Buenas Practicas en Desarrollo de Aplicaciones Reactivas

Investigador Edwin Morales*

2025, v-1.0.0

El desarrollo de aplicaciones reactivas es un paradigma de diseño y arquitectura de software enfocado en construir sistemas capaces de responder a eventos de forma asíncrona, garantizando un alto rendimiento y una experiencia de usuario fluida, incluso bajo cargas de trabajo pesadas. Se basa en el Manifiesto Reactivo, que establece cuatro principios fundamentales para crear aplicaciones robustas - La Reactividad, Contenido, Elasticidad y Orientado a Mensajes - permitiendo un buen rendimiento, escalabilidad de nuestras aplicacione, disponibilidad de los servicio y eficiencia en los procesos garantizando una buena experiencia al usuario.

Palabras-Clave: Desarrollo. Manifiesto, Principios, Respuesta.

Introducción

A través de este documento compartiremos conceptos que nos pueda ayudar a comprender lo que es inicialmente el tema de desarrollo de aplicaciones Reactivas; porque se les llama de esta forma.

Además estaremos compartiendo concepto generales de sobre como funcionan, basandose en los principios fundamentales para crear aplicaciones robustas que satisfagan las necesidades de nuestros clientes.

Además de comentar algunas herramientas con las cuales se disponen para la creación y manejo de las aplicaciones con este nuevo paradigma de programación el cual es un nuevo modelo de desarrollo que adicional apoyará a crear soluciones más resilientes, elásticas y reactivas que permitan una respuesta ágil de nuestras soluciones.

Podremos visualizar la relación de esta tecnología en los nuevos modelos en auge de las tecnologías.

[1].

*emoralesm@miumg.edu.gt

[1] <http://www.latex-project.org/lppl.txt>

1 Desarrollo del Tema

El desarrollo de aplicaciones reactivas es un paradigma de diseño y arquitectura de software enfocado en construir sistemas capaces de responder a eventos de forma asíncrona, garantizando un alto rendimiento y una experiencia de usuario fluida, incluso bajo cargas de trabajo pesadas

2 Manifiesto Reactivo

define cuatro principios clave: Reactividad, Resiliencia, Elasticidad y 4. Orientación a Mensajes

2.1 Reactividad (Responsive)

El sistema debe responder de manera rápida y consistente, proporcionando una experiencia de usuario fluida y sin retrasos perceptibles. Esto es la base del paradigma.

2.2 Resiliencia (Resilient)

El sistema se mantiene operativo incluso cuando falla. Los errores son aislados, gestionados de forma controlada y recuperados rápidamente sin afectar a la funcionalidad global. Se basa en el principio de "dejar que falle"(let it crash), manejando la recuperación en lugar de evitar el fallo.

2.3 Elasticidad (Elastic)

El sistema puede escalar o reducir sus recursos de forma elástica y automática. Puede manejar picos de carga aumentando los recursos y liberarlos cuando no son necesarios, utilizando de manera eficiente el hardware disponible.

2.4 Orientación a Mensajes (Message-Driven)

Los componentes del sistema se comunican entre sí de forma asíncrona mediante el intercambio de mensajes. Esto permite un desacoplamiento fuerte entre ellos, lo que es fundamental para lograr la resiliencia y la elasticidad.

3 Herramientas y Tecnologías para el Desarrollo Reactivo

Existen diversas tecnologías y frameworks que facilitan la implementación de los principios reactivos. Estos pueden ser tus "componentes o aliados"clave: Frameworks y Librerías de Programación Reactiva:
• Java/JVM:
• Spring WebFlux Project Reactor: La opción más popular en el ecosistema Java. Project Reactor es una librería que implementa el estándar Reactive Streams y se integra perfectamente con Spring WebFlux para construir APIs reactivas.
• Akka: Un toolkit para construir aplicaciones concurrentes, distribuidas y tolerantes a fallos en la JVM (usando Scala o Java). Se basa en el modelo de actores. • JavaScript/TypeScript:
• RxJS (Reactive Extensions for JavaScript): Una librería que utiliza "observables"para

manejar flujos de eventos de forma asíncrona y basada en push. Es fundamental en frameworks como Angular, y muy útil en React y Node.js.
• Node.js: Por su naturaleza de bucle de eventos (event loop) y arquitectura no bloqueante, es una plataforma ideal para el desarrollo reactivo.
• Kotlin:
• Coroutines: Un enfoque ligero para la concurrencia que permite escribir código asíncrono que parece síncrono, mejorando la legibilidad.

4 Bases de Datos y Mensajería

Entre las bases de datos podemos manejar como Reactivas las siguientes:

o MongoDB, Cassandra, Couchbase: Bases de datos NoSQL que, por su naturaleza distribuida y no relacional, se adaptan bien a arquitecturas reactivas y de alta escalabilidad.
o R2DBC (Reactive Relational Database Connectivity): Un estándar que permite usar bases de datos relacionales (PostgreSQL, MySQL) de manera reactiva, sin bloquear el hilo principal.

Buses de mensajes y Brokers
o Apache Kafka: Un bus de mensajes distribuido, ideal para manejar grandes volúmenes de eventos en tiempo real. Es el "aliado"perfecto para arquitecturas de microservicios basadas en eventos.
o RabbitMQ, ActiveMQ: Brokers de mensajería que permiten la comunicación asíncrona entre servicios.

5 Buenas Prácticas para Mejorar la Arquitectura y el Desarrollo

Entre las buenas practicas para este tipo de desarrollo podemos destacar:
• Arquitectura de Microservicios: Desacopla la lógica de negocio en servicios pequeños e independientes que se comunican de forma asíncrona. Esto potencia la resiliencia y la elasticidad.
• Arquitecturas Event-Driven: Diseña tus sistemas para que las acciones sean eventos que se publican y que otros servicios pueden consumir. Kafka es la herramienta ideal para esto.
• Uso de Circuit Breakers: Implementa el patrón "interruptor de circuito". Si un servicio dependiente falla, el circuito se "abre"y evita enviar más peticiones, lo que previene fallas en cascada y da tiempo al servicio para recuperarse.
• Programación Funcional y Flujos de Datos Inmutables: Trata los datos como flujos inmutables y utiliza funciones puras para procesarlos. Esto reduce los errores de concurrencia y simplifica la depuración.
• Gestión de Errores como Flujos de Datos: En la programación reactiva, los errores se manejan como eventos dentro del flujo de datos, permitiendo una gestión más robusta y declarativa.
• Monitoreo y Observabilidad: Implementa herramientas de monitoreo (como Prometheus, Grafana) para tener visibilidad de la salud, rendimiento y errores de cada servicio en tiempo real. Esto es vital para un sistema elástico y resiliente.

6 Conclusiones

Podemos destacar un gran beneficio en nuestras aplicaciones el implementar aplicaciones reactivas lo cual nos permite dar una mejor experiencia de servicio a nuestros clientes. Obtenemos un mejor rendimiento, alta disponibilidad, escalabilidad en nuestras aplicaciones y servicios que proveemos en general. Reducimos tiempos y costos ya que estamos enfocados en la disponibilidad y resilencia de nuestros servicios Podemoas proyectar el futuro del desarrollo de software en este nuevo paradigma de programación conociendo los beneficios que podemos obtener.

Desarrollo de aplicaciones Reactivas

Investigador Edwin Morales[†]

2025, v-1.0.0

Key-words: Desarrollo. Manifiesto, Principios, Respuesta.

Referências

Fuentes Primarias y Manifiestos El Manifiesto Reactivo (The Reactive Manifesto): URL: https://www.reactivemanifesto.org/es

Manifiesto del Microservicio (Microservice Manifesto): URL: https://www.microservice-manifesto.org/

Libro: Reactive Programming with RxJava de Tomasz Nurkiewicz y Ben Christensen

Sitio Oficial de Project Reactor: URL: https://projectreactor.io/

[†]emoralesm@miumg.edu.gt