TRABAJO CONSULTA PROGRAMACION FUNCIONAL Y REACTIVA(documento información)

NOMBRE: Jose David Caraguay Armijos

1.MANIFIESTO REACTIVO

Los sistemas construidos como Sistemas Reactivos son más flexibles, con bajo acoplamiento y escalables. Esto hace que sean más fáciles de desarrollar y abiertos al cambio. Son significativamente más tolerantes a fallos y cuando fallan responden con elegancia y no con un desastre. Los Sistemas Reactivos son altamente responsivos, dando a los usuarios un feedback efectivo e interactivo. Los Sistemas Reactivos son:

-Responsivos: El sistema responde a tiempo en la medida de lo posible. La responsividad es la piedra angular de la usabilidad y la utilidad, pero más que esto, responsividad significa que los problemas pueden ser detectados rápidamente y tratados efectivamente

-Resilientes: La resiliencia es alcanzada con replicación, contención, aislamiento y delegacion. Los fallos son manejados dentro de cada componente, aislando cada componente de los demás, y asegurando así que cualquier parte del sistema pueda fallar y recuperarse sin comprometer el sistema como un todo.

-Elásticos: Los Sistemas Reactivos pueden reaccionar a cambios en la frecuencia de peticiones incrementando o reduciendo los recursos asignados para servir dichas peticiones. -Orientados a Mensajes: Los Sistemas Reactivos confían en el intercambio de mensajes asíncrono para establecer fronteras entre componentes, lo que asegura bajo acoplamiento, aislamiento y transparencia de ubicación.

Manifiesto Reactivo: Responsive (Responsivo): El sistema debe responder de manera rápida a los cambios y a las interacciones del usuario.

Resilient (Resiliente): El sistema debe ser capaz de manejar errores y fallos de manera adecuada, proporcionando una experiencia de usuario sin interrupciones.

Elastic (Elástico): El sistema debe ser escalable y capaz de adaptarse a diferentes cargas de trabajo, permitiendo un uso eficiente de los recursos.

Message-Driven (Basado en Mensajes): Las comunicaciones entre componentes del sistema deben basarse en mensajes, lo que favorece la independencia y la capacidad de reacción a cambios.

2. REALICE UNA CONSULTA SOBRE LA PROGRAMACION REACTIVA

Decimos que algo es reactivo cuando reacciona, un ejemplo de esto son las celdas de excel que se calculan en base a otras, notas cómo cuando actualizamos el dato de una celda, el de la otra se actualiza automáticamente, luego por supuesto de ejecutar las operaciones deseadas, ese es el concepto abstracto de programación reactiva. La programación reactiva, es un paradigma de programación, tal como lo son la programación orientada a objetos o la programación funcional. En el centro de este paradigma están los datos y la ejecución asíncrona de las operaciones que realiza nuestro programa. La ejecución asíncrona de instrucciones trae incontables beneficios, mejoras a la experiencia de usuario, al rendimiento de tu código, etc. Sin embargo, es más complicada de razonar y puede presentar desafíos como el compartir información entre dos operaciones que se ejecutan en distintos puntos del tiempo. Uno puede pensar en la ejecución de un programa, como una línea del tiempo, como una flecha que va del inicio del programa, hasta el final. Durante este proceso, algunas operaciones se ejecutan en un orden previamente definido, primero a, luego b y luego c, a esto se le conoce como programación síncrona. Por otro lado, existen algunas operaciones que se ejecutan y sólo traen información al flujo, aunque no se sabe cuándo, a estos procesos los llamamos asíncronos. Aquí hay algunos conceptos clave asociados con la programación reactiva: Eventos y Flujos de Datos: En la programación reactiva, el flujo de datos y los eventos son fundamentales. Los eventos pueden ser cualquier cosa, desde clics del usuario hasta cambios en datos. Estos eventos se consideran como flujos de datos que pueden propagarse a través de la aplicación. Asincronía: La asincronía es esencial en la programación reactiva. Las operaciones no bloqueantes permiten que la aplicación continúe ejecutándose sin esperar a que se completen ciertas tareas. Esto es crucial para mantener la capacidad de respuesta de la aplicación, especialmente en entornos de interfaz de usuario.

3. PATRON OBSERVER

El patrón Observer es un patrón de diseño de software que se utiliza en programación reactiva para establecer una relación de dependencia uno a muchos entre objetos, de modo que cuando un objeto cambia su estado, todos sus dependientes son notificados y actualizados automáticamente. Este patrón es fundamental en la programación reactiva y se utiliza ampliamente en el desarrollo de sistemas que necesitan reaccionar a cambios en tiempo real también se puede decir que es un patrón de diseño en el que hay dos tipos de objetos: observadores y sujetos observables. Un observador es un objeto que observa los cambios de uno o más sujetos; un sujeto es un objeto que mantiene una lista de sus observadores y les notifica automáticamente cuando cambia de estado.

Los componentes del Patrón Observer son los siguientes:

IObservable:

-Interface que deben de implementar todos los objetos que quieren ser observados, en ella se definen los métodos mínimos que se deben implementar.

-Mantiene una lista de observadores y proporciona métodos para agregar, eliminar y notificar observadores.

ConcreteObservable

-Clase que desea ser observada, ésta implementa IObservable y debe implementar sus métodos.

- Los observadores registran su interés en el sujeto llamando a métodos proporcionados por el sujeto para registrarse y desregistrarse.

IObserver

-Interfaces que deben implementar todos los objetos que desean observar los cambios de IObservable.

- Implementa las operaciones específicas del dominio relacionadas con la gestión de observadores.

ConcreteObserver

-Clase concreta que está atenta de los cambios de IObserver, esta clase hereda de IObserver y debe de implementar sus métodos.

- Almacena una referencia al sujeto si necesita consultar detalles adicionales al recibir una notificación.

4. VENTAJAS/DESVENTAJAS

VENTAJAS

-Mejora la escalabilidad: La programación reactiva aprovecha los recursos de forma eficiente y puede escalar fácilmente para manejar un gran número de solicitudes.

-Alta disponibilidad: La programación reactiva se diseña para asegurar que los sistemas estén disponibles durante los períodos de alta demanda.

-Mejora la eficiencia: La programación reactiva se diseña de tal manera que no se desperdicien recursos. Esto mejora la eficiencia y reduce los costos de computación.

-Simplifica la programación de aplicaciones concurrentes: La programación reactiva simplifica la creación de aplicaciones concurrentes, lo que permite a los desarrolladores centrarse en la lógica de negocios en lugar de administrar los hilos.

-Mejora el rendimiento: La programación reactiva se diseña para minimizar el tiempo de respuesta y mejorar el rendimiento de la aplicación. Esto permite que la aplicación responda rápidamente a las solicitudes del usuario.

DESVENTAJAS

-Depuración Más Complicada: En algunos casos, la depuración de sistemas reactivos puede ser más complicada debido a la naturaleza asincrónica y a la propagación de eventos.

-Posible Sobrecarga: En algunos casos, la introducción de una arquitectura reactiva puede resultar en un código más complejo y, en ciertos contextos, puede haber una sobrecarga de abstracción.

-Posible Problema de Recursos: En sistemas mal diseñados, especialmente cuando se trata de grandes flujos de eventos, podría haber problemas de consumo excesivo de recursos.

-Mayor curva de aprendizaje: Debido a la mayor complejidad de la programación reactiva, también hay una mayor curva de aprendizaje para los desarrolladores. Esto significa que los desarrolladores pueden necesitar pasar más tiempo aprendiendo y practicando los conceptos básicos antes de poder comenzar a escribir código.

-Falta de soporte: Debido a que la programación reactiva es relativamente nueva, todavía hay una falta de soporte en comparación con otras formas de programación más establecidas. Esto significa que los desarrolladores pueden tener dificultades para encontrar una amplia gama de documentación y recursos útiles si necesitan ayuda para construir aplicaciones con programación reactiva.

5. FRAMEWORKS REACTIVOS

Un framework reactivo en programación reactiva es un conjunto de herramientas y patrones de diseño que facilitan el desarrollo de aplicaciones que responden de manera eficiente a cambios en los datos o eventos. La programación reactiva se centra en la propagación automática de cambios y en la gestión de eventos asíncronos. En el contexto de la programación reactiva, el término "reactivo" se refiere a la capacidad de reaccionar automáticamente a los cambios en los datos, eventos o condiciones del sistema, en lugar de depender de enfoques más tradicionales basados en la programación imperativa.

En cada framework de programación reactiva se llaman de una forma distinta, pero básicamente son un conjunto de hilos donde se van a ejecutar las operaciones, y unas reglas que optimizan su uso para ser lo más eficientes posibles.

¿Dónde se aplica la Programación Reactiva?

Uno de los aspectos más importantes, es dónde y cómo aplicar la Programación Reactiva en los lenguajes de Java y para quienes utilizan frameworks de Javascript.Para aquellos que utilizan Java, se destaca RxJava. Esta librería y su primera versión 1.x fue una de las pioneras en el desarrollo reactivo Java y actualmente, se está trabajando en la versión 3. Es importante distinguir que se encuentra integrada en diferentes frameworks como Spring MVC, Spring Cloud o Netflix OSS.

Por otro lado, la recomendación para aquellos que utilizan frameworks de Javascript es React + redux-saga: Este es un framework que se considera reactivo y se controla integrando un estado global a la aplicación, para ello, se actualiza constantemente con efectos colaterales.

Frameworks reactivos más conocidos en diferentes plataformas:

1.ReactiveX (Rx):

•Descripción: ReactiveX es una biblioteca que implementa el patrón Observer y proporciona un conjunto de operadores para manipular flujos de datos asíncronos y eventos. Existen implementaciones de ReactiveX para varios lenguajes, como RxJava (Java), RxJS (JavaScript), RxSwift (Swift), entre otros.

2.Project Reactor:

•Descripción: Project Reactor es un framework reactivo para el desarrollo de aplicaciones en Java. Está basado en el modelo de programación reactiva y proporciona abstracciones como Flux (para flujos de datos con cero o más elementos) y Mono (para flujos de datos con cero o un elemento). Se utiliza comúnmente en combinación con Spring Framework para el desarrollo de aplicaciones empresariales.

3.Akka Streams:

•Descripción: Akka Streams es parte del conjunto de herramientas Akka para el desarrollo de sistemas concurrentes y distribuidos en Scala y Java. Proporciona una API para procesar secuencias de elementos de manera reactiva y eficiente. Akka Streams se integra con el modelo de actores de Akka para manejar la concurrencia y la escalabilidad.

4.Vert.x:

•Descripción: Vert.x es un toolkit para el desarrollo de aplicaciones reactivas y escalables en la Máquina Virtual de Java (JVM). Proporciona una arquitectura basada en eventos y soporta lenguajes como Java, JavaScript, Kotlin y Groovy. Vert.x incluye módulos para manejar HTTP, WebSocket, bases de datos y otros protocolos.

6. OPERADORES

Un operando puede ser una constante, una variable o el resultado de una función. Los operadores son aritméticos, lógicos y relacionales. Al igual que sucede en C, la funcionalidad de un operador puede variar según el tipo de datos de los operandos especificados en la expresión. Estos operadores permiten realizar diversas operaciones en los eventos o datos que fluyen a través de un stream (flujo), y son fundamentales para trabajar de manera eficiente y expresiva en entornos reactivos.

Los operadores en programación reactiva se pueden clasificar en varias categorías según la funcionalidad que proporcionan estos son algunos ejemplos:

Operadores de Transformación:

Cambian los elementos de un flujo. Por ejemplo, el operador map transforma cada elemento de acuerdo con una función proporcionada.

Operadores de Filtrado:

Permiten seleccionar o excluir ciertos elementos del flujo basándose en ciertos criterios. Ejemplos incluyen filter y distinct.

Operadores de Combinación:

Combina múltiples flujos en uno solo o realiza operaciones con elementos de flujos diferentes. Ejemplos son merge, zip y concat.

Operadores de Tiempo:

Gestionan eventos en función del tiempo. Por ejemplo, debounce espera un tiempo después de la última emisión antes de permitir que un evento pase.

Operadores de Manejo de Errores:

Proporcionan manejo de errores en los flujos. Ejemplos incluyen catch y retry.

Operadores de Agregación:

Realizan operaciones de agregación en los elementos de un flujo. Por ejemplo, reduce combina todos los elementos en uno solo mediante una función acumulativa.

Operadores de Utilidad:

Ofrecen funcionalidades adicionales, como do para realizar acciones secundarias en los elementos del flujo sin afectar el flujo principal.

Operadores de Condiciones:

Permiten realizar operaciones basadas en condiciones. Por ejemplo, takeWhile emite elementos mientras se cumpla una condición dada.

7. ELABORE UN EJEMPLO

BIBLIOGRAFIA

-Goette, E. (s/f). Emanuel Goette, alias Crespo. Blogspot.com. Recuperado el 15 de enero de 2024, de https://emanuelpeg.blogspot.com/2021/09/el-patron-observador-en-la-programacion.html

-Programación Reactiva: Ventajas y Desventajas. (2022, diciembre 14). Radio Michi. https://radiomichi.com/programacion-reactiva-ventajas-y-desventa/

-Doonamis. (2020, diciembre 11). La programación reactiva: características y aplicación. Doonamis. https://www.doonamis.com/que-es-la-programacion-reactiva/

-Rubio, E. (2017, noviembre 14). Programación Reactiva, uso de la librería RxJs. Adictos al trabajo. https://www.adictosaltrabajo.com/2017/11/14/programacion-reactiva-uso-de-la-libreria-rxjs/