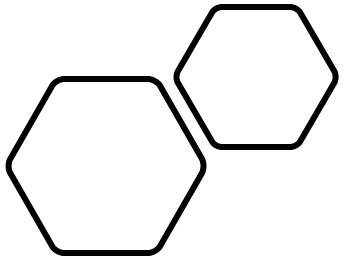
Arquitetura de Micros Serviços (P2)

Prof. Bernardo Copstein

Prof. Júlio Machado

Micros serviços: mecanismos de comunicação



Leituras recomendadas:

- Sommerville, Ian. Engineering Software Products – An Introduction to Modern Software Engineering. Pearson, 2019. Capítulo 6.
- Newman, Sam. Building Microservices. O'Reilly Media Inc, 2015.

Comunicação de micros serviços

- Os micros serviços se comunicam trocando mensagens.
- Uma mensagem enviada entre serviços inclui algumas informações administrativas, uma solicitação de serviço e os dados necessários para entregar o serviço solicitado.
- Os serviços retornam uma resposta às mensagens de solicitação de serviço.
  - Um serviço de autenticação pode enviar uma mensagem para um serviço de login que inclui o nome inserido pelo usuário.
  - A resposta pode ser um token associado a um nome de usuário válido ou pode ser um erro informando que não há usuário registrado.

Comunicação de microsserviços

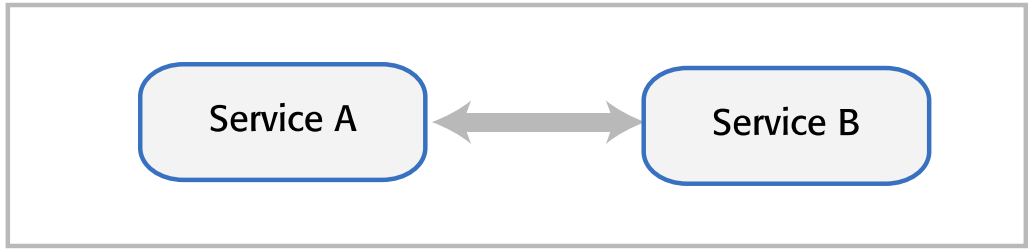
- Os serviços se comunicam trocando mensagens que incluem informações sobre o originador da mensagem, bem como os dados que são a entrada ou saída da solicitação.
- Ao projetar uma arquitetura de micros serviços, você precisa estabelecer um padrão de comunicação que todos os micros serviços devem seguir. Algumas das principais decisões que você precisa tomar são
  - A interação de serviço deve ser síncrona ou assíncrona?
  - Os serviços devem se comunicar diretamente ou por meio do middleware do agente de mensagens?
  - Qual protocolo deve ser usado para mensagens trocadas entre serviços?

Comunicação direta e indireta

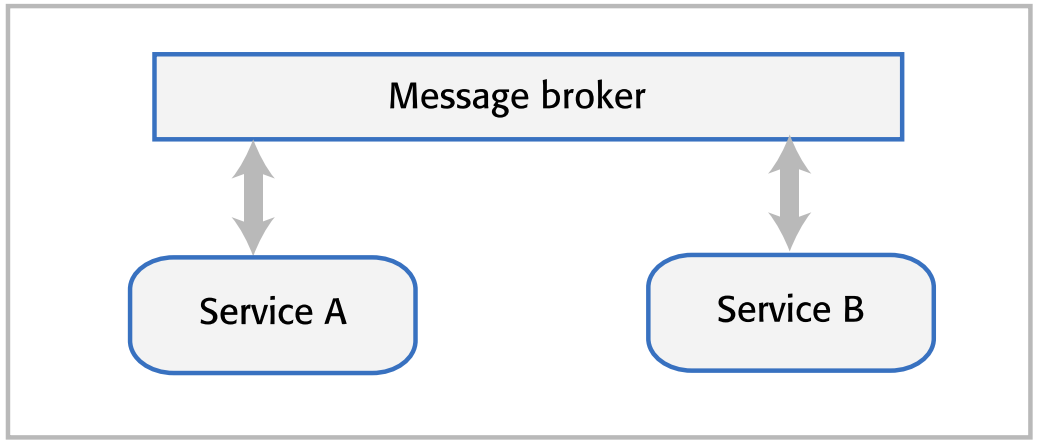
- A comunicação direta de serviço requer que os serviços que interagem conheçam o endereço um do outro.
- Os serviços interagem enviando solicitações diretamente para esses endereços.
- A comunicação indireta envolve nomear o serviço necessário e enviar essa solicitação para um agente de mensagens (às vezes chamado de barramento de mensagens).
- O agente de mensagens é então responsável por localizar o serviço que pode atender à solicitação de serviço.

Comunicação de serviço direta e indireta

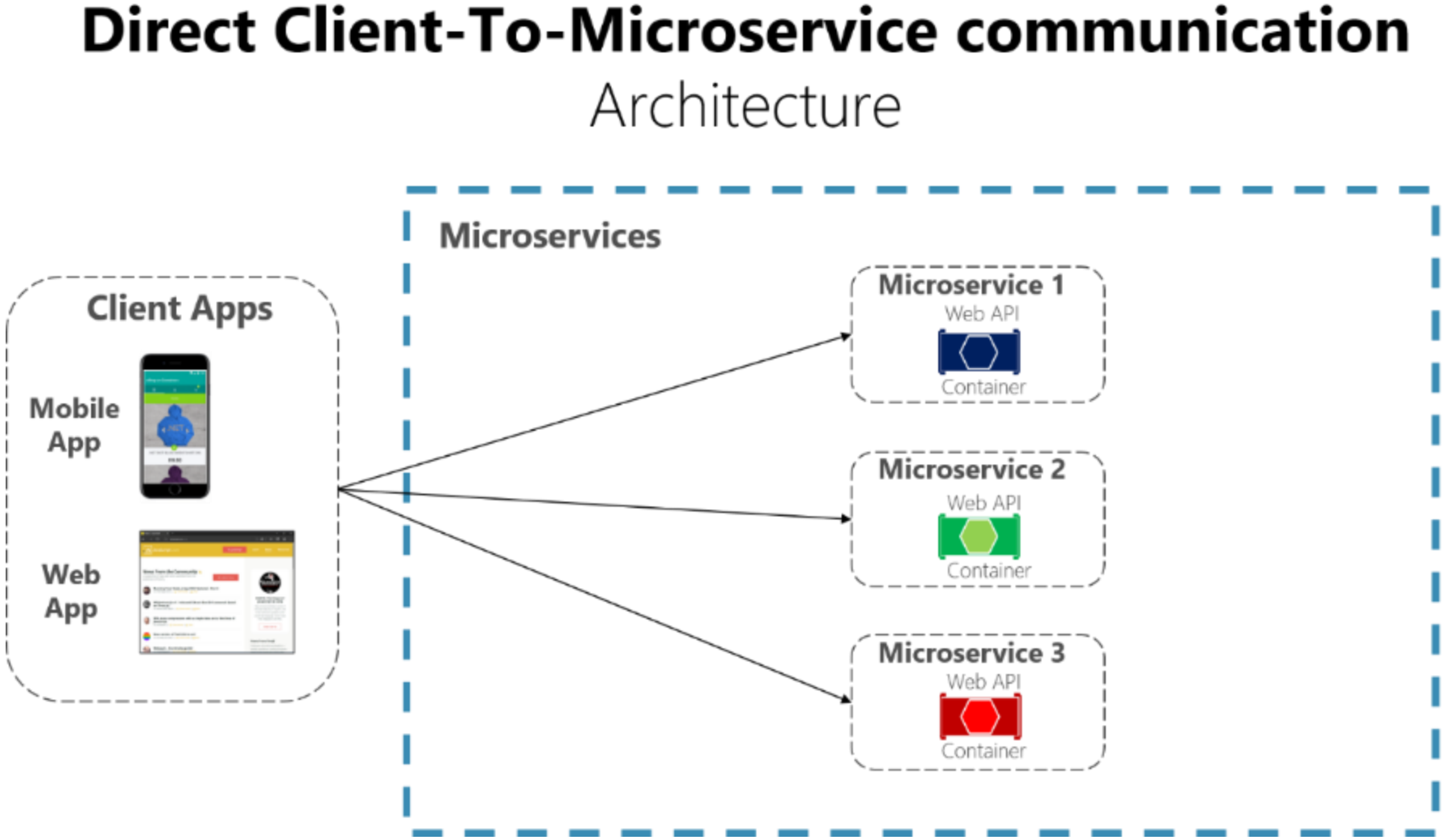
Direct communication - A and B send messages to each other



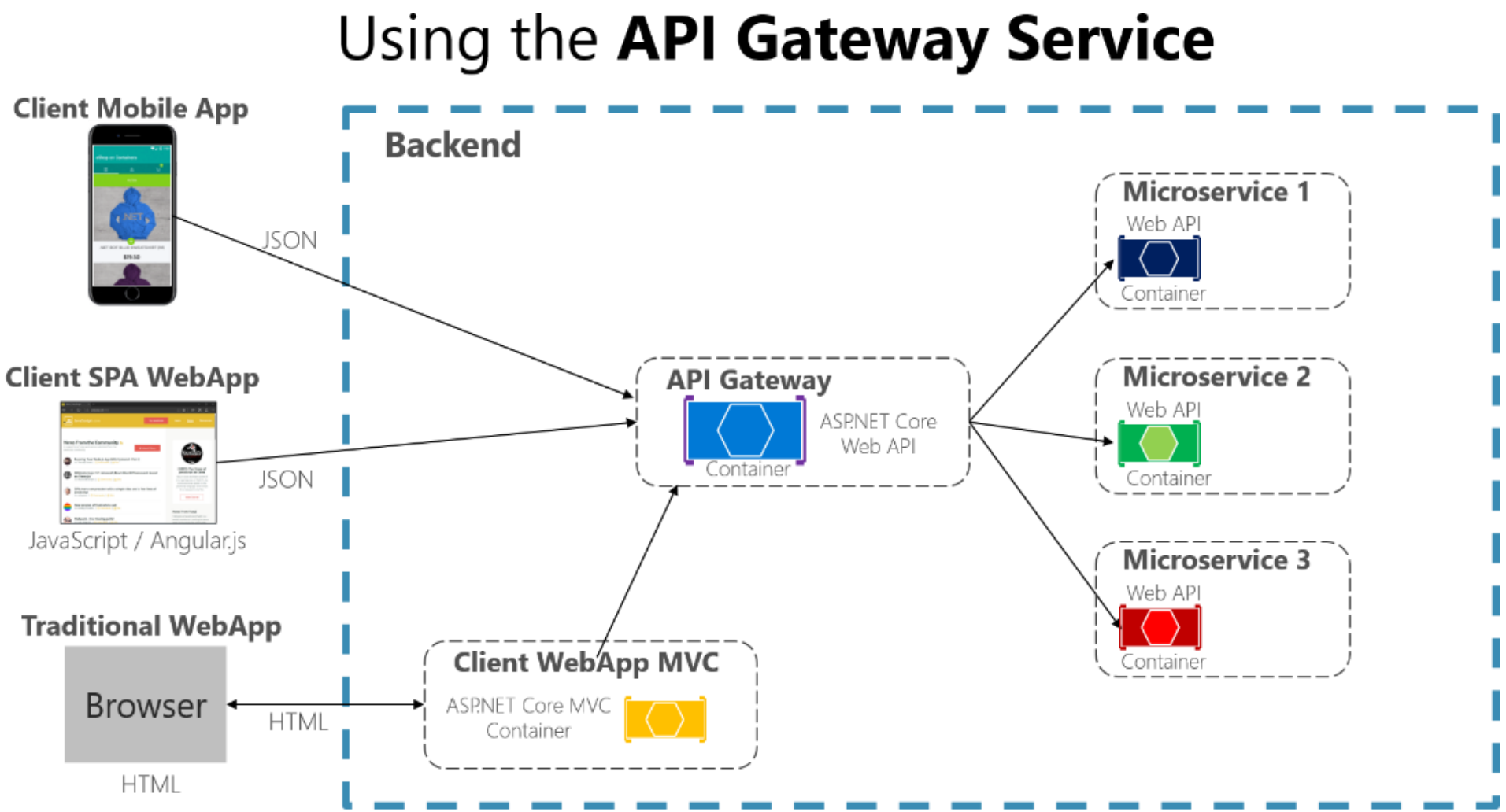
Indirect communication - A and B communicate through a message broker



Comunicação direta e indireta



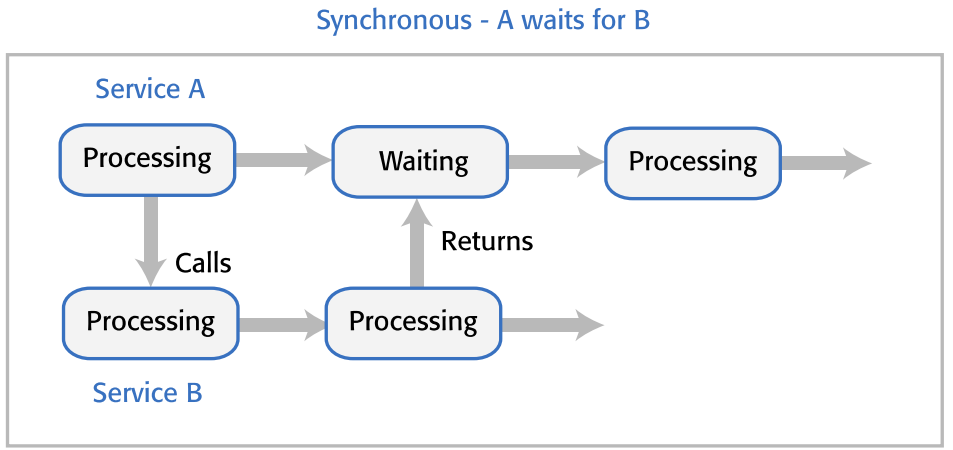
Comunicação direta e indireta

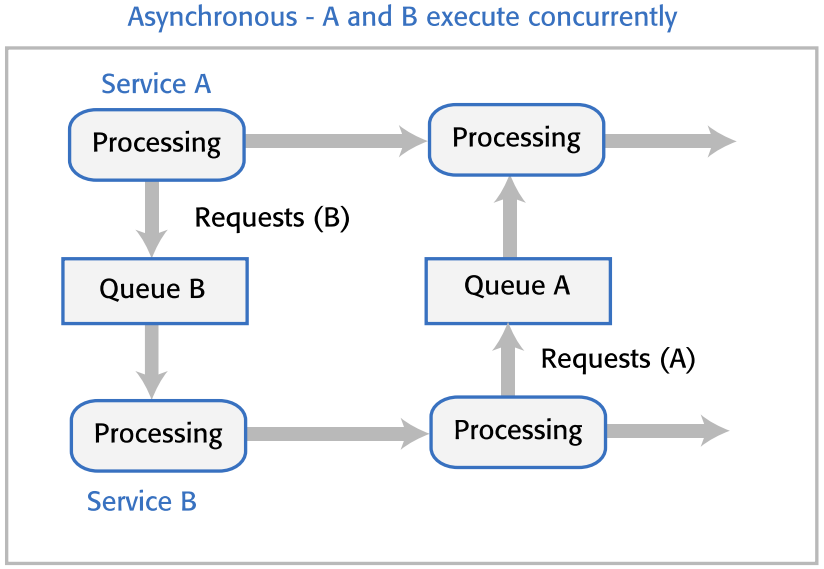


Interação síncrona e assíncrona

- Em uma interação síncrona, o serviço A emite uma solicitação ao serviço B. O serviço A então suspende o processamento enquanto B está processando a solicitação.
- Ele espera até que o serviço B retorne as informações necessárias antes de continuar a execução.
- Em uma interação assíncrona, o serviço A emite a solicitação que é enfileirada para processamento pelo serviço B. A então continua o processamento sem esperar que B termine seus cálculos.
- Algum tempo depois, o serviço B conclui a solicitação anterior do serviço A e enfileira o resultado a ser recuperado por A.
- O serviço A, portanto, precisa verificar sua fila periodicamente para ver se um resultado está disponível.

Interação de micros serviços síncrona e assíncrona



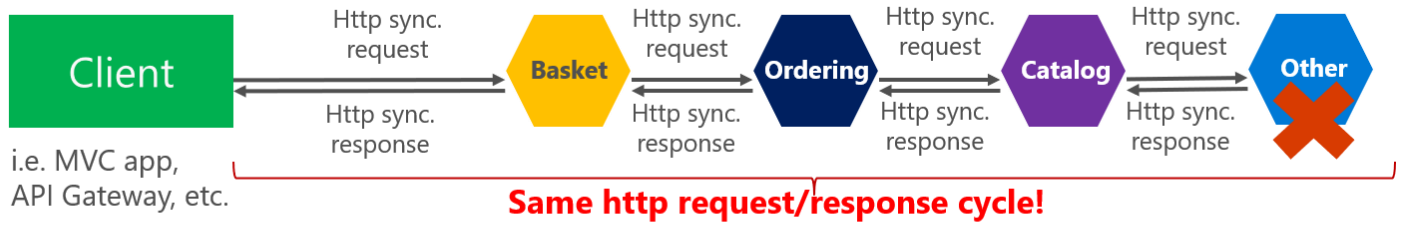


Interação síncrona e assíncrona

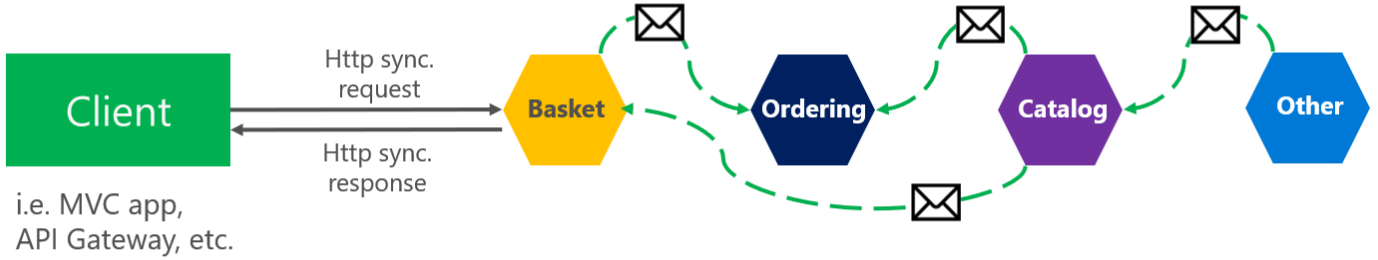
Synchronous vs. async communication across microservices

Anti-pattern

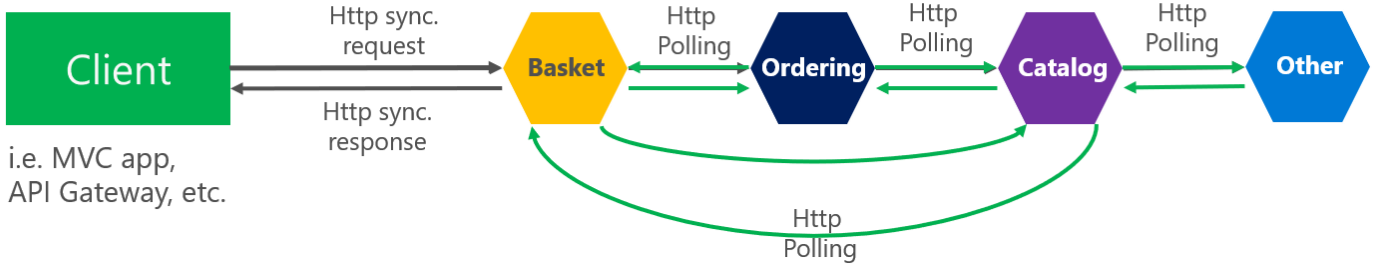
Synchronous
all req./resp. cycle



Asynchronous
Comm. across internal microservices (EventBus: i.e. **AMPQ**)



"Asynchronous"
Comm. across internal microservices (Polling: **Http**)

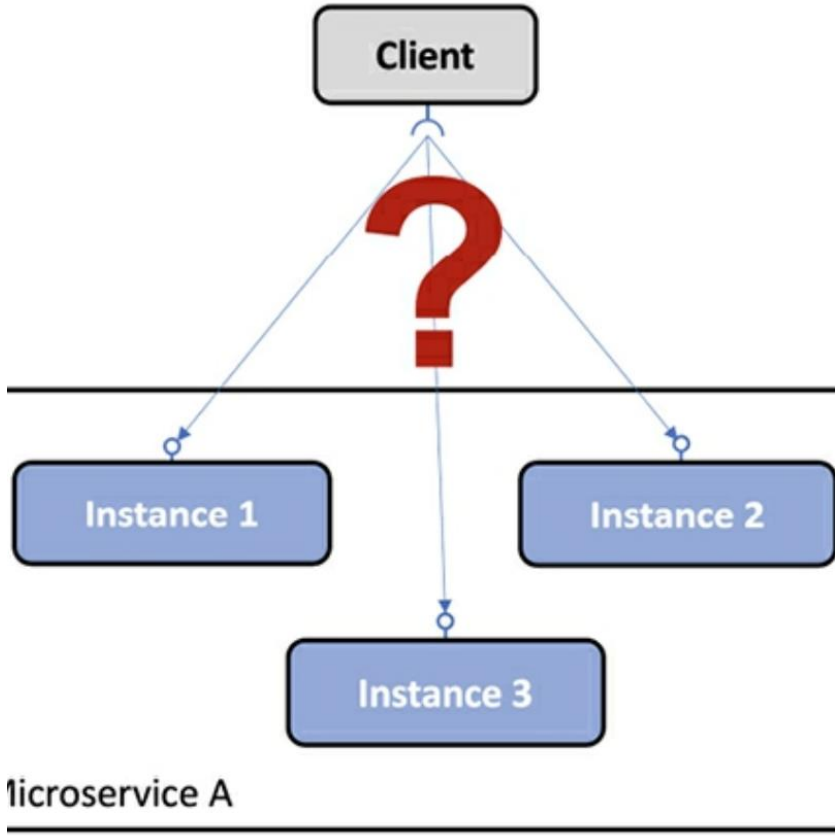


Padrões de projeto para microsserviços

- Alguns padrões de projeto são comumente usados para mitigar os principais problemas encontrados quando se desenvolve utilizando uma arquitetura de microsserviços. Entre estes podemos destacar:
  - Service Discover
  - Edge server
  - Reactive Microservices
  - Circuit breaker

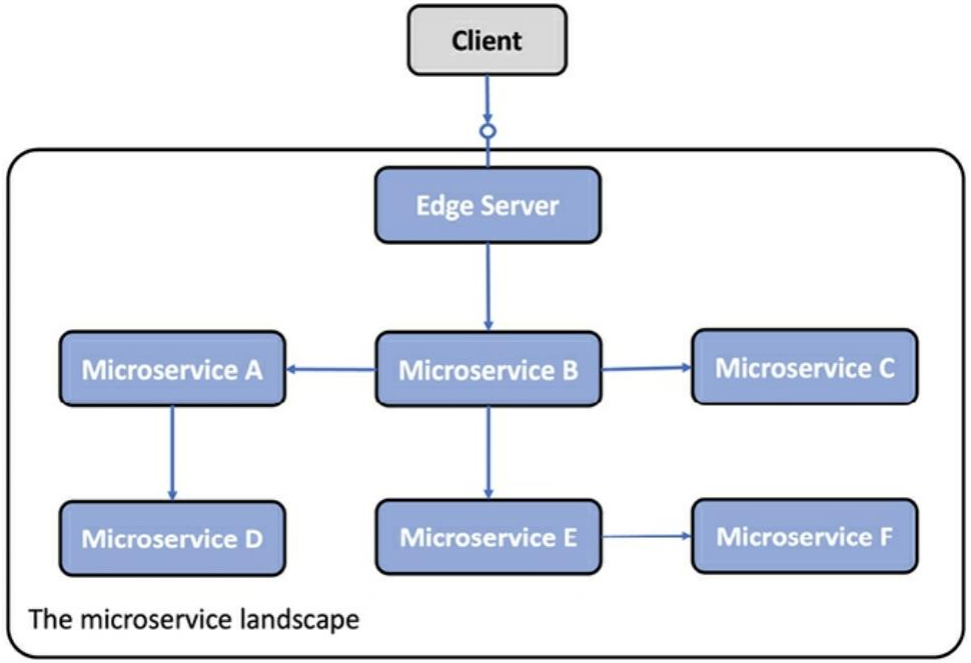
Padrão: service discover

- **Problema:** como os clientes encontram os microsserviços e suas instancias (microsserviços normalmente são associados a endereços IP dinamicamente alocados quando são iniciados)? **Solução:** Acrescente um novo componente – um serviço de descoberta – na aplicação. Que mantém o registro dos microsserviços disponíveis e os endereços IP de suas instâncias. Funcionalidades esperadas:
  - Capacidade de registar e cancelar o registro dos microsserviços e suas instancias conforme vem e vão.
  - Os clientes tem de poder fazer requisições para endpoints lógicos. As requisições devem ser roteadas para uma das instancias disponíveis.
  - As requisições para um microsserviço devem ser balanceadas entre as instancias disponíveis.
  - Deve ser possível detector as instancias que não estão saudáveis de maneira que as requisições não sejam encaminhadas para elas.



Padrão: edge server

- **Problema:** muitas vezes é interessante expor alguns microsserviços para consume externo enquanto outros devem permanecer “escondidos”.
- **Solução:** acrescentar um novo componente que irá receber todas as requisições externas. Deve ser integrado ao serviço de descoberta de maneira a prover balanceamento de carga. Deve atender aos seguintes requisitos:
  - Esconde os serviços internos que não devem ser expostos fora de seu contexto, isto é, deve rotear apenas para os microsserviços que estão configurados para receber solicitações externas.
  - Expõem os serviços externos protegendo-os de requisições maliciosas. Para isso usa protocolos padrao e as melhores práticas tais como OAuth, OIDC, JWT tokens e chaves de API para garantir o acesso Seguro.

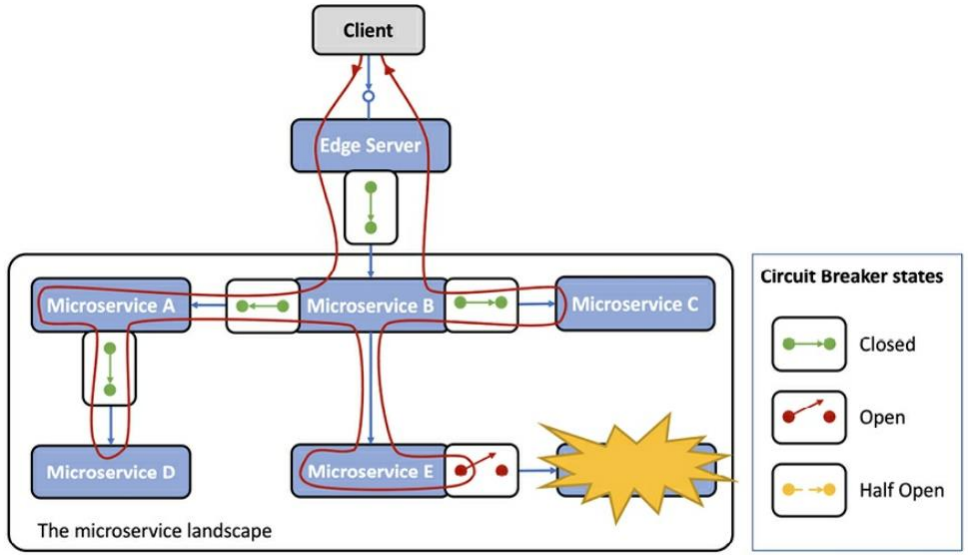


Padrão: reactive microsservices

- **Problema:** toda comunicação síncrona implica que uma thread fique alocada até que a comunicação seja atendida. Um número excessivo de comunicações síncronas pode levar ao sistema operacional não ter mais threads disponíveis.
- **Solução:** usar operações não bloqueantes para garantir que nenhuma thread fica alocada enquanto espera por processamento que ocorre no banco de dados ou em outros microsserviços.

Padrão: circuit breaker

- **Problema:** um sistema que usa comunicação síncrona entre os microsserviços fica exposto a uma cadeia de falhas. Se um microsserviço para de responder seus clientes podem ter problemas parando de responder também e assim sucessivamente.
- **Solução:** usar um circuit breaker para prevenir novas requisições de um cliente se for detectado problemas com o serviço que ele chama. A solução deve prover as seguintes funcionalidades:
  - Abrir o circuito e falhar rápido (sem timeout) se problemas com o serviço forem detectados. Monitorar quando o serviço é restabelecido.
  - Fechar o circuito quando for detectado que o serviço esta restabelecido fazendo com que o Sistema seja resiliente a este tipo de problema.



Aplicando os padrões

- Os roteiros 2, 3 e 4 mostram como aplicar alguns dos padrões apresentados aqui usando Spring-Boot:
  - Roteiro 2: name server → padrão service Discover
  - Roteiro 3: gateway → padrão edge server
  - Roteiro 4: circuit breaker → padrão circuit breaker