



UNIVERSIDAD EL BOSQUE

FACULTAD DE INGENIERÍA

INGENIERÍA DE SOFTWARE 2

ING. CARLOS HERNAN MEDINA AYALA

PROYECTO FINAL

PROYECTO BOLSA DE VALORES

EDISON MAURICIO BELTRÁN

JOHANN FELIPE TONCON

2024

ÍNDICE

- 1. Contexto del proyecto**
 - 1.1. Descripción del proyecto**
 - 1.2. Alcance del proyecto**
 - 1.3. Objetivos del proyecto**
 - 1.3.1. Objetivo general**
 - 1.3.2. Objetivos específicos**
- 2. Análisis desde el modelo biopsicosocial y cultural**
- 3. Levantamiento de requerimientos (ISO 29110)**
 - 3.1. Requerimientos funcionales**
 - 3.2. Requerimientos no funcionales**
- 4. Identificación de procesos de negocio que apoya**
- 5. Patrones de diseño**
- 6. Atributos de calidad del software**
- 7. Métricas**
- 8. EDT**
- 9. Cronograma**
- 10. Implementación de SCRUM**
- 11. Modelo canvas**
- 12. Arquitectura de software**
- 13. UML**
 - 13.1. Diagramas de casos de uso**
 - 13.2. Diagrama de componentes**
 - 13.3. Diagrama de despliegue**

13.4. Diagrama de clases

13.5. Diagramas de flujo

14. Usabilidad

15. Matriz de riesgos

16. Recomendaciones del arquitecto de TI

17. Plan de calidad

18. Referencias

1. Contexto del proyecto

1.1. Descripción del proyecto

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.

El sistema será implementado utilizando Spring Boot para el backend, Angular para el frontend, y se empleará JWT para la autenticación y autorización de usuarios. La persistencia de los datos se manejará con Spring Data JPA y una base de datos relacional (MySQL), asegurando un manejo eficiente de grandes volúmenes de transacciones bursátiles.

1.2. Alcance del proyecto

Este proyecto abarca el desarrollo de una plataforma que permitirá a los comisionistas de bolsa gestionar las transacciones bursátiles de sus clientes, proporcionando a los inversionistas una interfaz donde podrán visualizar la evolución de sus inversiones, realizar órdenes de compra/venta, y recibir reportes personalizados.

Se prevé la conexión del sistema con las bolsas de valores de los diferentes países en los que operará, garantizando transacciones en tiempo real. Además, se integrarán mecanismos de auditoría y recuperación de datos para asegurar la disponibilidad y fiabilidad del sistema.

1.3. Objetivos del proyecto

1.3.1. 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.

1.3.2. 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 auditoría para el registro y seguimiento de eventos clave del sistema.

2. Análisis desde el modelo biopsicosocial y cultural

Este modelo analiza la interacción entre factores biológicos, psicológicos, sociales y culturales en el desarrollo de sistemas de software, lo que permite adaptar el proyecto a los usuarios y contextos en los que se implementará.

El enfoque biopsicosocial y cultural abarca cuatro componentes clave: cultura, hábitos, medio y artefactos. En este análisis, se explorará cómo estos componentes afectan el diseño, desarrollo y la usabilidad del sistema, así como su capacidad para adaptarse a las particularidades del contexto en el que operará.

Componentes del modelo biopsicosocial y cultural

2.1 Cultura

La cultura, en el contexto de un proyecto de software como Andina Trading tiene un impacto significativo en la forma en que los usuarios perciben y utilizan el sistema, ya que se trata de un sistema que operará en varios países (Ecuador, Perú, Venezuela y Colombia), es crucial que el software se adapte a las diferencias culturales de cada región. Estos son algunos puntos clave relacionados con la cultura:

- **Lenguaje y localización:** El sistema debe estar disponible en español, pero también adaptarse a los modismos y terminología específica de cada país en relación con el mercado financiero. Esto mejorará la accesibilidad y facilitará la comprensión de los inversionistas y operadores.
- **Normas financieras locales:** Es importante que el sistema se configure según las regulaciones y normativas de cada país, lo que puede variar entre

las diferentes bolsas de valores. Adaptarse a estas normativas será fundamental para el éxito del sistema en múltiples regiones.

- **Comportamiento del usuario:** La cultura influye en la manera en que los inversionistas y comisionistas toman decisiones financieras. Por lo tanto, el sistema debe proporcionar interfaces intuitivas que reflejen las preferencias culturales en términos de visualización de datos y toma de decisiones.

2.2 Hábitos

Los hábitos de los usuarios también juegan un papel muy importante en el diseño del sistema. En el caso de Andina Trading los usuarios (inversionistas y comisionistas de bolsa) tendrán diferentes rutinas y prácticas que el sistema debe considerar para mejorar su efectividad:

- **Frecuencia de uso:** Los inversionistas experimentados tienden a realizar transacciones diarias o incluso por hora, mientras que los usuarios novatos pueden consultar sus inversiones de manera más esporádica. El sistema debe adaptarse a ambos perfiles, proporcionando alertas en tiempo real para usuarios frecuentes y reportes consolidados para los menos activos.
- **Preferencias de inversión:** En función de los hábitos de inversión de los usuarios, el sistema debe ser flexible para ofrecer opciones de inversión a corto, mediano y largo plazo, con reportes personalizables que se ajusten a sus estrategias.
- **Capacitación de usuarios:** Debido a la complejidad del mercado bursátil, es necesario que el sistema cuente con tutoriales o módulos de aprendizaje para que los nuevos usuarios comprendan los procesos de compra y venta de

acciones, lo cual es una práctica esencial para mejorar la usabilidad y garantizar un uso adecuado del sistema.

2.3 Medio

El medio ambiente o el entorno en el que se desenvuelve el proyecto influye directamente en la implementación y adopción del sistema. En este caso, se trata de un entorno financiero altamente regulado que abarca varios países con diferentes condiciones económicas y políticas. Los factores a considerar son:

- **Infraestructura tecnológica:** La conectividad a internet y el acceso a tecnologías avanzadas varían entre los países en los que operará el sistema. En algunas regiones es posible que se enfrenten limitaciones de conectividad, por lo que se recomienda desarrollar una versión optimizada para entornos con baja latencia.
- **Seguridad y privacidad de datos:** Dado que el sistema maneja información financiera sensible, debe cumplir con estrictas regulaciones de seguridad y protección de datos. Las leyes de protección de datos pueden variar entre los países, por lo que el sistema debe ser lo suficientemente flexible para ajustarse a las normativas locales, como la Ley de Protección de Datos Personales.
- **Entorno financiero:** Las condiciones del mercado financiero también varían entre países. Factores como la volatilidad del mercado, la inflación o las tasas de interés pueden influir en cómo los inversionistas toman decisiones, por lo que el sistema debe permitir la integración de análisis de datos en tiempo real para que los usuarios puedan tomar decisiones informadas.

2.4 Artefactos

El artefacto principal es el software que se busca crear para ayudar a los usuarios a invertir de una manera fácil y segura, también son las herramientas y tecnologías que facilitan la interacción entre los usuarios y el sistema. En este caso, los artefactos incluyen el software mismo, así como las plataformas tecnológicas que permiten la comunicación con las bolsas de valores y la gestión de las transacciones bursátiles.

- **Software y plataforma:** El sistema debe ser accesible desde diferentes dispositivos, incluidos ordenadores, tabletas y teléfonos móviles, dado que los inversionistas necesitan poder realizar transacciones en cualquier momento y desde cualquier lugar. Es importante que la interfaz de usuario esté optimizada para dispositivos móviles, ya que muchos usuarios acceden a los mercados financieros desde sus smartphones.
- **Tecnología de conectividad:** La integración en tiempo real con las bolsas de valores requiere una infraestructura robusta que permita la conexión constante y segura. El uso de tecnologías basadas en la nube para almacenar y procesar grandes volúmenes de datos también es esencial para garantizar la eficiencia del sistema.
- **Sistemas de simulación:** Los comisionistas de bolsa, como asesores de los inversionistas, dependen de herramientas de simulación para predecir el comportamiento de las acciones. Por lo tanto, el sistema debe ofrecer modelos de simulación que proyecten escenarios futuros basados en datos históricos y tendencias del mercado.

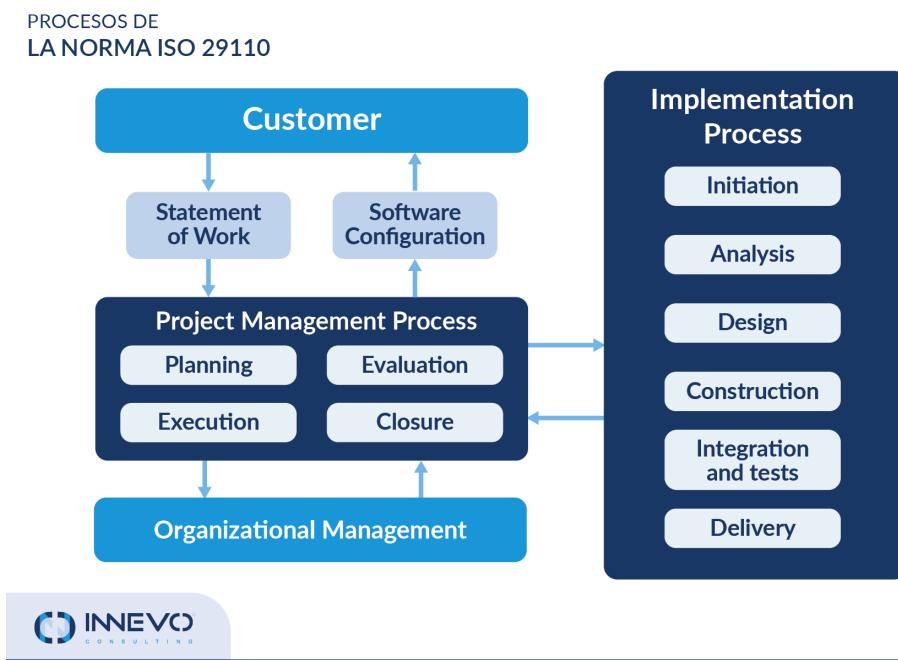
3. Levantamiento de requerimientos (ISO 29110)

Se van a identificar los requerimientos funcionales y no funcionales del proyecto donde se busca desarrollar un software de calidad (ISO 29110) siguiendo el caso de una empresa cuyo mercado es la bolsa de valores.

Implementación ISO 29110

Esta norma nos será útil, ya que somos una empresa de menos de 26 empleados y nuestras capacidades financieras son limitadas. El software que desarrollamos no es de una complejidad muy alta, sin embargo, deseamos usar en nuestro proyecto las buenas prácticas que se aplican en grandes proyectos exitosos.

Esta norma nos dará un marco de trabajo respecto a la gestión del proyecto y a la implementación del software, a continuación presentamos una imagen para facilitar la comprensión de cómo vamos a implementar esta normativa:



Los siguientes puntos responden a las fases de planificación del proyecto, y la fase de análisis de la implementación del software.

3.1. Requerimientos funcionales

Funciones del software que el cliente requiere para saciar su necesidad.

Debido a que este proyecto se fundamenta en metodologías ágiles, trataremos los requerimientos funcionales como historias de usuario, cuya complejidad es medida con la siguiente rúbrica:

Story point	Amount of effort required	Amount of time required	Task complexity	Task risk or uncertainty
1	Minimum effort	A few minutes	Little complexity	None
2	Minimum effort	A few hours	Little complexity	None
3	Mild effort	A day	Low complexity	Low
5	Moderate effort	A few days	Medium complexity	Moderate
8	Severe effort	A week	Medium complexity	Moderate
13	Maximum effort	A month	High complexity	High

Tomado de: <https://asana.com/es/resources/story-points>

- Configurar parámetros básicos, ciudades y países, auditorías, respaldos y recuperación. **(2 puntos)**
- 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)**
- 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 logs 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:
 - o Ver las indicaciones del inversionista en tiempo real. (**2 puntos**)
 - o 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**)

3.2. 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

4. Identificación de procesos de negocio que apoya

Los procesos de negocio se definen como aquellos conjuntos de tareas claves en la empresa, por ejemplo, los que se realizan en un departamento de contabilidad, servicio al cliente, etc.

Teniendo en cuenta esto, se identificaron procesos de negocio clave sobre los cuales este proyecto se fundamenta. Esta identificación es muy importante, porque nos ayudará a mejorar la usabilidad de la aplicación, aplicando el conocimiento impartido por el docente donde hay que capacitar al usuario en los procesos de negocio para luego llevarlo al software a que realice dichos procesos.

En este contexto, existen diferentes áreas en las cuales hay procesos de negocio.

Procesos de contabilidad y de finanzas

La contabilidad es el conjunto de tareas financieras que tanto el usuario como el administrador del software deben conocer, para realizar operaciones en la bolsa con resultados positivos.

- **Registro de transacciones:** Se lleva un registro detallado de todas las compras y ventas de acciones, se calculan las comisiones por transacción y también se actualizan los saldos de las cuentas de los inversionistas.
- **Valoración de carteras:** Se hace el cálculo del valor actual de las inversiones de cada cliente junto con la actualización en tiempo real basada en los precios del mercado.
- **Generación de estados financieros:** En este proceso se elaboran estados de cuenta para los inversionistas e informes financieros para las casas de bolsa.

Procesos de gestión de clientes

- **Registro de inversionistas:** Se registran los datos personales y financieros de los nuevos clientes.
- **Gestión de contratos:** Elaboración y firma de contratos de negociación.
- **Servicio al cliente:** Atención de consultas, reclamos y soporte en el uso del portal de inversionista.

Procesos de negociación y ejecución de órdenes

- **Recepción de órdenes:** Se capturan las órdenes de compra/venta de los inversionistas y se validan saldos y límites de inversión.
- **Ejecución de órdenes:** Se envían las órdenes al mercado.
- **Gestión de riesgos:** Se hace un monitoreo continuo de límites de inversión.

Procesos de análisis y asesoría

- **Análisis de mercado:** Se hace recopilación de datos de mercado para su análisis y generación de informes y recomendaciones.
- **Asesoría a inversionistas:** Recomendaciones personalizadas basadas en el perfil del inversionista.

Procesos de cumplimiento y auditoría

- **Monitoreo de cumplimiento normativo:** Se hace seguimiento de regulaciones locales e internacionales para detección y reporte de operaciones sospechosas.
- **Auditoría interna:** Revisiones periódicas de transacciones, con esto se generan logs y reportes.

Procesos de gestión de tecnología

- **Mantenimiento del sistema:** Actualizaciones y parches de seguridad para la optimización del rendimiento.

- **Gestión de la seguridad:** Control de acceso, encriptación de datos sensibles y respaldo de información.
- **Integración con bolsas de valores:** Mantenimiento de conexiones en tiempo real con las bolsas para mantener la sincronización de datos del mercado.

5. 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.

- **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.

6. Atributos de calidad del software

- **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.
- **Facilidad de mantenimiento:** La separación de responsabilidades mediante patrones como MVC y la utilización de capas para frontend y backend permite que el código sea más fácil de modificar y mantener. De igual manera, el uso de Spring Boot y Angular, que son frameworks que promueven buenas prácticas y estructuración clara del código.
- **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. También la implementación de formularios claros y accesibles para el registro, inicio de sesión y creación de solicitudes.
- **Disponibilidad:** La implementación garantizará que el sistema esté operativo en todo momento, con mínimos tiempos de inactividad.

7. Métricas

Puntos de función (PF)

Para realizar una medición de la funcionalidad del software, independientemente de su construcción y de las tecnologías a utilizar, usamos métricas basadas en puntos de función (PF) para el proyecto. Este enfoque nos permite cuantificar el esfuerzo y la complejidad del sistema, basándonos en cinco componentes clave: Entradas Externas (EE), Salidas Externas (SE), Consultas Externas (CE), Archivos Lógicos Internos (ALI) y Archivos de Interfaz Externos (AIE). Además, hay que aplicar los factores de ajuste de valor (FAV) para obtener un ajuste final en los puntos de función.

Ya conociendo los cinco componentes clave de este tipo de métricas, vamos a identificar uno a uno para este contexto:

- **Entradas Externas (EE)**

Las entradas externas son los datos que ingresan al sistema desde el usuario o de sistemas externos. En el caso de "Andina Trading", identificamos las siguientes entradas:

- **Registro de Inversionistas:** Cuando un nuevo inversionista se registra en el sistema.
- **Generación de Contratos:** Entrada para crear un contrato de negociación entre un inversionista y un comisionista.
- **Emisión de Órdenes de Compra/Venta:** El inversionista ingresa los datos necesarios para realizar órdenes de compra o venta.

Total de Entradas Externas: 3 Entradas.

- **Salidas Externas (SE)**

Las salidas externas son las respuestas que el sistema genera para el usuario o sistemas externos. Las salidas relevantes para este proyecto son:

- **Generación de Reportes:** Reportes financieros para los inversionistas, comisionistas, y el estado de las acciones.
- **Confirmación de Órdenes:** Mensajes de confirmación que indican el estado de las órdenes de compra y venta.

Total de Salidas Externas: 2 Salidas.

- **Consultas Externas (CE)**

Las consultas externas son solicitudes interactivas que producen una respuesta del sistema. Las consultas clave en "Andina Trading" incluyen:

- **Consulta de Estado de Inversionistas:** El comisionista o inversionista puede consultar el estado de los inversionistas.
- **Consulta de Transacciones en Tiempo Real:** Permite a los inversionistas consultar el estado actual de las transacciones en la bolsa de valores.

Total de Consultas Externas: 2 Consultas.

- **Archivos Lógicos Internos (ALI)**

Los archivos lógicos internos son archivos o bases de datos que son manejados y mantenidos dentro del sistema. Identificamos los siguientes archivos:

- **Base de Datos de Inversionistas:** Contiene la información detallada de los inversionistas.

- **Base de Datos de Transacciones:** Almacena las transacciones realizadas por los inversionistas y comisionistas.

Total de Archivos Lógicos Internos: 2 ALI.

- **Archivos de Interfaz Externos (AIE)**

Los archivos de interfaz externos son archivos que interactúan con otros sistemas externos. En el caso de "Andina Trading", tenemos:

- **Interfaz con las Bolsas de Valores:** Archivo que envía y recibe datos de las bolsas de valores.

Total de Archivos de Interfaz Externos: 1 AIE.

Como ya se identificaron los componentes, ahora se tiene que asignar una ponderación según su complejidad que puede ser baja, media o alta (tal como se ha descrito en las clases de Ingeniería de Software 2):

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

Teniendo en cuenta la tabla anterior (creación propia), pasamos a hacer la sumatoria de todos los resultados de las multiplicaciones entre cantidad de cada fila por el factor de ponderación asignado.

Total = 12 + 10 + 6 + 20 + 7 = 55 puntos de función

Ahora, siguiendo el paso a paso de las métricas de puntos de función, se debe ahora aplicar los factores de ajuste de valor (FAV) para refinar el cálculo de puntos de función. 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

La sumatoria de todos los factores es **53**.

Por último, pasamos al cálculo de los puntos de función (PF), la fórmula es la siguiente:

$$PF = \text{Conteo total} * [0.65 + 0.01 * \sum F_i]$$

Por lo tanto:

$$PF = 55 * [0.65 + (0.01 * 53)]$$

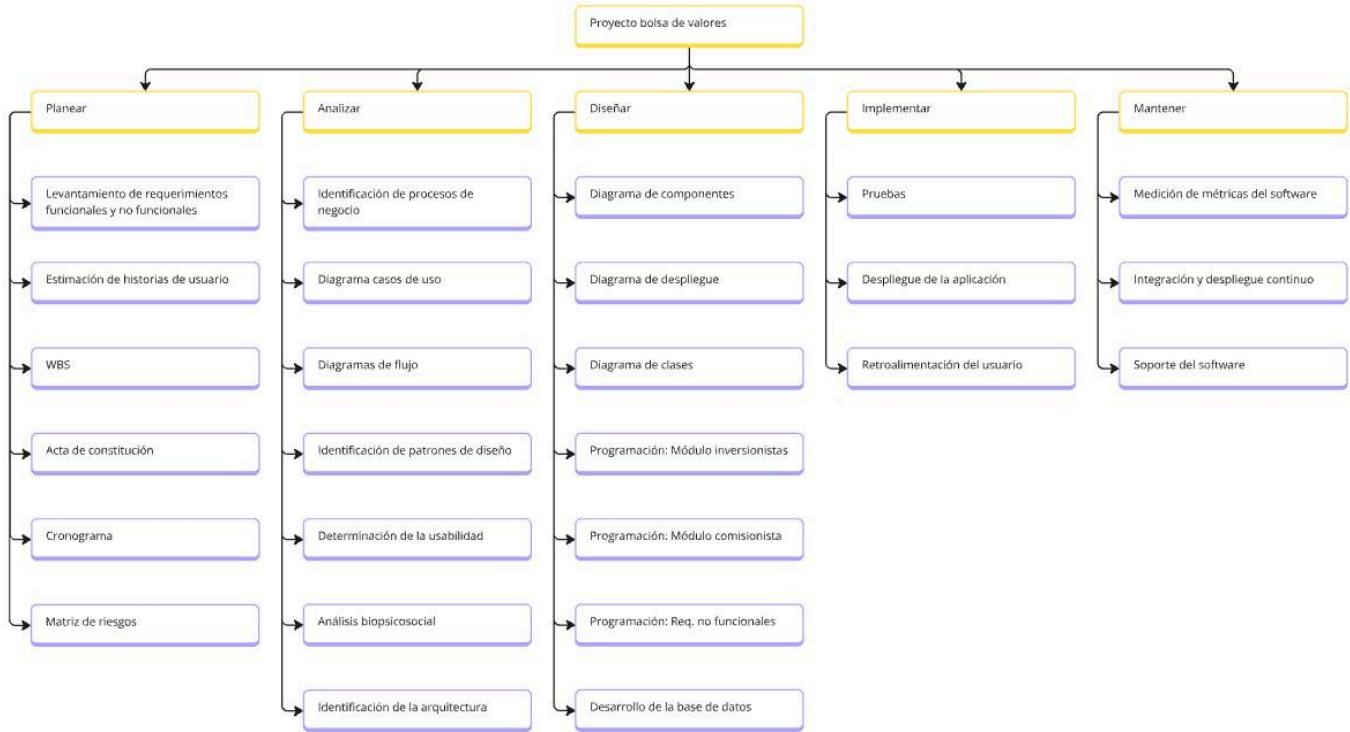
$$PF = 55 * [0.65 + 0.53]$$

$$PF = 55 * 1.18 = \mathbf{64.9}$$

Ahora hay que analizar el resultado (64.9). 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.

8. EDT



9. 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...		█		

10. Implementación de SCRUM

SCRUM es una metodología ágil ampliamente utilizada en proyectos de desarrollo de software debido a su enfoque iterativo, adaptable y centrado en la entrega continua de valor. A lo largo de este apartado, se busca desarrollar una descripción detallada de los roles, artefactos, eventos y procesos de SCRUM que se utilizarán en el proyecto, con el fin de garantizar una implementación efectiva.

11.1. Descripción General de SCRUM

SCRUM es un marco de trabajo ágil que facilita la organización y el desarrollo de proyectos complejos y se basa en ciclos cortos de desarrollo llamados sprints, en los que se realiza la planificación, desarrollo, revisión y mejora continua del producto. SCRUM permite adaptarse a cambios, priorizar características que añaden más valor al cliente y mejorar la calidad general del producto a lo largo del proceso de desarrollo.

11.2. Roles en SCRUM para "Andina Trading"

Para implementar SCRUM en este proyecto es esencial definir los diferentes roles clave que estarán presentes en el desarrollo del proyecto:

11.2.1. Product Owner

Principalmente, el product owner es quien va a representar al cliente y a los interesados teniendo interacción directa con ellos, y con esto, poder crear y priorizar las tareas en el product backlog según el valor que aporten al sistema. Debe estar pendiente que el equipo de desarrollo esté trabajando en las funcionalidades más importantes o que generen más valor.

Para Andina Trading, el Product Owner debe ser una persona con amplio conocimiento en mercados bursátiles, que entienda las necesidades de los inversionistas, comisionistas y otros actores clave.

11.2.2. Scrum Master

Debe guiar al equipo en la correcta implementación de los principios SCRUM, ya que él es quien conoce todo el marco de trabajo que requiere esta metodología y también, debe facilitar las reuniones garantizando que sean efectivas.

El Scrum Master debe ser alguien con experiencia en metodologías ágiles, comunicación efectiva y manejo de equipos. No es un rol jerárquico, sino un facilitador del equipo.

11.2.3. Equipo de Desarrollo

Como lo dice el nombre, es el equipo que busca desarrollar y entregar incrementos de producto al final de cada sprint colaborando estrechamente con el Product Owner para lograr entender todos los requerimientos.

El equipo de desarrollo debe estar compuesto por ingenieros de software, diseñadores y especialistas en pruebas que tengan habilidades técnicas sólidas y una buena capacidad de trabajo en equipo.

11.3. Artefactos de SCRUM para "Andina Trading"

Los artefactos de SCRUM son elementos que proveen transparencia y oportunidades de inspección y adaptación en el proyecto.

11.3.1. 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.

Contenido:

- Módulo de gestión de inversionistas.
- Módulo de conexión en tiempo real con la bolsa de valores.
- Portal del inversionista.
- Módulo de generación de reportes financieros.
- Funcionalidades de seguridad y autenticación.
- Soporte para múltiples idiomas.

11.3.2. 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.

Ejemplo de Sprint Backlog

- Crear la funcionalidad de registro de inversionistas.
- Implementar conexión con la bolsa de valores de Ecuador.
- Desarrollar el módulo de autenticación con JWT.

11.3.3. 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.

11.4. Eventos en SCRUM para "Andina Trading"

Los eventos en SCRUM estructuran el trabajo y promueven la transparencia e inspección del progreso del proyecto.

11.4.1 Sprint

Un Sprint es un ciclo de desarrollo que dura entre 2 y 4 semanas y en cada uno, el equipo debe entregar un incremento funcional del software.

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

11.4.2 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.

Busca asegurar que todos los miembros del equipo comprendan el trabajo a realizar y acuerden cómo se va a desarrollar.

11.4.3 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?

Esta reunión busca garantizar la sincronización del equipo y detectar problemas o bloqueos rápidamente.

11.4.4. Sprint Review

Al final de cada sprint, el equipo presenta el incremento de software al Product Owner y otros interesados para recibir retroalimentación.

Para evaluar si se han cumplido los objetivos del sprint y obtener sugerencias para mejorar en el siguiente.

11.4.5. 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.

11.5. Proceso de Implementación de SCRUM en "Andina Trading"

11.5.1. Iniciar con la planificación del proyecto:

- Definir claramente el Product Backlog con la ayuda del Product Owner.
- Priorizar las funcionalidades más críticas (por ejemplo, conexión en tiempo real con la bolsa de valores).

11.5.2. Formación del equipo SCRUM:

- Designar el Product Owner, Scrum Master y los miembros del equipo de desarrollo.

11.5.3. Primera Sprint Planning:

- Definir las tareas para el primer sprint, comenzando con la implementación del módulo de registro de inversionistas.

11.5.4. Sprints Continuos:

- Realizar Daily Scrums para sincronizar el trabajo.
- Al final de cada sprint, realizar una revisión del incremento y una retrospectiva.

11.5.5. Inspección y Adaptación:

- Revisar constantemente el Product Backlog y ajustar las prioridades según las necesidades del cliente y los cambios en el mercado bursátil.

11. Modelo Canvas

Business Model Canvas

Edison Beltrán Garzón
Johann Toncon Poveda



12. Arquitectura de software

El proyecto llamado Andina Trading busca desarrollar un sistema de software que gestione las transacciones de acciones en una bolsa de valores que opera en varios países de Latinoamérica. Este sistema debe permitir la conexión en tiempo real con las bolsas de valores locales, la gestión de inversionistas y comisionistas, y la generación de reportes financieros, asegurando la confiabilidad y eficiencia del software. A lo largo de este trabajo se quiere dar a conocer la arquitectura de software en la cual se quiere desarrollar el sistema y justificar su elección.

Descripción del Proyecto

El software debe integrar funcionalidades para conectar con bolsas de valores en tiempo real, permitir a los comisionistas gestionar las transacciones de acciones, y brindar a los inversionistas un portal donde puedan monitorear sus inversiones y ejecutar órdenes de compra/venta. El sistema también va a consolidar información financiera para el análisis y generación de reportes. Dado el alcance y la complejidad del sistema, es crucial seleccionar una arquitectura adecuada para garantizar su escalabilidad, modularidad y rendimiento.

Propuesta de Arquitectura

- Arquitectura seleccionada

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, despliegue y mantenimiento.

- **¿Por qué microservicios?**

La arquitectura de microservicios permite escalar cada componente de manera independiente. Por ejemplo, si la demanda en el módulo de conexión en tiempo real aumenta, se puede escalar solo esa parte sin afectar los demás módulos. Cada funcionalidad se implementa como un servicio independiente (microservicio), lo que mejora la organización del código y permite una gestión más efectiva de las responsabilidades. Los microservicios pueden ser actualizados y desplegados de manera independiente, reduciendo los riesgos asociados a las actualizaciones y mejorando la continuidad del servicio. La separación de funcionalidades en microservicios simplifica la localización y resolución de problemas, ya que se pueden aislar errores en módulos específicos sin afectar al resto del sistema.

Diseño de la Arquitectura: Componentes Principales

- **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.

13. UML

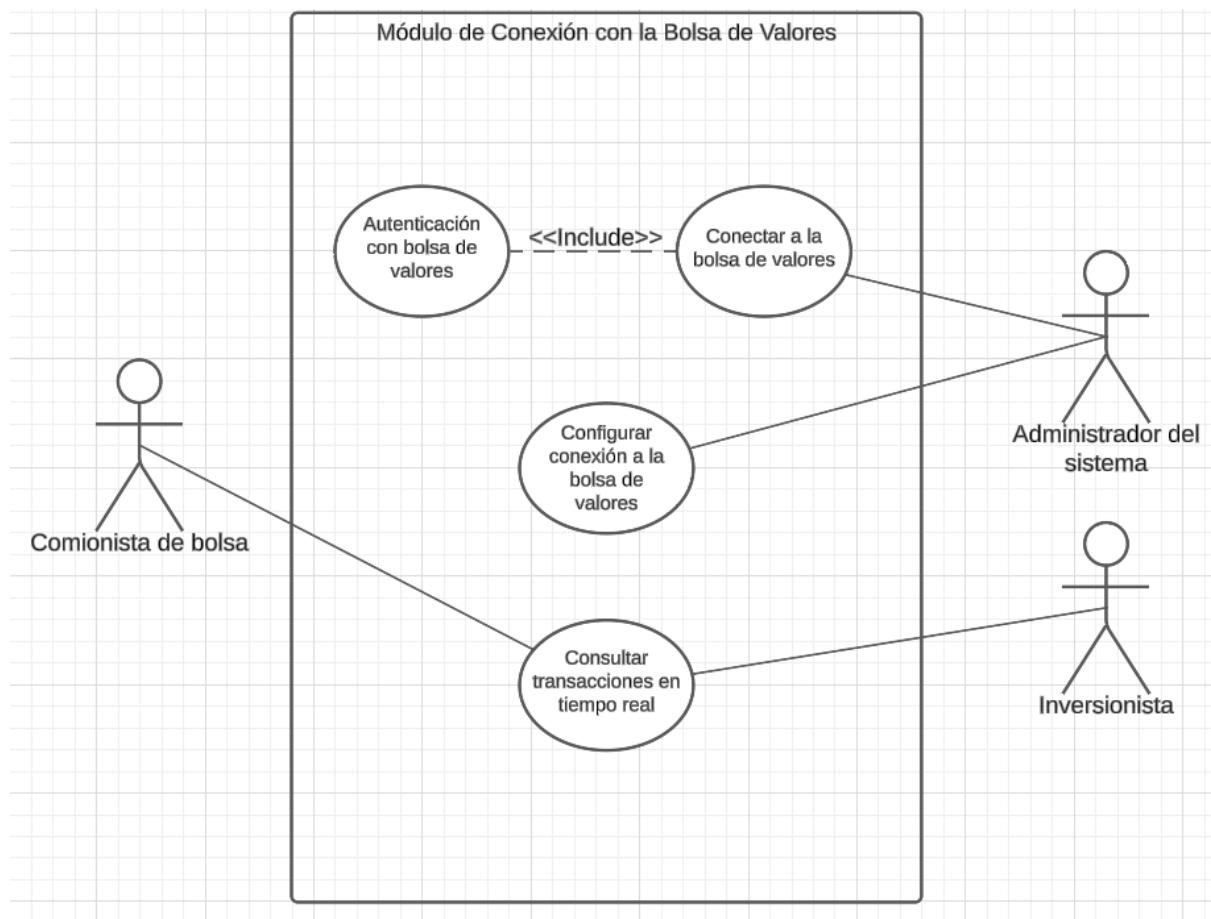
13.1. Diagramas de casos de uso

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 microservicios o módulos.

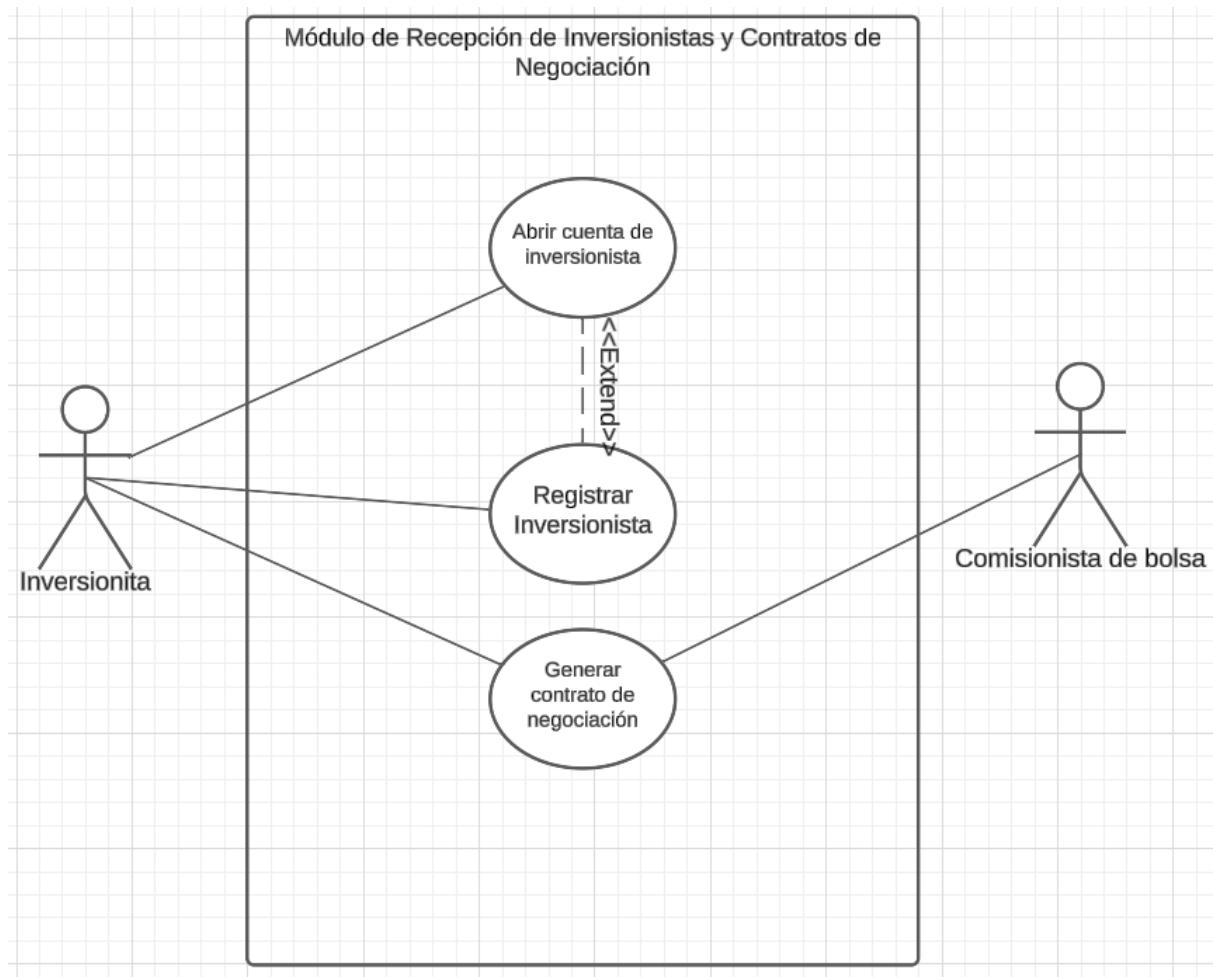
- Módulo de gestión del sistema



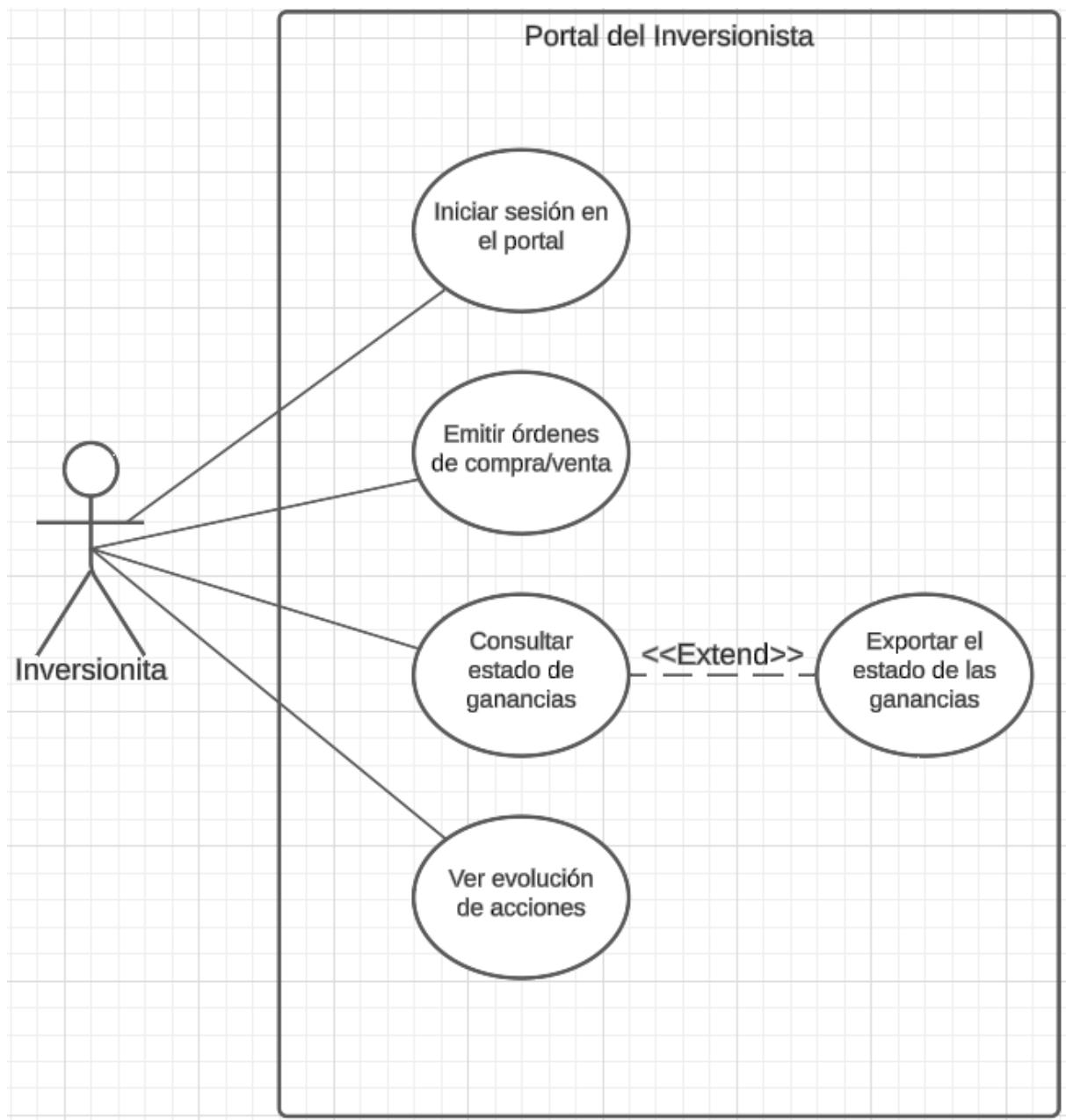
- Módulo de conexión con la bolsa de valores



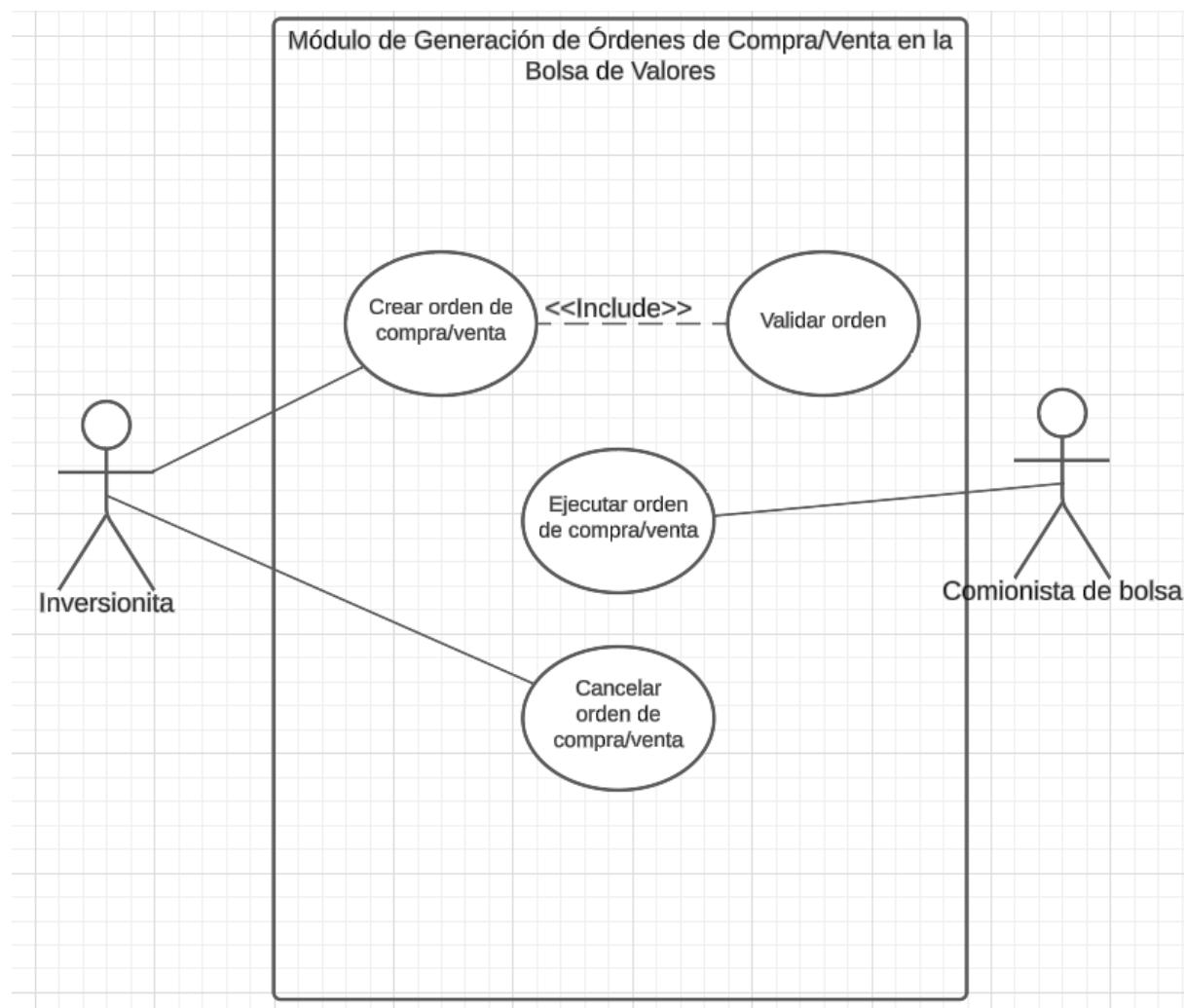
- **Módulo de recepción de inversionistas y contratos de negociación**



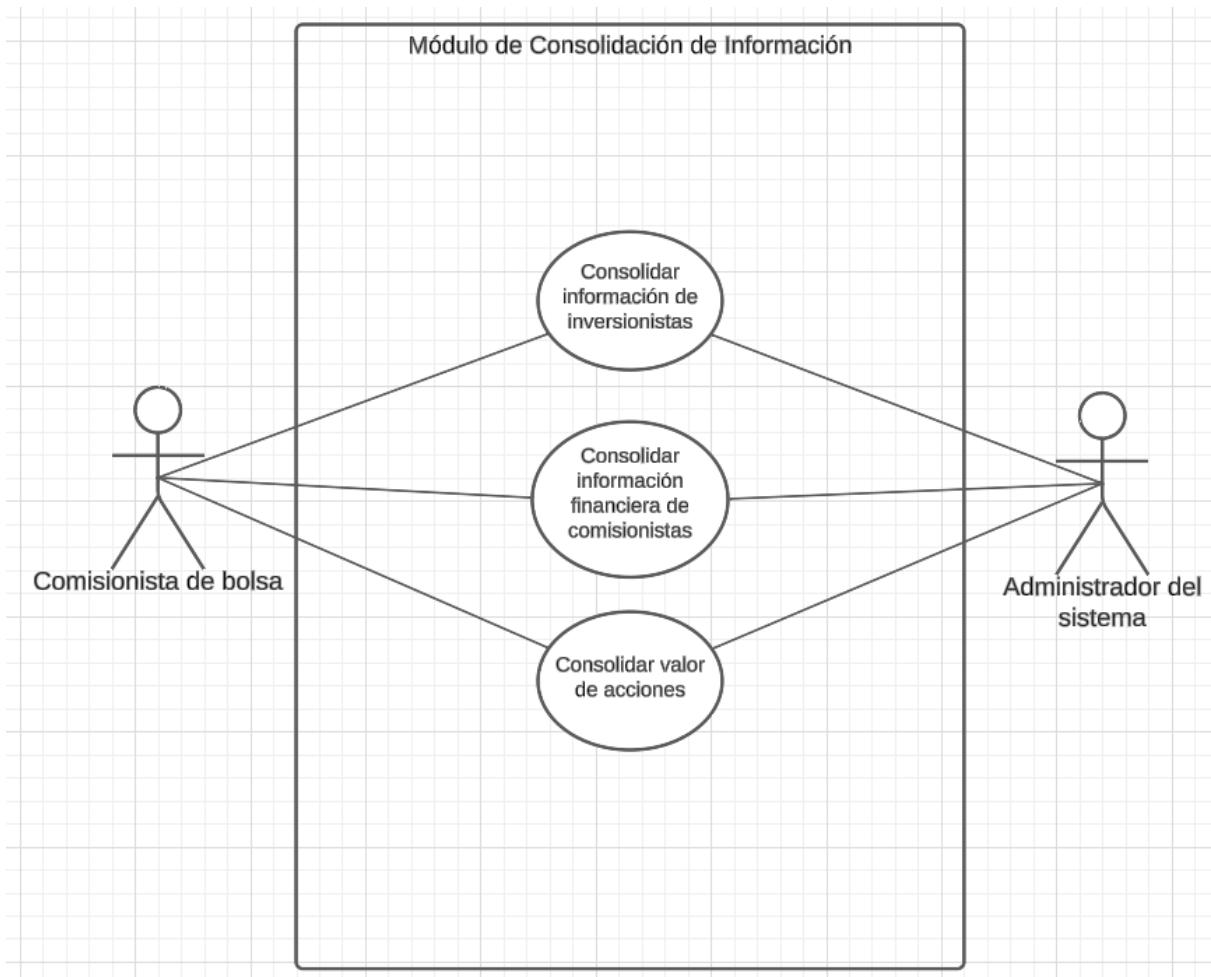
- Portal inversionistas



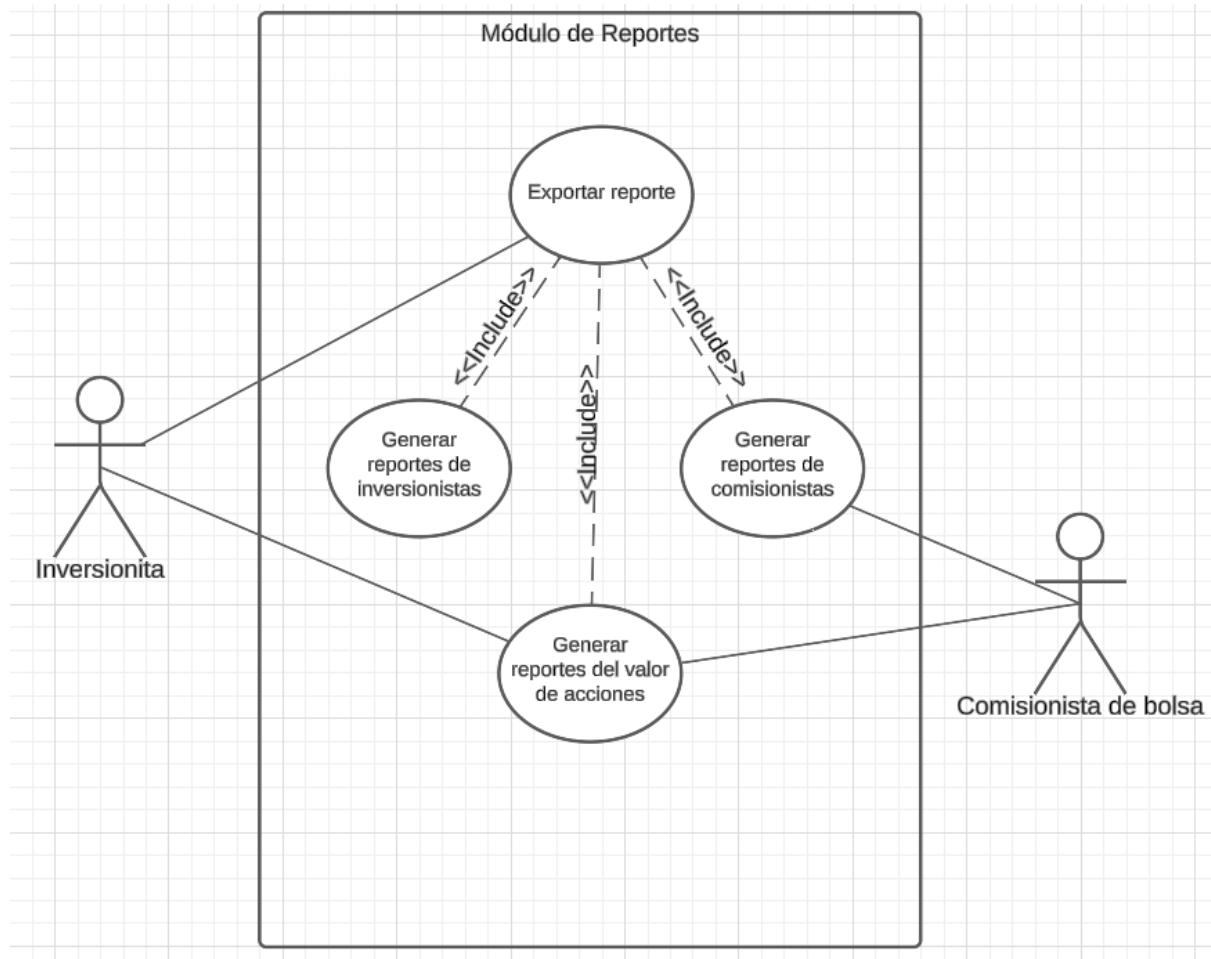
- Generación de órdenes de compra/venta en la bolsa de valores



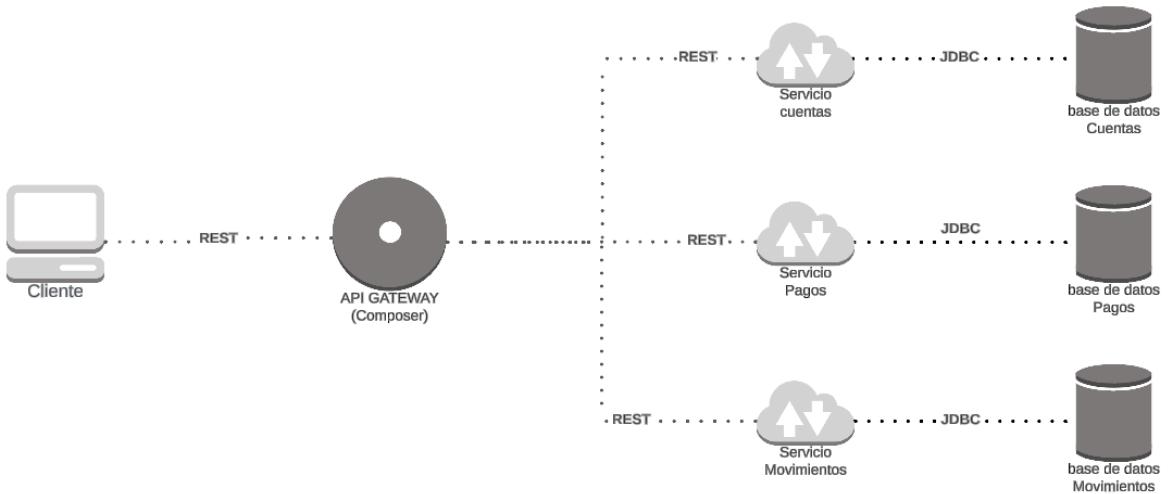
- **Consolidación de información**



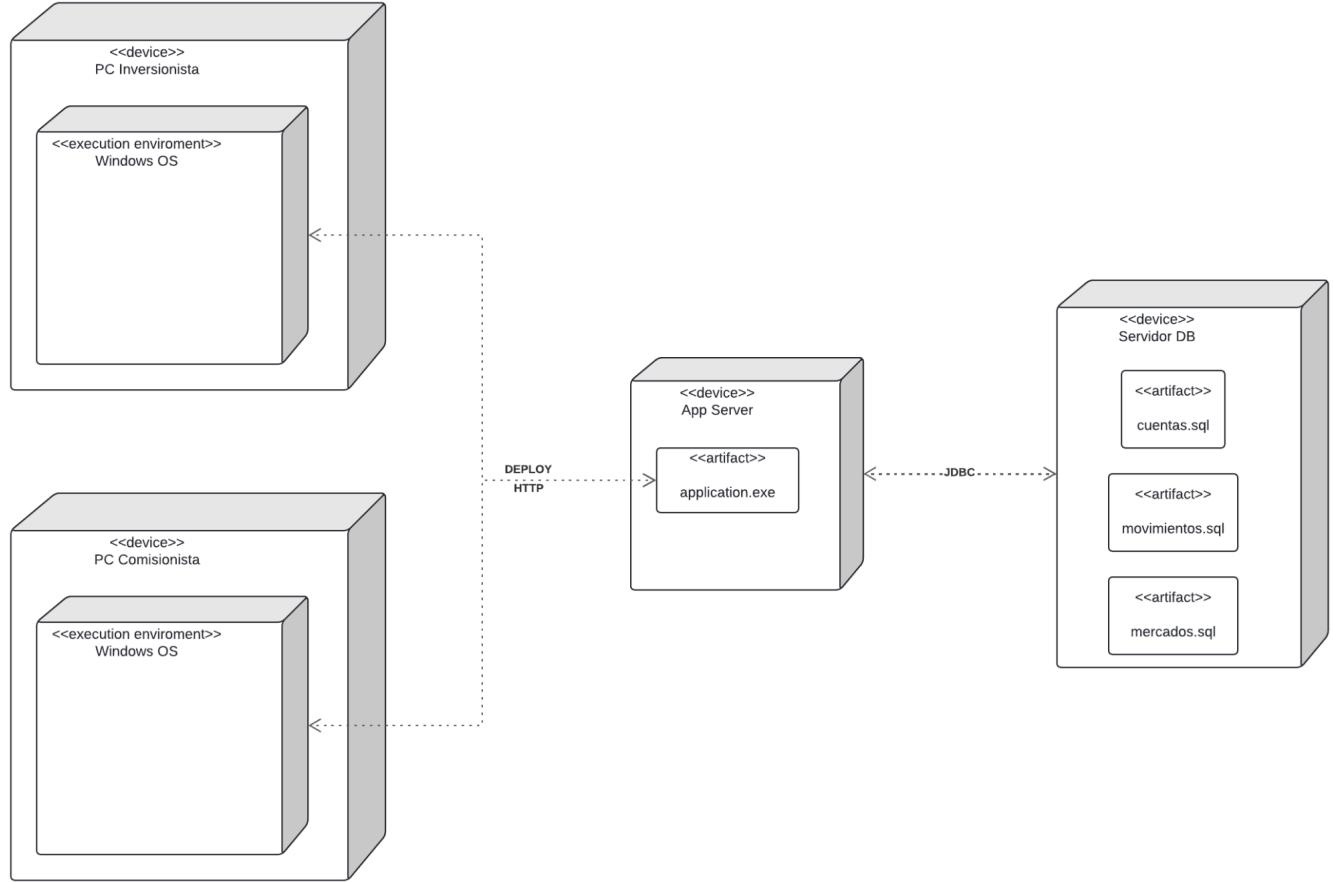
- **Módulo de reportes**



13.2. Diagrama de componentes

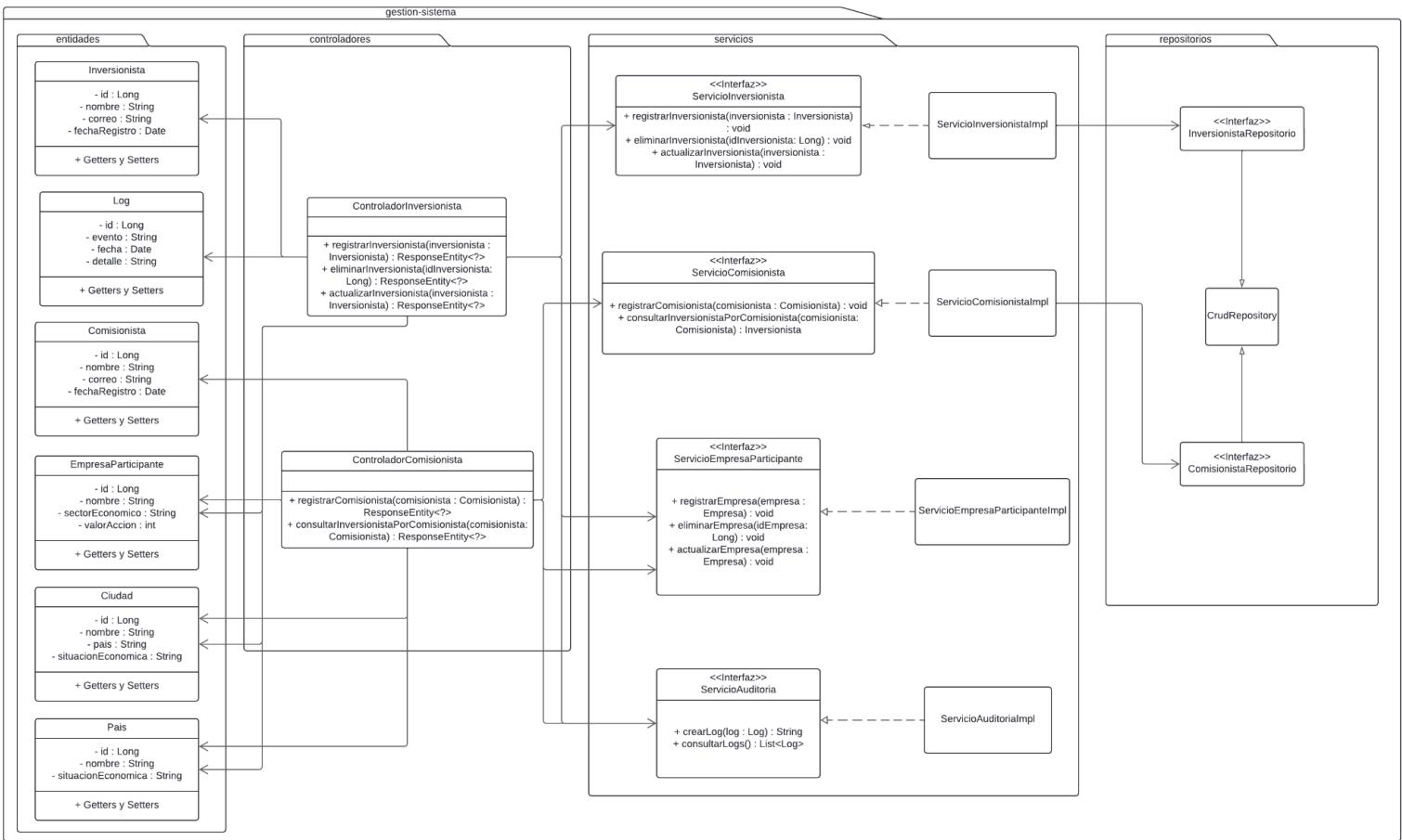


13.3. Diagrama de despliegue

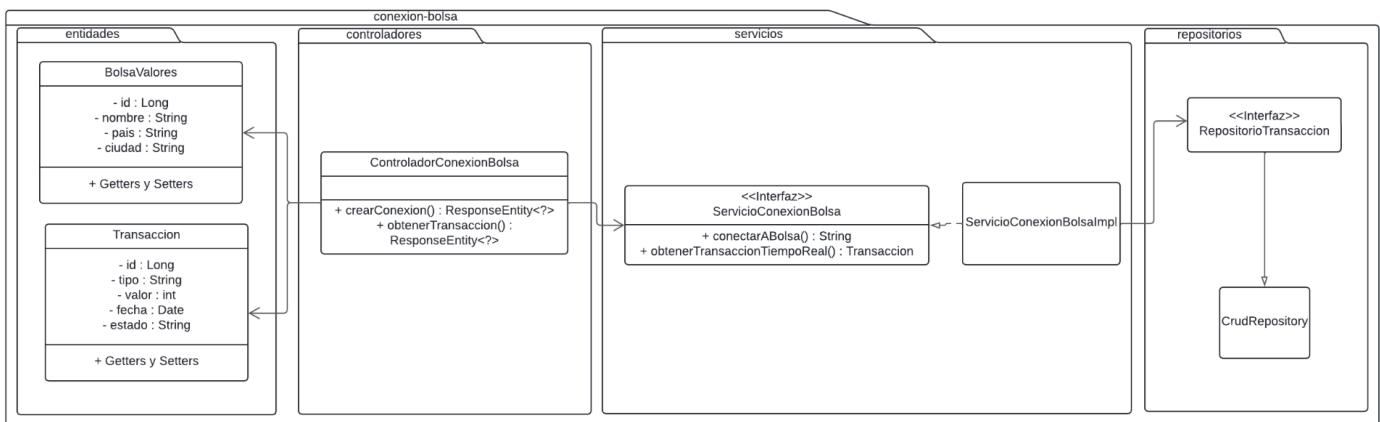


13.4. Diagrama de clases

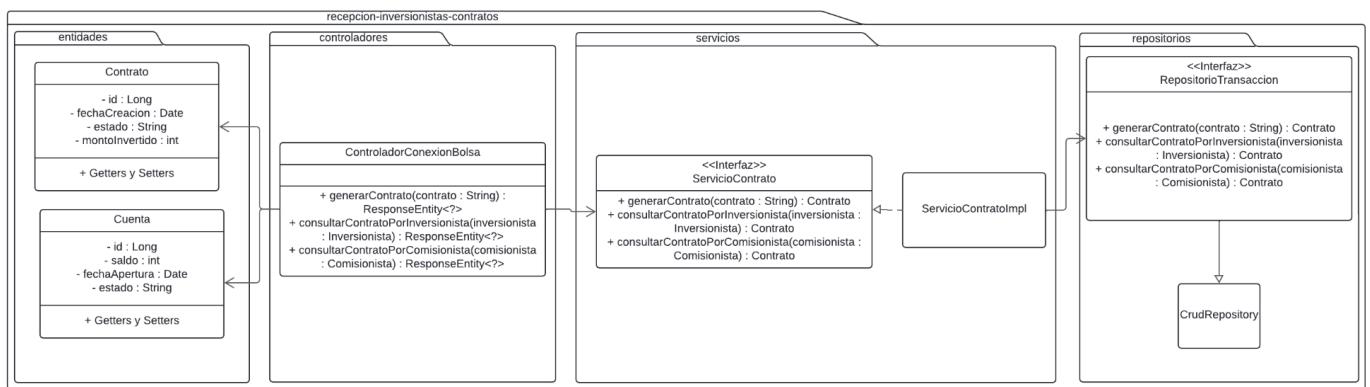
13.4.1. Módulo Gestión del Sistema



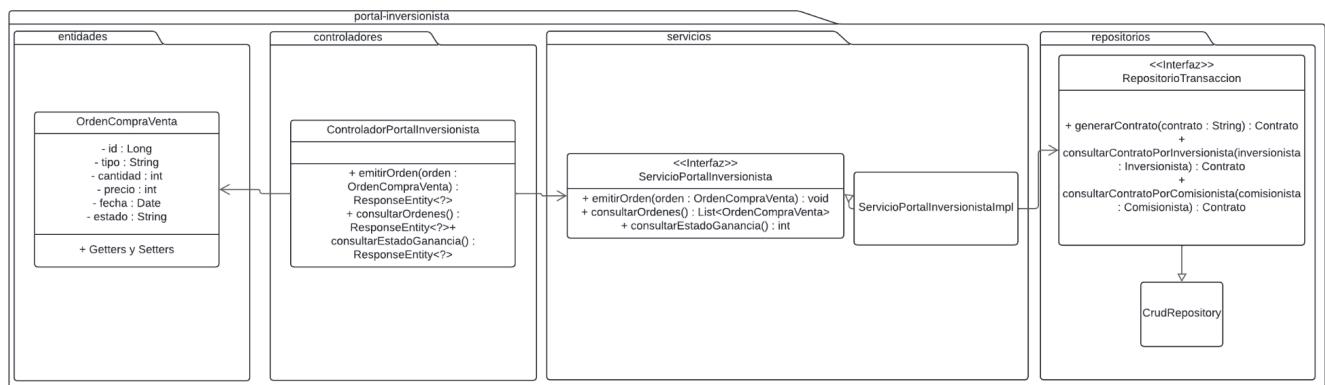
13.4.2. Módulo de Conexión con la Bolsa de Valores



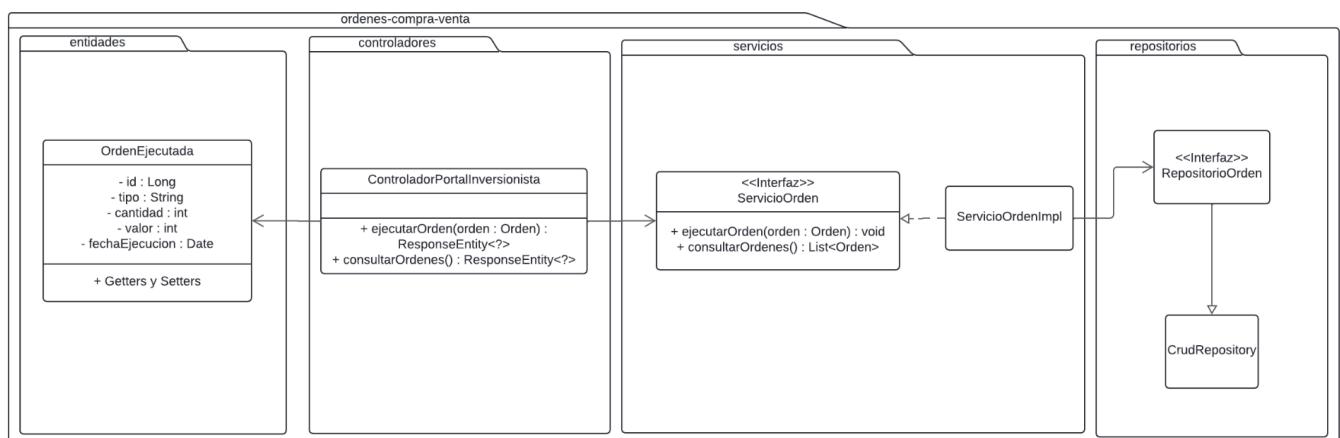
13.4.3. Módulo de Recepción de Inversionistas y Contratos de Negociación



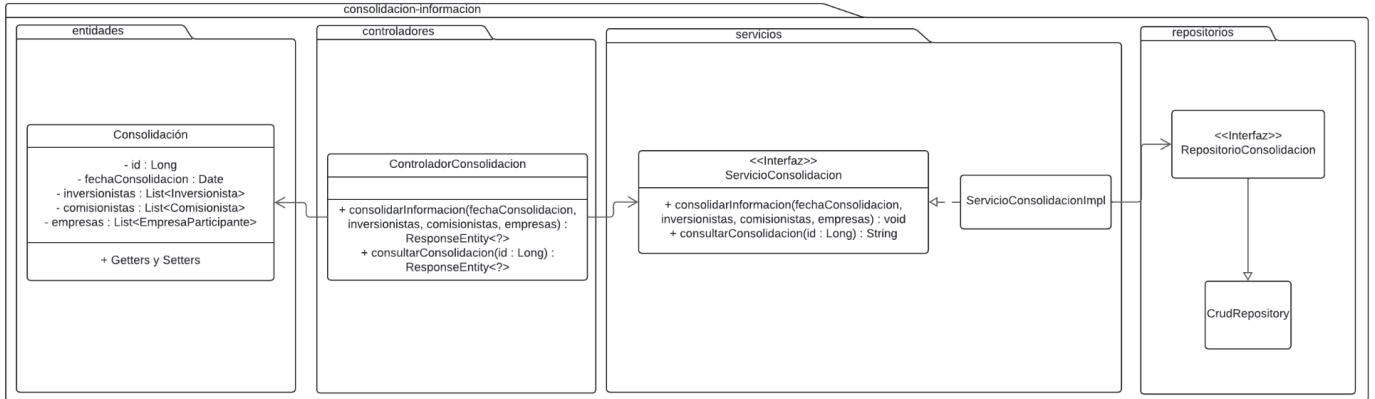
13.4.4. Portal del Inversionista



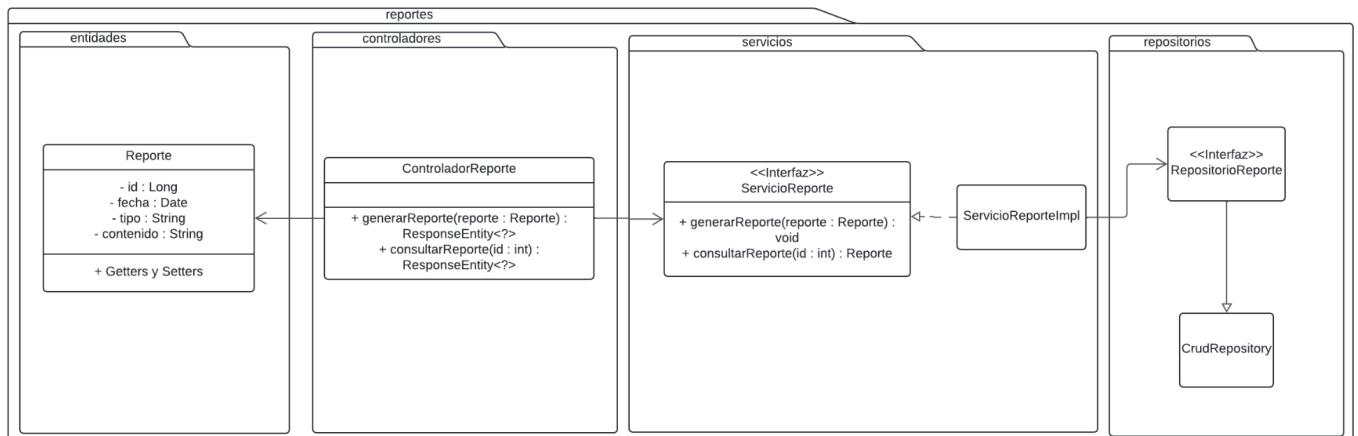
13.4.5. Módulo de Generación de Órdenes de Compra/Venta en la Bolsa de Valores



13.4.6. Módulo de Consolidación de Información

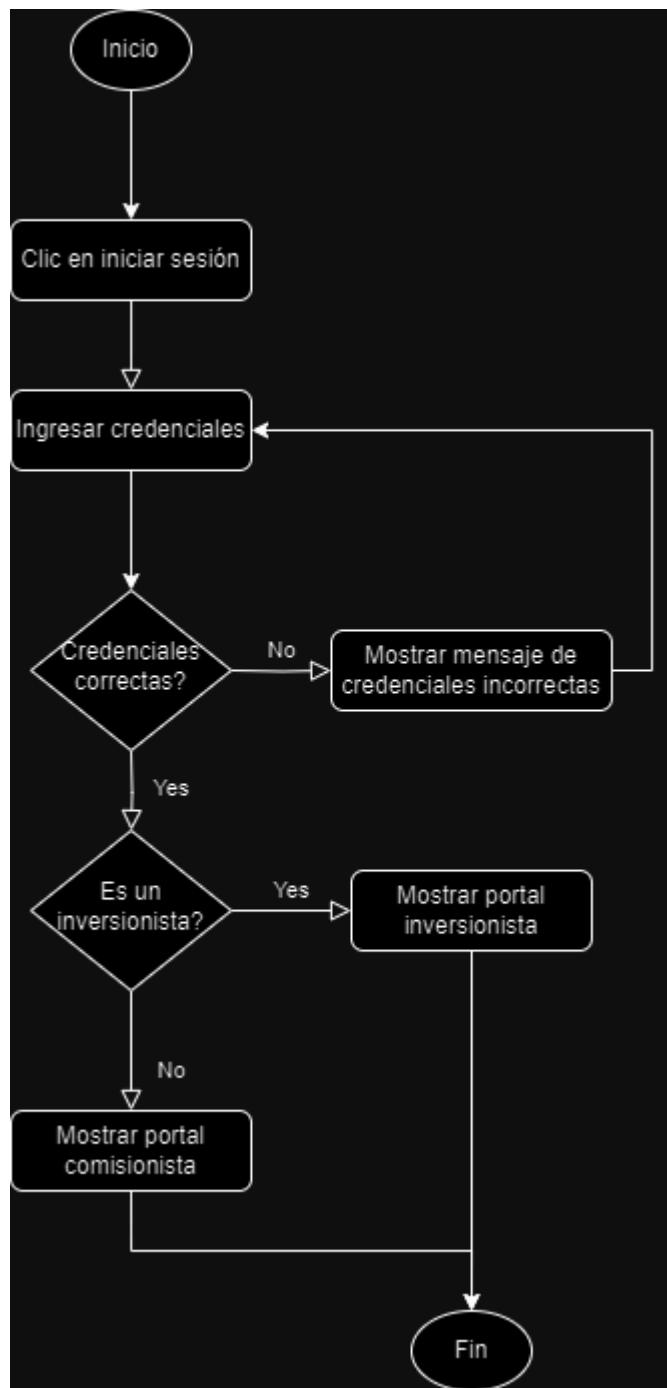


13.4.7. Módulo de Reportes

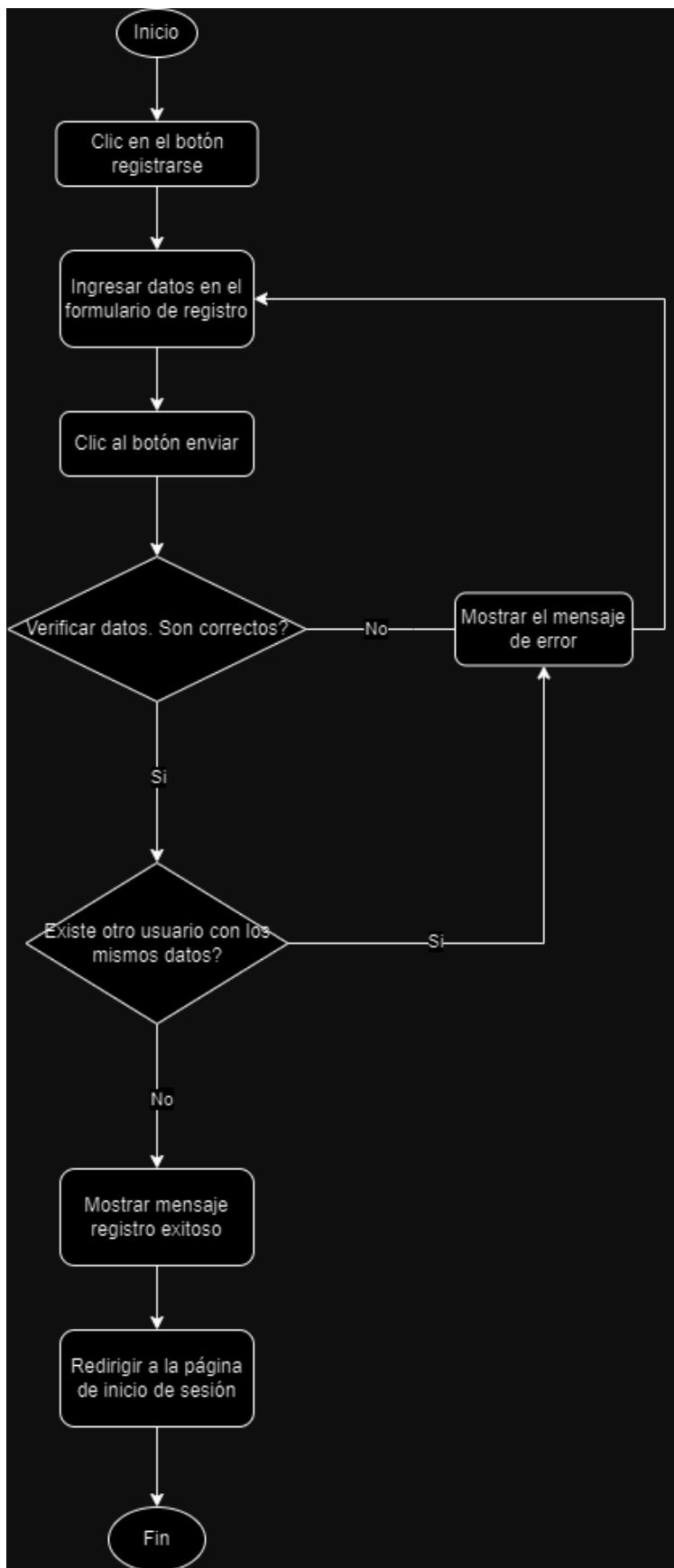


13.5. Diagramas de flujo

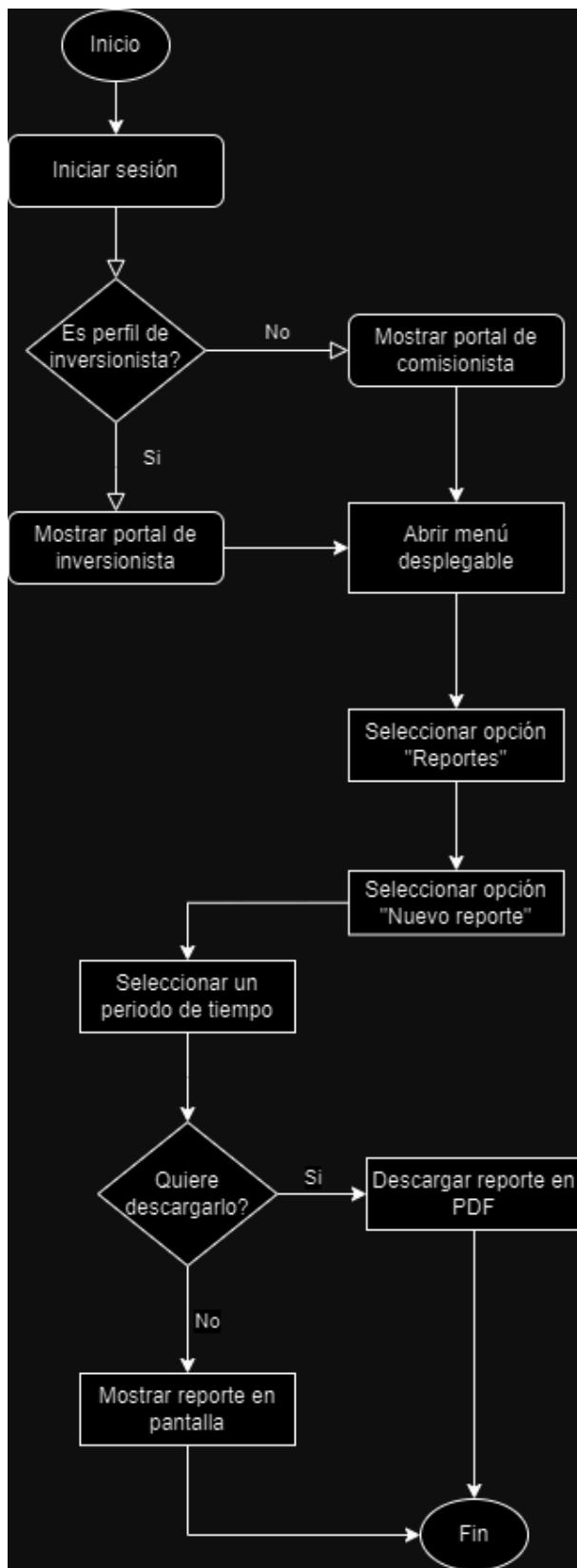
- Inicio de sesión



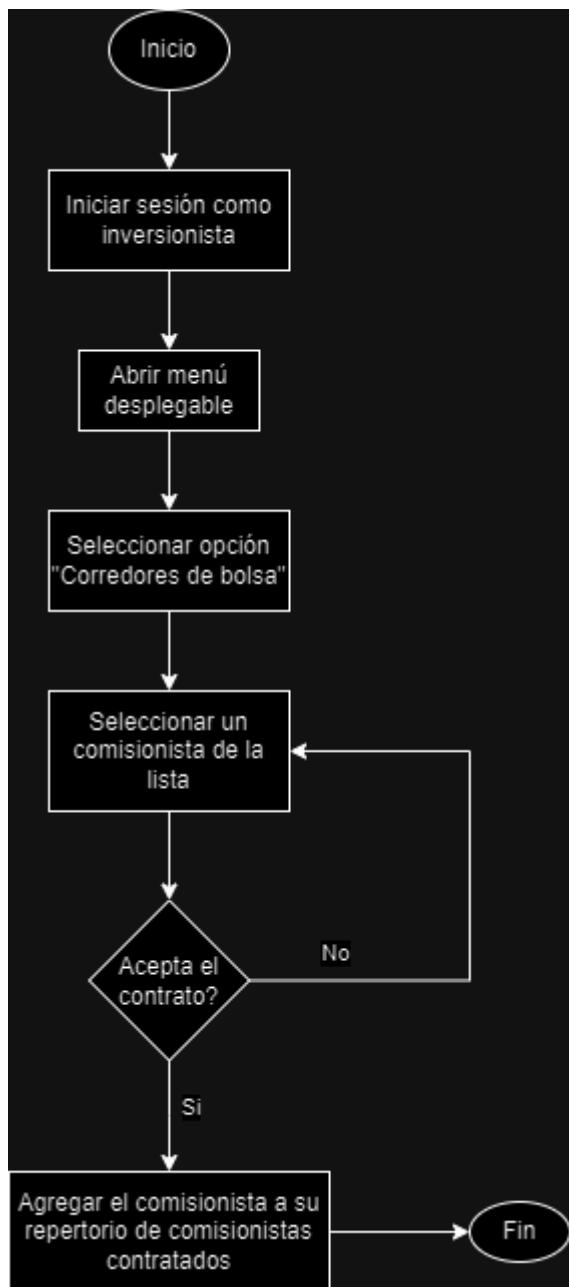
- **Registro:**



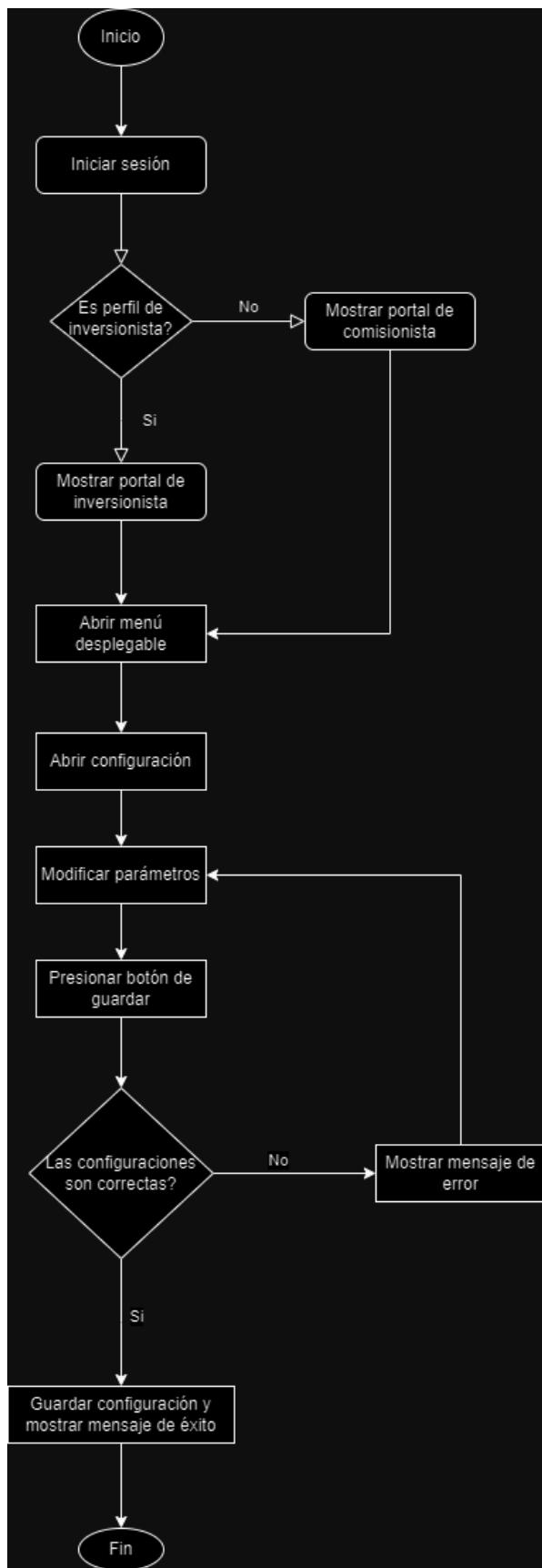
- Generar reportes



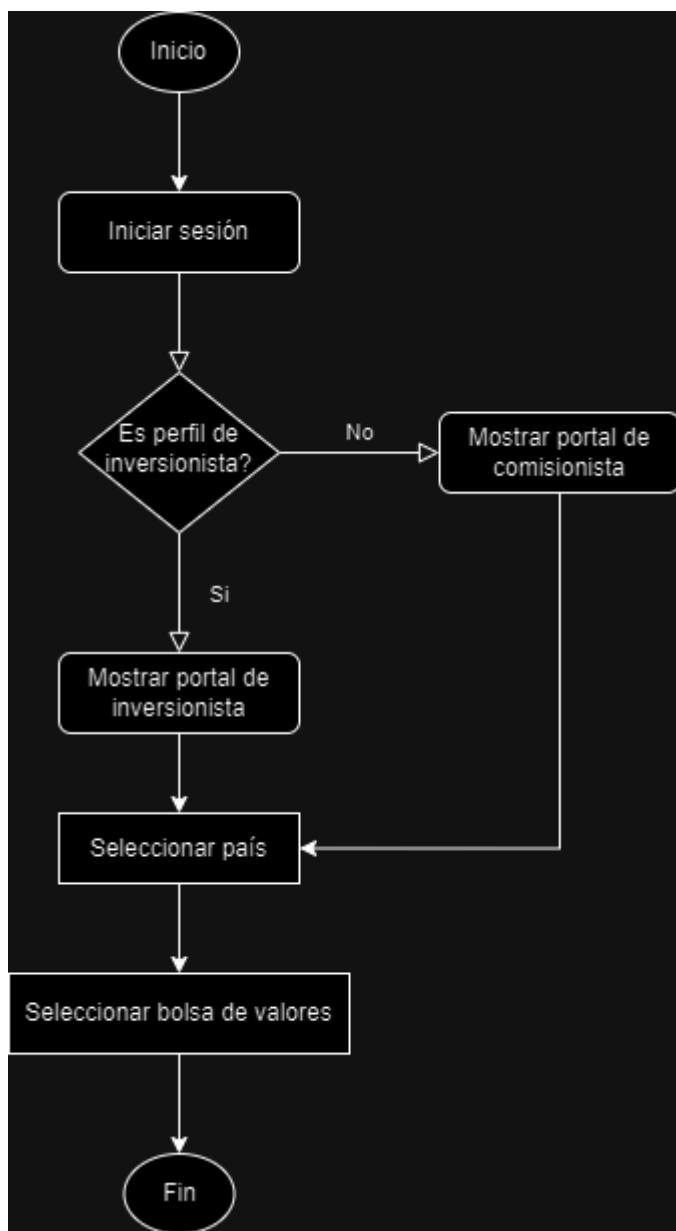
- Contratar comisionistas



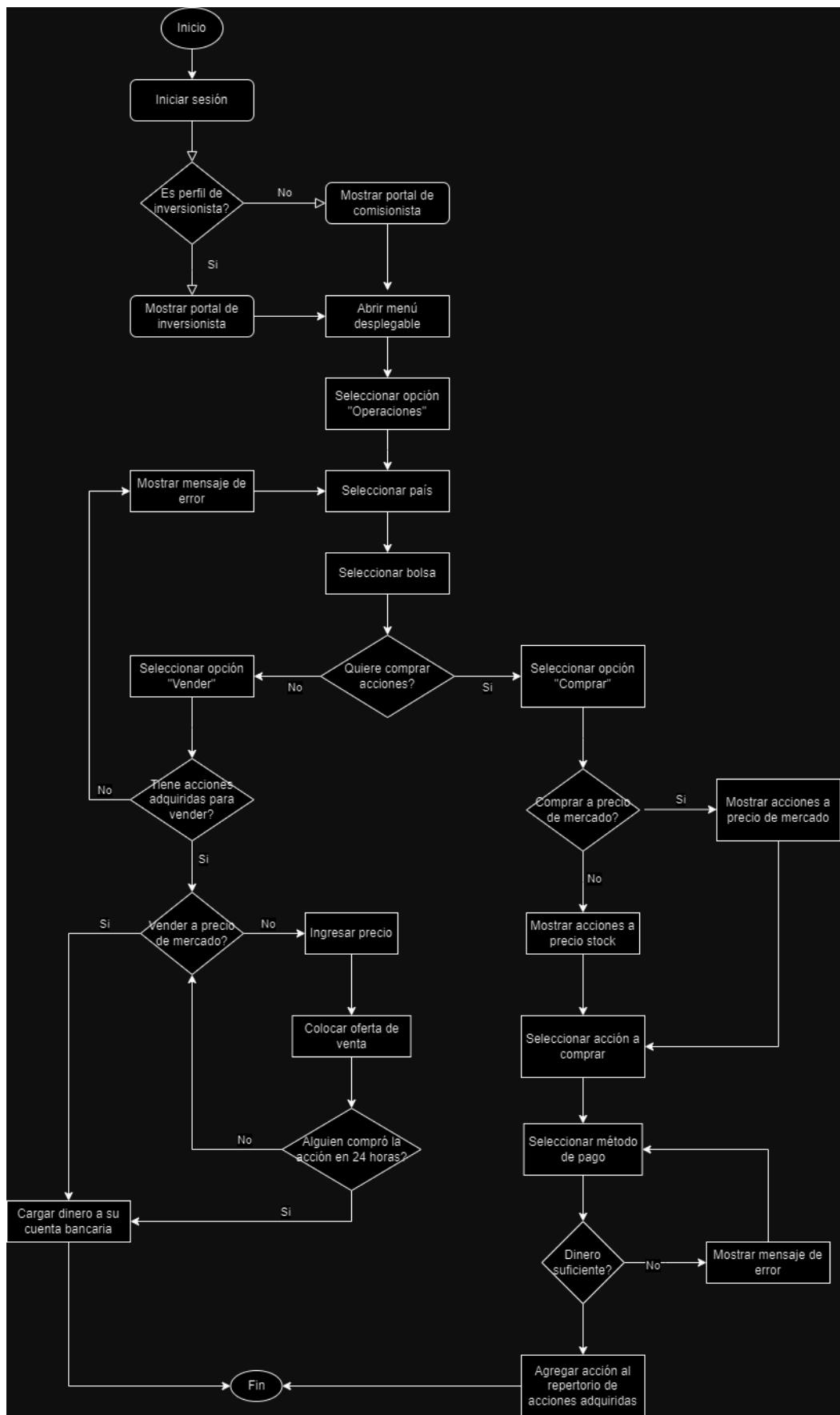
- Configuración de parámetros



- Entrar a una bolsa de valores



- Compra y venta de acciones



14. Usabilidad

Este punto tiene como objetivo establecer el marco de usabilidad aplicado en el proyecto basado en un software de Bolsa de Valores, basado en la ISO 9126.

Explicaremos la heurística utilizada para el diseño de la herramienta, los principios aplicados, y el impacto que esto tiene en el proyecto. De igual manera, daremos a conocer los mockups diseñados para este proyecto en términos de mejorar la usabilidad.

Procuramos acondicionar el producto al cliente, los cuales son todos los interesados en invertir en la bolsa de valores para obtener utilidades, siguiendo la regla aprendida en clase “Si todo es llamativo, nada es llamativo”.

Principios aplicados

- **Facilidad de aprendizaje:** Nuestro software será diseñado pensando en lo que hace el cliente, no en lo que dice (Diseño centrado en el usuario). Esto permite que el cliente se familiarice de una forma más rápida y efectiva con el programa, reduciendo en un 40% los costos de capacitación.
- **Facilidad de uso:** Nuestro software será eficaz y eficiente, con esto 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. Esto

siempre pensando en que el software se hace para facilitar la vida del cliente, no para hacerla más compleja.

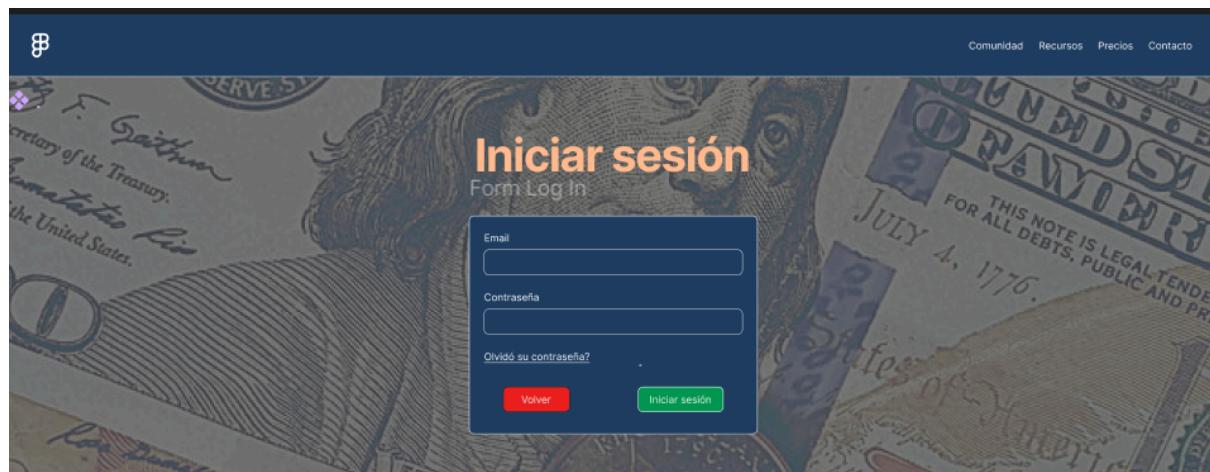
- **Satisfacción:** Al aplicar estos principios de la usabilidad, el usuario no se verá frustrado o detenido por procesos complejos o indocumentados. Esto asegura que el usuario esté satisfecho con la herramienta, lo que tiene un efecto directamente proporcional en la calidad de su trabajo.

Mockup

- **Vista principal**



- **Inicio de sesión**



- Portales Inversionista/Comisionista

A screenshot of a stock market investment platform. At the top, there's a dark header with a logo and a menu icon. Below it, a large green section displays "Tus ganancias" (Your earnings) in a white box, showing "7'234.190 COP". A yellow rounded rectangle contains a table for "Acciones populares" (Popular Stocks). The table has three columns: "Empresa" (Company), "Precio" (Price), and "Comportamiento 1 hora" (1-hour behavior). It lists two companies: "Manzana Inc." with a price of "153.021 COP" and a green "+ 0,4%", and "RootSoftWorks Inc." with a price of "825.124.021 COP" and a red "- 0,1%". To the right of the table is a vertical bar chart with an upward-pointing triangle at the top and a downward-pointing triangle at the bottom.

- Menú desplegable portal

The screenshot displays a mobile application interface. At the top, there is a dark blue header bar with a small logo on the left and a "Menú" button with three horizontal lines on the right. Below the header is a green section titled "Tus ganancias" (Your earnings) containing a large white box with the text "7'234.190 COP". To the right of this green section is a vertical dark blue sidebar menu. The menu items are "Perfil" (Profile), "Operaciones" (Operations), "Generar reporte" (Generate report), and "Cerrar sesión" (Close session). At the bottom of the sidebar, it says "v0.1" and "@rootsoftworks". The main content area below the green section shows a table titled "Acciones populares" (Popular stocks) with two rows of data:

Empresa	Precio	Comportamiento 1 hora
Manzana Inc.	153.021 COP	+ 0,4%
RootSoftWorks Inc.	825.124.021 COP	- 0,1%

Below the table is a small downward-pointing arrow icon.

15. Matriz de riesgos

Riesgo	Descripción	Probabilidad	Impacto	Mitigación
Cambio inesperado de las condiciones del proyecto	El cliente solicita nuevos requerimientos, modificar los existentes o cambiar las condiciones (alcance, tiempo y/o costo) del proyecto luego del 80% de avance	Medio	Alto	Realizar una reunión con el cliente para negociar el alcance, costo y/o tiempo
Problemas de integración con las bolsas de valores	El equipo de desarrollo no logra crear las integraciones de las bolsas de valores con el software	Medio	Medio	Consultar con el personal de cada bolsa de valor las integraciones disponibles y la documentación para utilizarlas
Los stakeholders pierden interés en el proyecto	El equipo de stakeholders está en desacuerdo en la forma como se lleva a cabo el proyecto y/o no tienen interés en continuar	Bajo	Medio	Reunirse con los stakeholders para conocer su perspectiva del proyecto para negociar
El equipo de desarrollo no cumple los requerimientos del proyecto	El equipo de desarrollo no logra cubrir el alcance del proyecto debido a la desorganización o la falta de capacitación	Bajo	Alto	Contratar desde el inicio a personal cualificado y realizar ceremonias durante el proyecto para hacer seguimiento
El cliente está insatisfecho con los resultados	Al presentarle al cliente los avances del proyecto, muestra una inconformidad en la forma como se está construyendo el software en términos de usabilidad ISO9126	Medio	Bajo	Consultar con el cliente si los desea revisar de nuevo los requerimientos del software y negociar costo y tiempo adicional
El proyecto no cumple con las normas ISO/IEEE implementadas	El equipo no pasa las auditorías de control para la certificación en normas de calidad ISO29110 e ISO12207	Bajo	Medio	Asegurarse que el equipo conoce los procesos de negocio que el software trabaja, y las normas de calidad.
El software presenta errores luego de su implementación	Luego de implementar el software en un ambiente de producción, este presenta fallas o no cumple con el comportamiento esperado	Medio	Alto	Tener un plan de calidad según lo indica la ISO1005 para seguir las mejores prácticas y asegurar que se cumpla usando la ISO730-2002

16. Recomendaciones del arquitecto de TI

Se busca presentar diferentes recomendaciones como un arquitecto de software para el proyecto Andina Trading, el cual consiste en desarrollar un sistema de gestión de transacciones en una bolsa de valores operando en varios países de Latinoamérica. Estas recomendaciones se centran en asegurar la eficiencia, escalabilidad, seguridad y mantenibilidad del sistema, garantizando su perfecto desempeño y calidad de código.

Recomendaciones para la arquitectura del software

Adopción de una arquitectura basada en microservicios

Por ser un proyecto de naturaleza distribuida y modular del proyecto, se recomienda usar una arquitectura basada en microservicios. Este tipo de arquitectura de software permite que cada componente del sistema (módulo de gestión de inversionistas, módulo de conexión con la bolsa, portal del inversionista, etc.) se implemente como un servicio independiente, facilitando su desarrollo, prueba y despliegue. Las ventajas de esta arquitectura están mejor descritas en el documento en que se aclara la arquitectura de software del proyecto.

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.

Utilización de patrones de diseño eficientes

Para mejorar la eficiencia y la organización del código, se recomiendan los siguientes patrones de diseño:

- **Patrón MVC:** Se aplicará este patrón para separar la lógica de presentación del frontend (Angular) y la lógica del backend (Spring Boot).
- **Patrón RESTful:** Las APIs RESTful son altamente compatibles con el uso de frontend y backend separados. Permite que el frontend en Angular pueda consumir los recursos y datos proporcionados por el backend de forma eficiente a través de HTTP.
- **Patrón Singleton:** Utilizado en el backend para la gestión de servicios clave como la autenticación y el manejo de sesiones.
- **Patrón Repository:** Manejará la persistencia de los datos de las transacciones y la información financiera mediante Spring Data JPA.
- **Patrón DTO (Data Transfer Object):** Optimiza la transferencia de datos entre el cliente y el servidor. En lugar de enviar todo el objeto de dominio, se pueden enviar solo los datos necesarios, lo que reduce la sobrecarga de la red.
- **Patrón Factory:** Este patrón puede ser utilizado para crear instancias de objetos de forma flexible, como los servicios de autenticación y validación de

JWT. Permite una creación controlada de instancias, lo que simplifica la integración de diferentes tipos de autenticaciones en el futuro.

Elección de tecnologías y frameworks escalables

Para asegurar el rendimiento y escalabilidad del proyecto es muy importante seleccionar tecnologías y frameworks adecuados:

- **Backend:** Utilizar un framework ligero y modular como Spring Boot, que se ajusta bien a la arquitectura de microservicios y proporciona herramientas para seguridad, manejo de datos, y configuración de API RESTful.
- **Frontend:** Desarrollar el portal del inversionista utilizando Angular, ya que ofrecen un alto rendimiento y una experiencia de usuario interactiva.
- **Base de Datos:** Optar por bases de datos relacionales como MySQL, ya que son excelentes para manejar datos estructurados y garantizar la integridad de los datos.

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.

Recomendaciones para el código y la calidad del software

Estrategias de testing automatizado

Para asegurar la calidad del software, se deben implementar diferentes tipos de pruebas automatizadas:

- **Pruebas Unitarias:** Se recomienda el uso de herramientas como JUnit para el backend y Jest para el frontend. Estas pruebas deben cubrir todas las funcionalidades críticas de cada microservicio y del portal del inversionista.
- **Pruebas de Integración:** Asegurar que los microservicios se comuniquen correctamente entre sí y con las bases de datos. Esto incluye pruebas de API para validar el correcto flujo de información.
- **Pruebas de Carga y Rendimiento:** Para validar que el sistema soporta la carga esperada en tiempo real, se deben realizar pruebas de carga utilizando herramientas como JMeter.

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 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.

17. Plan de calidad

Para el desarrollo del plan de calidad, utilizamos el marco para el desarrollo y gestión de planes de calidad ISO 10005.

Primero debemos establecer un propósito y un alcance para este plan:

- **Propósito:** Establecer las actividades y controles necesarios para asegurar la calidad en el desarrollo, pruebas, implementación y mantenimiento del software.
- **Alcance:** Este plan cubre los portales de comisionistas e inversionistas, y se centra en cumplir con los requisitos de funcionalidad, seguridad, rendimiento y usabilidad definidos para ambos usuarios.

Ahora debemos implementar una política de calidad en el proyecto:

- **Política de calidad:** Asegurar un sistema seguro y confiable que cumpla con los estándares de seguridad financiera (PCI DSS) y la normativa de datos personales, ley 1581 de 2012.
 - **Objetivos Generales de Calidad:**
 - Proveer una interfaz intuitiva y fácil de usar tanto para comisionistas como para inversionistas.
 - Cumplir con un tiempo de disponibilidad del sistema de al menos el 99.9%.
 - Proteger la información de los usuarios mediante las directivas de la ISO 27001

Los encargados de velar por el cumplimiento de este plan de calidad se detallan a continuación:

- **Gerente de Proyecto:** Responsable de la implementación general del plan de calidad.
- **Equipo de Desarrollo:** Responsable de aplicar las políticas de calidad en el código y asegurar su funcionalidad siguiendo la norma ISO 12207.
- **Equipo de ciberseguridad:** Responsable de implementar y monitorear prácticas de seguridad como lo establece la ISO 27001.
- **Equipo de Pruebas:** Responsable de validar funcionalidad, seguridad y rendimiento mediante pruebas específicas establecidas en la IEEE 730-2002, de QA y Testing. Para esto, el equipo deberá estar conformado por un Ingeniero QA para desarrollar pruebas automatizadas y un analista QA para documentarlas.

Los requisitos de calidad del proyecto vienen definidos por los requerimientos funcionales y no funcionales de este documento.

Respecto a la gestión de recursos para este plan de calidad, evaluamos 3 aspectos:

- **Recursos humanos:** Equipo de desarrollo (5 personas), equipo de seguridad (2 personas), testers (3 personas), auditor de calidad (1 persona).
- **Recursos materiales:** Uso de herramientas de control de versiones, en este caso Git y GitHub, herramientas de prueba de seguridad, en este caso OWASP, y por último infraestructura de servidores en la nube (IaaS).
- **Recursos económicos:** Presupuesto para capacitación en prácticas de seguridad y para adquirir herramientas de automatización de pruebas. Estimado: 17'000.000, lo que incluye las licencias de las herramientas y el desarrollo de las capacitaciones.

Para asegurar el cumplimiento de este plan, se establece la siguiente estrategia:

- **Control de Calidad:** Revisiones de código semanales para mantener estándares de desarrollo. Por otro lado, la documentación y resolución de errores críticos detectados en cada ciclo de desarrollo será fundamental.
- **Aseguramiento de Calidad:** Auditorías de calidad internas cada tres meses y el análisis de métricas de calidad, como la cantidad de errores críticos, tiempos de respuesta y cumplimiento de SLAs.

Es importante conocer la opinión del cliente respecto al proyecto para evaluar la efectividad de este plan, para ello planeamos evaluaciones trimestrales por parte del cliente y de los usuarios para medir la satisfacción de los mismos.

Por último, en caso de que el cliente recomiende un cambio durante la revisión del plan de calidad trimestral, establecimos un proceso formal utilizando un formulario para evaluar el impacto de dicho cambio en materia de alcance, tiempo y costo.

18. Referencias

- Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide*. URL:
<https://www.scrumguides.org>
- Rubin, K. S. (2012). *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. Addison-Wesley.
- Larman, C. (2004). *Agile and Iterative Development: A Manager's Guide*. Addison-Wesley.
- Engel, G. L. (1977). The need for a new medical model: A challenge for biomedicine. *Science*, 196(4286), 129-136.
- Universidad El Bosque. (2021). *Modelo Biopsicosocial*. URL:
<https://www.unbosque.edu.co>
- CMF Chile. (2021). *Participantes del Mercado de Valores*. URL:
<https://www.cmfchile.cl>