Other Challenges

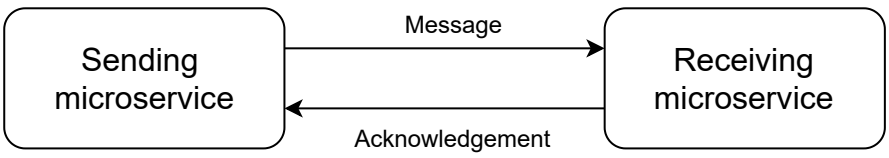
In this lesson, we'll continue to discuss the challenges that an asynchronous approach poses.

WE'LL COVER THE FOLLOWING

Guaranteed delivery

In an asynchronous system, the delivery of messages can be guaranteed if the system is appropriately implemented.

The sender has the messaging system confirm that it received the message. Afterwards, the messaging system has the recipient of the message acknowledge the receipt. However, if the recipient never picks up the data and thus prevents delivery, the sender has an acknowledgement, but the message still does not arrive at the recipient.



Guaranteed delivery

It is difficult to guarantee delivery when the recipient is anonymous. In this case, it is unclear who is supposed to receive the message and whether there are any recipients at all who should get the message. Therefore, it is also unclear who has to issue receipts.

Idempotency

If the messages are not acknowledged by the recipient, they are sent again. When the receiving microservice processes the message but is unable to acknowledge the message due to a problem or a failure, **the recipient receives the message a second time** although the recipient processed the message already.

This is an **at least once strategy**: the messages are sent at least once, and, in the described failure scenario, more often.

> Therefore, one tries to design distributed systems in such a way that the microservices are **idempotent**. This means that a message can be processed more than once, but the state of the service no longer changes.

For example, when creating an invoice:

- The *invoice* microservice can first check in its own database whether an invoice has already been created.
- In this manner, an invoice is created only the first time the message is received.
- If the message is transferred again, it will be ignored.

One recipient

In addition, it can be necessary that only **one instance of a microservice processes a message**.

- For example, it would be incorrect from a domain perspective when multiple instances of the *invoice* microservice receive the order and all of them generate an invoice.
- This would generate multiple invoices.

> For this, messaging systems normally have an option to send messages only to a single recipient. This recipient then has to confirm the message and process it. Such a communication type is called **point to point communication**.

Unfortunately, the rules for processing can be complex. When changes are made to customer data, parallel processing should be carried out as far as possible to ensure high performance.

However, changes to the data of a specific customer probably have to follow a sequence. For example, it would not be good if changes to the billing address are processed after the invoice has been written; the invoice would still contain the wrong address.

For this reason, it may be important to guarantee the order of messages.

Test

With asynchronous microservices, the **continuous delivery pipelines must be independent** to enable independent deployment.

To do this, **the testing of the microservices must be independent** of other microservices.

With asynchronous communication, a test can send a message to the microservice and check whether the system behaves as expected. **Timing can be difficult** because it is not clear when the microservice has processed the message and how long the test should wait for processing. The test can then check whether the microservice sends the correct messages in response.

This **allows very simple black box tests**, a test based on the interface without knowing about the internal structure of the microservice.

These tests do not place particularly high demands on the test environment. The environment just needs to be able to transmit messages. It is not necessary to install a large number of other microservices in the test environment. Instead, the messages that other microservices send or expect from the tested microservice can be the basis for the tests.

QUIZ

1

Consider the following scenario: Sending the same order details (identified by a unique order ID) multiple times to the shipping microservice results in **only one** instance of the order being shipped.

What concept is being demonstrated here?

COMPLETED 0%

1 of 3

In the next lesson, we'll discuss some advantages and variations of asynchronous communication.