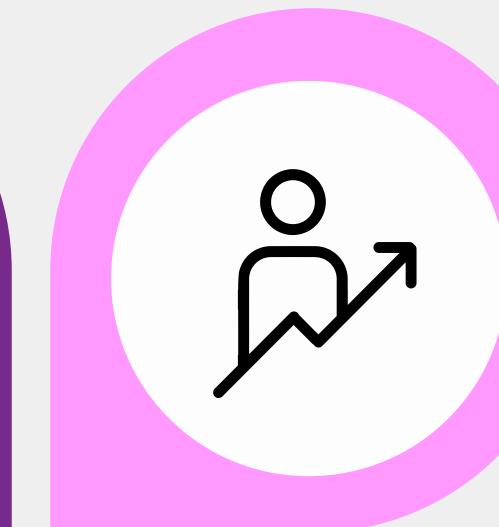
**(Ver documento del proyecto)**

# USABILIDAD

ISO 9126

## Facilidad de aprendizaje

- Nuestro software será diseñado pensando en lo que hace el cliente, no en lo que dice (Diseño centrado en el usuario)
- - 40% de gastos en capacitaciones



## Facilidad de uso

- lograremos reducir la tasa de errores hasta aproximadamente 50% ya que los procesos de negocio están mapeados de la forma más eficiente posible, mejorando la eficacia de los empleados.



## Robustez

- Nuestro software asegura la integridad, disponibilidad y confidencialidad de la información con una arquitectura bien definida pensada para apoyar los procesos de negocio, entregando valor al cliente.

## Satisfacción

- Al aplicar estos principios de la usabilidad, el usuario no se verá frustrado o detenido por procesos complejos o indocumentados.

# **USABILIDAD**

**Mockups**

**(Ver documento del proyecto)**

# **MATRIZ DE RIESGOS**

**(Ver documento del proyecto)**

# RECOMENDACIONES

## ARQUITECTOS DE IT

### Implementación de un API Gateway

Para gestionar las peticiones hacia los microservicios y centralizar el acceso, se recomienda la implementación de un API Gateway ya que este componente actúa como un punto de entrada único para todas las solicitudes externas, permitiendo:

- **Seguridad centralizada:** Autenticación y autorización de usuarios en un único lugar.
- **Gestión de peticiones:** Reducción de la carga en los microservicios al filtrar y enrutar peticiones de manera eficiente.
- **Monitoreo y logging:** Un API Gateway facilita la captura de datos sobre el tráfico y el rendimiento del sistema, ayudando en la detección de fallos y en el monitoreo general.

# RECOMENDACIONES

## ARQUITECTOS DE IT

### Recomendaciones para Seguridad

Dado que el sistema va a manejar información financiera y personal sensible, se deben implementar las siguientes medidas de seguridad:

- **Cifrado de Datos:** Todos los datos sensibles, como las transacciones y la información de los usuarios, deben ser cifrados tanto en tránsito (usando HTTPS/TLS) como en reposo.
- **Autenticación y Autorización:** Utilizar tecnologías como Spring Security junto con JWT (JSON Web Tokens) para asegurar que solo usuarios autorizados accedan a las funciones críticas del sistema.
- **Monitoreo y Auditoría:** Implementar un sistema de monitoreo continuo que registre las actividades en el sistema para detectar posibles amenazas y ataques en tiempo real.

# RECOMENDACIONES

## ARQUITECTOS DE IT

### Adopción de Buenas Prácticas de Programación

- **Documentación del Código:** Cada microservicio y componente debe contar con una documentación clara que detalle su propósito, funciones y conexiones con otros módulos.
- **Revisión de Código:** Implementar un sistema de revisión de código en equipo, donde los desarrolladores revisen y comenten el trabajo de sus compañeros para asegurar la calidad y detectar errores.
- **Uso de Principios SOLID:** Aplicar principios como el de Responsabilidad Única y el de Inversión de Dependencias para asegurar que el código sea modular, fácil de mantener y ampliar.

# RECOMENDACIONES

## ARQUITECTOS DE IT

### **Manejo eficiente de datos y transacciones**

El sistema de Andina Trading va a procesar un alto volumen de transacciones en tiempo real, por lo cual se recomienda:

- **Implementar transacciones ACID** (Atomicidad, Consistencia, Aislamiento, Durabilidad) en los microservicios críticos para asegurar que las operaciones en la bolsa se realicen de manera segura y sin inconsistencias.

# **PLAN DE CALIDAD**

## **ISO 10005**

**(Ver documento del proyecto)**



Borcelle

# MUCHAS GRACIAS