PARCIAL FINAL SOFTWARE II

Presentado por:
Jesus David Cardenas Sandoval
Jesus Gabriel Parra Dugarte
Elkin Ariel Morillo Quenguan

Presentado a:
Julio Ariel Hurtado Alegria
Ricardo Zambrano

Ingeniería de Software II

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Programa de Ingeniería de sistemas

Popayán, Cauca

10/Noviembre/2023

Historias Épicas Mercado de valores

- **HE_001 Gestión de acciones**
  - HU001- Creación de acciones
  - HU002- Edición de acciones
  - HU003- Eliminación de acciones
  - HU004- Edición del precio una acción
  - HU005- Consultar todas las acciones
  - HU006- Filtro de acciones por id

Historias de usuario y criterios de aceptación

Mercado Valores HU/CA

Atributos de calidad

Atributos de calidad - Bolsa de valores

Backlog

Esquema del Backlog

Repositorio

Link GitHub

Demo

Video Demo

Tácticas de Escalabilidad

Retención del desempeño: Para lograr la táctica de retención del desempeño, tenemos la arquitectura de cliente-servidor y publicador-suscriptor, pues cada uno ayuda a mantener un alto rendimiento dentro de la bolsa de valores, mejorar la capacidad del sistema y el manejo de la concurrencia. Por ejemplo, al distribuir el trabajo entre el cliente y el servidor, se logra una mejor capacidad de respuesta y una gestión eficiente de concurrencia, ya que en un entorno financiero la velocidad de respuesta es algo muy crucial y esencial, así mismo en publicador-suscriptor nos ofrece mantener un rendimiento óptimo en el sistema a largo plazo, ya que proporciona una comunicación asíncrona. Esto es importante dentro de la bolsa de valores, pues al permitir suscribirse a un evento de interés, facilita la notificación eficiente de cambios y actualizaciones a los usuarios.
Rentabilidad: En este caso tenemos la arquitectura orientada a servicios que permite desplegar servicios de forma independiente, ofreciendo como resultado mejorar la eficiencia en el sistema, así puede adaptarse de una forma más fácil a cambios en los datos o a los servicios sin afectar al sistema en conjunto. De la misma manera tenemos la arquitectura de microservicios, cada servicio puede ser desplegado en su servidor virtual, siendo ideal para virtualizar los recursos del servidor. Esto es muy útil en la bolsa de valores ya que cada acción puede considerarse como un servicio independiente, lo que facilita la gestión, escalabilidad y mantenimiento de cada componente de manera única.

Tácticas de Modificabilidad

Reducir acoplamiento: Las arquitecturas que mejor se adaptan, es la de capas y la de microkernel, cada capa es independiente y la comunicación entre las capas es a través de interfaces, dando como resultado lo que buscamos, minimizar las dependencias entre los módulos y el microkernel, cada plugin es independiente entre sí, cada uno solo se conecta al núcleo del sistema, ofreciendo un acoplamiento mínimo y reduciéndolo. Esto resulta en un acoplamiento mínimo entre módulos y de esa manera permite cambios en una capa sin afectar las demás, brindando flexibilidad y facilitando futuras modificaciones sin impactar toda la aplicación. Además de eso la reducción del acoplamiento es esencial dentro del sistema para su fácil modificación, ya que si se busca escalar a nivel departamental o nacional la independencia de capas facilita la gestión de cambios sin afectar otras áreas críticas del sistema. Así mismo la arquitectura de microkernel permite incorporar nuevas funcionalidades de manera modular sin afectar el núcleo central minimizando el núcleo del sistema a funciones esenciales, delegando servicios adicionales a plugins.
Reducir el tamaño de un módulo: Las arquitecturas que se pueden considerar son la de modelo-vista-controlador, siendo ideal para separar un módulo en varios módulos más pequeños y permitiendo una modificación de manera independiente. La capacidad de modificar la interfaz de usuario, la lógica de negocio o la gestión de datos de manera independiente facilita la evolución del sistema sin afectar otras áreas críticas. La de microservicios, cada microservicio es pequeño y se encarga de una funcionalidad, de tal manera que se puede reducir el módulo en varios microservicios con una única responsabilidad, esto permite reducir el tamaño de un módulo a través de la encapsulación de funcionalidades específicas en servicios pequeños.

Atributo a descomponer: Escalabilidad

Perspectivas

Perspectiva de componentes
La perspectiva de componentes se enfoca en descomponer el sistema en módulos o componentes y la interacción entre ellos, de tal manera que que se puedan modificar sin afectar a los demás y al estar distribuidos mejoren la capacidad de respuesta del sistema, un patrón arquitectónico que facilita esta perspectiva, es el de microservicios , ya que permite que partes específicas del sistema se desarrollen e implementen de manera independiente o individual facilitando la escalabilidad en que cada componente.

Perspectiva de módulos
La perspectiva de módulos, tiene un enfoque sobre la organización modular y evaluar cómo se comunican los módulos, la arquitectura publicador-suscriptor nos ofrece una comunicación asíncrona, además una gran escalabilidad en cada módulo y desacoplamiento, cada módulo se debe interesar a eventos específicos de tal manera que tendrá solo una responsabilidad y se encargará de ella, ofreciendo una gran escalabilidad en el sistema y una independencia entre módulos para permitir el desacoplamiento entre ellos.

Perspectiva de localización
La perspectiva de localización se centra en la organización espacial y disposición física de los elementos, por lo tanto, el patrón arquitectónico a utilizar será n-tier, dicho patrón nos permite desplegar las capas en ubicaciones geográficas específicas, y nos brinda utilidad para optimizar el rendimiento y latencia ubicando dichas capas donde se necesite un procesamiento más intensivo.

MODELO C4

Contexto
En el primer nivel tenemos el contexto, en el cual identificamos nuestro sistema principal este sistema será una aplicación de seguimiento del mercado de valores , tendremos el frontend y el backend de la aplicación. Además servicios externos como Postman y RabbitMQ que nos apoyaran en la implementación.
DiagramaC4-Diagrama de Contexto.drawio.png

Contenedor
En el segundo nivel encontramos dos contenedores por parte del backend de la aplicación: la aplicación de gestión de acciones, que se realizará en Java Spring Boot y una base de datos relacional que inicialmente será MYSQL.
Diagrama de Contenedor Sistema Bolsa de Valores.drawio.png

Por parte del frontend tenemos un solo contenedor que es la aplicación desktop que permite la interacción del cliente con el sistema.
_Diagrama de Contenedor MiAgenteFinanciero.drawio.png

Componente
La arquitectura del sistema estará basada en la arquitectura hexagonal o microservicios,
por lo que tendremos el núcleo de la aplicación que contendrá el todo el funcionamiento, en un nivel superior tendremos del núcleo tendremos el dominio , contendrá el modelo y los repositorios correspondientes , y por último tendremos la infraestructura, tiene todo el funcionamiento externo de la aplicación , que funcionara como adaptadores para conectar los servicios requeridos en la aplicación.
Diagrama de Componentes-MercadoValores.drawio.png

Por la otra parte de la aplicación , tendremos un montaje básico para su funcionamiento , ya que solo se encargará de mostrar mensajes a los usuarios.
Diagrama de Componentes-MiAgenteFinanciero.drawio.png

Clases
En este último nivel tenemos un detalle sobre todas las clases que tiene el sistema, muestra cada clase, sus atributos y métodos y las relaciones entre ellas.
DiagramaUMLMercadoValores.png
DiagramaUMLMiAgenteFinanciero.png

Diagrama de secuencia
Diagrama de secuencia Registrar Accion.png

Diagrama de Módulos
Diagrama_Modulos.png

Diagrama de componentes
Diagrama_Componentes.jpg