Comunicación asincrónica basada en mensajes

28/09/2021 • Tiempo de lectura: 4 minutos •  

¿Le ha resultado útil esta página?

En este artículo

La mensajería asincrónica y la comunicación controlada por eventos son fundamentales para propagar cambios entre varios microservicios y sus modelos de dominio relacionados. Como se mencionó anteriormente en la descripción de los microservicios y los contextos delimitados (BC), los modelos (de usuario, cliente, producto, cuenta, etc.) pueden tener diferentes significados para distintos microservicios o BC. Esto significa que, cuando se producen cambios, se necesita alguna manera de conciliarlos entre los diferentes modelos. Una solución es la coherencia final y la comunicación controlada por eventos basada en la mensajería asincrónica.

Cuando se usa la mensajería, los procesos se comunican mediante el intercambio de mensajes de forma asincrónica. Un cliente ejecuta una orden o una solicitud a un servicio mediante el envío de un mensaje. Si el servicio tiene que responder, envía un mensaje diferente al cliente. Como se trata de una comunicación basada en mensajes, el cliente asume que la respuesta no se recibirá inmediatamente y que es posible que no haya ninguna respuesta.

Un mensaje está compuesto por un encabezado (metadatos como información de identificación o seguridad) y un cuerpo. Normalmente, los mensajes se envían a través de protocolos asincrónicos como AMQP.

La infraestructura preferida para este tipo de comunicación en la comunidad de microservicios es un agente de mensajes ligero, que es diferente a los agentes grandes y orquestadores que se usan en SOA. En un agente de mensajes ligero, la infraestructura suele ser "simple" y solo actúa como un agente de mensajes, con implementaciones sencillas como RabbitMQ o un Service Bus escalable en la nube como Azure Service Bus. En este escenario, la mayoría de las ideas "inteligentes" siguen

existiendo en los puntos de conexión que generan y consumen mensajes, es decir, en los microservicios.

Otra regla que debe intentar seguir, tanto como sea posible, es usar la mensajería asincrónica solo entre los servicios internos y la comunicación sincrónica (como HTTP) solo desde las aplicaciones cliente a los servicios front-end (puertas de enlace de API y el primer nivel de microservicios).

Hay dos tipos de comunicación de mensajería asincrónica: la comunicación basada en mensajes de receptor único y la comunicación basada en mensajes de varios receptores. En las siguientes secciones se proporcionan detalles sobre los dos tipos.

Comunicación basada en mensajes de receptor único

La comunicación asincrónica basada en mensajes con un receptor único significa que hay una comunicación punto a punto que entrega un mensaje a exactamente uno de los consumidores que está leyendo en el canal, y que el mensaje solo se procesa una vez. Pero hay situaciones especiales. Por ejemplo, en un sistema de nube que intenta recuperarse automáticamente de los errores, el mismo mensaje se podría enviar varias veces. Debido a problemas de red o de otro tipo, el cliente tiene que poder volver a intentar el envío de los mensajes y el servidor tiene que implementar una operación que sea idempotente para procesar un mensaje concreto una sola vez.

La comunicación basada en mensajes de receptor único es especialmente idónea para enviar comandos asincrónicos de un microservicio a otro, como se muestra en la figura 4-18.

Una vez que se inicia el envío mediante la comunicación basada en mensajes (ya sea a través de comandos o eventos), no se debe mezclar con la comunicación sincrónica de HTTP.

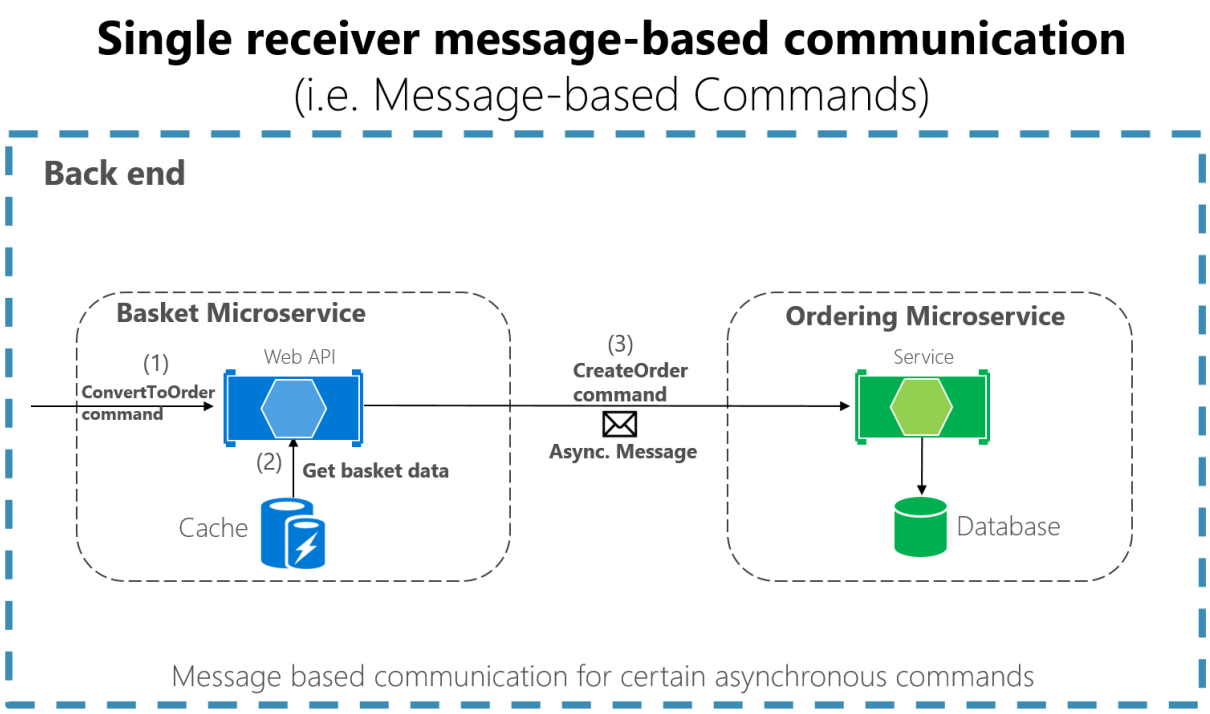

Figura 4-18. Un único microservicio en el que se recibe un mensaje asincrónico

Cuando los comandos proceden de aplicaciones cliente, se pueden implementar como comandos sincrónicos de HTTP. Use comandos basados en mensajes cuando necesite una mayor escalabilidad o cuando ya se encuentre en un proceso empresarial basado en mensajes.

Comunicación basada en mensajes de varios receptores

Como un enfoque más flexible, es posible que también le interese usar un mecanismo de publicación y suscripción para que la comunicación desde el remitente esté disponible para microservicios de suscriptor adicionales o para aplicaciones externas. Por tanto, le ayudará seguir el principio de abierto y cerrado en el servicio de envío. De este modo, se pueden agregar suscriptores adicionales en el futuro sin necesidad de modificar el servicio del remitente.

Cuando se usa una comunicación de publicación y suscripción, es posible que se use una interfaz de bus de eventos para publicar eventos en cualquier suscriptor.

Comunicación asincrónica controlada por eventos

Cuando se usa la comunicación asincrónica controlada por eventos, un microservicio publica un evento de integración cuando sucede algo dentro de su dominio y otro

microservicio debe ser consciente de ello, como por ejemplo un cambio de precio en un microservicio del catálogo de productos. Los microservicios adicionales se suscriben a los eventos para poder recibirlos de forma asincrónica. Cuando esto sucede, es posible que los receptores actualicen sus propias entidades de dominio, lo que puede provocar la publicación de más eventos de integración. Este sistema de publicación/suscripción se realiza mediante una implementación de un bus de eventos. El bus de eventos se puede diseñar como una abstracción o interfaz, con la API que se necesita para suscribirse o cancelar la suscripción a los eventos y para publicarlos. El bus de eventos también puede tener una o más implementaciones basadas en cualquier agente entre procesos y de mensajería, como una cola de mensajes o un Service Bus que admita la comunicación asincrónica y un modelo de publicación y suscripción.

Si un sistema usa la coherencia final controlada por eventos de integración, se recomienda que este enfoque sea transparente para el usuario final. El sistema no debe usar un enfoque que imite a los eventos de integración, como SignalR o sistemas de sondeo desde el cliente. El usuario final y el propietario de la empresa tienen que adoptar explícitamente la coherencia final en el sistema y saber que, en muchos casos, la empresa no tiene ningún problema con este enfoque, siempre que sea explícito. Este enfoque es importante porque los usuarios pueden esperar ver algunos resultados inmediatamente y es posible que esto no pase con la coherencia final.

Como se indicó anteriormente en la sección Desafíos y soluciones para la administración de datos distribuidos, se pueden usar eventos de integración para implementar tareas de negocio que abarquen varios microservicios. Por tanto, tendrá coherencia final entre dichos servicios. Una transacción con coherencia final se compone de una colección de acciones distribuidas. En cada acción, el microservicio relacionado actualiza una entidad de dominio y publica otro evento de integración que genera la siguiente acción dentro de la misma tarea empresarial descentralizada.

Un punto importante es que es posible que le interese comunicarse con varios microservicios que estén suscritos al mismo evento. Para ello, puede usar la mensajería de publicación y suscripción basada en la comunicación controlada por eventos, como se muestra en la figura 4-19. Este mecanismo de publicación y suscripción no es exclusivo de la arquitectura de microservicios. Es similar a la forma en que deben comunicarse los contextos delimitados en DDD o a la forma en que se propagan las actualizaciones desde la base de datos de escritura a la de lectura en el modelo de arquitectura Command and Query Responsibility Segregation (CQRS) (Segregación de responsabilidades de comandos y consultas). El objetivo es tener coherencia final entre varios orígenes de datos en el sistema distribuido.

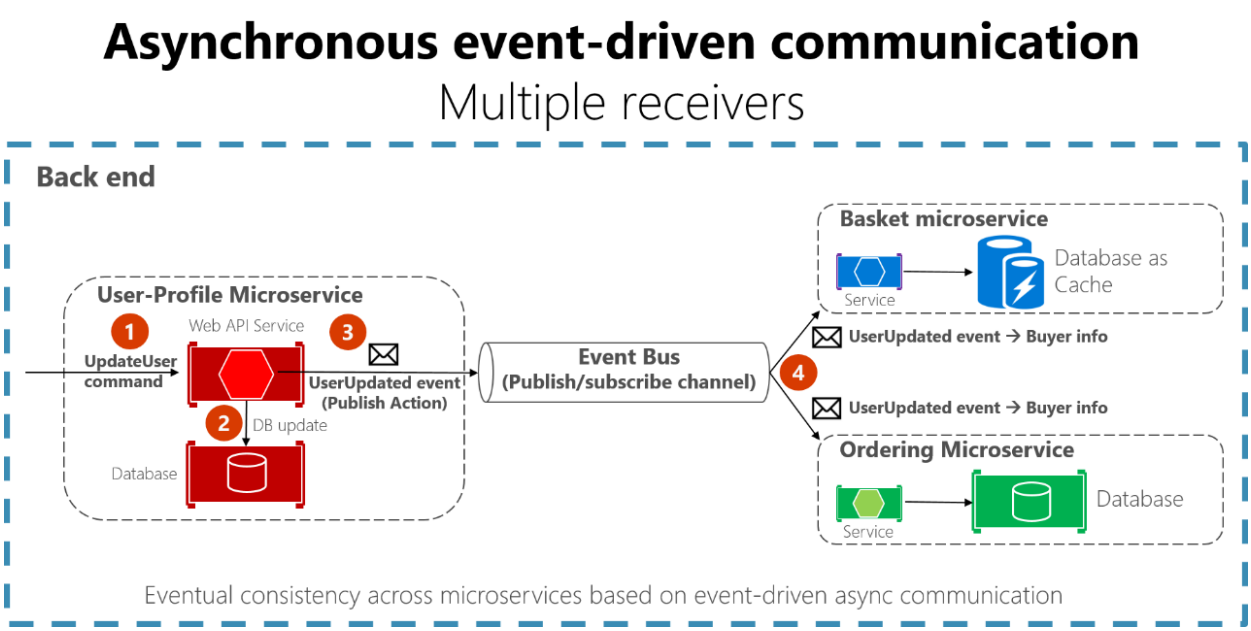

Figura 4-19. Comunicación asincrónica de mensajes controlada por eventos

En la comunicación controlada por eventos asincrónica, un microservicio publica los eventos en un bus de eventos y muchos microservicios pueden suscribirse a él para recibir una notificación y actuar en consecuencia. La implementación determinará qué protocolo se va a usar para las comunicaciones basadas en mensajes y controladas por eventos. AMQP puede ayudar a lograr una comunicación en cola confiable.

Al usar un bus de eventos, es posible que le interese usar una capa de abstracción (como una interfaz de bus de eventos) basada en una implementación relacionada en las clases con código que use la API de un agente de mensajes como RabbitMQ o un Service Bus como Azure Service Bus con Topics. Como alternativa, es posible que le interese usar un Service Bus de nivel superior como NServiceBus, MassTransit o Brighter para articular el bus de eventos y el sistema de publicación y suscripción.

Una nota sobre las tecnologías de mensajería destinadas a sistemas de producción

Las tecnologías de mensajería disponibles para implementar el bus de eventos abstractos se encuentran en distintos niveles. Por ejemplo, productos como RabbitMQ (un transporte de agente de mensajería) y Azure Service Bus se colocan en un nivel inferior a otros productos como NServiceBus, MassTransit o Brighter, que pueden trabajar sobre RabbitMQ y Azure Service Bus. La elección depende de la cantidad de características enriquecidas en el nivel de aplicación y de la escalabilidad de serie que necesite para la aplicación. Para implementar solamente un bus de eventos de prueba de concepto para el entorno de desarrollo, como se ha hecho en el ejemplo

eShopOnContainers, una implementación sencilla sobre RabbitMQ que se ejecute en un contenedor de Docker podría ser suficiente.

Pero para sistemas decisivos y de producción que necesiten una gran escalabilidad, es posible que quiera probar Azure Service Bus. Para las abstracciones generales y las características que facilitan el desarrollo de aplicaciones distribuidas, se recomienda evaluar otros Service Bus comerciales y de código abierto, como NServiceBus, MassTransit y Brighter. Por supuesto, puede crear sus propias características de Service Bus sobre tecnologías de nivel inferior como RabbitMQ y Docker. Pero ese trabajo podría ser muy costoso para una aplicación empresarial personalizada.

Publicación de forma resistente en el bus de eventos

Un desafío al implementar una arquitectura controlada por eventos entre varios microservicios es cómo actualizar de manera atómica el estado en el microservicio original mientras se publica de forma resistente su evento de integración relacionado en el bus de eventos, en cierta medida en función de las transacciones. Las siguientes son algunas maneras de lograr esta funcionalidad, aunque podría haber enfoques adicionales.

- Uso de una cola transaccional (basada en DTC) como MSMQ. (Pero es un método heredado).
- Uso de la minería del registro de transacciones.
- Uso del patrón de orígenes de eventos completo.
- Uso del patrón de bandeja de salida : una tabla de base de datos transaccional como una cola de mensajes que será la base para un componente de creador de eventos que creará el evento y lo publicará.

Temas adicionales que se deben tener en cuenta al usar la comunicación asincrónica son la idempotencia y la desduplicación de los mensajes. Estos temas se describen en la sección Implementación de la comunicación basada en eventos entre microservicios (eventos de integración) más adelante en esta guía.

Recursos adicionales

- **Mensajería controlada por eventos**
  https://patterns.arcitura.com/soa-