Organizador de Tareas

ARQUITECTURA IMPLEMENTADA

UNIVERSIDAD MARIANO GALVEZ DE GUATEMALA | SAN JUAN SACATEPEQUEZ

Proyecto Final

Análisis de Sistemas II

Gestor de tareas

Dennis Iván Santizo Pocon 7590-19-2524

Moises Gabriel Coc Estrada 7590-14-6250

Oscar Giovanni Culajay Tocay 7590-18-381

Juan Pablo Sagastume 7590-19-23736

Universidad Mariano Gálvez.

San Juan Sacatepéquez

Análisis de Sistemas II

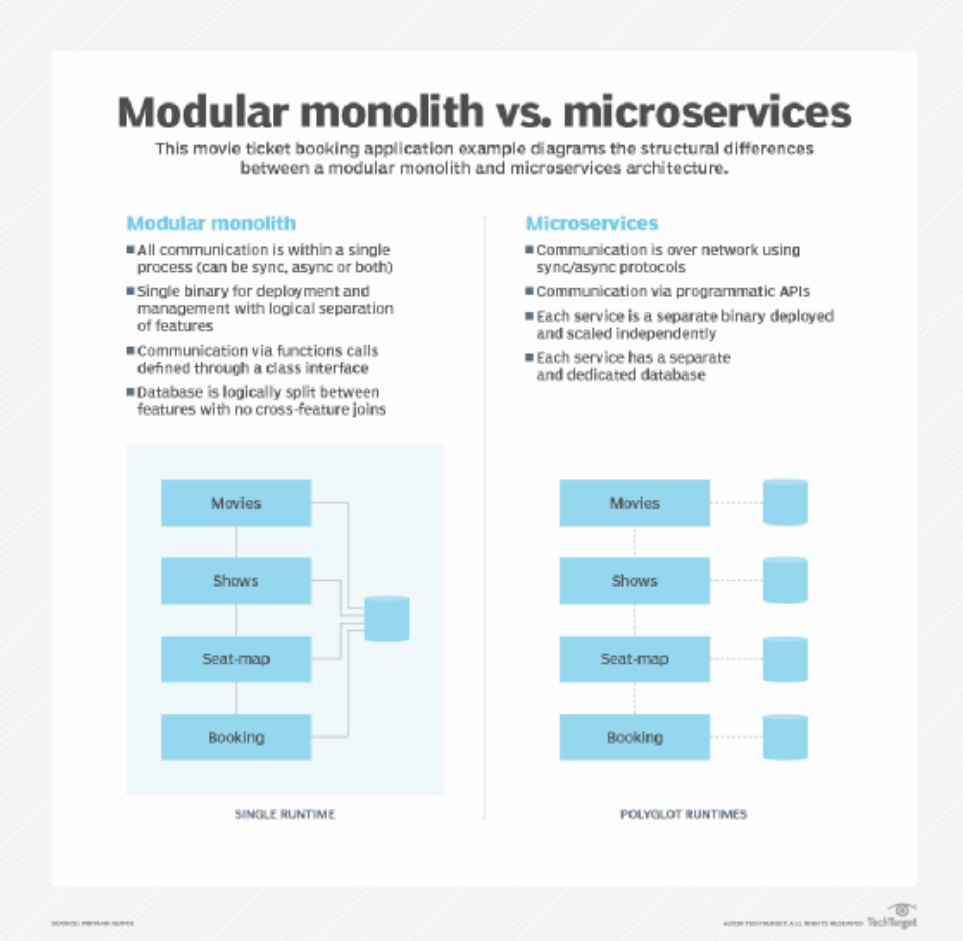
A. Arquitectura implementada en la aplicación.

Arquitectura Modular monolito

Las arquitecturas monolíticas convencionales se centran en colocar el código en capas horizontalmente a través de los límites y dependencias funcionales, lo que inhibe su capacidad para separarse en componentes funcionales. El monolito modular revisa esta estructura y la configura para combinar la simplicidad de la comunicación de un solo proceso con la libertad de la creación de componentes.

A diferencia del monolito tradicional, los monolitos modulares intentan establecer un contexto limitado segmentando el código en módulos de características individuales. Cada módulo expone una definición de interfaz de programación a otros módulos. La definición alterada puede provocar que sus dependencias cambien a su vez.

Gran parte de esto se basa en definiciones de interfaz estables. Sin embargo, al limitar las dependencias y aislar el almacén de datos, la arquitectura establece límites dentro del monolito que se asemejan a la alta cohesión y el bajo acoplamiento que se encuentran en una arquitectura de microservicios. Los equipos de desarrollo pueden comenzar a analizar la funcionalidad, pero pueden hacerlo sin preocuparse por el bagaje de administración vinculado a múltiples tiempos de ejecución y comunicación asincrónica.



Un beneficio del monolito modular es que la encapsulación lógica permite una alta reutilización, mientras que los datos permanecen consistentes y los patrones de comunicación simples. Es más fácil administrar un monolito modular que decenas o cientos de microservicios, lo que mantiene bajos los costos operativos y la complejidad de la infraestructura subyacente.

Sin embargo, los equipos de desarrollo deben comprender que los monolitos modulares no brindan todos los beneficios de los microservicios, particularmente cuando se trata de diversificar la tecnología y las opciones de idioma. Dado que las aplicaciones necesitan ejecutar código dentro de un solo tiempo de ejecución, existe una oportunidad limitada para mezclar esos entornos de tiempo de ejecución.

Estos tipos de pilas de tecnología políglota son útiles cuando las tecnologías heredadas crean un cuello de botella en el rendimiento o si los desarrolladores desean tener la capacidad de trabajar y experimentar con el lenguaje de su elección. En casos como estos, el monolito modular probablemente no cumplirá sus objetivos de arquitectura.

¿Cuál es la estructura de código de un monolito modular?

El diseño y la disposición del código correcto son fundamentales para el monolito modular. A continuación, la Figura 2 muestra el código detrás de una aplicación de reserva de entradas de cine que sigue el modelo modular. El código se ha estructurado con múltiples módulos de funciones y cada uno tiene una interfaz que representa su definición pública. SeatMapService actúa como interfaz para la función SeatMap. Esta interfaz es implementada por una clase bajo el paquete de servicio llamado SeatMapServiceImpl.

El módulo se presenta a aplicaciones y servicios externos como una API, utilizando protocolos como REST HTTP o GRPC. Un proxy inverso, una puerta de enlace API u otro tipo de agente de mensajería puede administrar estas llamadas API y hacerlo entre varios módulos simultáneamente.

Los servicios y aplicaciones internos interactúan con el módulo a través de un componente de interfaz abstraído. Esta interfaz permite que las aplicaciones solicitantes recopilen la información que necesitan. Sin embargo, no pueden acceder a la implementación real y deben depender únicamente de las abstracciones. Esto mantiene una separación adecuada de preocupaciones sin comprometer los procesos de solicitud.

¿Cuándo tienen sentido los monolitos modulares?

Las organizaciones ni siquiera deberían considerar los microservicios a menos que tengan un sistema que sea demasiado complejo para administrar como un monolito, según Kent Beck, creador de desarrollo impulsado por pruebas y programación extrema. Las decisiones de arquitectura siempre deben girar en torno al estado actual de su entorno, y la estrategia de monolito modular no es una excepción.

Hay algunos escenarios que se adaptan naturalmente a los monolitos modulares, como:

- Una implementación totalmente nueva, como una nueva plataforma de préstamos entre pares en las primeras etapas de crecimiento para convertirse en un conjunto completo de servicios financieros.
- Un sistema con una escala baja a moderada, como una plataforma de reserva de entradas de cine que atiende a varios miles de usuarios y gestiona unos pocos millones de transacciones por semana.
- Plataformas de software empresarial no complejas, como una plataforma de gestión y sincronización de notas / documentos para aplicaciones de consumo.
- Aplicaciones heredadas de gran tamaño, como un monolito de plataforma bancaria a gran escala existente que necesita dividirse en microservicios, pero no está listo para dividirse en servicios completamente independientes.

Pero, como la mayoría de los patrones de arquitectura, también hay escenarios en los que el monolito modular puede producir rendimientos reducidos. Éstas incluyen:

Cuando partes de la plataforma están siendo construidas y administradas por múltiples equipos independientes, especialmente si cada equipo gobierna su elección de tecnología subyacente y tiempos de ejecución;

Nuevas plataformas a gran escala creadas en entornos maduros donde se encuentran disponibles plantillas de servicio o chasis de servicio, que reducen significativamente el costo de entrada de los microservicios; y

Cuando la escala y el volumen de negocios ya son consistentemente altos, el personal de TI tiene una comprensión sólida de la arquitectura y tiene sentido sumergirse directamente en los microservicios.

Refuerce la coherencia a través de herramientas

En los sistemas monolíticos modulares, ayuda a mantener la distribución del código, los métodos de acceso y las dependencias asociadas con cada módulo de manera relativamente consistente. Por lo general, esto se logra mediante la creación de convenciones de orientación en torno a los procesos de tiempo de compilación y tiempo de compilación.

Hay muchas herramientas disponibles para ayudar a construir estas barandillas requeridas. SonarQube proporciona resaltado de análisis de código estático y evita la proliferación de dependencias cruzadas y la complejidad del código a lo largo del tiempo. Herramientas como Checkstyle, JDepend y CodeAssert de GitHub están diseñadas para imponer un diseño y una estructura de acceso en el momento de la compilación utilizando reglas y configuraciones definidas.