DESARROLLO EFICAZ DE APPS REACTIVAS

V.I. Quino Quino
7690-17-6433 Universidad Mariano Gálvez
Seminario de Tecnologías de Información
vquinoq@miumg.edu.gt

Resumen
Las aplicaciones reactivas se consideran una solución efectiva al problema de manejar grandes volúmenes de datos y proporcionar respuestas rápidas y sobre todo confiables. Este artículo presenta un análisis que busca profundizar de lo que se considera mejores practicas para el desarrollo de aplicaciones reactivas siguiendo los principios del Manifiesto Reactivo: responsividad, resiliencia, elasticidad y comunicación basada en mensajes. Se examina técnicas como el procesamiento asíncrono para mejorar en temas de responsividad, mecanismos de tolerancia a fallas para mejorar la resiliencia, estrategias de escalabilidad horizontal para adaptarse a cargas variables. Por medio de este artículo se pueden adquirir un entendimiento profundo de cómo aplicar técnicas en sus proyectos, mejorando así la calidad y eficiencia de sus aplicaciones reactivas, la adopción disciplinada de estas prácticas permite un desarrollo profesional en la implementación de soluciones a sus proyectos.

Palabras claves: aplicaciones reactivas, sistemas distribuidos, programacion asincrona

Desarrollo del tema
Las aplicaciones reactivas son sistemas que se caracterizan por su capacidad para responder de manera oportuna, seguir operando ante fallos, ajustarse a cargas variables y comunicarse de forma asíncrona mediante mensajes. Estos atributos las hacen particularmente adecuadas para entornos modernos de computación, donde la distribución, la escalabilidad y la tolerancia a fallos son cruciales. Estos principios no solo se aplican al desarrollo de backend, sino también al frontend, donde frameworks como React y Angular utilizan conceptos reactivos para gestionar el estado de la interfaz de usuario de manera eficiente. El concepto de aplicaciones reactivas fue publicado inicialmente en el 2014. El manifiesto define cuatro principios fundamentales que deben seguir los sistemas actuales y son los siguientes:

1. **Responsivo:** El sistema debe proporcionar respuestas rápidas y consistentes, mejorando la experiencia del usuario y facilitando la detección de problemas.

2. **Resiliente:** Debe ser capaz de recuperarse de fallos, utilizando técnicas como la replicación y el aislamiento para evitar que un fallo afecte a todo el sistema.

3. **Elastico:** Debe poder escalar horizontalmente para manejar aumentos en la carga, ajustando los recursos de manera eficiente.

4. **Basado en mensajes:** La comunicación entre componentes debe ser asíncrona y basada en mensajes, permitiendo un desacoplamiento que facilita la gestión de la carga y la tolerancia a fallos.

Para lograr que una aplicación sea responsiva, es fundamental utilizar procesamiento asíncrono y programación no bloqueante. Esto permite que el sistema maneje múltiples solicitudes concurrentemente sin que una operación lenta bloquee a las demas. Frameworks como Spring WebFlux en Java facilitan la implementación de estos patrones mediante el uso de tipos reactivos como Mono y Flux.

La resiliencia se logra implementando mecanismos que permitan al sistema recuperarse de fallos sin comprometer su funcionamiento general. Patrones como el circuit breaker ayudan a prevenir fallos en cascada al interrumpir temporalmente las solicitudes a servicios que están fallando. Asimismo, el uso de fallbacks proporciona respuestas alternativas cuando un servicio no está disponible. El aislamiento de componentes es crucial; cada parte del sistema debe poder fallar de manera independiente sin afectar a los demás. Esto se puede lograr mediante contención y delegación, donde los fallos son manejados localmente siempre que sea posible.

Para que una aplicacion sea elastica, debe ser capaz de escalar horizontalmente, anadiendo mas instancias segun sea necesario para manejar aumentos en la carga. Esto requiere un diseno que evite puntos de contencion y permita la distribucion uniforme de la carga. El uso de balanceo de carga y autoescalado en entornos de nube, como AWS Auto Scaling, permite ajustar dinamicamente los recursos en funcion de la demanda, optimizando asi los costos y el rendimiento.

En sistemas distribuidos, mantener la consistencia de los datos puede ser desafiante. A menudo, se opta por la consistencia eventual, donde los datos convergen a un estado consistente con el tiempo, en lugar de garantizar consistencia fuerte en todo momento. Tecnicas como los CRDT (Conflict-free Replicated Data Types) o el event sourcing pueden ser utiles en este contexto. Es crucial elegir el modelo de consistencia adecuado segun las necesidades del negocio, balanceando entre rendimiento y garantias de consistencia.

Finalmente, es importante mantener una estructura de código limpia y modular. La separación de responsabilidades, el uso de patrones como Repository, Service y Handler, y la documentación clara de los flujos reactivos contribuyen a la mantenibilidad del sistema. Ademas, se recomienda realizar ´ pruebas unitarias y de integración utilizando herramientas como ´ JUnit 5, Mockito y Testcontainers para validar el comportamiento reactivo bajo diferentes condiciones. Las pruebas deben simular escenarios de carga, errores y recuperación para garantizar que el sistema responde adecuadamente ´ en situaciones reales.

Observaciones y comentarios
La implementación de estas mejores prácticas requiere un cambio de mentalidad en el desarrollo de software, pasando de enfoques tradicionales a paradigmas más adaptados a los sistemas distribuidos y las altas exigencias de rendimiento. Aunque puede haber una curva de aprendizaje inicial, los beneficios en términos de escalabilidad, resiliencia y mantenibilidad son significativos. Es importante que los equipos de desarrollo se capaciten adecuadamente y utilicen las herramientas y frameworks apropiados.

Conclusiones

1. Las aplicaciones reactivas ofrecen una solución efectiva para los desafíos de los sistemas distribuidos modernos, proporcionando responsividad, resiliencia, elasticidad y comunicación basada en mensajes.
2. La programación no bloqueante y el manejo eficiente de errores son fundamentales.
3. La adopción disciplinada de buenas prácticas garantiza aplicaciones reactivas sostenibles.

Bibliografía

- Boner, J., Farley, D., Kuhn, R., & Thompson, M. (2014). The Reactive Manifesto. Recuperado de https://www.reactivemanifesto.org/
- Reactive Foundation. (2022). The Reactive Principles. Recuperado de https://principles.reactive.foundation/
- Platzi. (s.f.). Programacion reactiva: Guia para desarrolladores. Recuperado de https://platzi.com/blog/progr reactiva/
- Profile.es. (2017). ¿Que es la programacion reactiva? Recuperado de https://profile.es/blog/que-es-la-programacion-reactiva-una-introduccion/