

O que iremos aprender nessa unidade?

Marco Mendes

Estilo Arquitetural de Microsserviços

Marco Mendes

Microsserviços

- Uma forma particular de projetar aplicações de software como **suítes de serviços implantáveis de forma independente**.
- O conceito nasce da indústria com o exemplo de modernização de legados em empresas como Netflix, Uber, Amazon, entre outras, a partir de 2009.
- O conceito de populariza com a escrita do artigo de Microsserviços de James Lewis e Martin Fowler em 2015
 - <https://martinfowler.com/articles/microservices.html>

Características

- Embora não exista uma definição única deste estilo arquitetural, existem certas características comuns tais como
 - a organização em torno da capacidade de negócios,
 - implantação automatizada
 - inteligência nos *endpoints* e
 - controle descentralizado de linguagens e de bases de dados.

Razões para uso

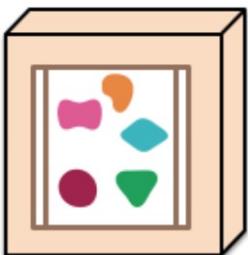
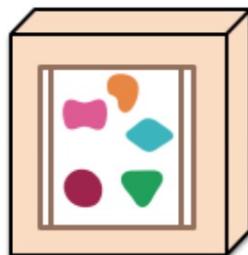
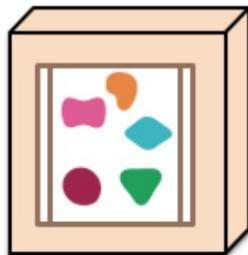
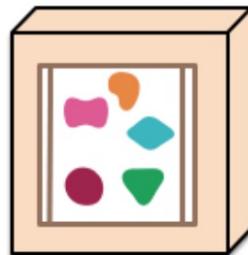
- Aplicações com topologia monolíticas com alto custo para alterar e implantar em produção.
- Servidores Web e de aplicação pesados
- Longos ciclos de mudança
- Dificuldades de implantação
- Custo de evolução do legado
- Barramentos de serviços que falharam em suas promessas

Aplicações Monolíticas

Uma aplicação monolítica coloca toda sua funcionalidade em um único processo...



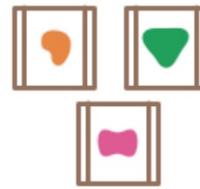
... e escala replicando a aplicação monolítica em vários servidores



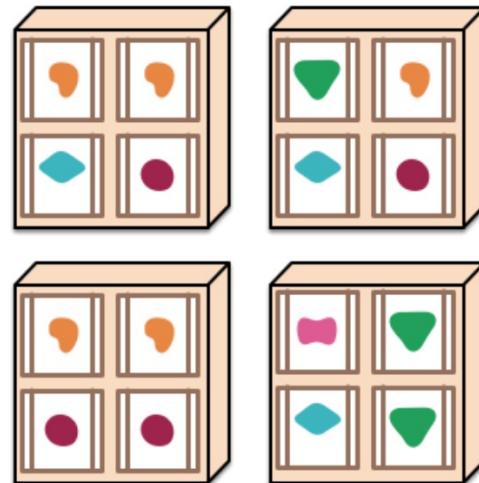
Fonte: Microservices Architecture, James Lewis e Martin Fowler, 2015

Aplicações de Microsserviços

Uma arquitetura em microsserviços põe cada elemento de uma funcionalidade em um serviço separado ...



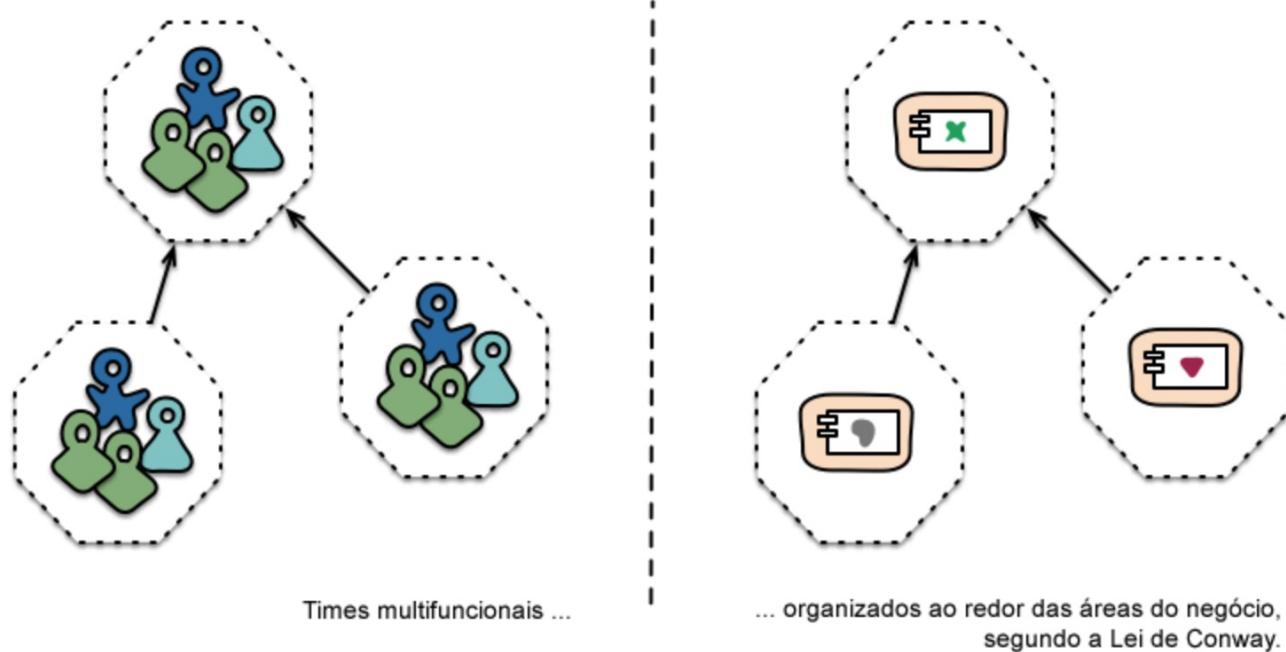
... e escala distribuindo estes serviços entre os servidores, replicando quando necessário.



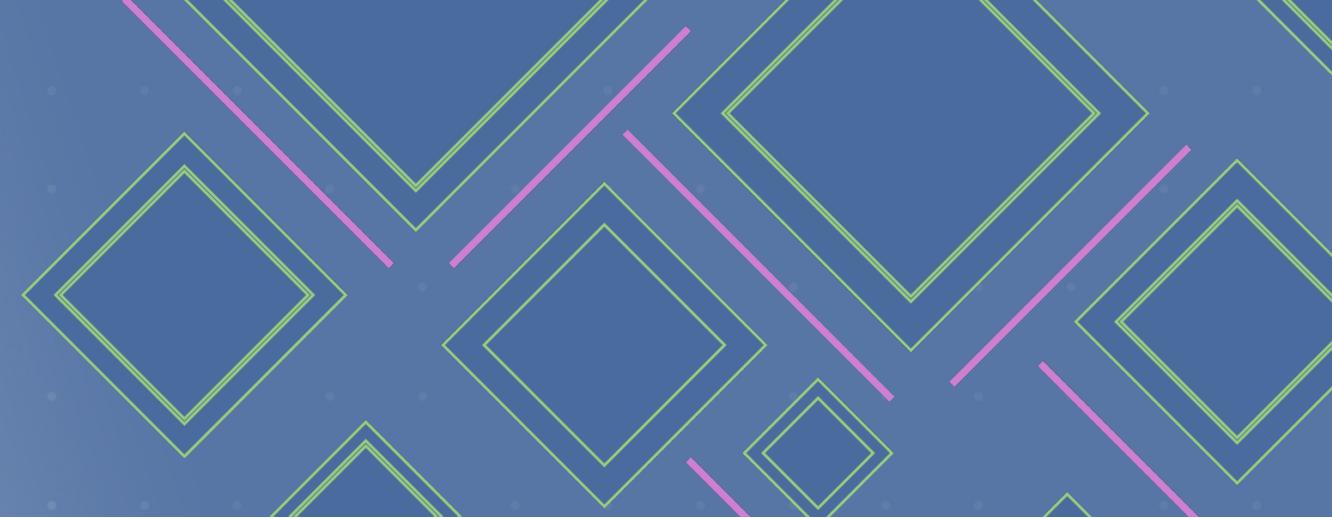
- Arquitetura distribuída
- Os serviços são implantados e escalam de forma independente.
- Cada serviço também provê uma fronteira bem definida entre os módulos, permitindo até mesmo que diferentes serviços sejam escritos em diferentes linguagens de programação. Eles podem inclusive serem administrados por times diferentes.

Fonte: Microservices Architecture, James Lewis e Martin Fowler, 2015

Aplicações de Microserviços e a Lei de Conway

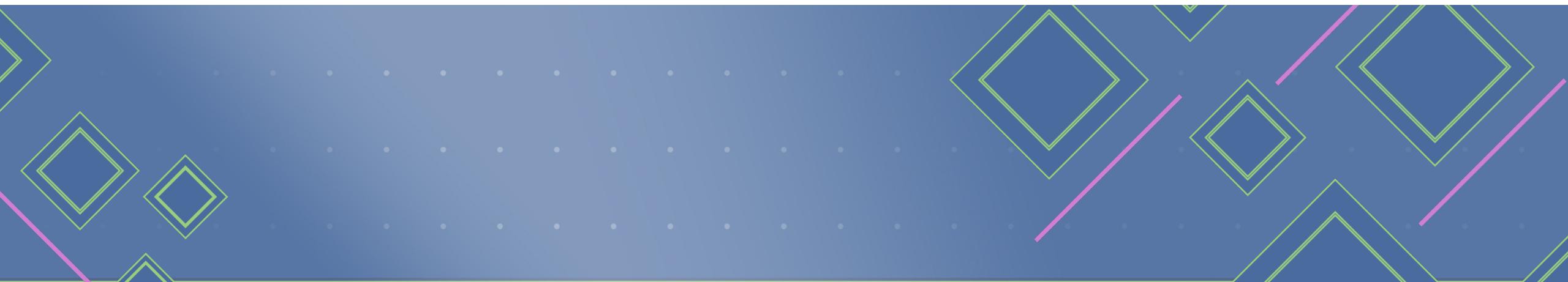


Fonte: Microservices Architecture, James Lewis e Martin Fowler, 2015

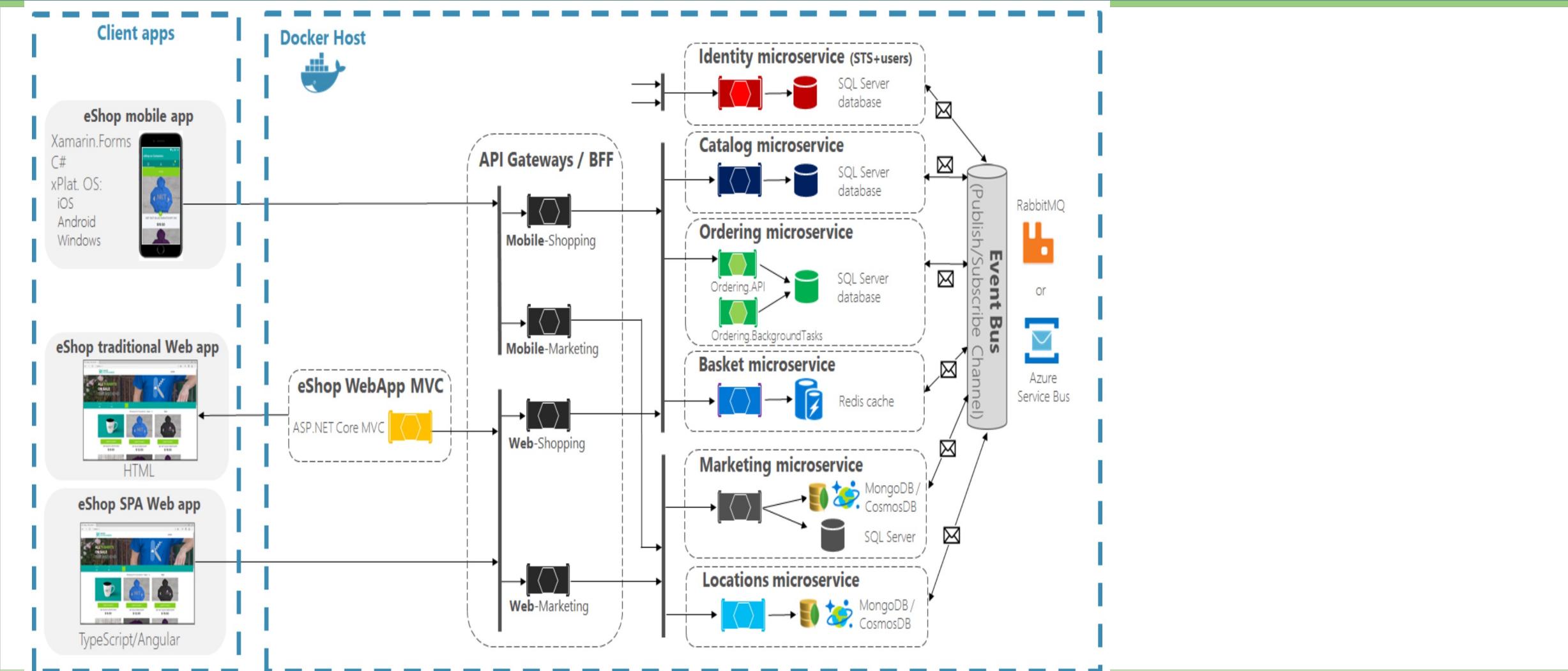


Microsserviços - Exemplo

Marco Mendes



Fonte: <https://github.com/dotnet-architecture/eShopOnContainers>



Microserviços - Características

Marco Mendes

Implantação como Componentes

- Se você tem uma aplicação que consiste em diversas bibliotecas em um único processo, uma mudança em qualquer componente resulta em ter que republicar toda sua aplicação.
- Uma das principais razões para usar serviços como componentes (ao invés de bibliotecas) é que serviços são publicados de maneira independente.

Implantação como Componentes

- Mas se esta aplicação é dividida em múltiplos serviços, você pode esperar que diversas mudanças em um único serviço exijam uma republicação somente no serviço alterado.
- Isso não é algo absoluto, pois algumas mudanças criam alterações nas interfaces entre os serviços, mas o dever de uma boa arquitetura em microsserviços é minimizar este impacto, criando interfaces coesas e mecanismos para evolução entre os serviços.

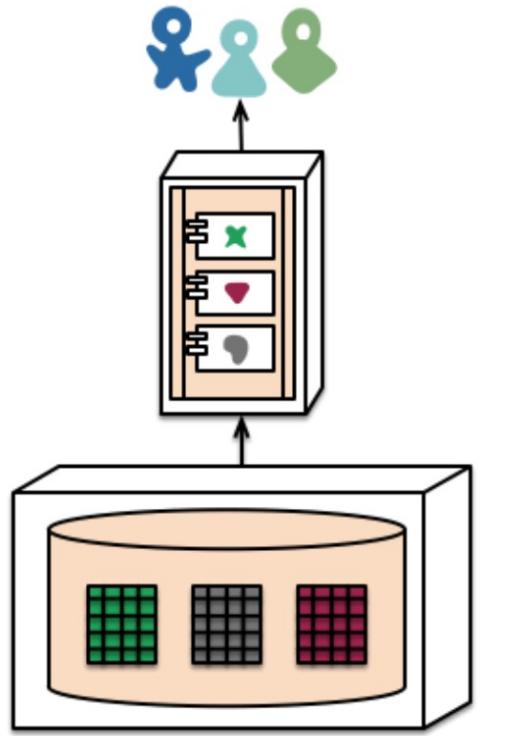
Implantação como Componentes

- Uma das principais razões para usar serviços como componentes (ao invés de bibliotecas) é que serviços são publicados de maneira independente.
- Um componente é uma unidade de software que é substituída ou atualizada de maneira independente.
- Na prática, tecnologias como o Docker e o Kubernetes permitem esse nível de componentização.

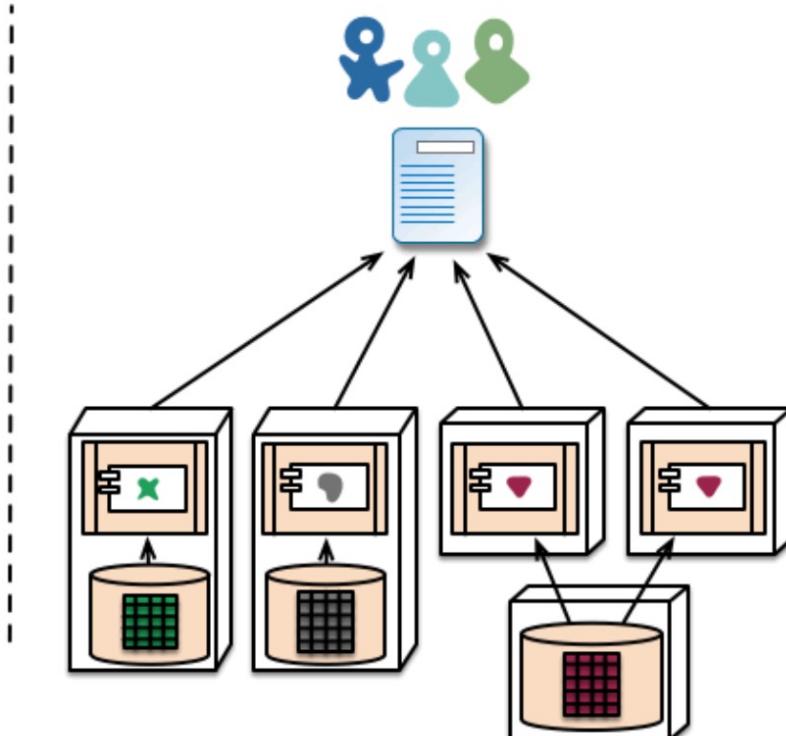
Interfaces publicadas

- A consequência em usar serviços como componentes é ter uma interface mais explícita. A maioria das linguagens não tem uma boa forma de definir explicitamente uma interface *do tipo Published Interface*.
- Serviços ajudam a evitar esse problema usando mecanismos de chamadas remotas através de APIs baseadas em REST, RPC, GraphQL ou outros protocolos.
- Mas usar serviços desta forma tem alguns efeitos colaterais. Chamadas remotas são mais custosas que chamadas dentro do mesmo processo e APIs remotas precisam ser granulares, o que torna ainda mais complicado para usar.
- E Se você precisa mudar as responsabilidades entre os componentes, tais mudanças de comportamento são mais difíceis de fazer do que quando você consegue ultrapassar as fronteiras entre os processos.

Bases de Dados próprias

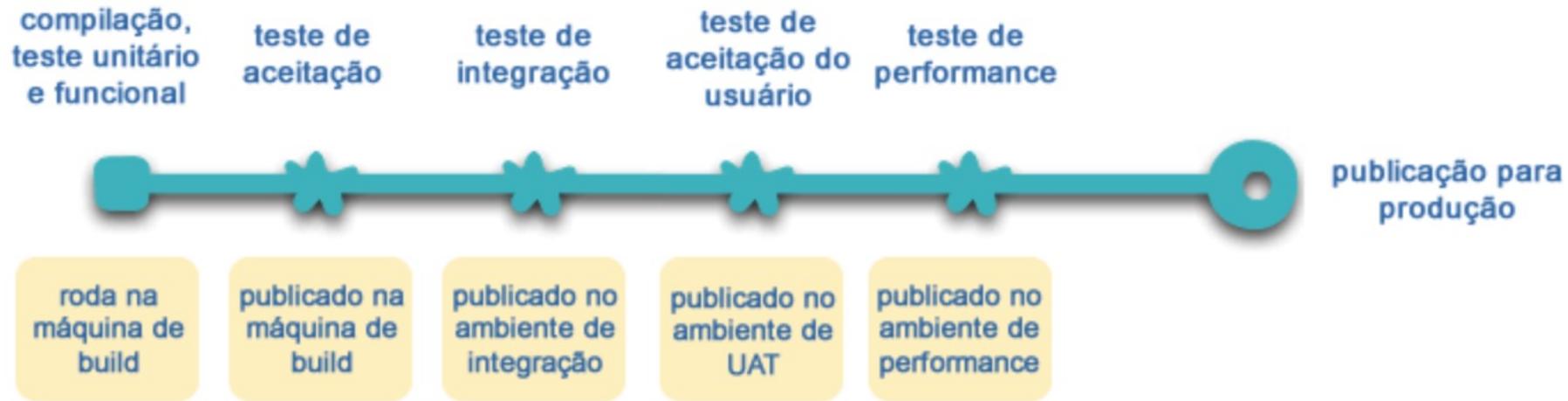


Ambiente monolítico - único banco de dados



Microsserviços - banco de dados por aplicação

Implantação Automatizada



Fonte: Microservices Architecture, James Lewis e Martin Fowler, 2015

Resumo de Características

- Serviços como componentes
- Organização em torno da capacidade de negócios
- Interfaces publicadas (APIs)
- Implantação automatizada
- Inteligência nos *endpoints*
- Controle descentralizado de linguagens e de bases de dados

Governança de Microsserviços

Marco Mendes

Governança Técnica Tradicional

- Uma das consequências de governanças centralizadas é a tendência de padronizar tudo em uma única plataforma tecnológica.
- Exemplos:
 - Um único banco de dados
 - Uma única linguagem de programação.

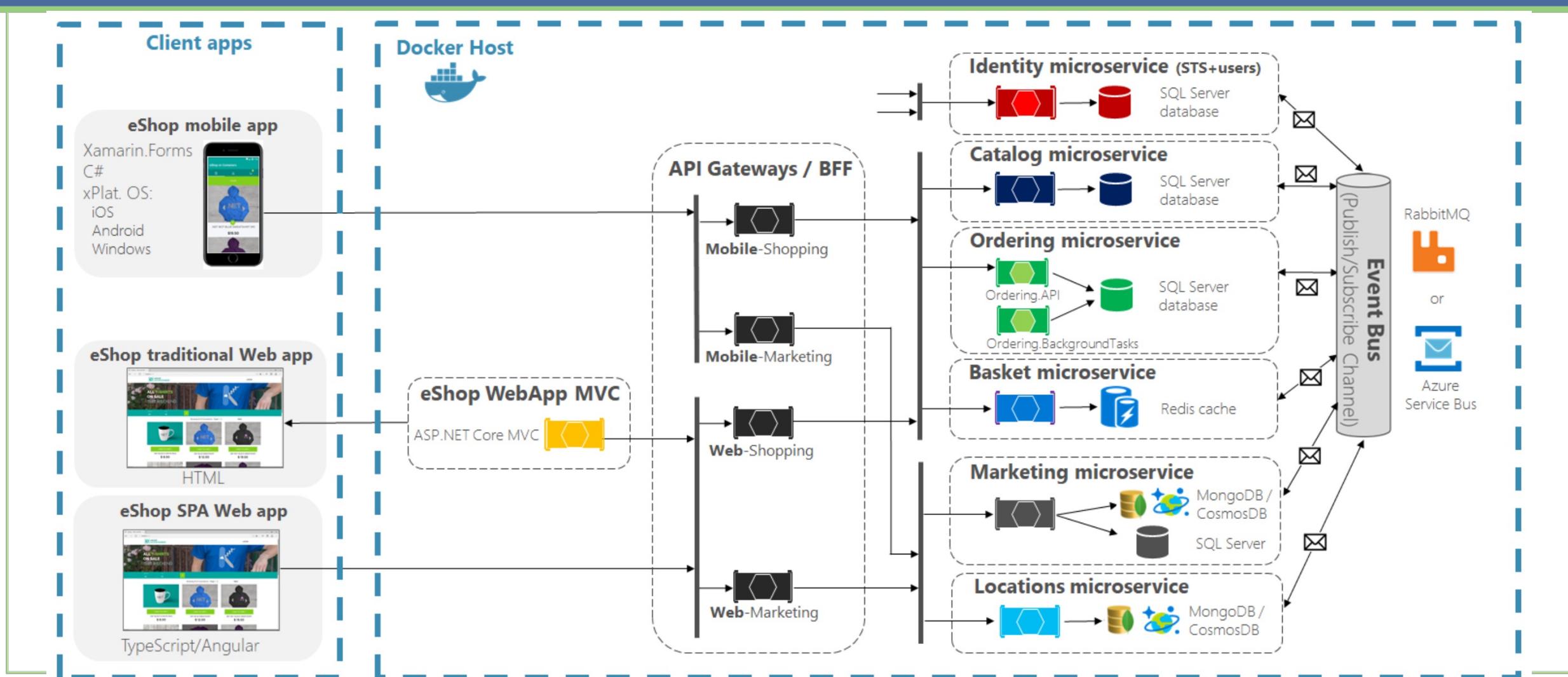
Limites da centralização de tecnologias

- A experiência mostra que esta abordagem é limitada – nem todo problema é um prego, nem toda solução é um martelo.
- Por exemplo, problemas de ciência de dados tendem ser melhor resolvidos por linguagens funcionais.
- E aplicações de telemetria tendem ser melhor resolvidas por banco de dados NoSQL.

Governança Técnica de Microsserviços

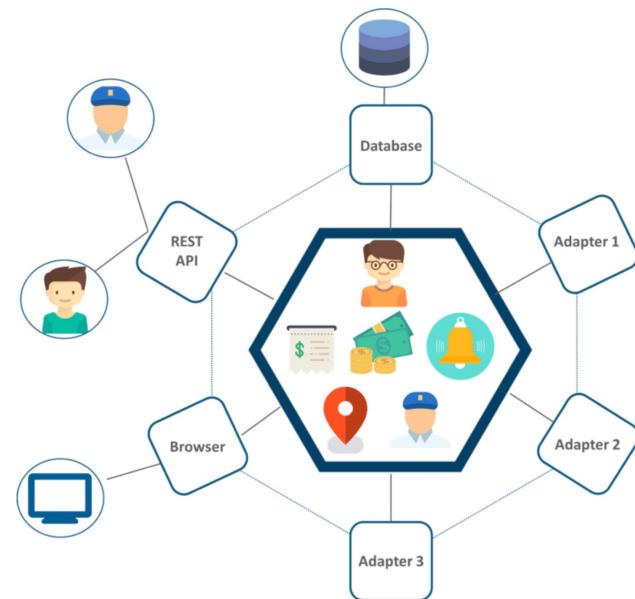
- Times construindo microsserviços também preferem uma **abordagem descentralizada**.
- Ao invés de usar um conjunto definido de padrões escritos em algum papel, eles preferem a ideia de escolher suas próprias linguagens e produzir ferramentas úteis que outros desenvolvedores possam usar para resolver problemas similares aos que eles têm enfrentado.
- Estas ferramentas são extraídas geralmente das próprias implementações e compartilhadas com um grupo maior, algumas vezes usando modelos *open sources* públicos.

Revisão



O caso Uber

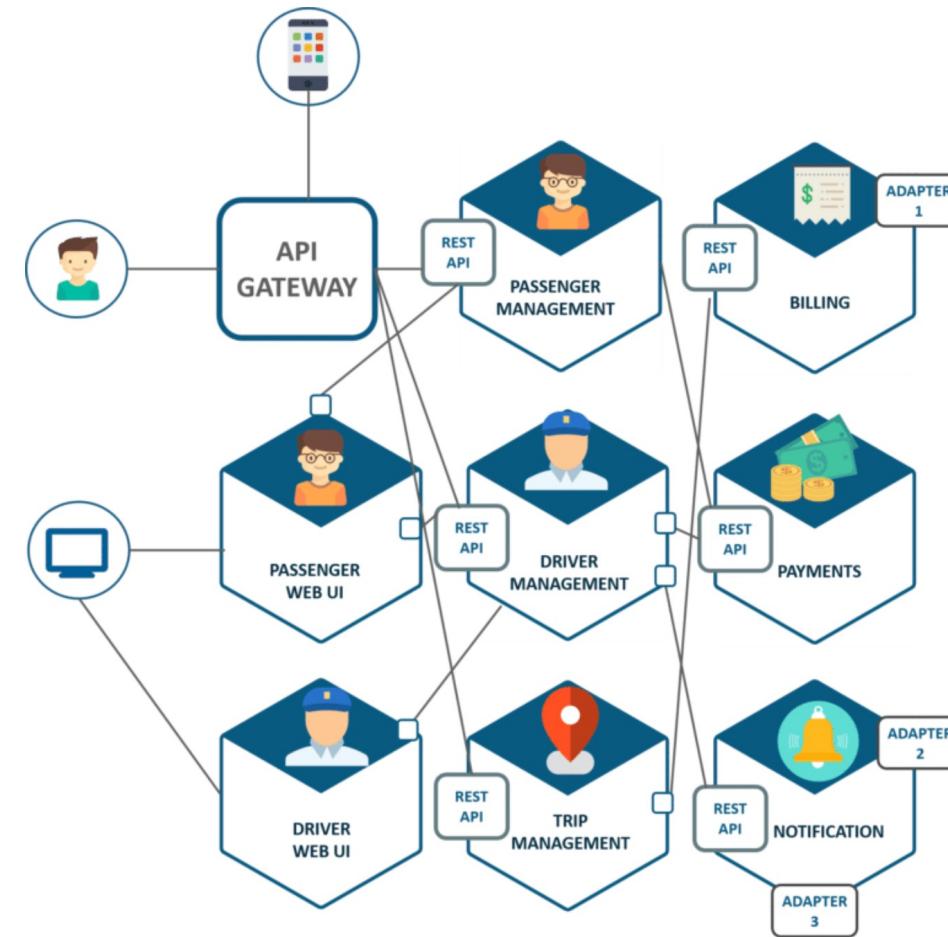
- No início de sua operação em uma única cidade, o Uber possuía uma arquitetura monolítica.
- Havia um repositório de código e um único banco de dados baseado em MySQL.



O caso Uber

Após adotar uma arquitetura de microserviços para escalar seus serviços por todo o planeta, ele optou por uma governança descentralizada.

Exemplo: Os times usam linguagens diferentes (Python, Node.js, Go e Java).



O caso Netflix

- As equipes de engenharia da Netflix produziram dezenas de ferramentas nos últimos anos para suportar a sua arquitetura de microsserviços.
- Essas ferramentas foram disponibilizadas para a comunidade em um modelo *open source*.

Referência: <https://netflix.github.io>

Desafios da descentralização tecnológica

A governança técnica descentralizada traz desafios de comunicação entre os times para que os aspectos técnicos compartilhados entre os serviços sejam equacionados.

Descentralização tecnológica em microsserviços

- Provedores de identidade (IAM)
- API Gateway
- Formatos de mensagens (REST/HTTP, AMQP, MQTT, STOMP)
- Bancos de dados e mecanismos de tratamento de dados
- Geração de conteúdo estáticos (CDN)
- Descoberta de serviços
- Health Check
- Monitoração, *Log* e *Tracing*

Resumo

- A adoção de uma arquitetura de microsserviços precisa fornecer autonomia técnica para os times.
- Governança técnica descentralizada é abordagem recomendada para equipes operando com microsserviços

A Plataforma CNCF

Marco Mendes

Gestão de Dados Descentralizada

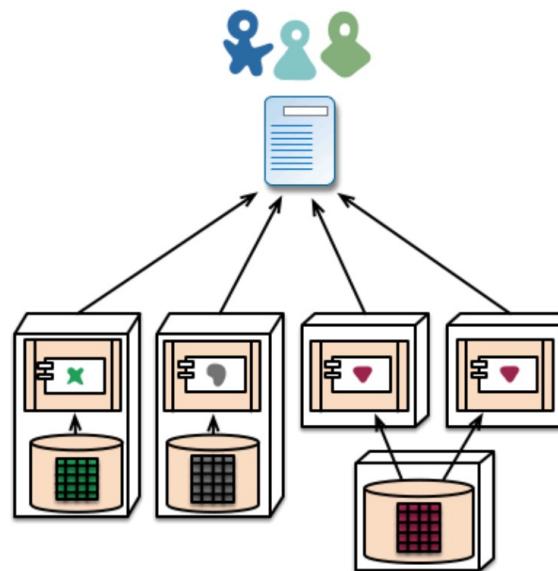
Marco Mendes

Gestão de dados centralizada

- Normalmente temos um banco de dados monolítico com centenas de tabelas.
- A alteração nos esquema de dados é mediado por um AD (administrador de dados).
- O suporte em produção é mediado por um DBA (administrador do banco de dados).

Gestão de dados descentralizada em microsserviços

Microsserviços preferem permitir que cada serviço gerencie sua própria base de dados, quer através de diferentes instâncias usando a mesma tecnologia de banco de dados, ou até mesmo usando diferentes sistemas de banco de dados – uma abordagem chamada Polyglot Persistence.



Microsserviços - banco de dados por aplicação

Gestão de dados descentralizada em microsserviços

- Uma base de dados proprietária para cada microsserviço **não implica que cada microsserviço use um servidor de banco de dados dedicado.**
- Saber diferenciar a visão lógica da visão física é importante na arquitetura de microsserviços para evitar complexidade acidental na infraestrutura de dados.

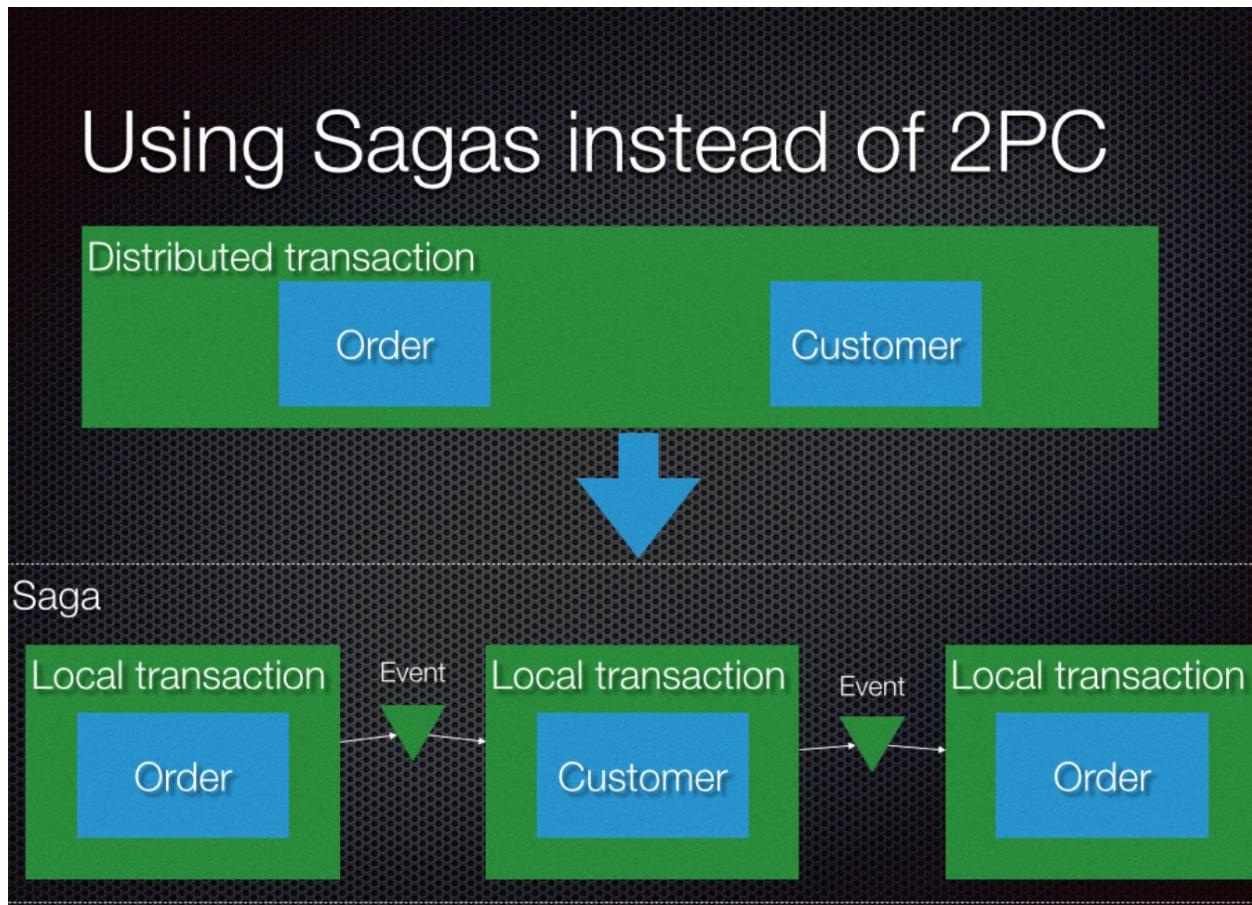
Implicações para o controle transacional

- A responsabilidade descentralizada para os dados através dos microsserviços tem implicações para a administração de atualizações.
- A abordagem comum para lidar com atualizações tem sido usar **transações** para garantir a atualização de múltiplos recursos. Esta abordagem é usada frequentemente em estruturas monolíticas.
- O uso de transações como estas ajuda com a consistência, mas impõem um acoplamento temporário significante, que é problemático quando existem múltiplos serviços.

Implicações para o controle transacional

- Transações distribuídas são nitidamente difíceis para implementar e como consequência, arquiteturas em microserviços enfatizam a coordenação sem transação entre serviços.
- Existe o reconhecimento explícito que consistência pode ser somente eventual e os problemas gerados por isto são lidados com operações que compensem esta questão.
- Um padrão comum para lidar com isso é o SAGA.

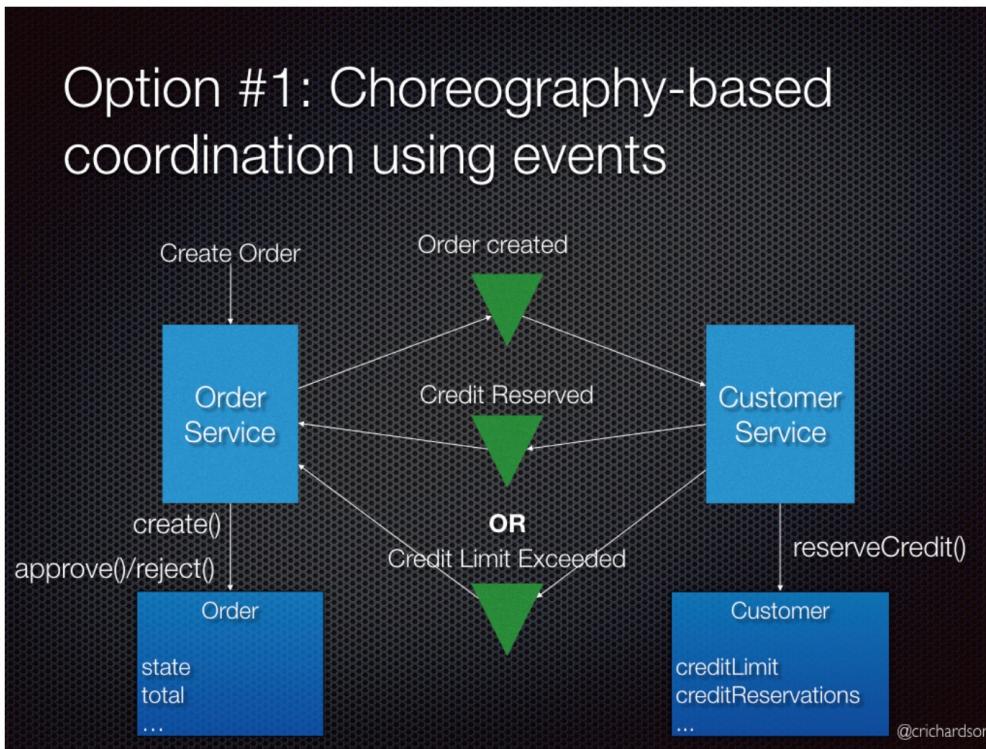
O padrão SAGA



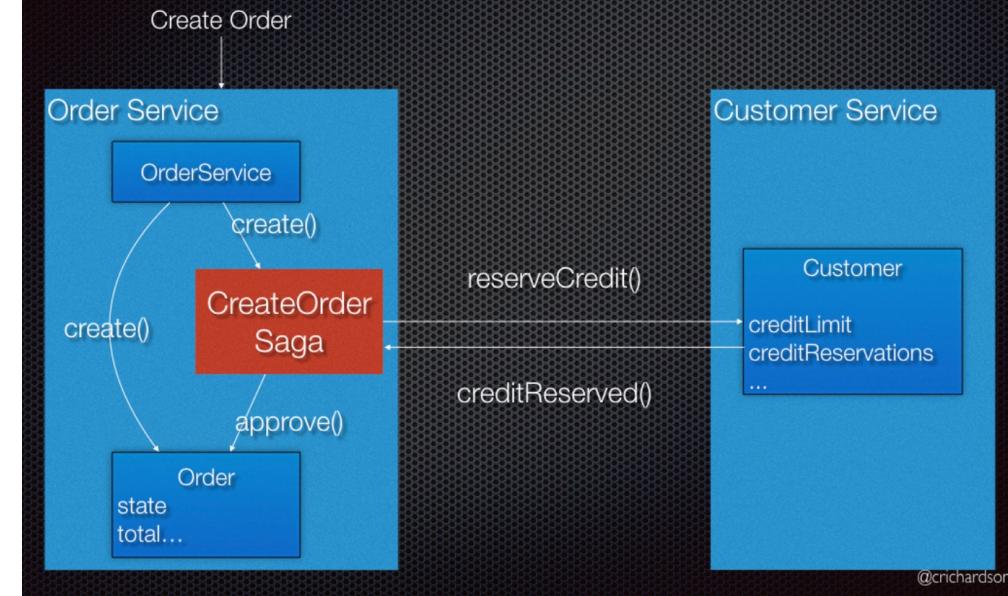
Fonte: <https://microservices.io/patterns/data/saga.html>

O padrão SAGA

Option #1: Choreography-based coordination using events



CreateOrderSaga orchestrator

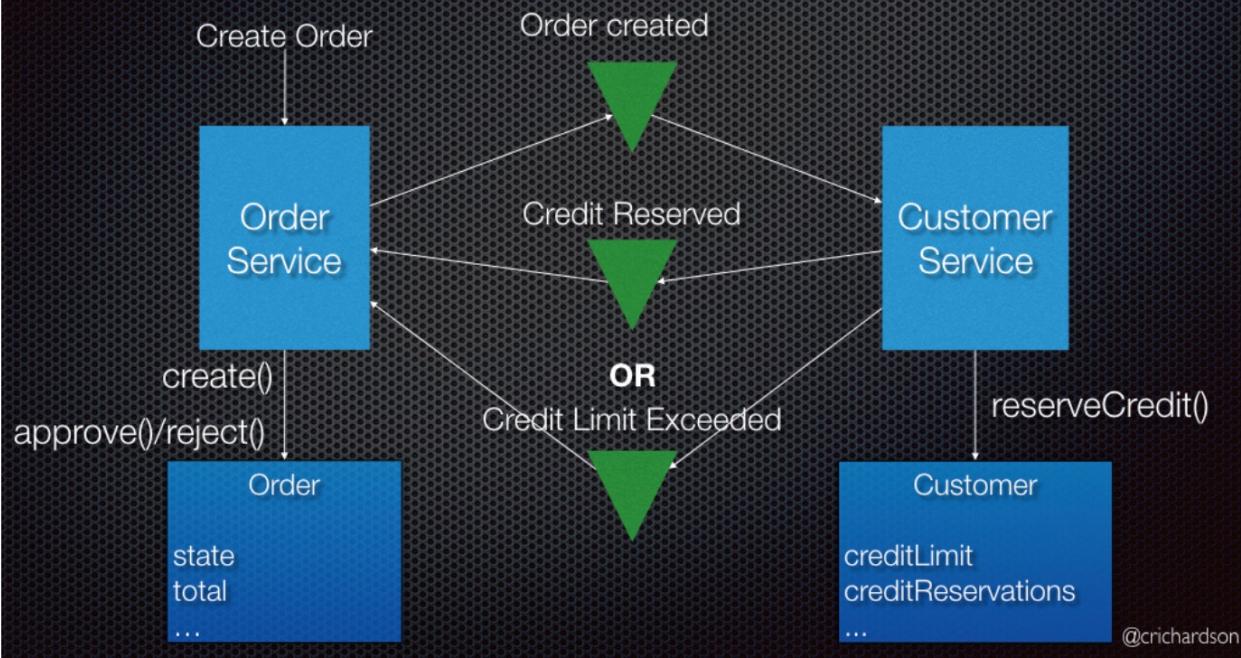


Fonte: <https://microservices.io/patterns/data/saga.html>

PUC Minas Virtual

O padrão SAGA com Coreografia

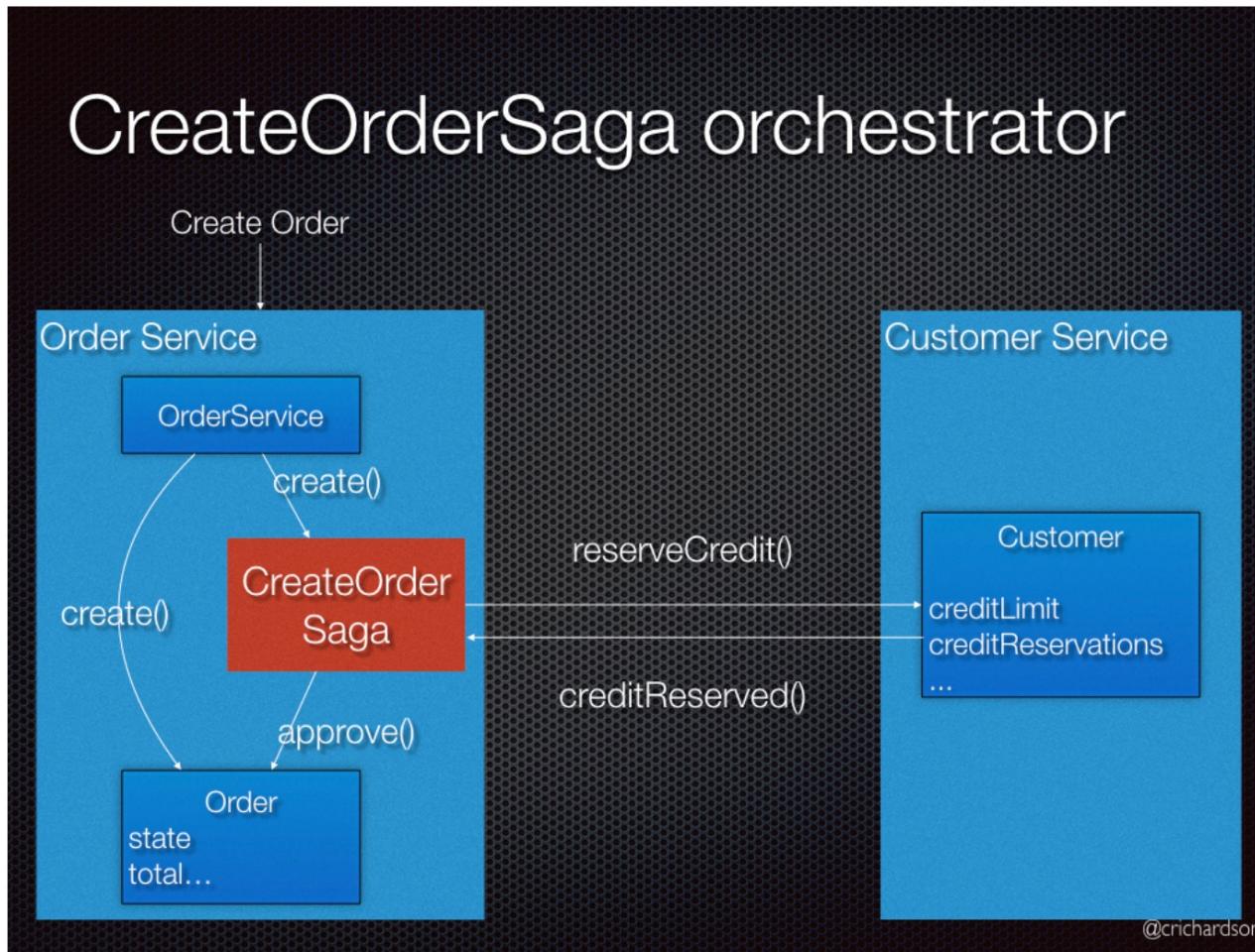
Option #1: Choreography-based coordination using events



Fonte: <https://microservices.io/patterns/data/saga.html>

PUC Minas Virtual

O padrão SAGA com Orquestração



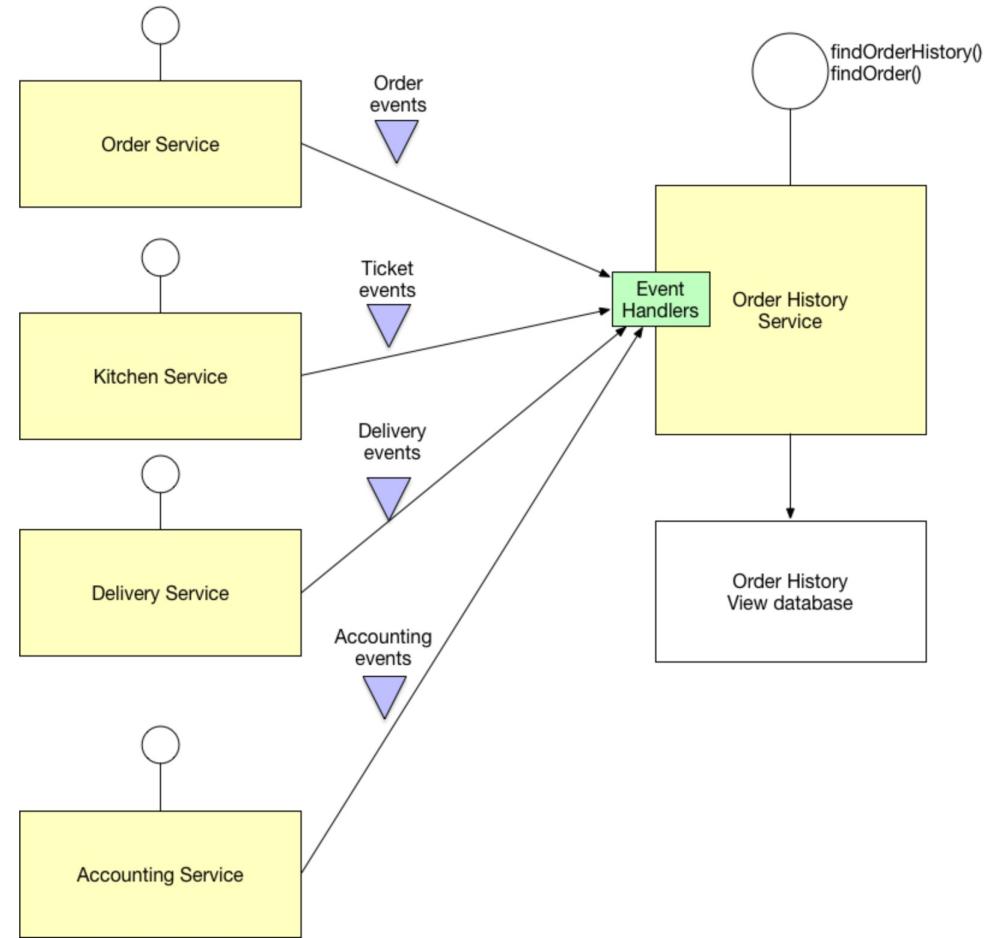
Fonte: <https://microservices.io/patterns/data/saga.html>

PUC Minas Virtual

Implicações para o acesso a dados

- Muitas vezes, dados precisam ser recuperados a partir de diversos serviços e bancos de dados distribuídos em servidores distintos.
- Como resultado, não é mais fácil implementar consultas que associam dados de vários serviços.
- Múltiplas chamadas de rede podem tornar o desempenho de relatórios pesados inviáveis.
- O padrão arquitetural CQRS pode ser usado para lidar com essa situação.

O padrão CQRS



<https://microservices.io/patterns/data/cqrs.html>

Banco de dados por serviço – Pros e Contras

Potenciais benefícios:

- Ajuda a garantir que os serviços sejam fracamente acoplados. As alterações no banco de dados de um serviço não afetam nenhum outro serviço.
- Cada serviço pode usar o tipo de banco de dados mais adequado às suas necessidades.
- Por exemplo, um serviço que faz pesquisas de texto pode usar o ElasticSearch.
- Um serviço que manipula um grafo social poderia usar o Neo4j.

Banco de dados por serviço – Pros e Contras

Pontos de atenção:

- A implementação de transações distribuídas que abrangem vários serviços não é simples. É melhor evitar transações distribuídas por causa do teorema de CAP e usar padrões como o SAGA.
- A implementação de consultas que associam dados que agora estão em vários bancos de dados é um desafio.
- Complexidade do gerenciamento de vários bancos de dados SQL e NoSQL

Resumo

- Em microserviços a gestão de dados se torna descentralizada.
- Isso envolve modelos mentais diferentes para o arquiteto e o time.
- Complexidades accidentais devem ser resolvidas com padrões como o SAGA e o CQRS.

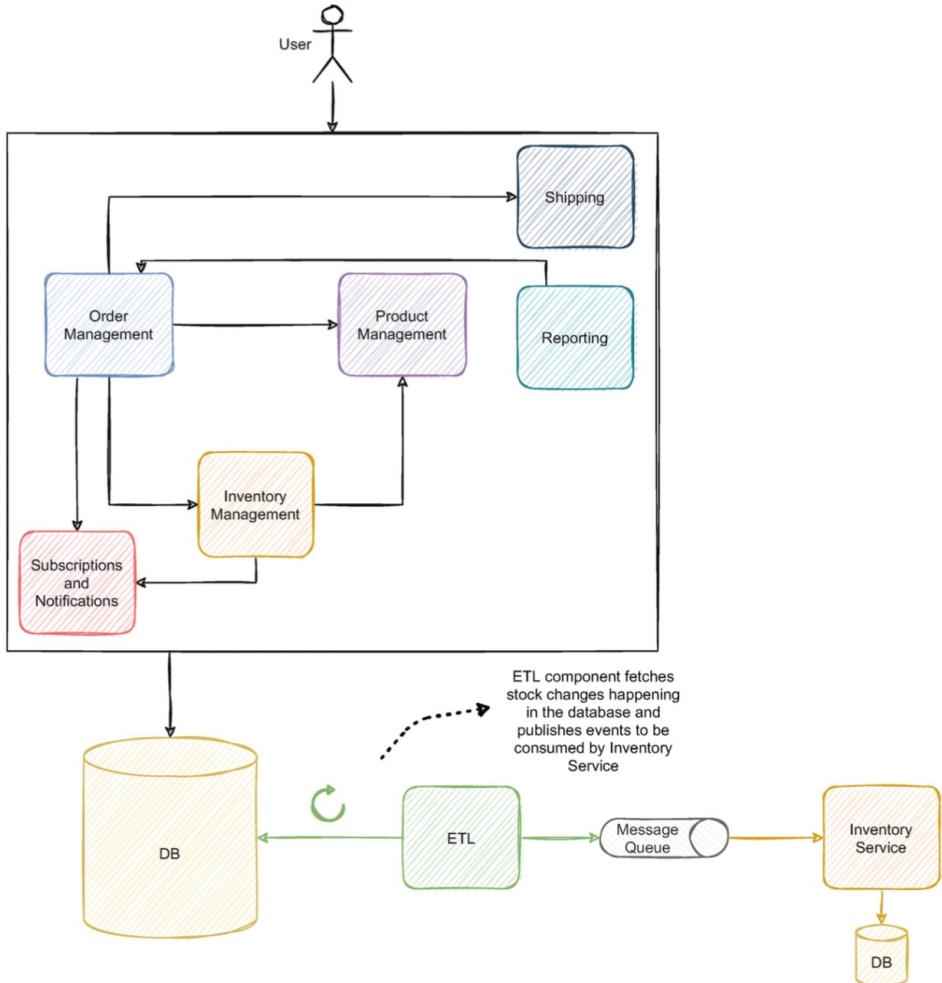
Padrões Arquiteturais de Microsserviços

Marco Mendes

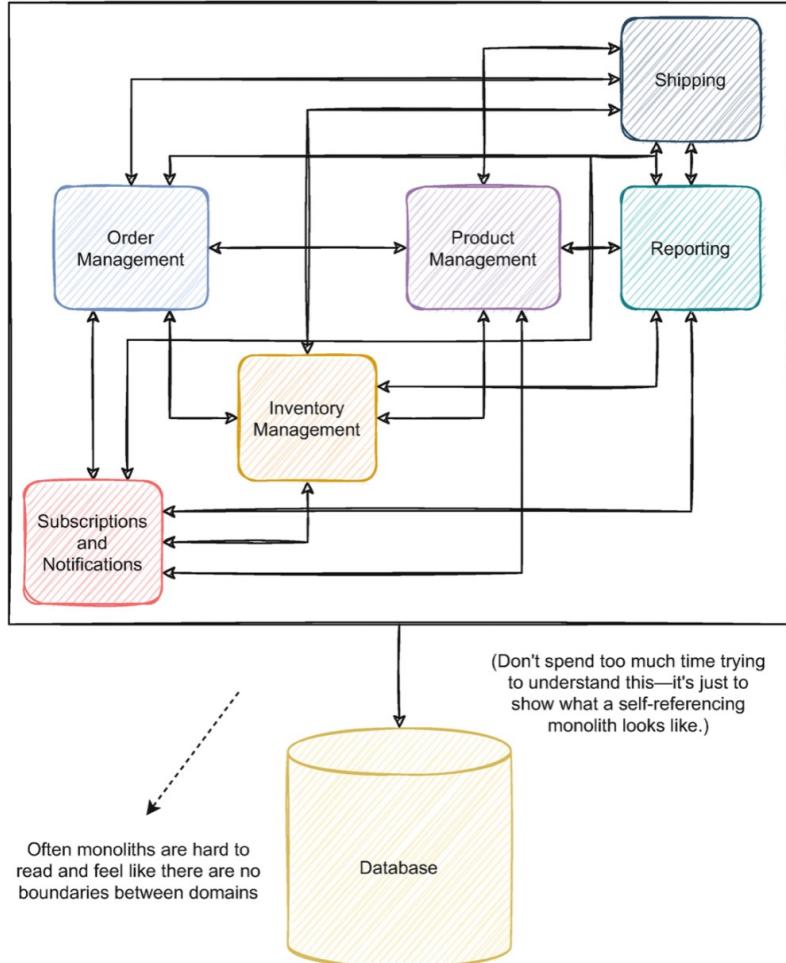
Cenários de Uso de Microsserviços

Marco Mendes

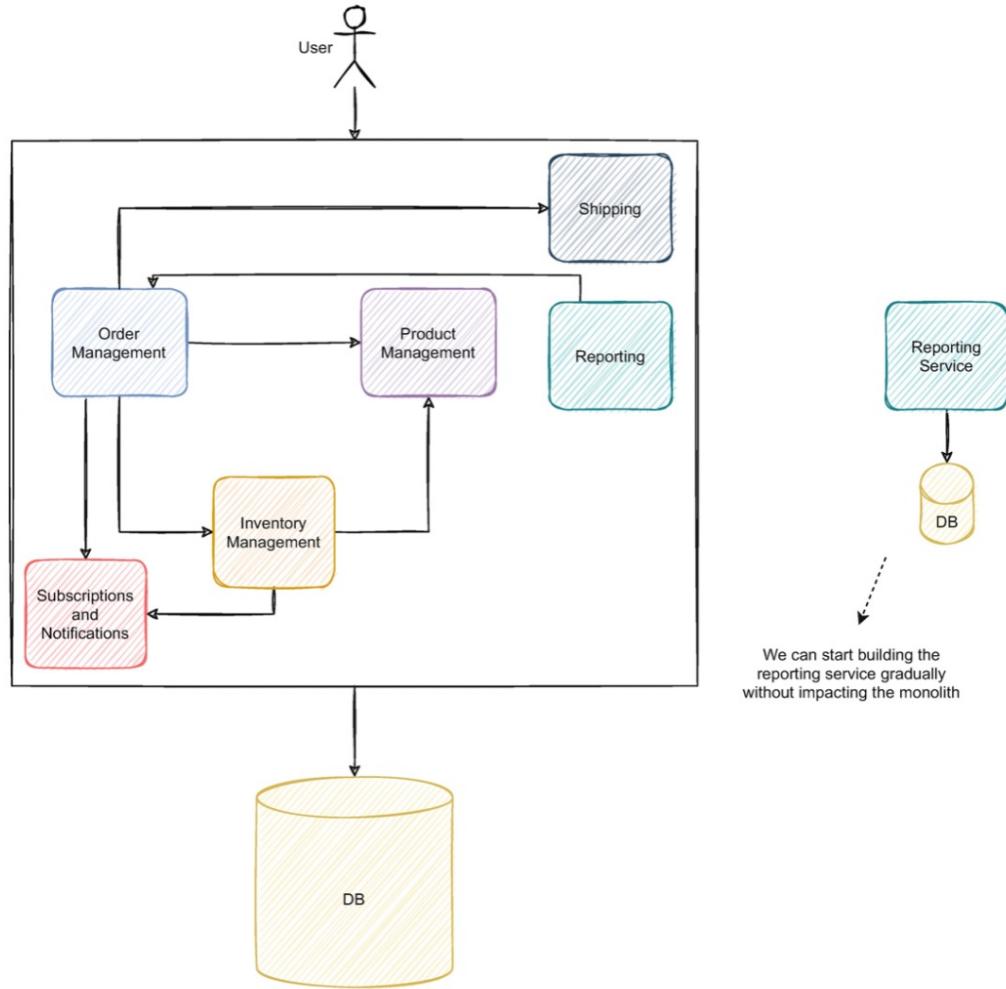
Integração com Código Legado



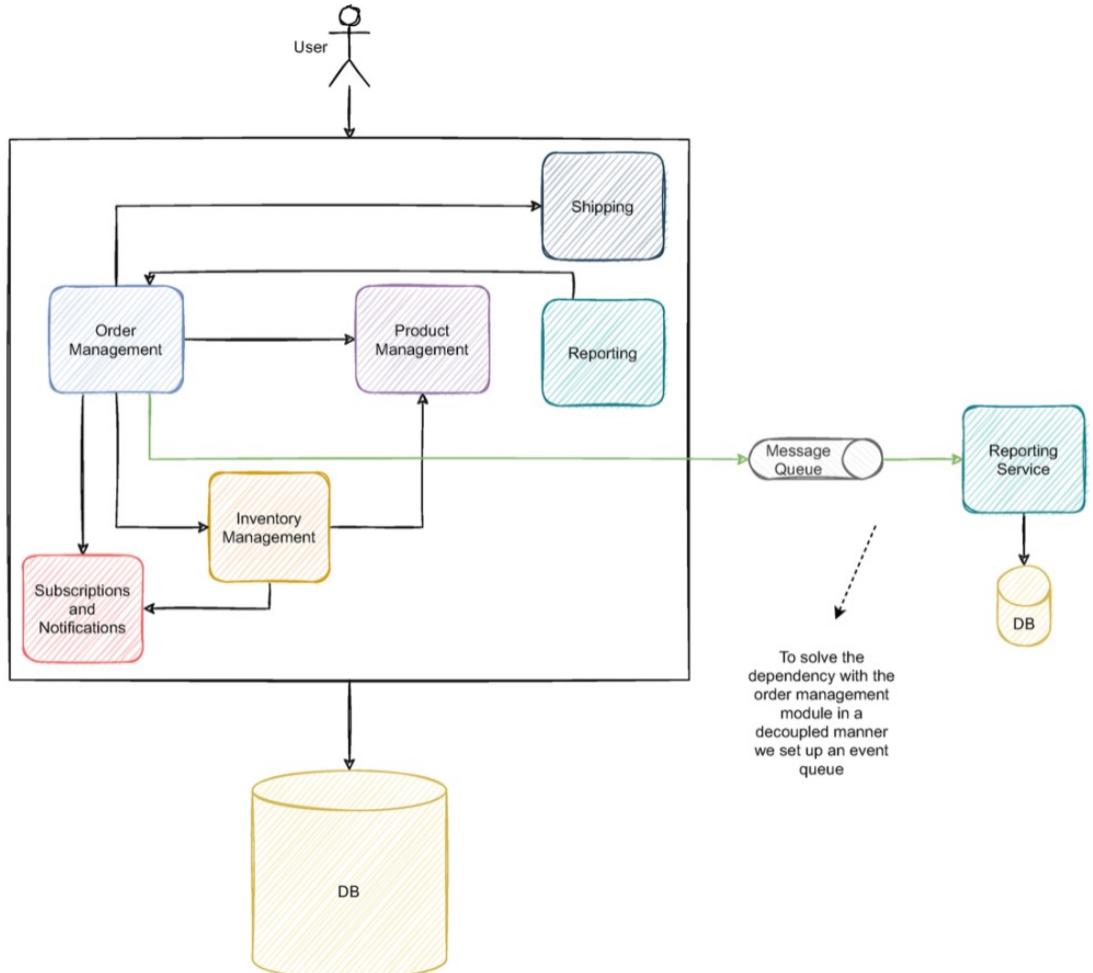
Estrangulamento de Monolitos



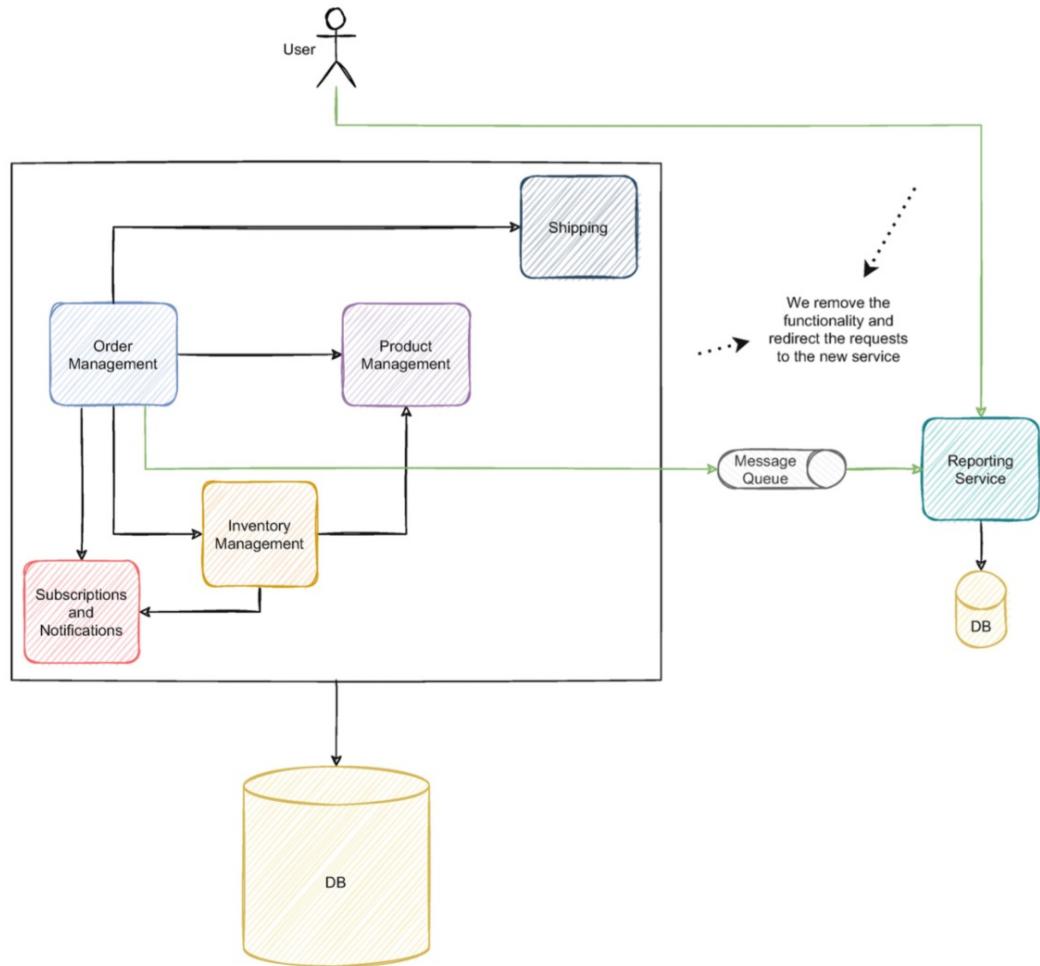
Estrangulamento de Monolitos



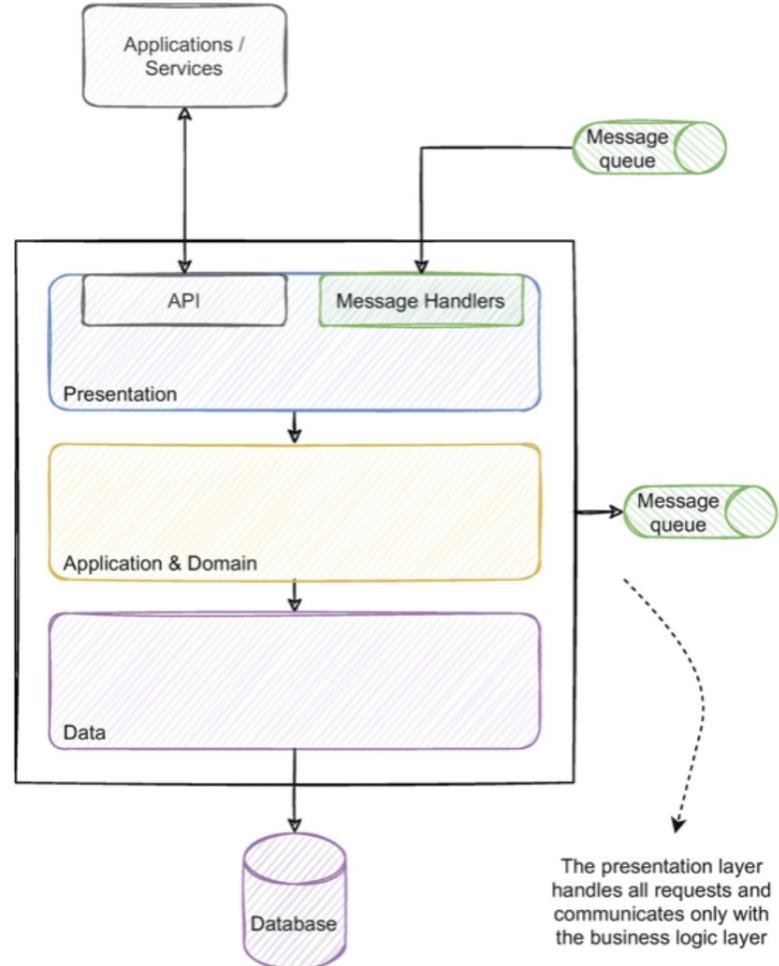
Estrangulamento de Monolitos



Estrangulamento de Monolitos



Microserviços com Arquiteturas de Eventos



Microserviços com Arquiteturas de Eventos

