Miguel González Navarro UO282337@uniovi.es

Saúl Tuñón Fernández UO277490@uniovi.es

José Jiménez García UO276008@uniovi.es

ARQUITECTURA INMUTABLE

Introducción

Michael L. Perry comienza discutiendo su último libro, ***The Art of Immutable Architecture***, en el que usa ejemplos familiares como git y blockchain, diferencia las arquitecturas inmutables de otros enfoques y aborda posibles conceptos erróneos sobre el diseño de arquitecturas inmutables. La mayoría de las personas a menudo confunden el término arquitectura inmutable con la infraestructura inmutable, que es un concepto que define cómo se configura una máquina para que no pueda cambiar esa configuración.

Por lo tanto, la inmutabilidad es el valor predeterminado en un lenguaje de programación funcional, pero necesita una sintaxis adicional para decir que algo es un lenguaje de programación mutable orientado a objetos y, a menudo, al revés. Esta es una de las razones por las que la gente se refiere a la inmutabilidad como funcional en lugar de orientada a objetos.

La arquitectura inmutable es un enfoque de diseño de software que se basa en la idea de que una vez que se crea un objeto o estructura de datos, no se puede cambiar su estado. En otras palabras, los objetos son inmutables y no pueden ser modificados después de su creación.

Beneficios/ventajas

- Puede razonar sobre sus sistemas y comprender lo que deben hacer, lo que es casi imposible de hacer con los sistemas distribuidos. También puede conocer las restricciones que debe cumplir para lograr ciertas propiedades y cuáles no se pueden lograr en un sistema distribuido.

- Los objetos inmutables son más seguros porque una vez que se crean, no se pueden modificar. Esto reduce la posibilidad de errores debido a cambios accidentales de datos.

- Puede trabajar de manera más eficiente en un entorno paralelo porque no tiene que tomar medidas especiales para evitar que los objetos compartidos sean modificados por varios subprocesos de ejecución.

- Los objetos inmutables tienen un comportamiento predecible y no cambian su estado después de la creación, lo que hace que su código sea más legible, comprensible y fácil de mantener. Esto hace que el software sea más estable y menos propenso a errores.

- Escalabilidad: los objetos inmutables se pueden compartir fácilmente entre múltiples procesos, lo que facilita la creación de sistemas distribuidos y de alta disponibilidad.

Inconvenientes

- La implementación es a menudo una experiencia dolorosa, aterradora y que consume mucho tiempo sin una automatización bien probada. Las arquitecturas inmutables pueden ser más difíciles de implementar y comprender que las arquitecturas mutables tradicionales. Esto puede aumentar el tiempo y el costo de desarrollo.

- Las arquitecturas inmutables pueden ser más difíciles de implementar en sistemas grandes y complejos, como árboles o gráficos, ya que hacer copias de objetos puede requerir más memoria y

recursos. Sin embargo, para sistemas que requieren alta confiabilidad y tolerancia a fallas, una arquitectura inmutable puede ser una buena opción.

- La creación de nuevas copias de objetos inmutables puede consumir mucha memoria, especialmente en sistemas grandes y complejos.

Cambios en Arquitectura inmutable

Algo que puede llamar la atención de esta arquitectura inmutable es el cómo se realizan los cambios en los objetos una vez creados, respecto a esto, Michael en el podcast explica que los objetos en la arquitectura inmutable no cambian una vez son creados. Por lo tanto, para realizar un cambio se crea una copia del objeto con las modificaciones necesarias, y una vez modificado se sustituye por el antiguo. Esta técnica proporciona un código más fácil de entender y manejar, ya que los objetos no pueden ser modificados desde ninguna parte de la aplicación, además, proporciona una mayor seguridad y escalabilidad, ya que los objetos inmutables son menos propensos a errores y más fáciles de distribuir en sistemas distribuidos.

Patrones

Inmutabilidad en cascada

Este patrón, mencionado en el podcast, es un patrón de diseño que se utiliza para crear objetos inmutables a partir de objetos existentes. Se basa en la idea de que un objeto inmutable se puede construir a partir de otro objeto inmutable y que cada objeto inmutable creado es una versión modificada del objeto anterior. Este nuevo objeto se puede utilizar para crear un objeto inmutable adicional, y así sucesivamente, lo que lleva a una cascada de objetos inmutables.

El patrón de inmutabilidad en cascada puede ayudar a mantener la coherencia y la integridad de los datos en sistemas distribuidos, ya que cada objeto inmutable creado es una versión consistente y completa del objeto anterior. También puede mejorar la eficiencia, ya que cada objeto inmutable creado se puede reutilizar en lugar de crear nuevos objetos desde cero.

Cambio estructural

En la arquitectura inmutable, los objetos no pueden modificarse una vez creados. Si se necesita cambiar un objeto inmutable, se debe crear una nueva versión del objeto que contenga los cambios deseados. Sin embargo, si se tienen muchos objetos inmutables que dependen de un objeto común, cambiar ese objeto puede resultar en la necesidad de recrear muchos otros objetos.

El patrón de cambio estructural aborda este problema al proporcionar una manera de actualizar los objetos inmutables sin tener que recrearlos todos. En lugar de crear una nueva versión de cada objeto que depende del objeto que se va a cambiar, se crea una nueva versión del objeto común y se actualizan todos los objetos que dependen de él.

Este patrón se puede implementar utilizando una estructura de datos compartida, como un árbol o un grafo, y creando nuevas versiones de los nodos que necesitan ser actualizados. De esta manera, se puede cambiar la estructura del árbol o grafo sin tener que recrear todos los objetos que dependen de él.

El patrón de cambio estructural puede mejorar la eficiencia y la escalabilidad de los sistemas inmutables al reducir la cantidad de objetos que necesitan ser recreados en caso de cambios en la estructura de datos compartida.

El Futuro y el por qué de la arquitectura inmutable

Un gran potencial que tiene la arquitectura inmutable es su potencial para mejorar la seguridad de los sistemas. Al construir sistemas inmutables, se reduce la superficie de ataque para posibles vulnerabilidades de seguridad. Como las piezas inmutables no pueden ser modificadas después de ser creadas, se evita la posibilidad de que un atacante pueda modificar el estado del sistema o inyectar código malicioso.

Además, el uso de funciones puras y la separación de la lógica del estado pueden ayudar a prevenir errores comunes de seguridad, como los errores de acceso a datos o la modificación accidental del estado. Al mantener el estado en piezas inmutables separadas, se puede aplicar seguridad a nivel de datos y controlar el acceso a ellos.

La arquitectura inmutable también puede ser útil en sistemas de blockchain, donde la inmutabilidad es un requisito crítico. En sistemas de blockchain, cada transacción se registra en bloques de datos inmutables que no pueden ser modificados después de ser creados. El enfoque de arquitectura inmutable se alinea bien con los requisitos de la tecnología blockchain, lo que lo hace una opción popular para su implementación.

También es destacable su capacidad para mejorar la capacidad de recuperación de los sistemas. Debido a que las piezas inmutables no pueden ser modificadas después de su creación, los errores no pueden propagarse a través del sistema y los problemas en una parte del sistema no afectan a otras partes.

Esto significa que, en el caso de una falla en el sistema, se puede recuperar fácilmente a un estado anterior al problema, simplemente cargando una versión anterior de las piezas inmutables. Además, como las piezas inmutables son más fáciles de probar y verificar, es menos probable que se produzcan errores en el sistema en primer lugar.

Otra cosa interesante de la arquitectura inmutable es su impacto en el rendimiento del sistema. Debido a que las piezas inmutables son inmutables, pueden ser almacenadas en caché de manera segura, lo que puede mejorar significativamente el rendimiento. Al tener menos cambios de estado en el sistema, también se reduce la cantidad de trabajo que debe realizarse para mantener y actualizar el estado del sistema, lo que también puede mejorar el rendimiento.

Por último, la arquitectura inmutable también puede simplificar la implementación de ciertos patrones de diseño comunes, como el patrón Observador o el patrón Estrategia. En un sistema inmutable, los objetos pueden ser pasados como argumentos a otras funciones sin preocuparse por los efectos secundarios, lo que hace que sea más fácil implementar estos patrones de diseño.

Bibliografía

- https://www.se-radio.net/2021/02/episode-447-michael-perry-on-immutable-architecture/
- https://galvarado.com.mx/post/beneficios-retos-y-como-lograr-infraestructura-inmutable-con-packer-ansible-y-terraform/