

MICROSERVICES HANDS-ON

API GATEWAY, WEBHOOKS & ELASTIC APM

P
ANDRÉ PONTES SAMPAIO
profandre.sampaio@fiap.com.br

2019

Prof. André Pontes Sampaio

<https://www.linkedin.com/in/andre-pontes-sampaio/>

Formação

Mestrado em
Engenharia de
Teleinformática

MBA em
Gerenciamento
de Projetos

Graduação em
Ciência da
Computação

Segmento de atuação

SEFAZ-SP:
Diretor de
Operações e
Infraestrutura

Banco:
Consultor
Técnico

Telecom:
Engenharia
Comercial

Certificações

Service
Offering and
Agreement

Planning
Protection and
Optimization

Operational
Support and
Analysis

Objetivos do Curso

O que são e como funcionam os Microserviços?

Quais ferramentas podem ser utilizadas?

Arquiteturas de Projetos e Monitoração de Microserviços



Objetivos da Disciplina

Aulas com explicações prática

Adoção de ferramentas DevOps

Fornecimento de microserviços com API e KAFKA

Laboratórios com hands-on

Laboratórios baseado em Javascript, Python, NodeJS

Fluxo CI/CD utilizando Docker

Padrões de Projetos

Arquitetura REST e principais padrões de aplicações



Módulo 01

- **Microserviços:**
 - Definindo Microservices
 - SOA versus Microservices
 - A arquitetura de Microservices
 - Consumindo Microservices via API Webhooks **[LAB1 e LAB 2]**

Módulo 02

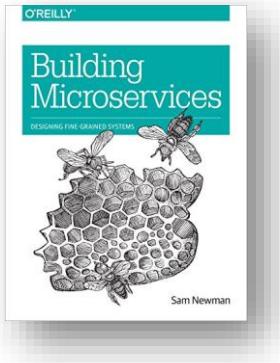
- **Desenvolvendo Microserviços:**
 - Construindo o Microservices da camada de apresentação ao Backend **[LAB 2 - continuação]**
 - Containers e Microservices: fluxo CI/CD **[LAB 3]**
 - Microservices e Databases: queries e consistency

Módulo 03

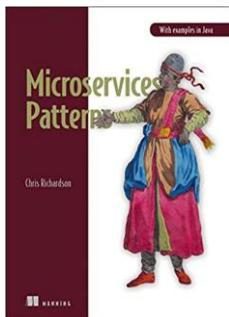
- **Microserviços no dia a dia:**
 - Dividindo o Monolítico
 - Monitoramento de Microservices
 - Elastic APM, elasticsearch e Kibana **[LAB 4]**

Materiais de Estudo

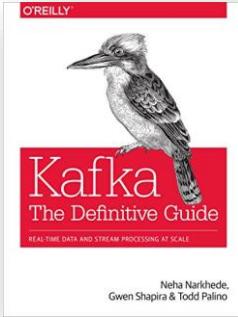
FIAP



Building Microservices: Designing Fine-Grained Systems
1st Edition [2015], *Sam Newman*



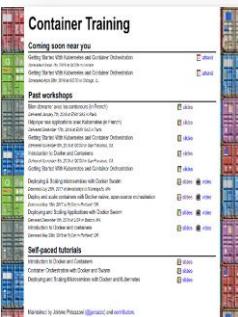
Microservices Patterns: With examples in Java
1st Edition [2018], *Chris Richards*
Disponível em: <https://microservices.io/patterns/>



Kafka: The Definitive Guide: Real-Time Data and Stream Processing at Scale

1st Edition [2017], Neha Narkhede, Gwen Shapira, Todd Palino

Disponível em: <https://www.confluent.io/wp-content/uploads/confluent-kafka-definitive-guide-complete.pdf>



Self-paced tutorials: Introduction to Docker and Containers

Disponível em <https://container.training/>

Maintained by Jérôme Petazzoni and contributors

MICROSERVIÇOS

Como lidar com as questões abaixo?

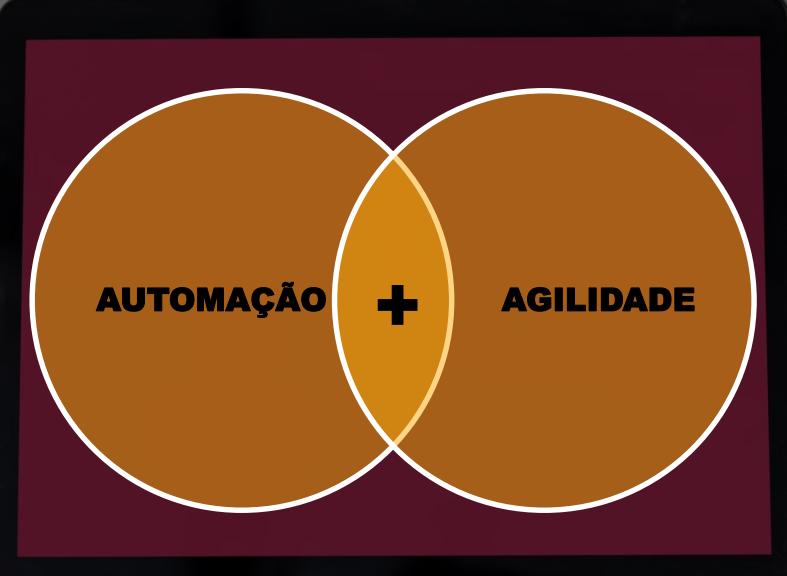
FIAP

Serviços de TI devem atender as necessidades atuais e futuras

Ambientes híbridos mais complexos de gerenciar

Restrições de orçamento e equipe
Aumento dos riscos

Questionamentos sobre os retornos dos investimentos realizados em TI



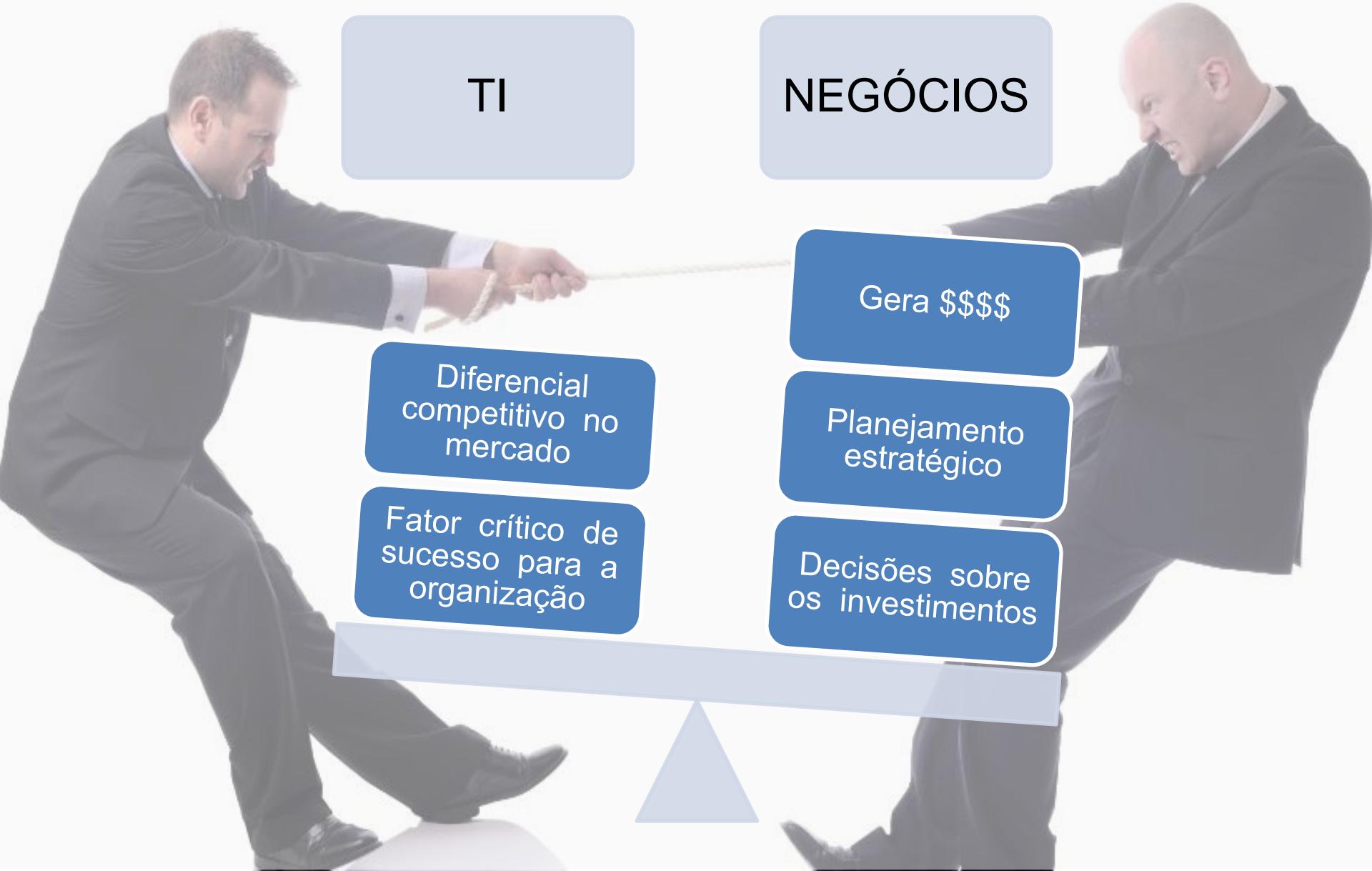
Expectativa de reunião executiva...



Tempo gasto nas atividades



Queda de braço



CUSTOMER
FEEDBACK
SUPPORT
INNOVATIVE
QUALITY
EXCELLENT
FRIENDLY



Definição de Serviços

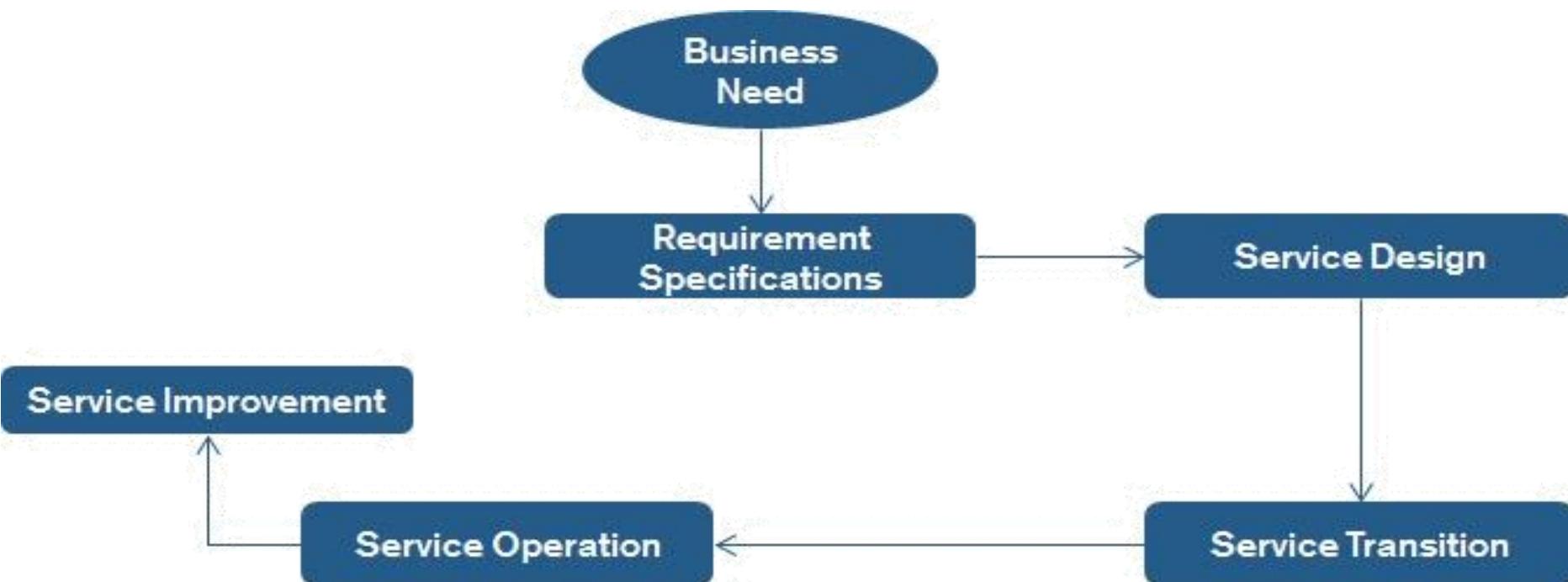
FIAP

Utilidade relaciona-se com as funcionalidades que o cliente quer no serviço (O QUE)

Garantia está associada ao nível de qualidade na entrega do serviço (COMO)

Serviço é uma forma de **ENTREGAR VALOR** ao cliente, facilitando a **OBTENÇÃO DE RESULTADOS** que os clientes querem alcançar, sem que eles assumam a propriedade dos riscos inerentes.

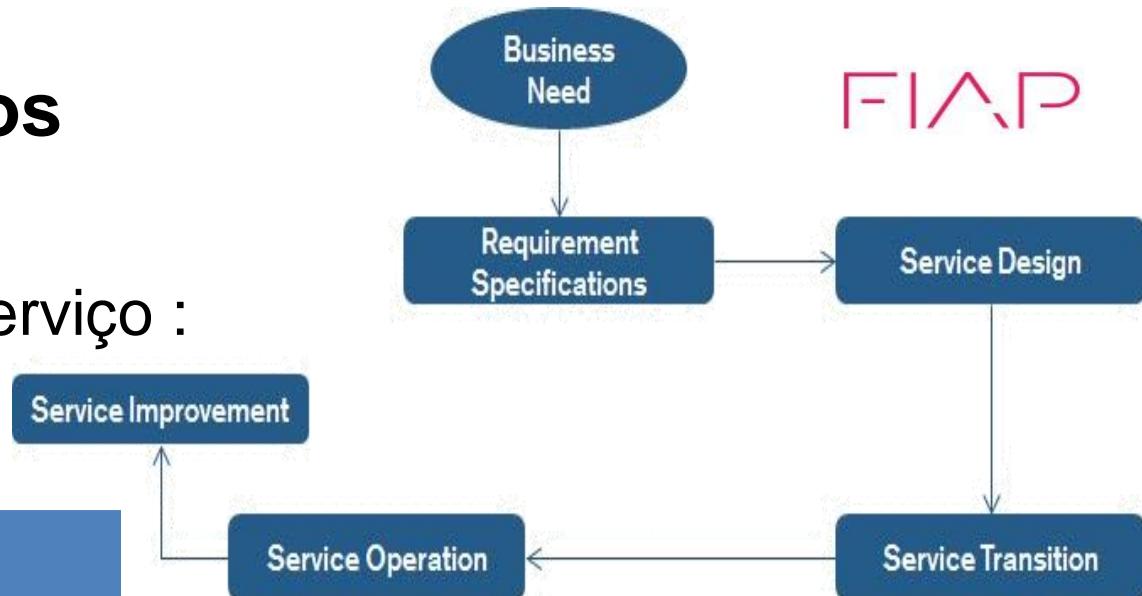
Ciclo de Vida de um Serviço :



Gestão de Serviços

FIAP

Ciclo de Vida de um Serviço :



Como eles nascem?

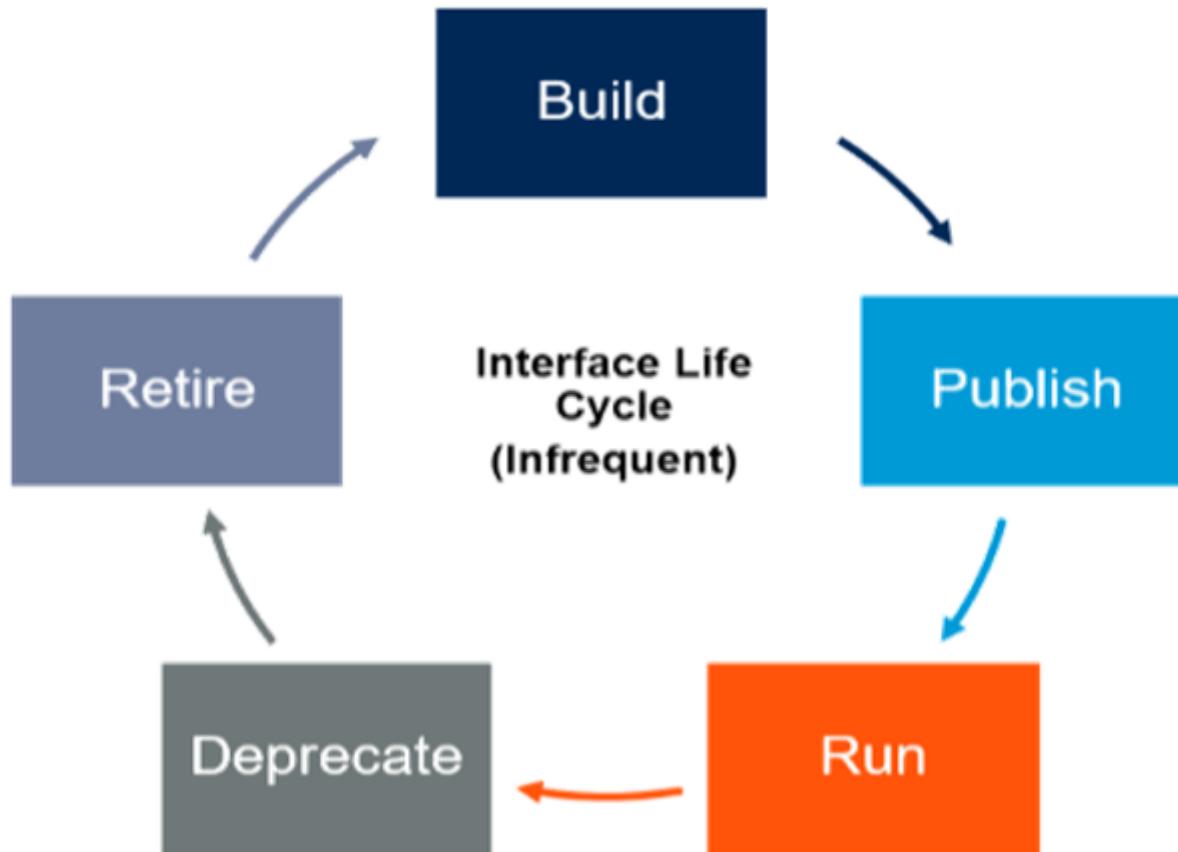
Qual a relação entre um novo serviço e os Projetos?

Onde é possível aplicar metodologia Ágil?

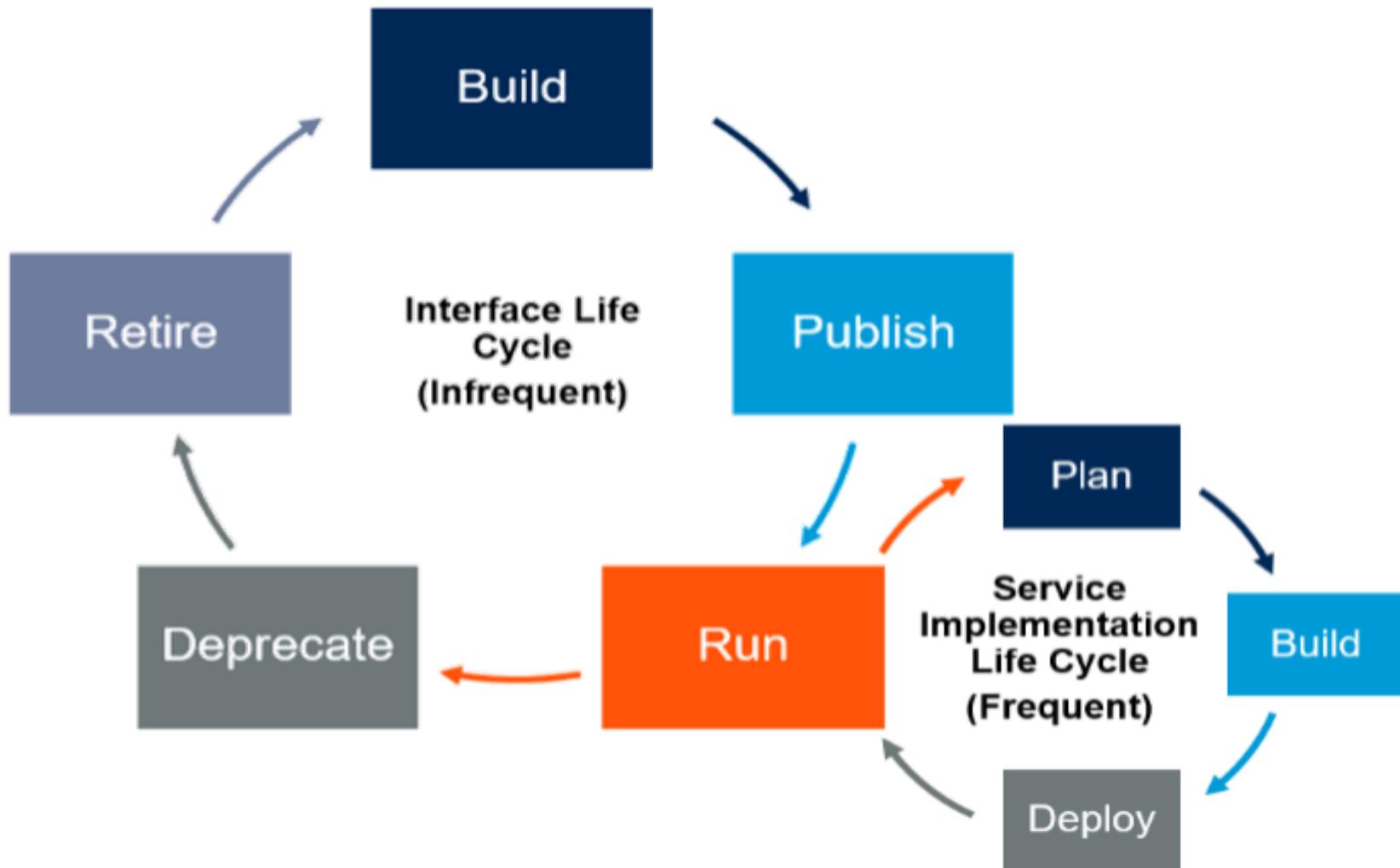
Em que momento do ciclo de vida o serviço gera valor?

Onde está o fluxo para aposentar um serviço?

Ciclo de vida de um código de software



Ciclo de vida de um código de software





Microservices

***Microserviços são
pequenos serviços
autonômios que
funcionam de forma
integrada.***



Microserviços são **pequenos**:

Focado em fazer 1 coisa bem feita

Quando precisar mudar algo, que seja em um único lugar

Alta coesão: agrupa componentes relacionados no mesmo código

Como saber quão pequeno? Dependendo do contexto.

Para uma startup: podem ser reescritos em até 2 semanas?

Código é muito grande para ser gerenciado por um time de 3 pessoas?

Microserviços são **autônomos**:

Alterados de forma independente uns dos outros

Seu deploy não pode afetar outros serviços

Possibilitam uma abstração de como eles funcionam

Detalhes internos ficam dentro do próprio serviço

Escalabilidade: aumento de capacidade granular

Tecnologia utilizada deve evitar acoplamento

Microserviços funcionam de forma **integrada**:

Comunicação por eventos/API

Funcionalidades consumidas por outros serviços

Permite o uso através de diversos canais: Web, Mobile, Tablets, Comandos de voz, Relógios, TVs

Novas tecnologias podem ser testadas em componente menos crítico

Resiliência: falha deve ficar isolada e não impactar o funcionamento global

REGRA DE OURO

**Deploy do microserviço pode ser feito sem afetar
nenhum outro?**



**Se precisar alterar serviços externos, não estaremos
aproveitando as vantagens dessa arquitetura.**

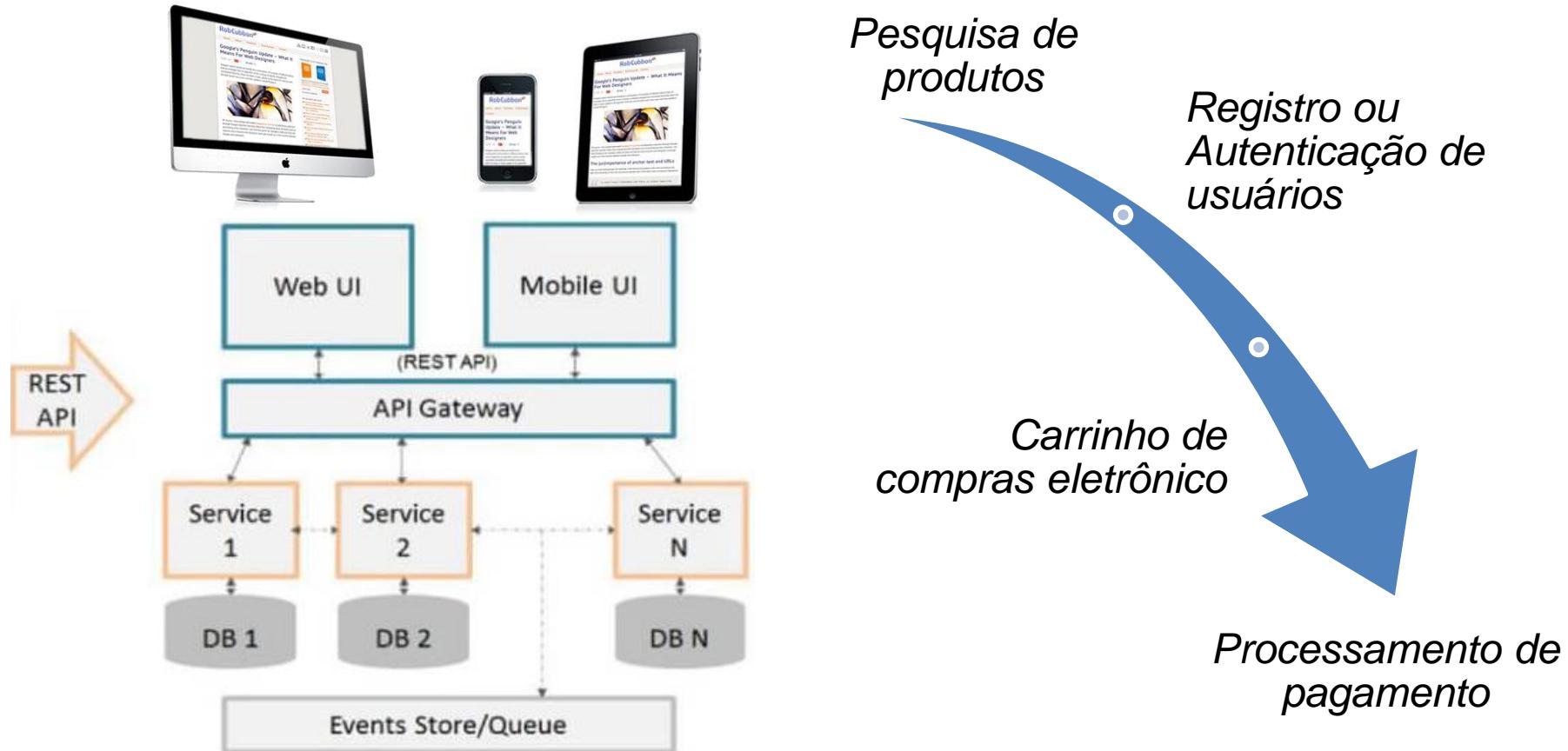
Visão geral do uso de Microserviços



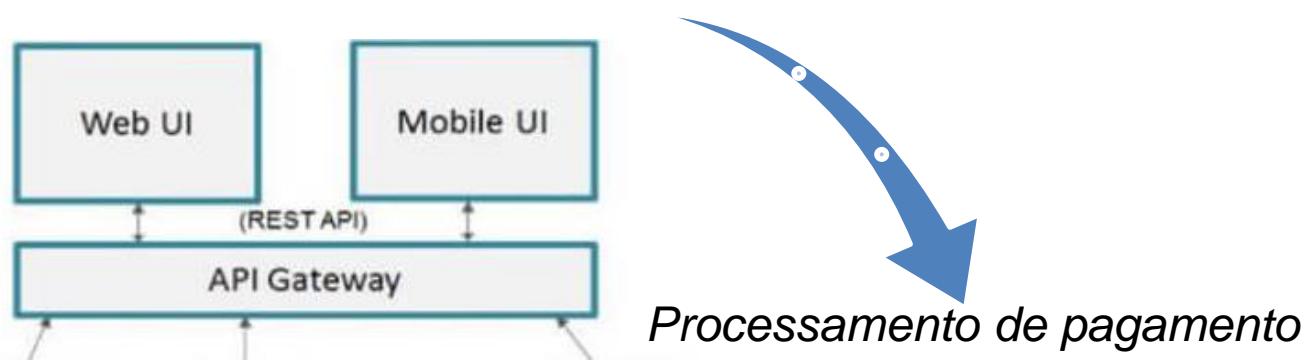
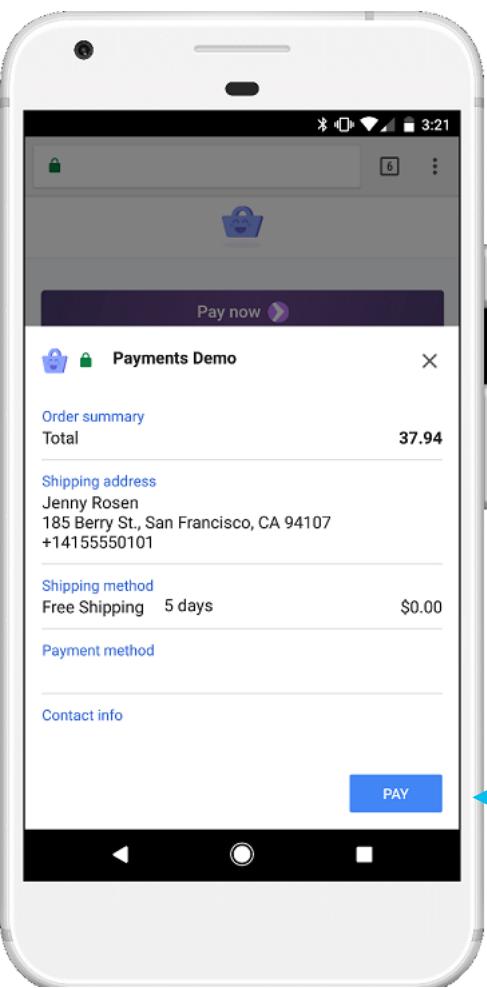
Gestão de Microserviços

FIAP

Exemplo: MICROSERVIÇOS DE COMÉRCIO DIGITAL



Chamadas síncronas de API: request X response:



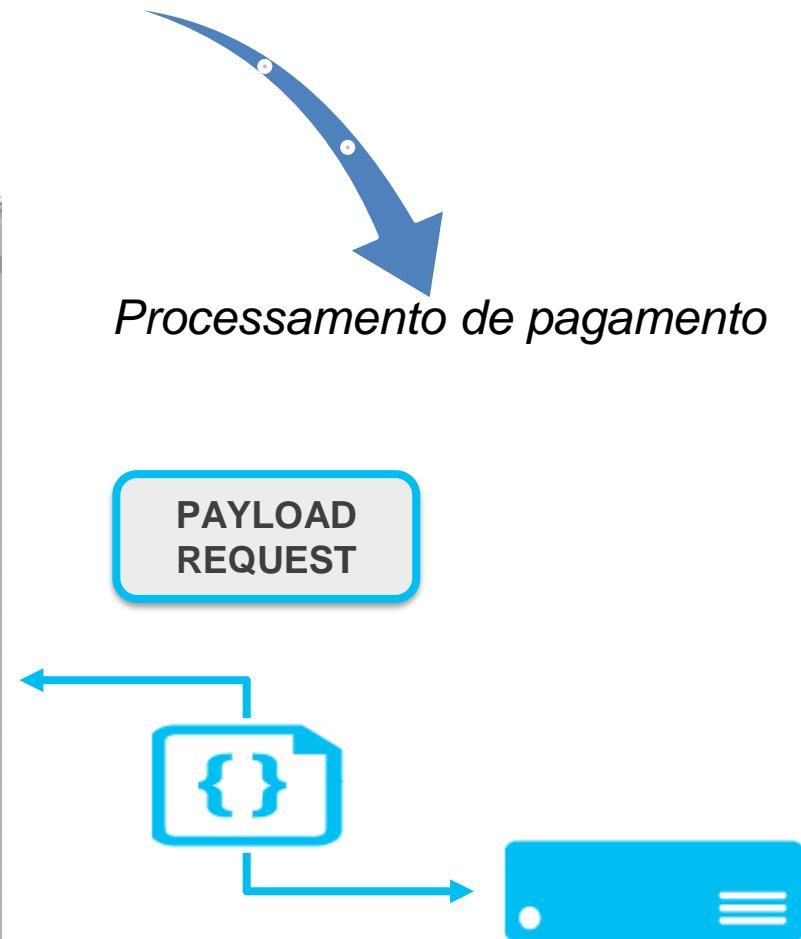
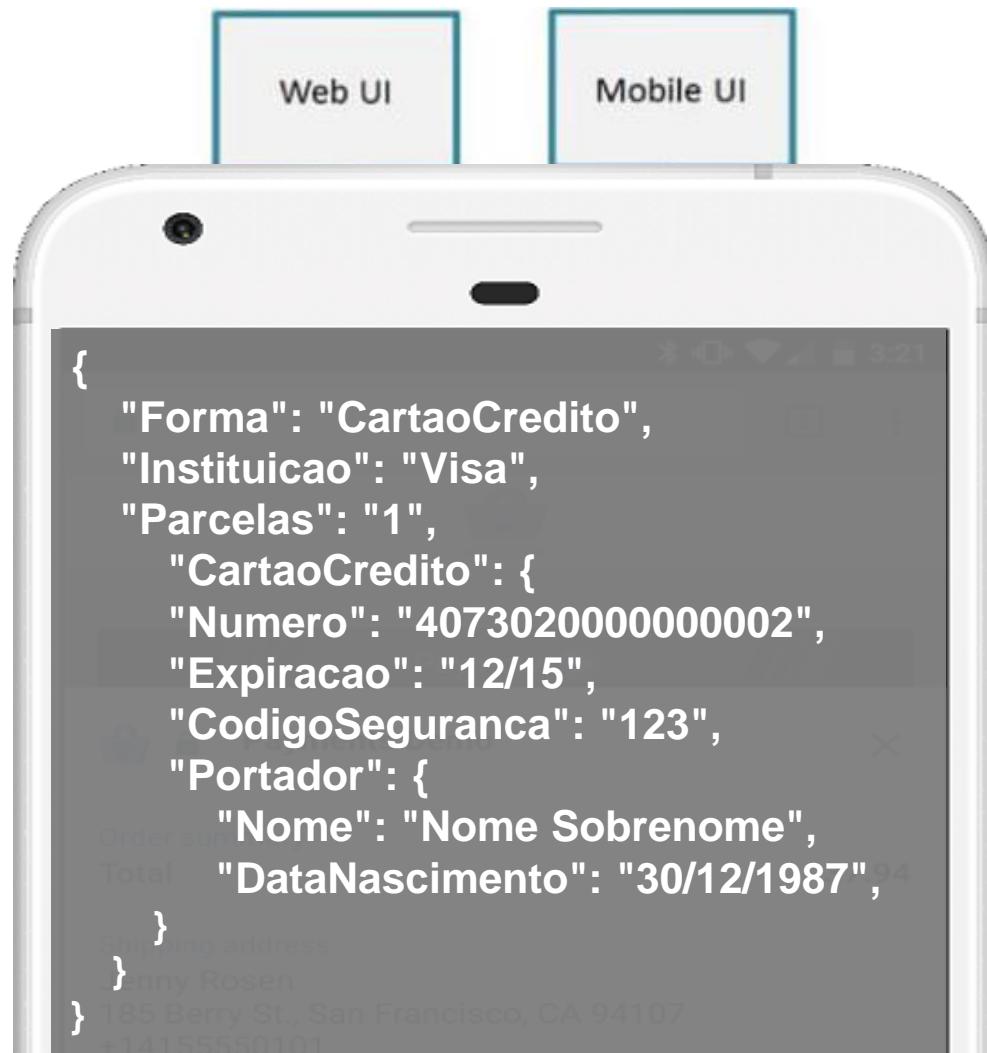
HTTP POST
JSON



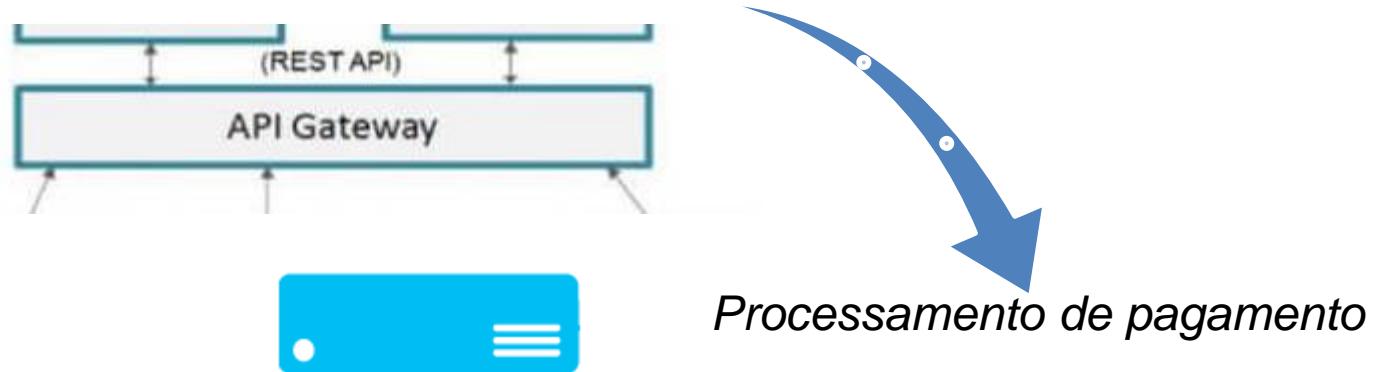
`http://servidor/api/v1/pagamento/`



Fazendo a solicitação



Resposta de uma solicitação



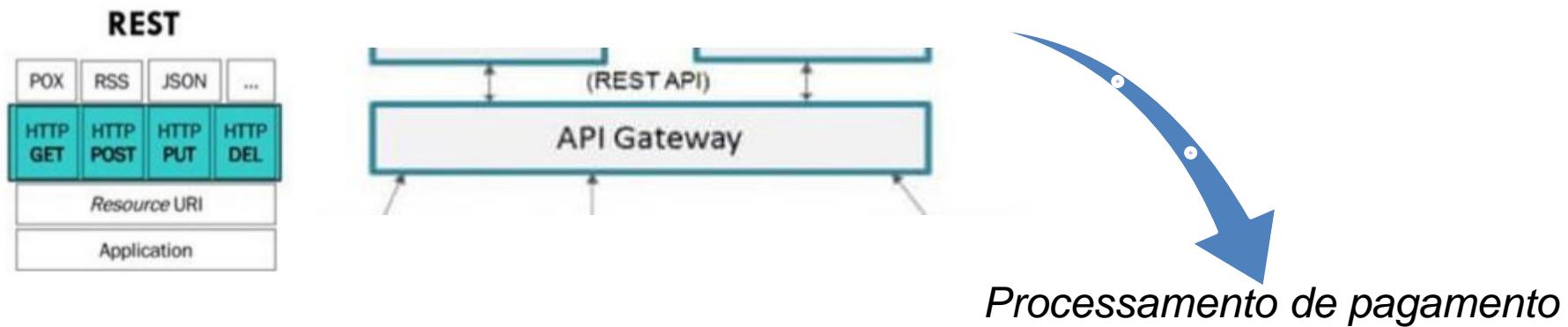
RESPONSE HEADER

Request URL: <http://servidor/api/pagamento>
Request Method: POST
Status Code: 201 CREATED
X-Requested-With: XMLHttpRequest

PAYLOAD RESPONSE

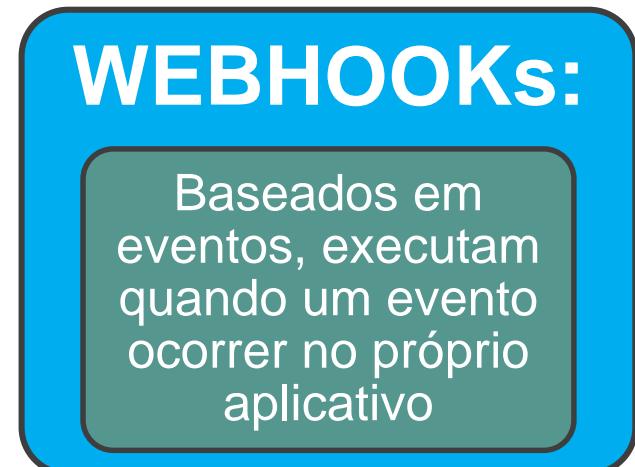
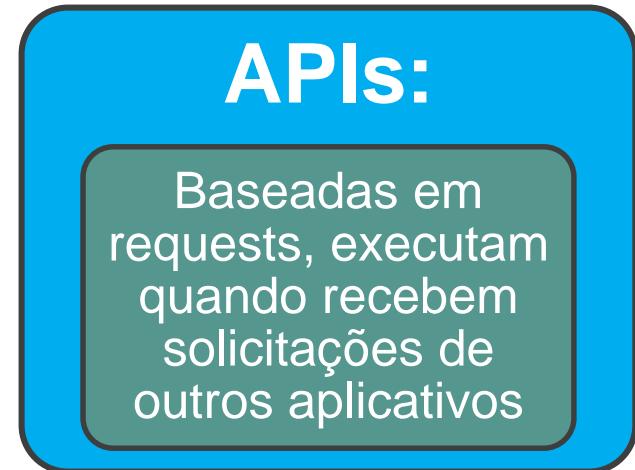
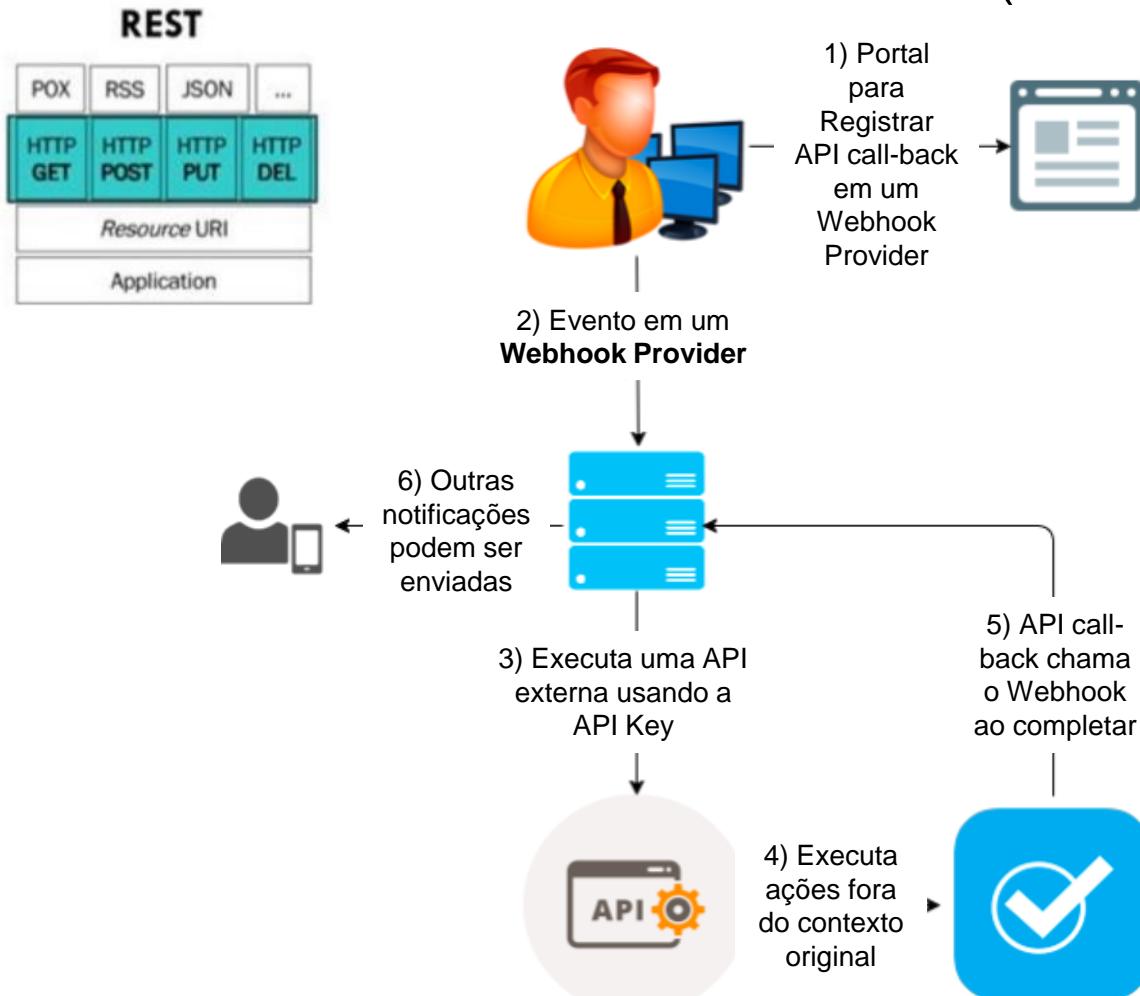
```
{  
  "StatusPagamento": "Sucesso",  
  "StatusInterno": "EmAnalise",  
  "ValorPago": "37.94",  
  "Mensagem": "Pago com sucesso",  
  "CodigoRetorno": "012345",  
}
```

Serviço RESTful: arquitetura REST com API utilizada via HTTP



Ação (CRUD)	Verbos HTTPs	Recurso (Endpoint)	Descrição
Create	POST	/api/v1/pagamento/	Criar novo pagamento
Read	GET	/api/v1/pagamento/	Listar todos os pagamentos
	GET	/api/v1/pagamento/:id	Informações de um pagamento
Update	PATCH	/api/v1/pagamento/:id	Atualizar alguns campos de um pagamento
	PUT	/api/v1/pagamento/:id	Atualizar todos os campos de um pagamento
Delete	DELETE	/api/v1/pagamento/:id	Excluir um pagamento

WEBHOOKS: integração de APIs de vários aplicativos baseada em eventos (assíncronos)



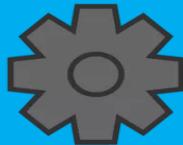
Formas de funcionamento:

APIs:

Request de outra APP



Chamada HTTP para
o Endpoint da API



Execução da API



Evento

WEBHOOKS

(também chamado de HTTP Callbacks):



POST

API / Aplicações de 3os
Listener / URL



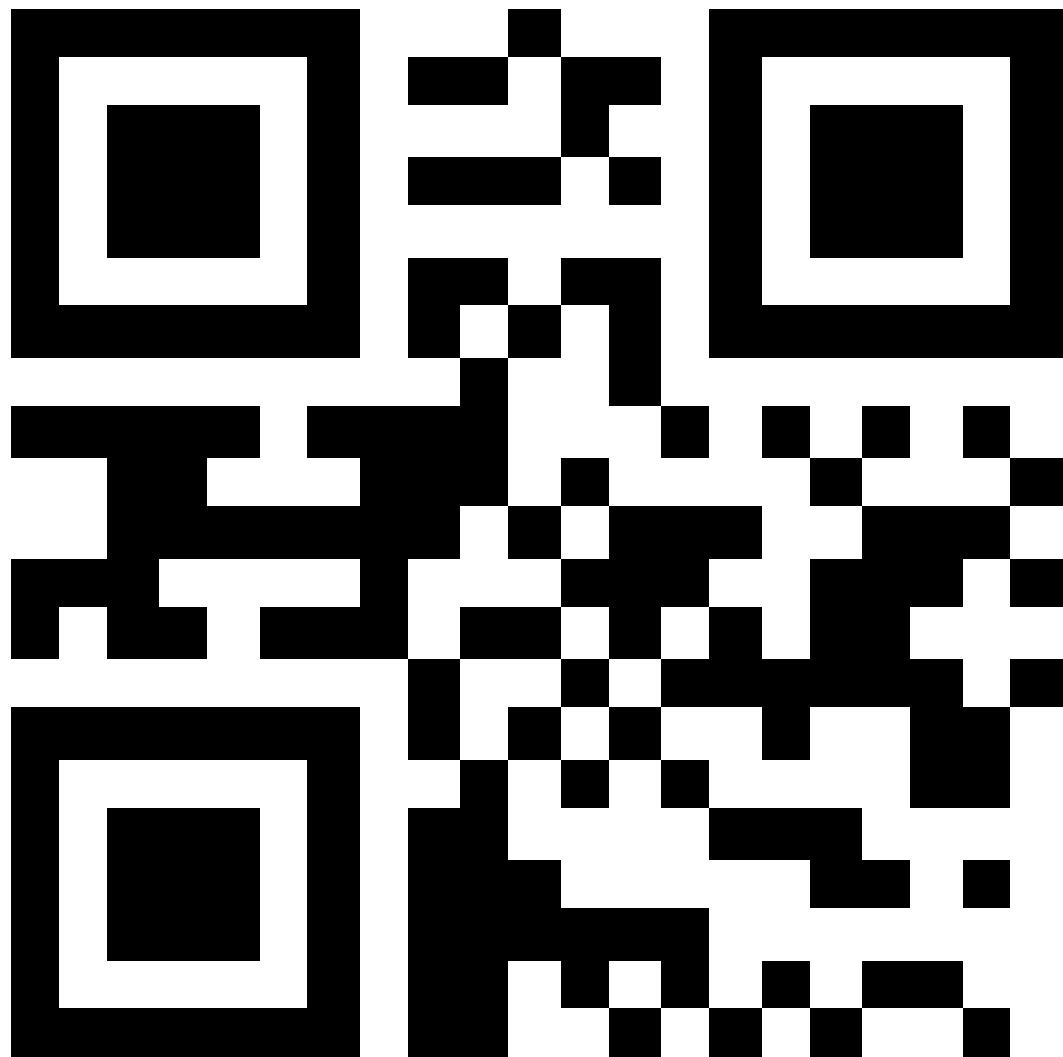
Execução de ações

Workspace Colaborativo

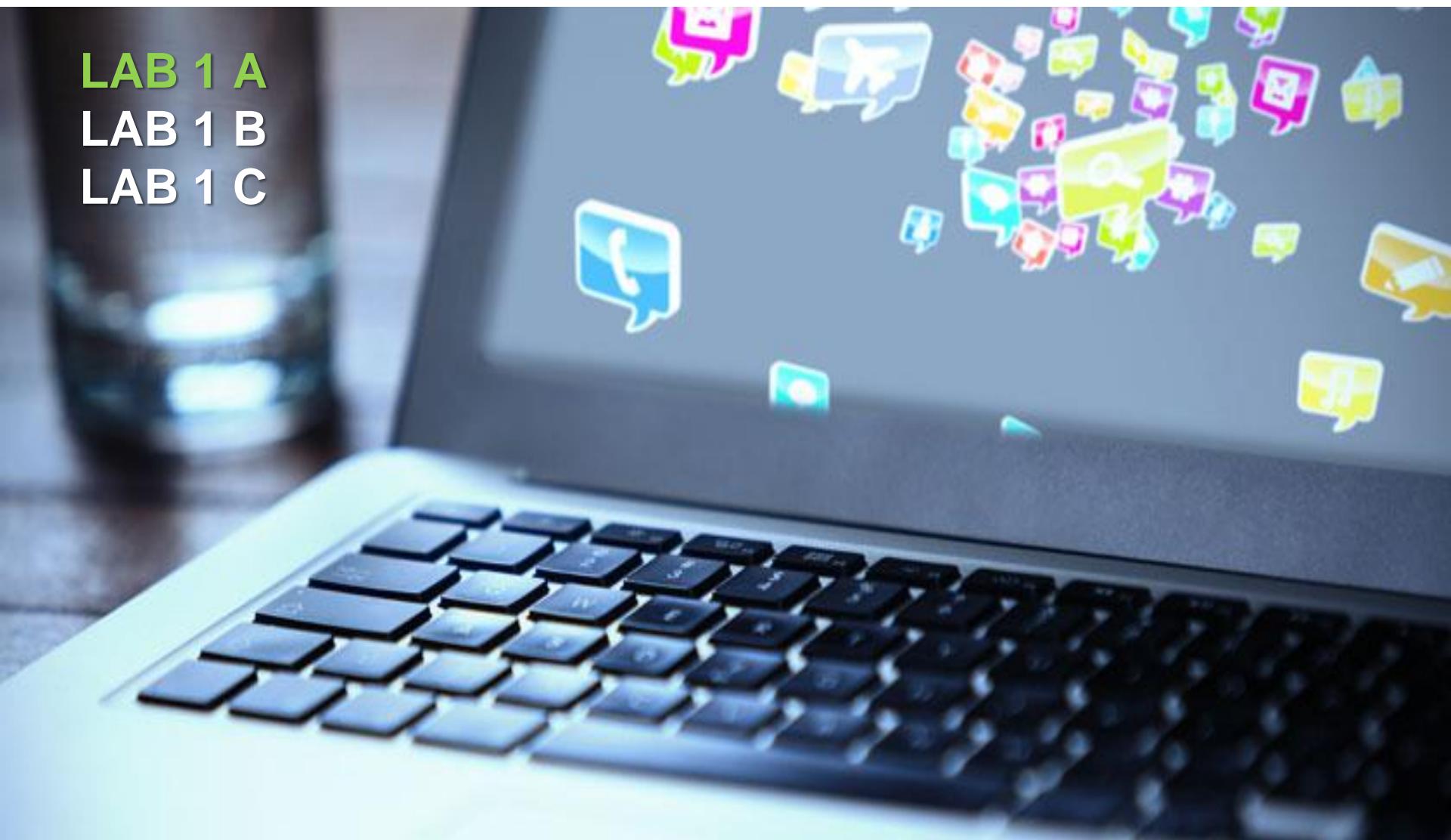
FIAP

Ferramenta
colaborativa para
agilidade

bit.ly/slackshift



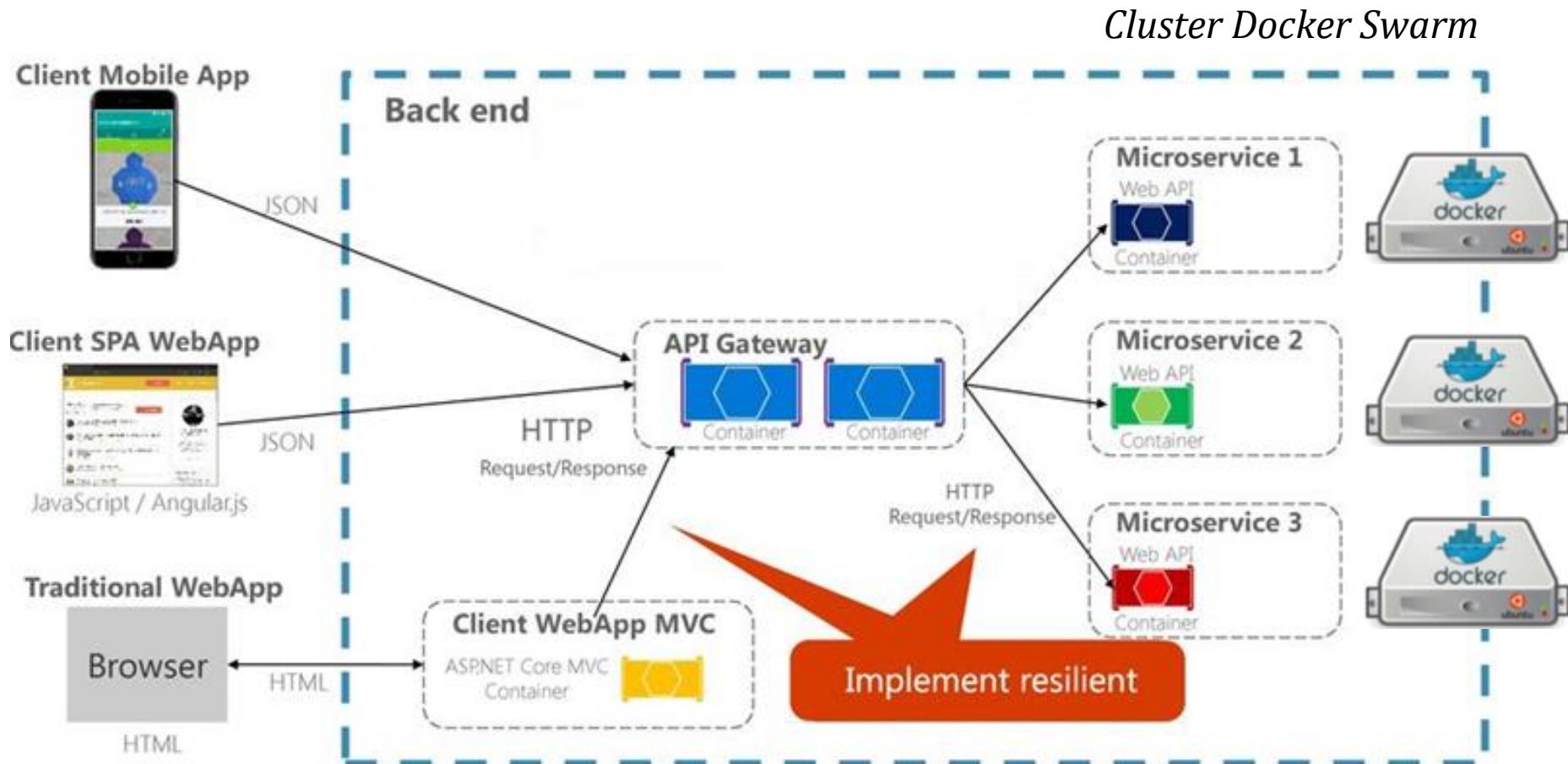
LAB 1 A
LAB 1 B
LAB 1 C



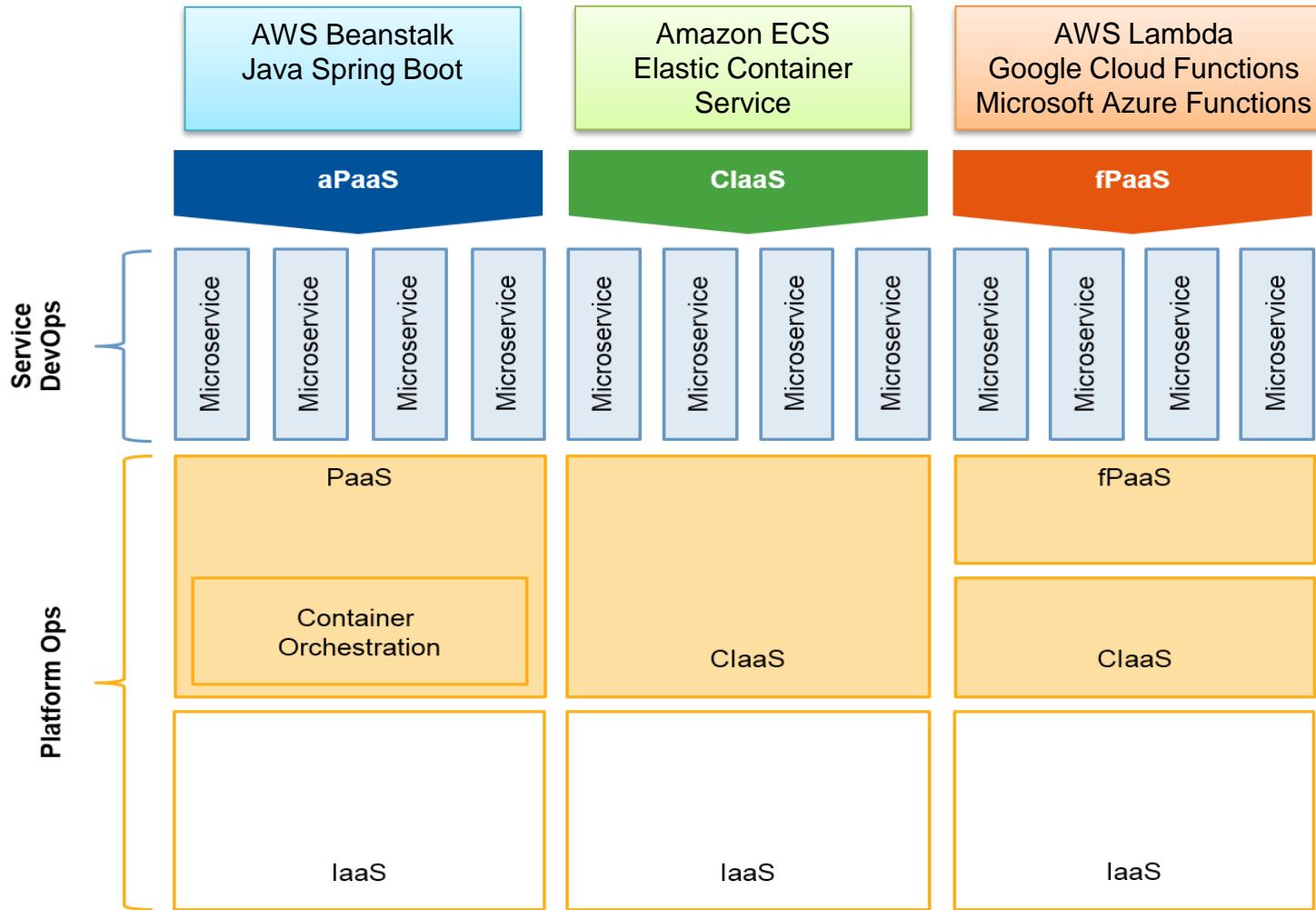
Gestão de Microserviços

FIAP

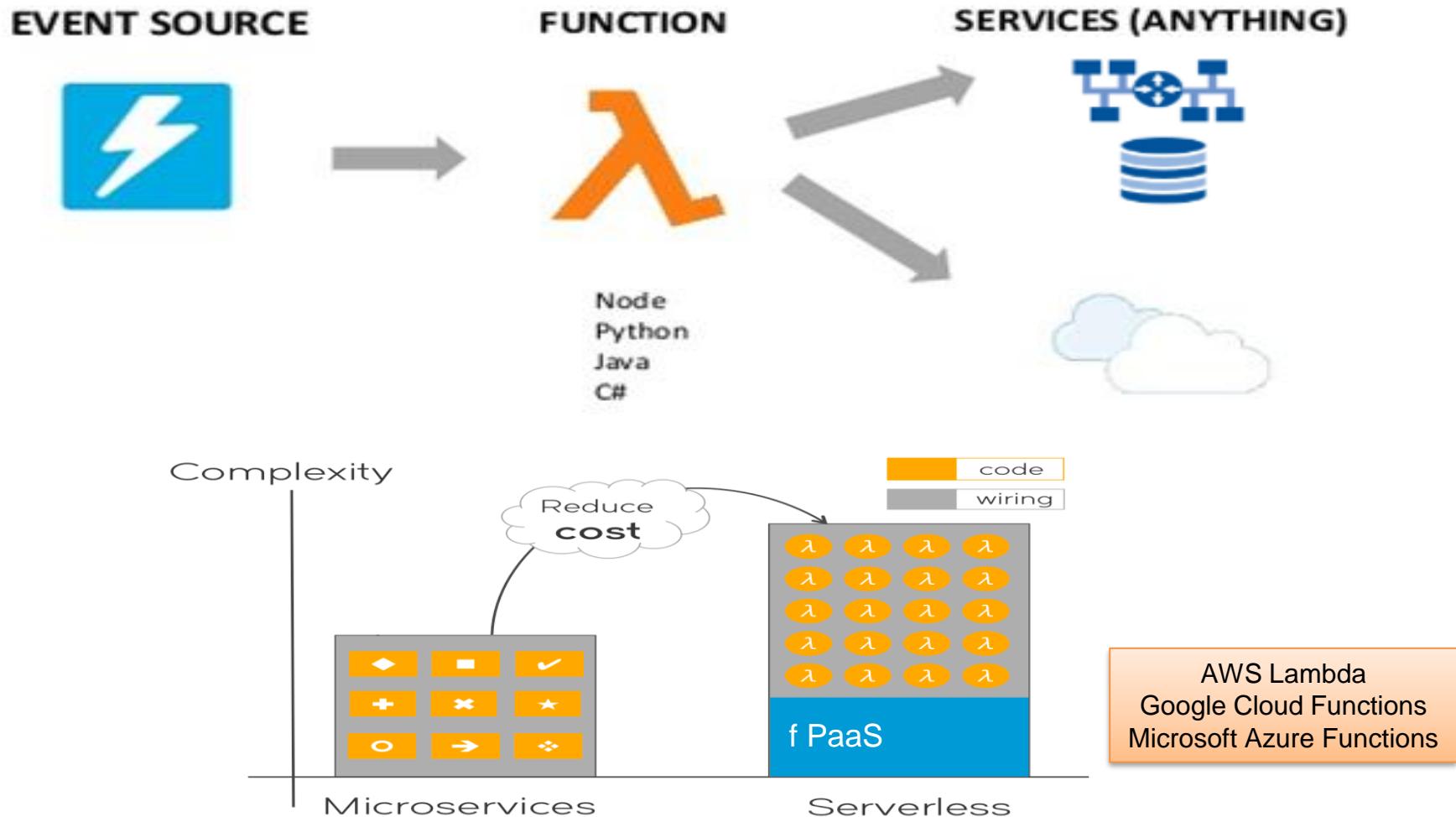
Hospedando nosso COMÉRCIO DIGITAL em um Cluster



Novas Plataformas : Aplicação X Conteineres X Functions



Arquitetura sem servidores (Serverless)



Evolução de Microserviços

FIAP

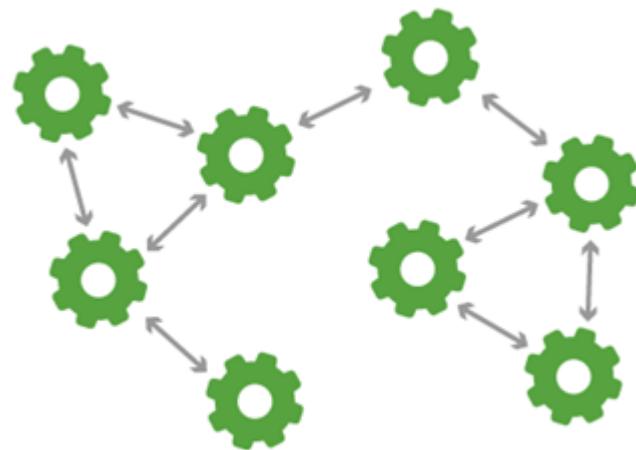
2000's SERVICE ORIENTED ARCHITECTURE



SOA:

Aplicações em componentes que se comunicam por ESB

2010's MICROSERVICES ARCHITECTURE



MICROSERVIÇOS:

Aplicações se comunicam de forma independente entre si

Governança SOA

Garante que a estratégia seja implementada com sucesso

Evita duplicidade e aumenta o reuso nos projetos

Mantém e garante padrões arquiteturais

Auxilia a análise de impacto nas mudanças organizacionais

SOA

ESB pode se tornar um ponto único de falha que afeta todo o sistema

Orquestração dos formatos pelo ESB

Governança e padrões comuns

Microserviços

Se um microserviço falhar, apenas ele será afetado

Usam APIs HTTP REST leves para se comunicarem entre si

Forte foco em DevOps e entrega contínua

SOA

Popular com plataforma on-premise para serviços implantados

Compartilham o armazenamento de dados

Adequada para ambientes complexos e integração de aplicativos

Microserviços

Uso de plataformas em nuvem com contêineres

Cada microsserviço pode ter seu BD independente

Adequado para sistemas modernos baseados na Web



**O pivô das mudanças:
TRANSFORMAÇÃO DIGITAL**

Transformação Digital



Inovação →	API	Respostas mais rápidas
Custos →	Speedometer	Ganhos econômicos em escala global
Agilidade →	Clouds	Elasticidade e flexibilidade das nuvens
Canais padronizados →	Mobile device	Plataformas abertas, seguras e flexíveis

Padrões de projetos modernos

Padrões de projetos podem ser utilizados em qualquer plataforma

Melhores práticas para resolver problemas comuns

Combinam componentes em arquitetura de alto nível

Componentes mais flexíveis e fáceis de manter



Padrões de Arquitetura

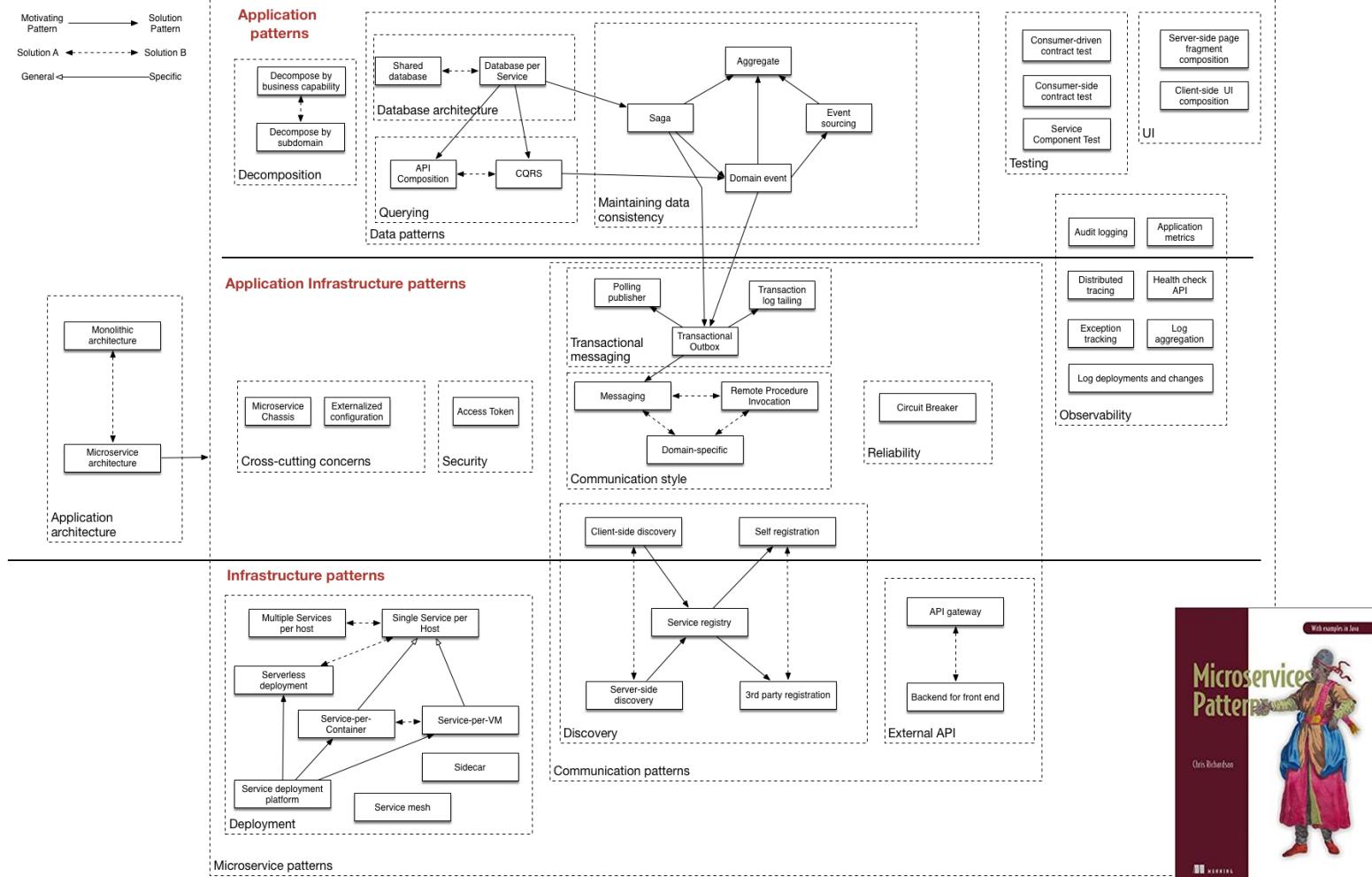
FIAP

Arquitetura Front-end

Arquitetura de Aplicações

Infraestrutura de Aplicações

Arquitetura de Infraestrutura



Arquitetura Front-end

Arquitetura de Aplicações

Infraestrutura de Aplicações

Arquitetura de Infraestrutura

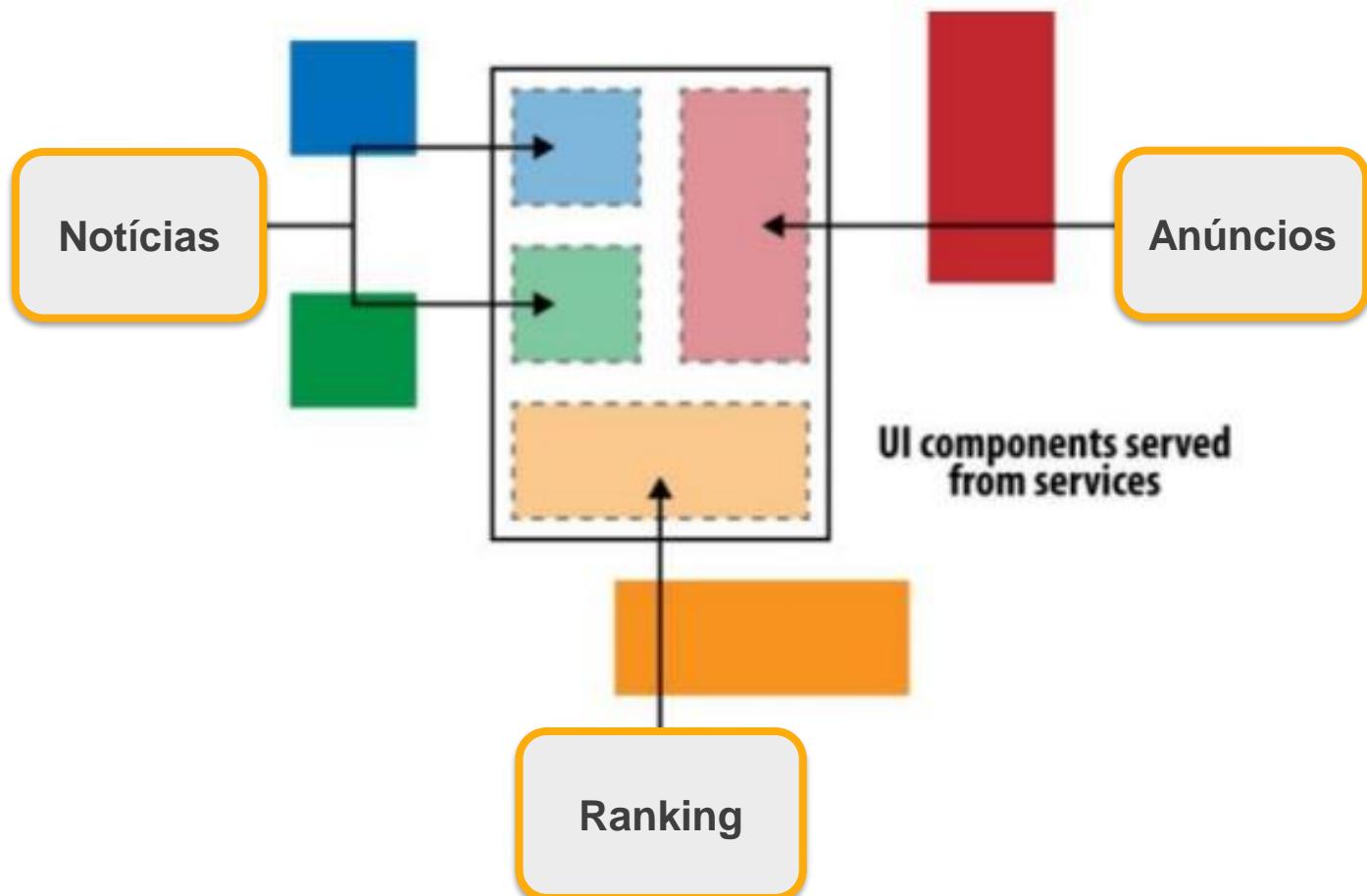
Interface do Usuário

Server-side page fragment composition

Client-side UI composition

UI

Componentes da página: Servidor X Cliente



Arquitetura Front-end

Arquitetura de Aplicações

Infraestrutura de Aplicações

Arquitetura de Infraestrutura

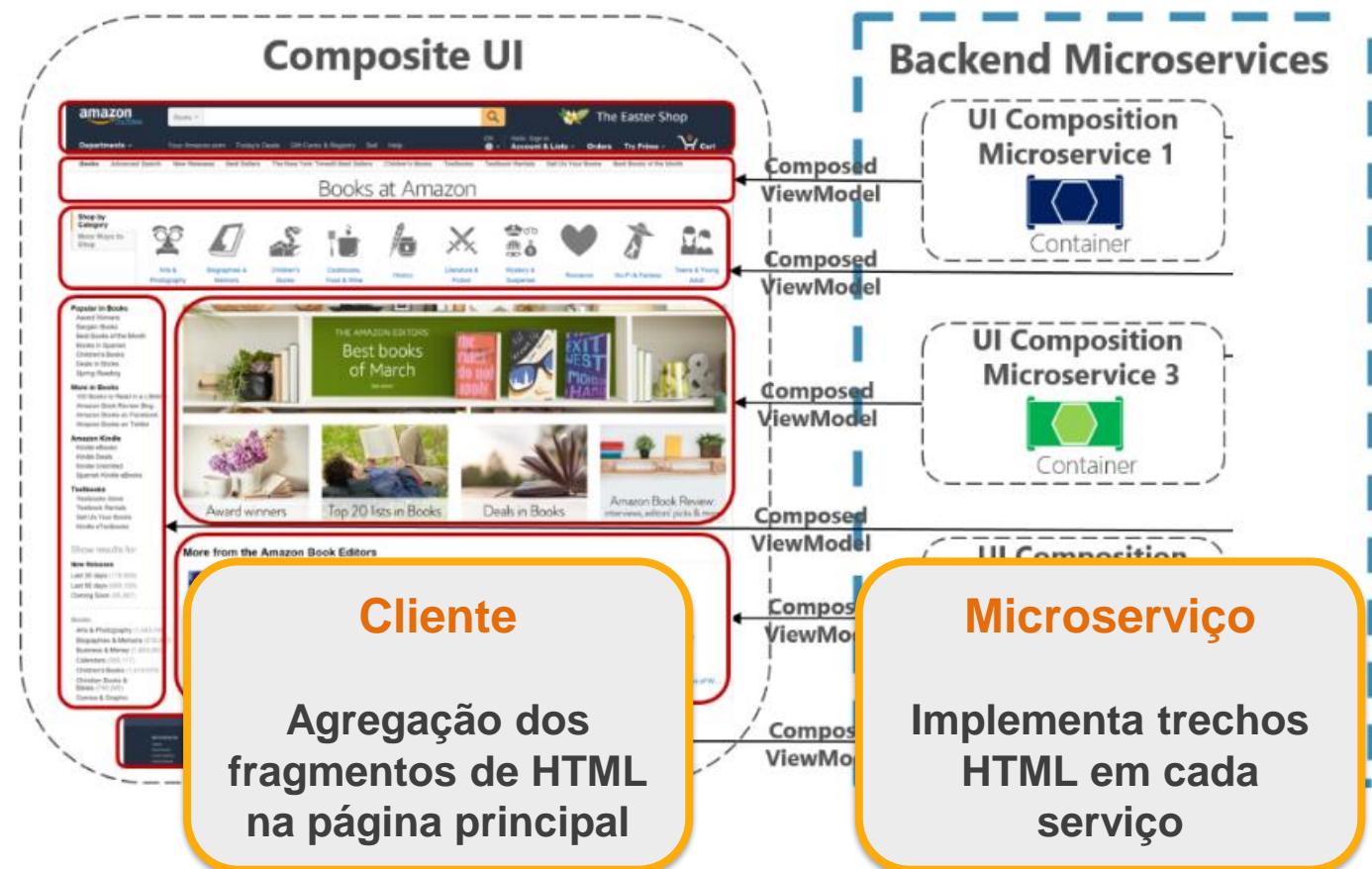
Interface do Usuário

Server-side page fragment composition

Client-side UI composition

UI

Servidor: envia ao cliente trechos HTML



Arquitetura
Front-end

Arquitetura de
Aplicações

Infraestrutura de
Aplicações

Arquitetura de
Infraestrutura

Interface do
Usuário

Server-side page
fragment
composition

Client-side UI
composition

UI

Cliente: monta e exibe toda a página
ex: Progressive Web App



Semelhante a
aplicativos

Funciona
off-line com
cache local

Acessado de
qualquer
dispositivo e
navegador

Supora
mensagens
push, HTTPS e
atualização

Arquitetura
Front-end

Arquitetura de
Aplicações

Infraestrutura de
Aplicações

Arquitetura de
Infraestrutura

Interface do
Usuário

Server-side page
fragment
composition

Client-side UI
composition

UI

Componentes da página

Problema

Como criar
interfaces e
telas
flexíveis?

Solução

Modelo de
página
preenchido por
trechos
recebidos do
servidor
pelo próprio
cliente

Vantagens

Evitar o
acoplamento
entre os
componentes

Melhor
desempenho
e uso de
cache

Vamos nos Divertir!

FIAP

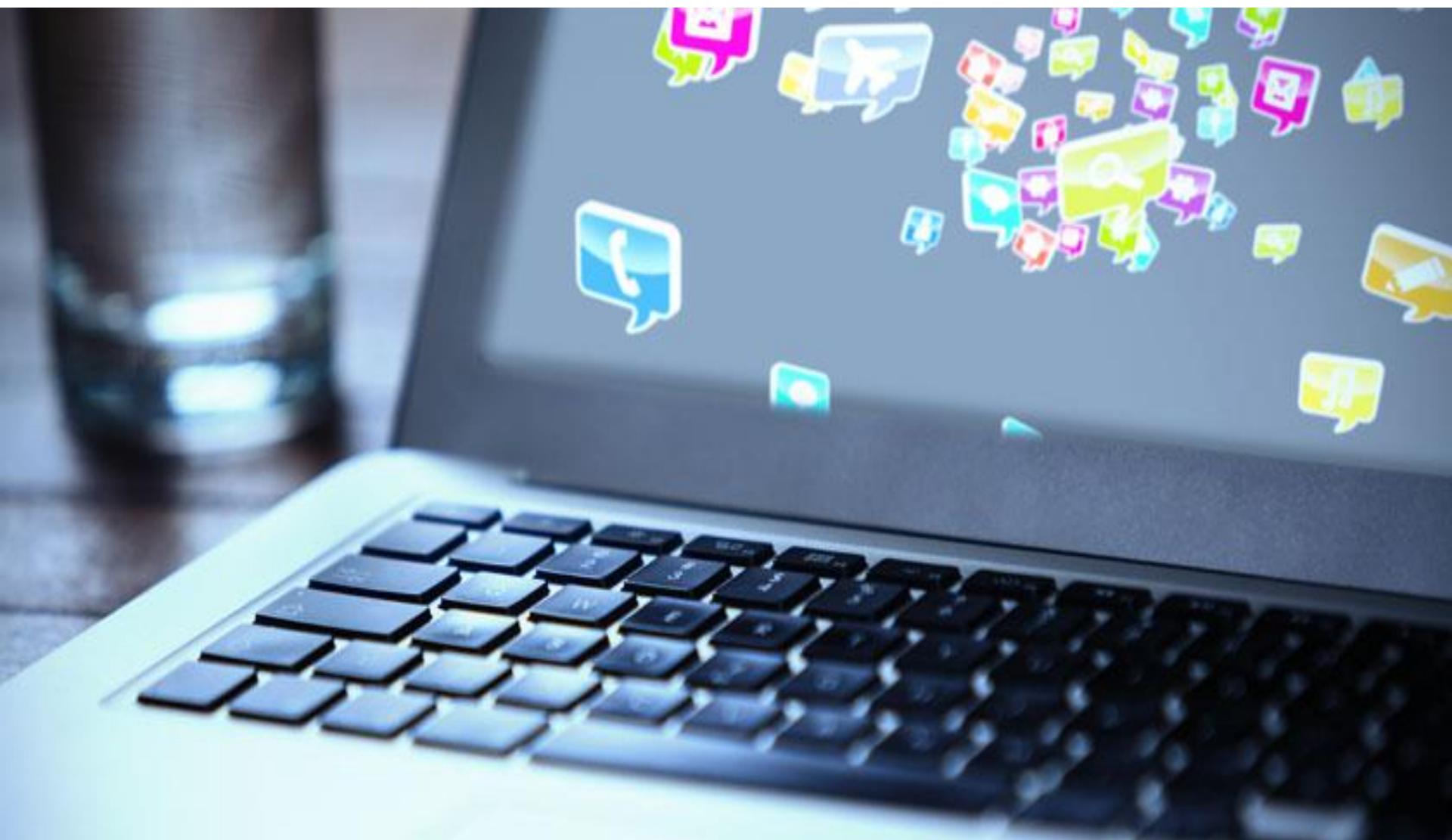


Vamos nos Divertir!

FIAP

1. Kahoot.it ou instalar o APP !
2. Baixar o volume do Celular
3. Ao entrar: clicar “Next”
4. Escolher perfil “As a **Student**”
5. Idade > 18 ?
6. Clicar: ENTER PIN
7. Colocar Nome (Sobrenome)



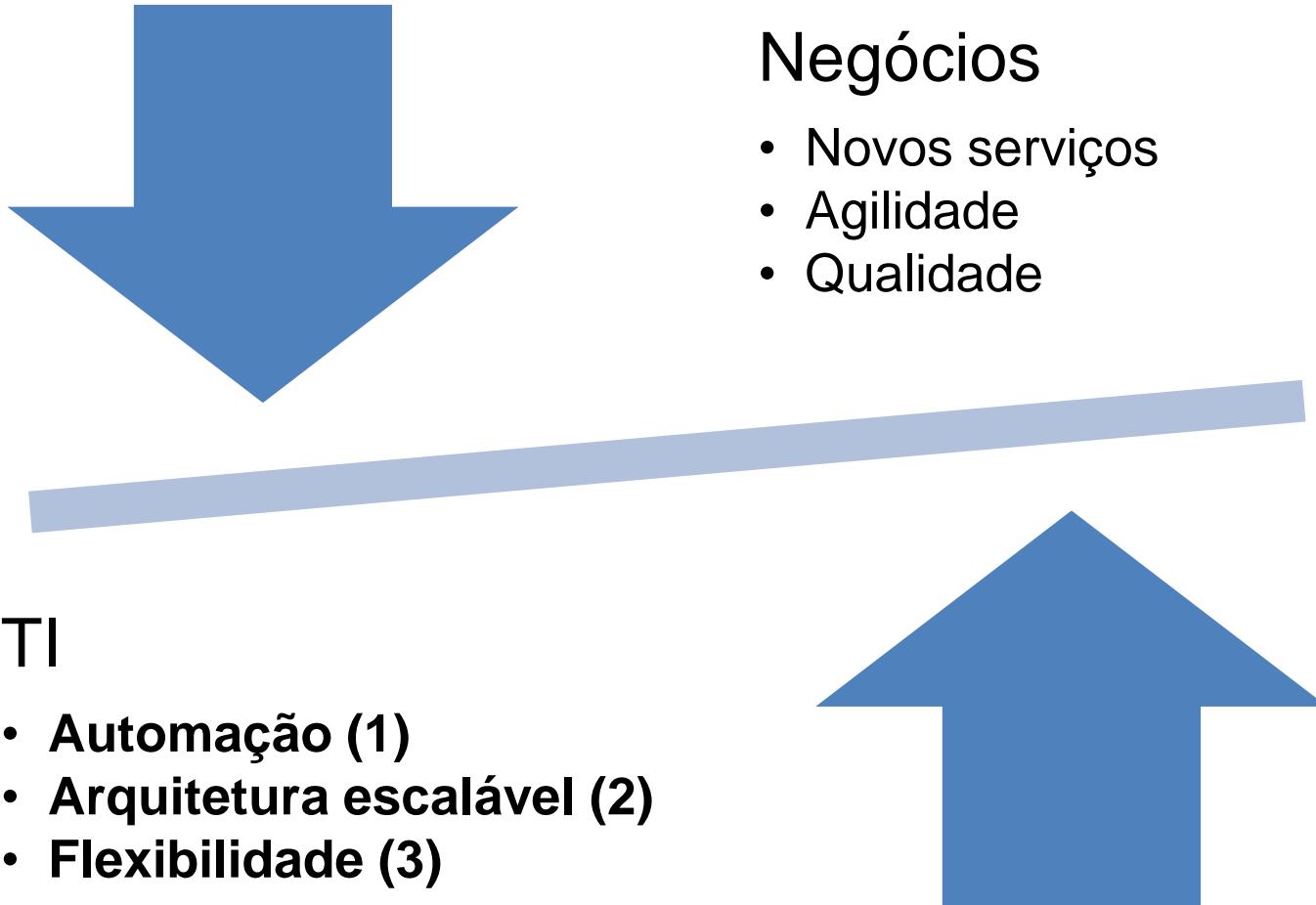


O que vimos até aqui:

- Microserviços (conceitos e cenários de uso)
- Ciclo de vida
- Agilidade com Microserviços
- Arquitetura de Projetos
 - Interface do Usuário

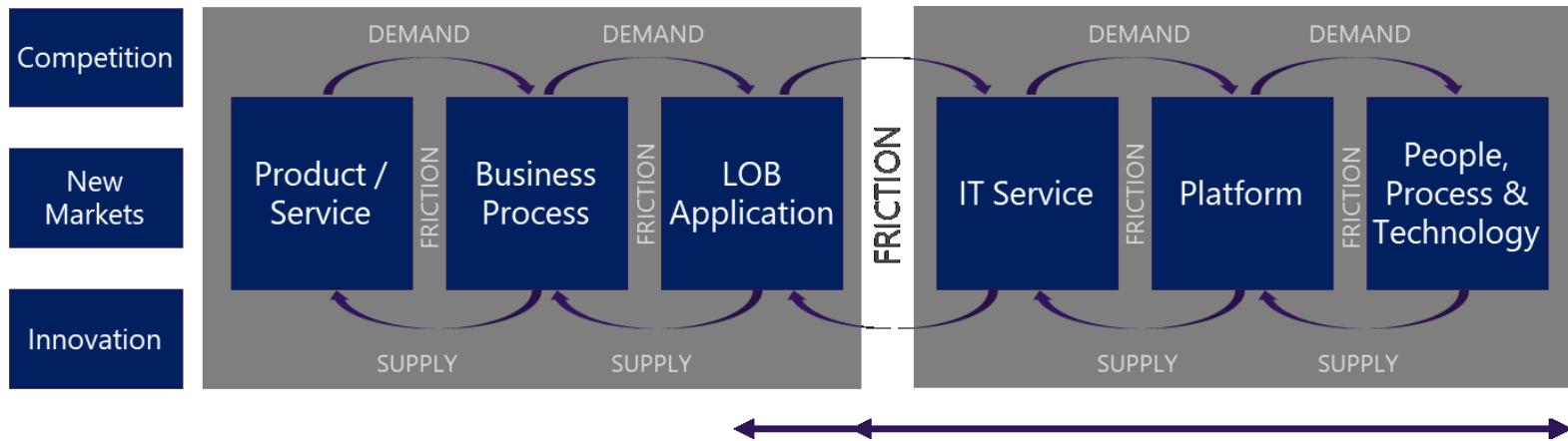
DESENVOLVENDO MICROSERVIÇOS

Gerenciamento ágil de microserviços

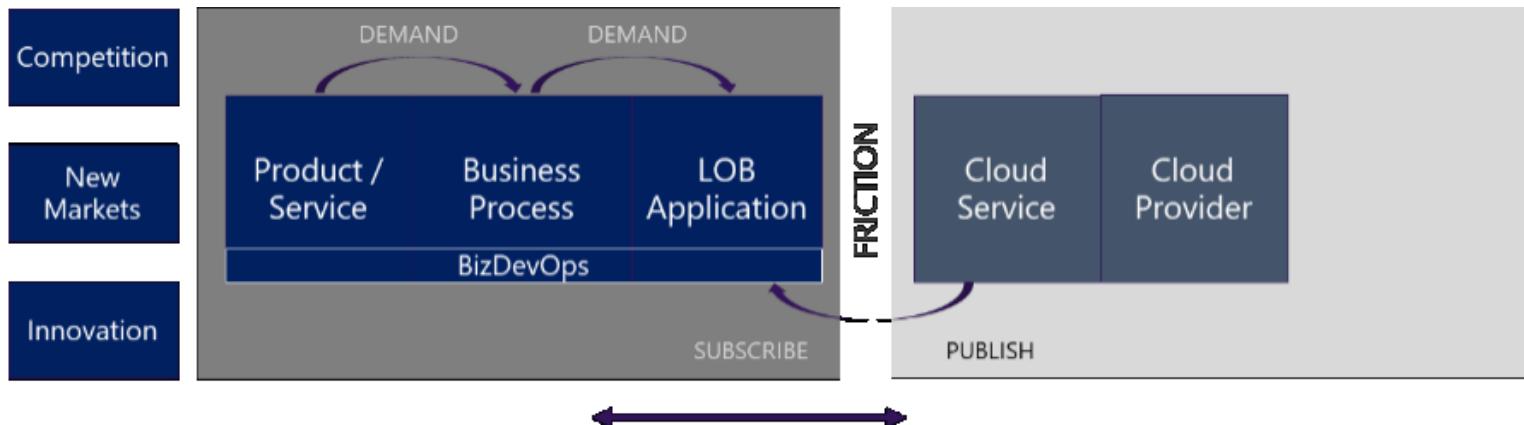


Ponto de atrito

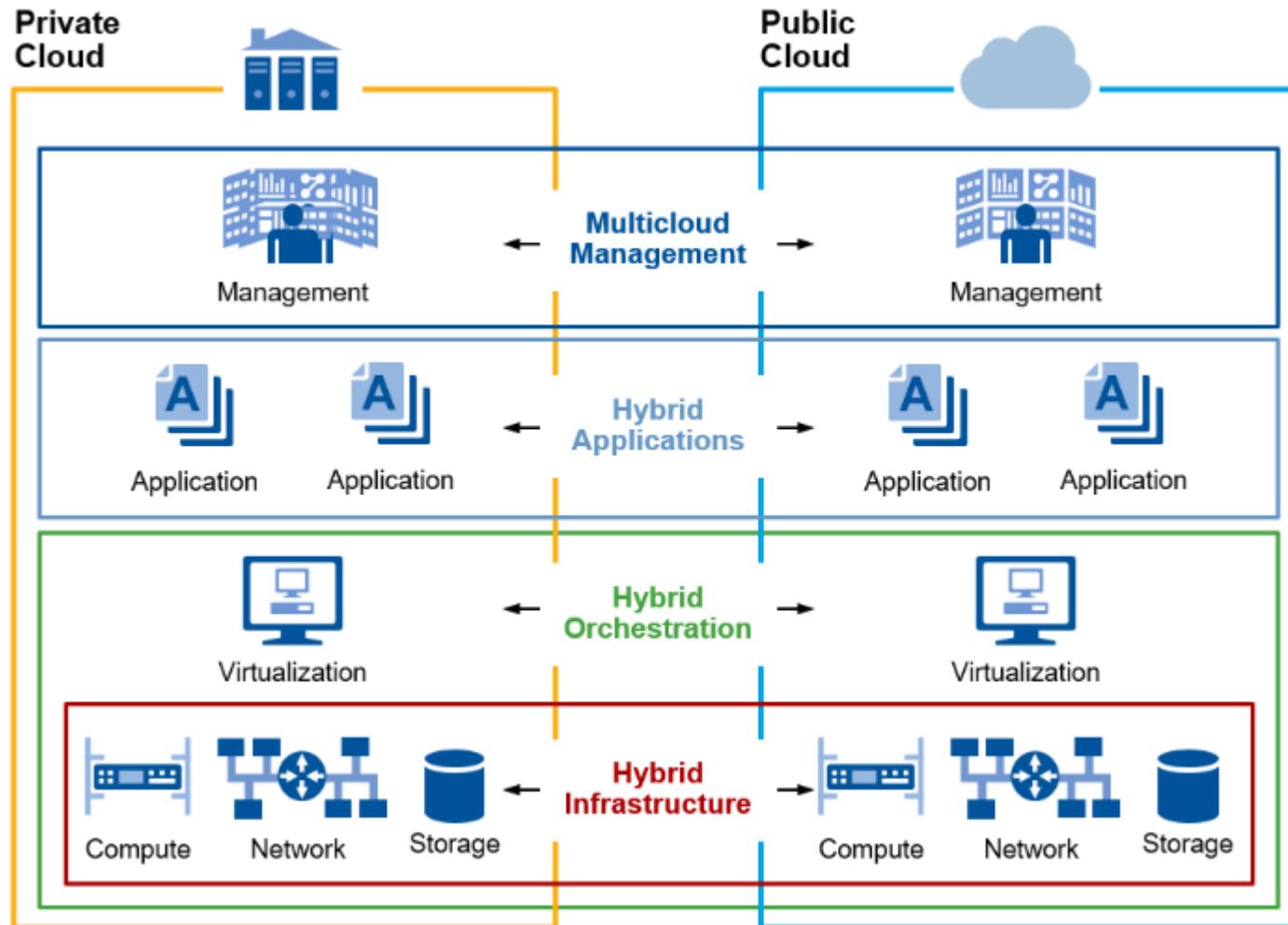
- Outsourcing clássico da TI



- Outsourcing ágil baseado em nuvem



Controles híbridos necessários na era da nuvem:



Padrões de Arquitetura

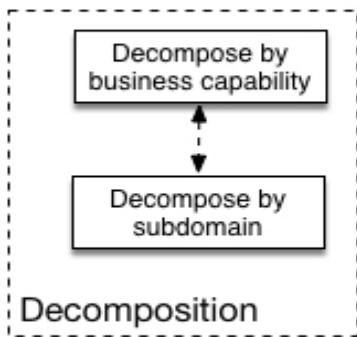
FIAP

Padrões de
Aplicativos

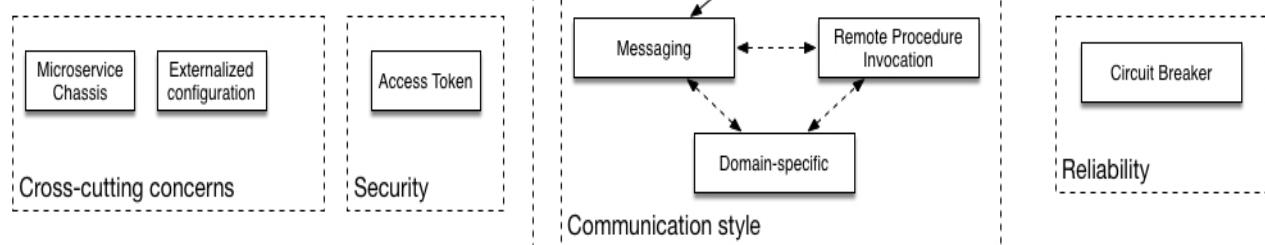
Infraestrutura de
Aplicações

Padrões de
Infraestrutura

Application
patterns



Application Infrastructure patterns



Padrões de Aplicativos

Infraestrutura de Aplicações

Padrões de Infraestrutura

Decomposição

Application patterns

Decompose by business capability

Decompose by subdomain

Decomposition

Relacionar serviços aos objetivos de negócios

Vendas e Pagamentos

Web
e-commerce

Mobile
e-commerce

Venda em
eventos

...

Loja física

Cobrança

NFe

Frete

Devolução

Produtos

Microserviços

Padrões de Aplicativos

Infraestrutura de Aplicações

Padrões de Infraestrutura

Decomposição

Application patterns

Decompose by business capability

Decompose by subdomain

Decomposition

Problema

Como decompor um Software em microserviços

Solução

Relacionar componentes:
Capacidades de Negócio
Subdomínios

Vantagens

Alterações na regra de negócios afetam um único serviço

Padrões de Arquitetura

FIAP

Padrões de Aplicativos

Infraestrutura de Aplicações

Padrões de Infraestrutura

Decomposição

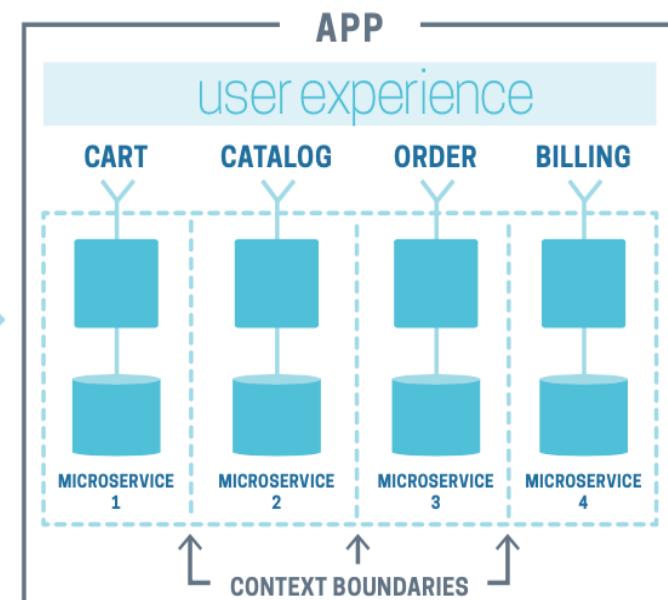
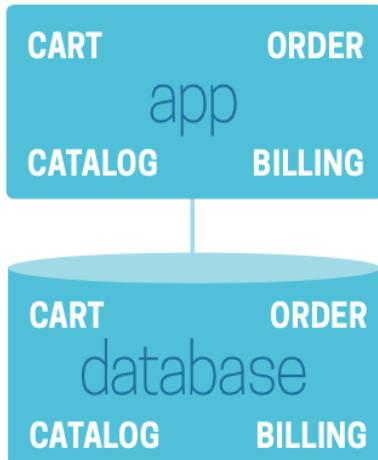
Application patterns

Decompose by business capability

Decompose by subdomain

Decomposition

Relacionar serviços aos subdomínios DDD



Padrões de Arquitetura

FIAP

Padrões de Aplicativos

Infraestrutura de Aplicações

Padrões de Infraestrutura

Decomposição

Application patterns

Decompose by business capability

Decompose by subdomain

Decomposition

Problema

Como decompor um Software em microserviços

Solução

Componentes baseados em Domain Driver Design (DDD):
Core
Suporte Genérico

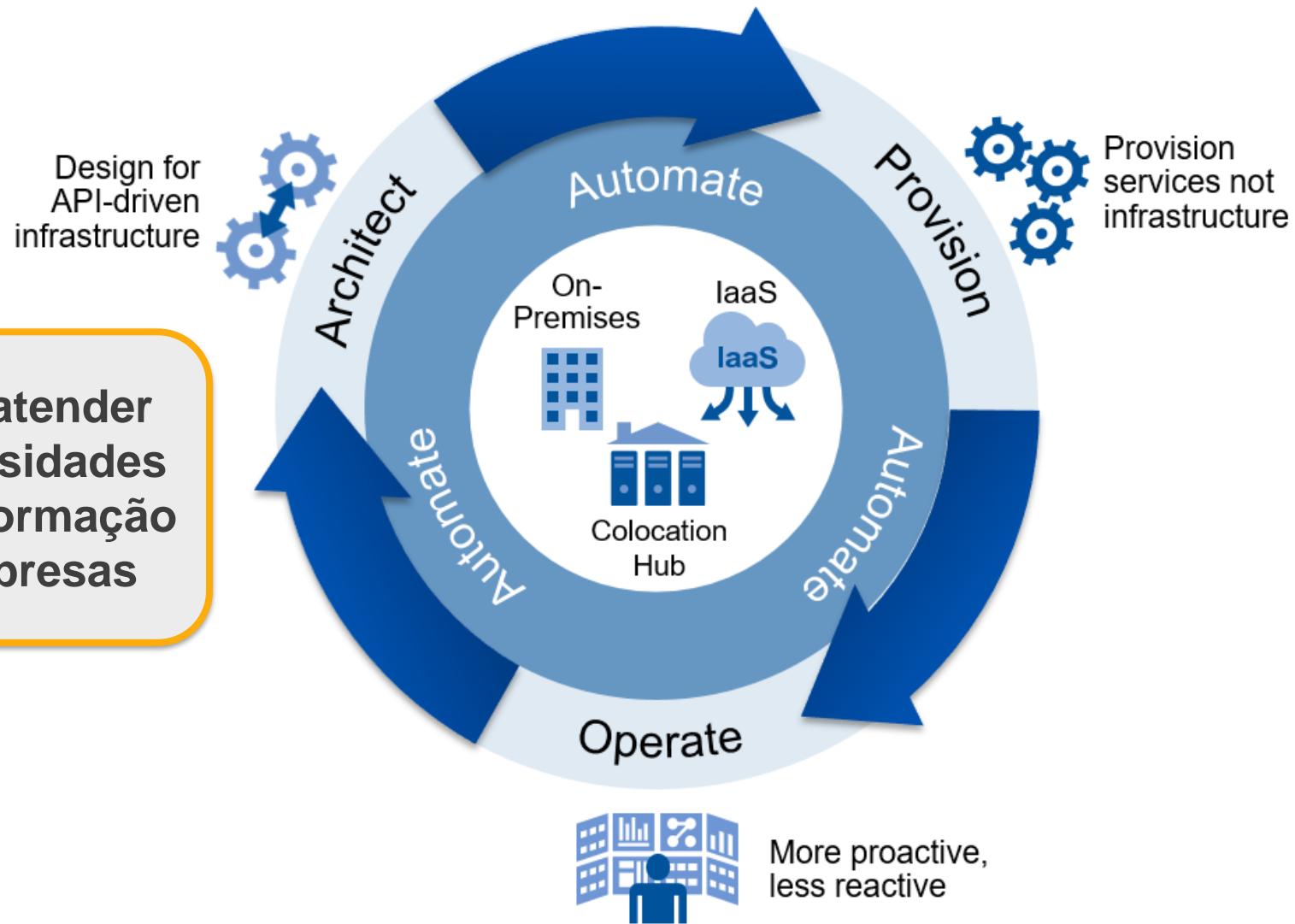
Vantagens

Arquitetura estável já que os subdomínios mudam pouco

Serviços coesos

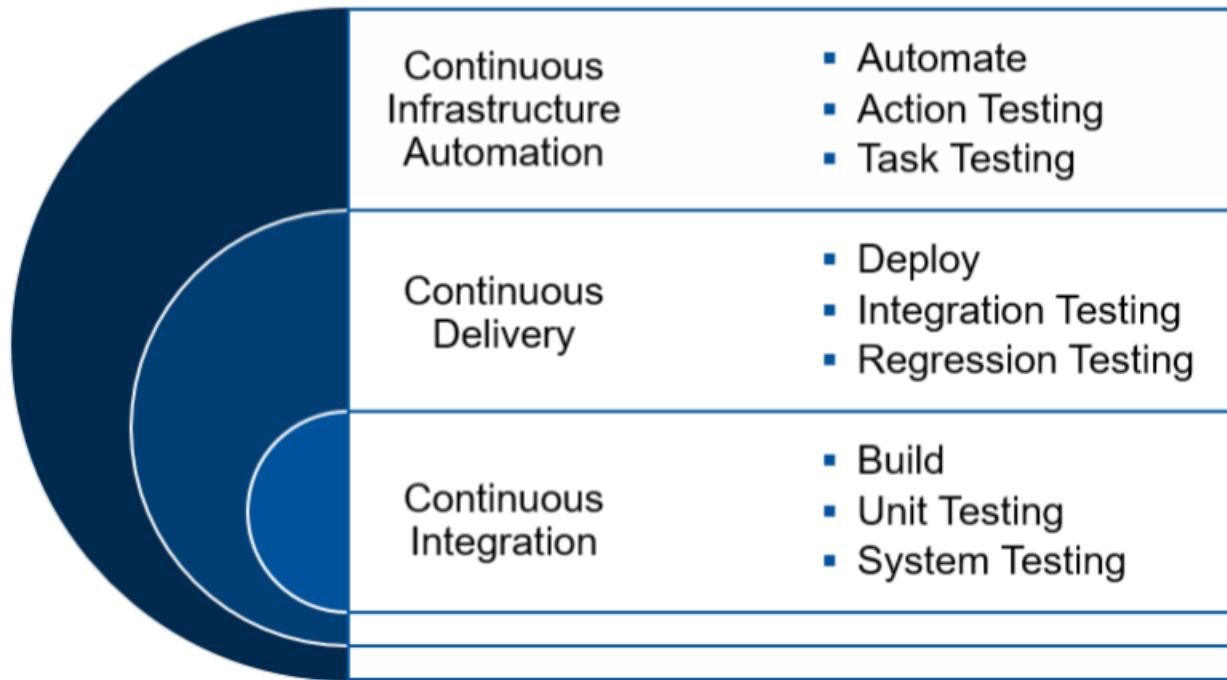
Decompor serviços em subdomínios DDD

Ambientes e necessidades modernas



Automação : acelerando a inovação

Continuous Infrastructure Automation vs. CI/CD

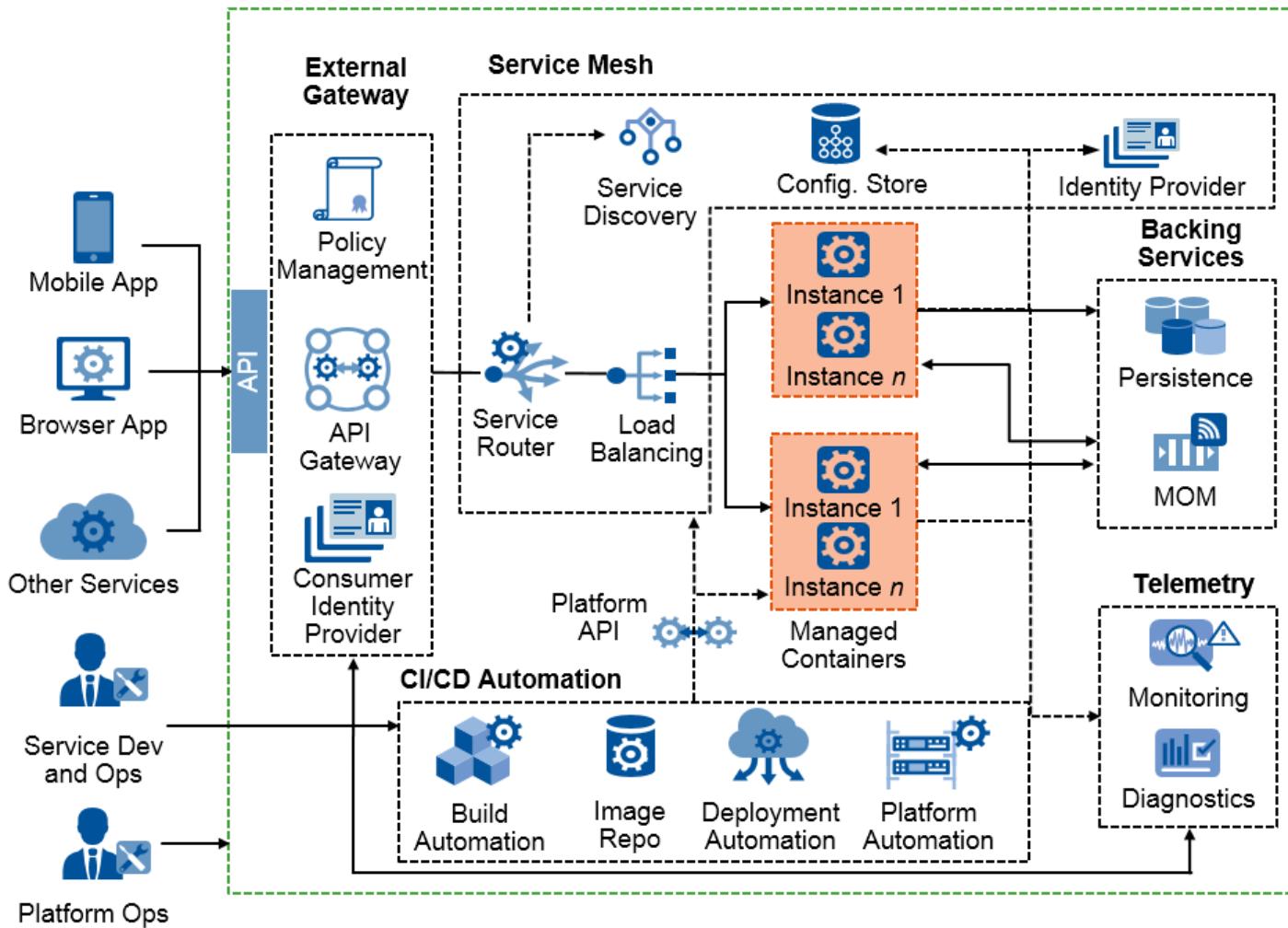


- # Instalações disparadas ao mesmo tempo
- # Repositórios utilizados para inspeção e verificação

Desenvolvendo Microserviços

FIAP

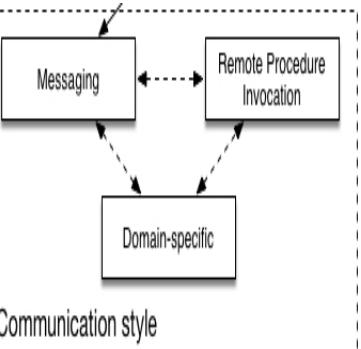
Visão geral da Arquitetura de Microserviços



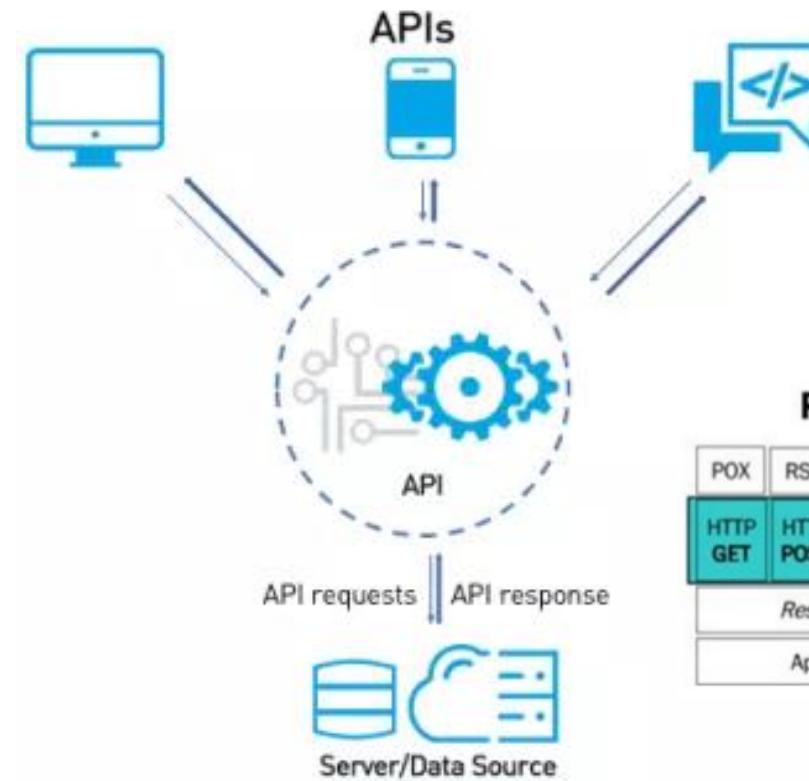
Infraestrutura de Aplicações

Padrões de Infraestrutura

Comunicação



RPI: comunicação síncrona via request/response



REST

POX	RSS	JSON	...
HTTP GET	HTTP POST	HTTP PUT	HTTP DEL
Resource URI			
Application			

SOAP

SOAP (WS-*)
SMTP
HTTP POST
MQ...

Endpoint URI
Application

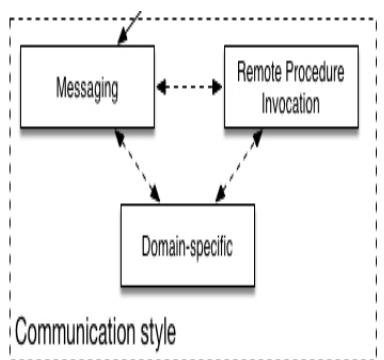
Padrões de Arquitetura

FIAP

Infraestrutura de Aplicações

Padrões de Infraestrutura

Comunicação



RPI: comunicação síncrona via request/response

Problema

Troca de informações que precisam de **resposta imediata**

Solução

RPC: chamada executada remotamente

REST: chamada de API utilizando **verbos HTTP**

Vantagens

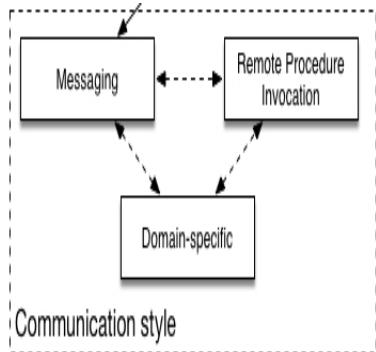
RPC: mais **fácil de desenvolver e executar**

REST: **baixo acoplamento e melhor desempenho**

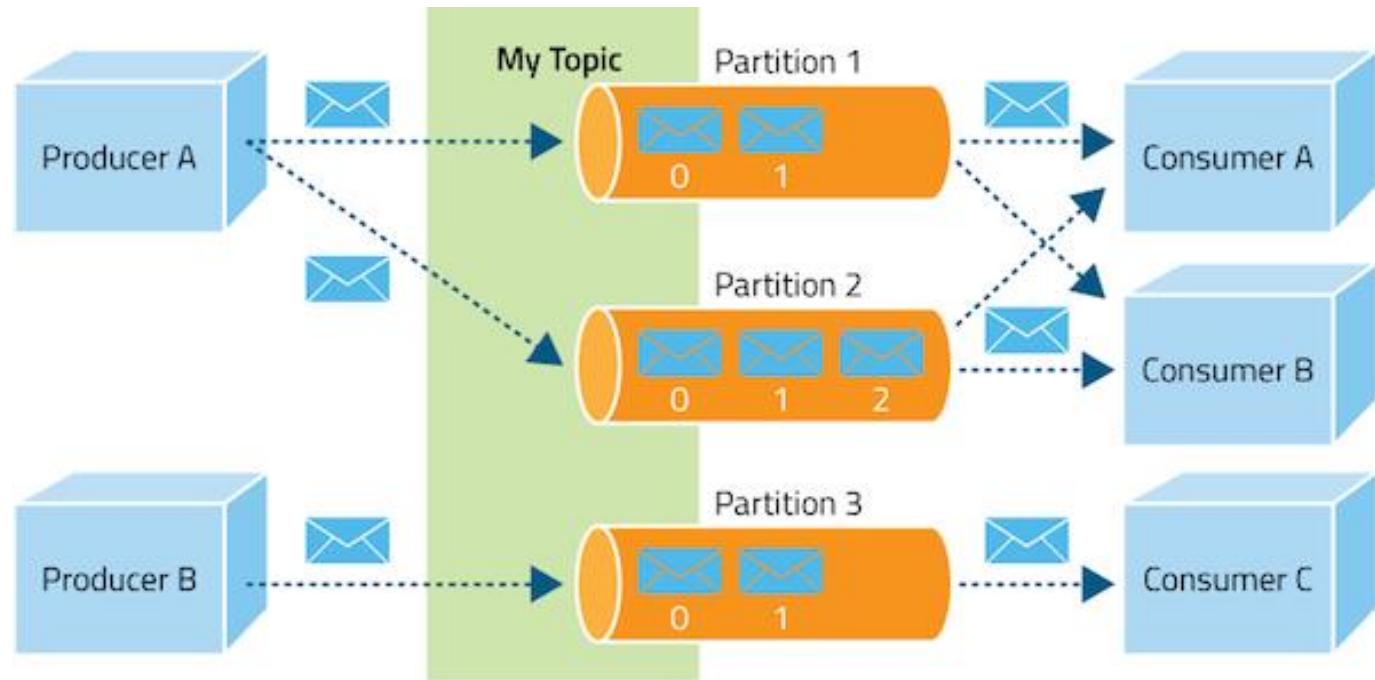
Infraestrutura de Aplicações

Padrões de Infraestrutura

Comunicação



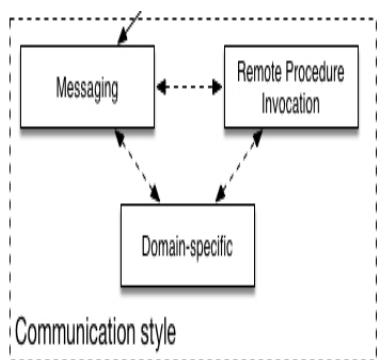
Mensagens: comunicação assíncrona entre serviços



Infraestrutura de Aplicações

Padrões de Infraestrutura

Comunicação



Mensagens: comunicação assíncrona entre serviços

Problema

Transações síncronas podem impactar alguns casos de uso

Solução

Envio de mensagens assíncronas entre os componentes do serviço

Vantagens

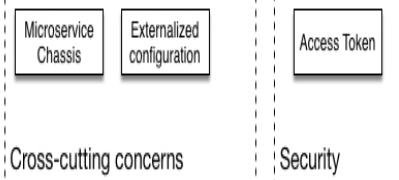
Melhorar a experiência do usuário

Possibilita a comunicação baseada em eventos

Infraestrutura de Aplicações

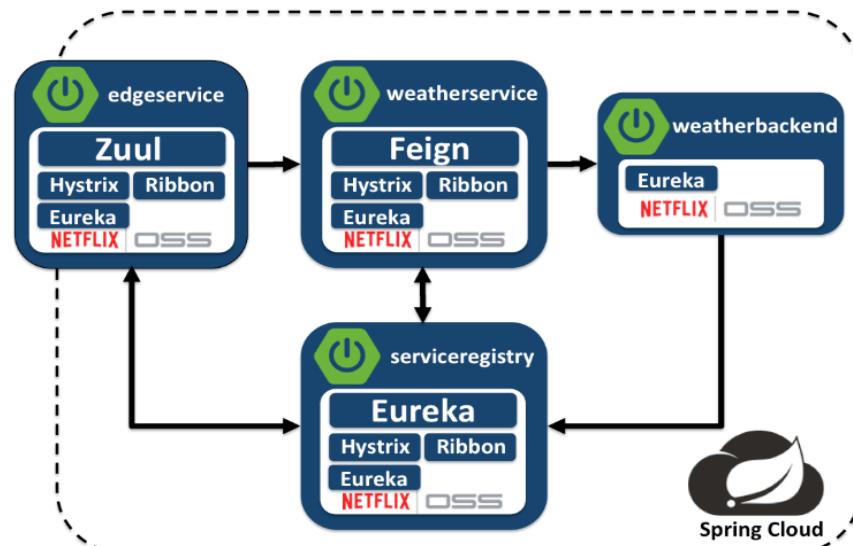
Padrões de Infraestrutura

Configurações comuns



Chassi: estrutura de suporte para outros componentes

Framework/Componente	Função
Spring Boot	Java Core framework
Spring Cloud Netflix (Eureka)	Registration & Discovery
slf4j and Logback	Logging
Spring Cloud Sleuth + Zipkin	Monitoring

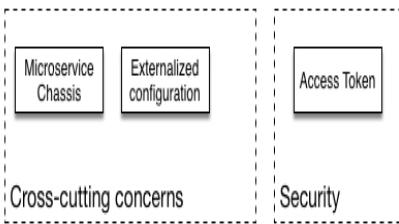


Padrões de Arquitetura

FIAP

Infraestrutura de Aplicações

Configurações comuns



Padrões de Infraestrutura

Chassi: estrutura de suporte para outros componentes

Problema

Tempo gasto
em configurações secundárias

logging,
autenticação,
monitoração

Solução

Criação de arquitetura reutilizável para economizar tempo em **requisitos não funcionais**

Vantagens

Consistência entre as equipes

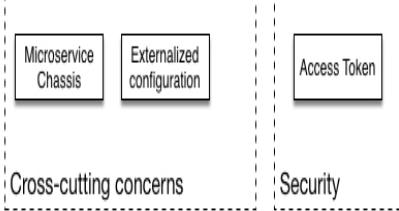
Reuso das mesmas características entre serviços

Padrões de Arquitetura

FIAP

Infraestrutura de Aplicações

Configurações comuns



Padrões de Infraestrutura

Configurações externas: ex: variáveis de ambiente e docker-compose.yml

Configurações específicas de ambiente (DEV/QA/PROD)

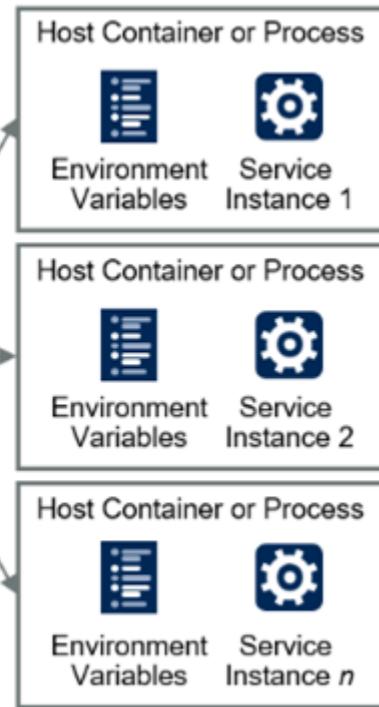
```
"SERVICE_DB": "prod.db.internal.domain:27017"  
"EVENT_BROKER": "prod.evt.internal.domain:9093"  
"EVENT_PUBLISH_TOPIC": "service_updates"
```



Configuration Distribution and Replication



```
version: '3'  
services:  
  web:  
    build: .  
    ports:  
      - "5000:5000"
```



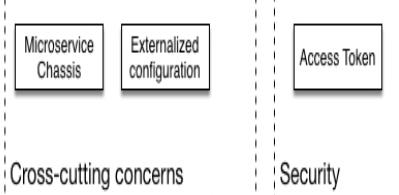
Padrões de Arquitetura

FIAP

Infraestrutura de Aplicações

Configurações comuns

Padrões de Infraestrutura



Problema

Como possibilitar a **execução** do serviço em **qualquer ambiente** sem modificação?

Solução

Serviço **efetua sua configuração** de acordo com uma **configuração externa**

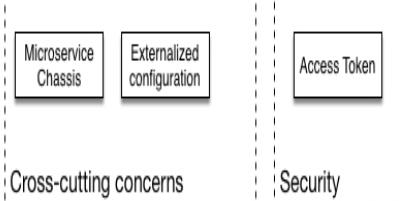
Vantagens

Consistência entre os diversos ambientes

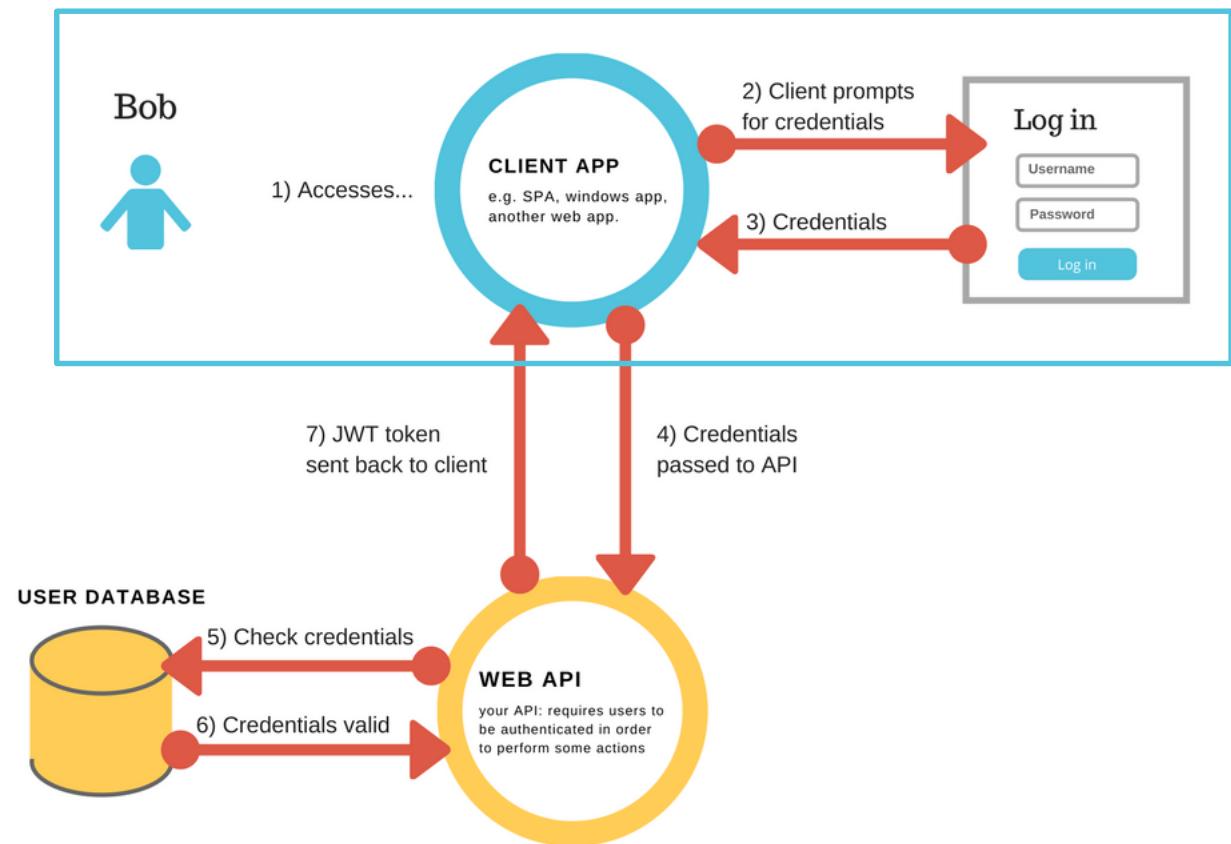
Infraestrutura de Aplicações

Padrões de Infraestrutura

Segurança



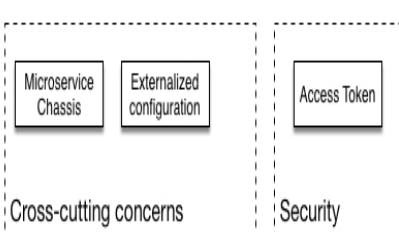
Token de acesso: autenticação



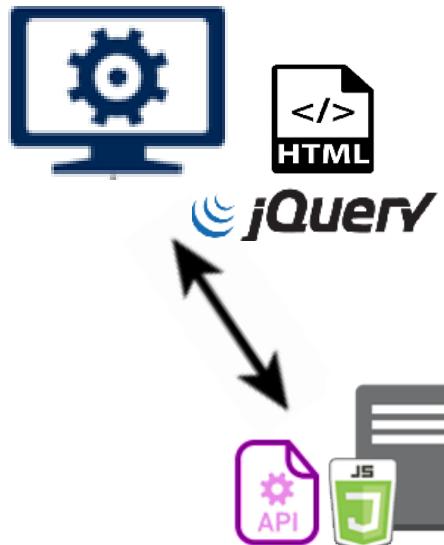
Infraestrutura de Aplicações

Padrões de Infraestrutura

Segurança



CORS: Cross-Origin Resource Sharing



Bloqueia

domínio diferente
url.com → chama → api.com

subdomínio diferente
url.com → chama → api.url.com

porta diferente
url.com → chama → url.com:3001

protocolo diferente
<https://url.com> → <http://url.com>

GET
OPTIONS
X
RESPONSE
HEADER

```
HTTP/1.1 200 OK
Date: Sat, 31 Jan 2015 23:05:44 GMT
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT, PATCH, OPTIONS
Access-Control-Allow-Headers: Content-Type, api_key, Authorization
Content-Type: application/json
Content-Length: 0
```

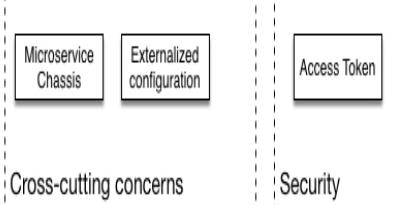
Padrões de Arquitetura

FIAP

Infraestrutura de Aplicações

Padrões de Infraestrutura

Segurança



Token de acesso: autenticação de identidade

Problema

Como autenticar a identidade do solicitante para acesso aos recursos?

Solução

Serviço verifica as credenciais e devolve um token para ser utilizado nas transações subsequentes

Vantagens

Verificação e controle de acesso para executar uma operação

Infraestrutura de Aplicações

Padrões de Infraestrutura

Confiabilidade

Circuit Breaker

Reliability

Resposta OK



Falhas ultrapassaram o limite definido



Testes com Fluxo controlado durante um período de tempo



Infraestrutura de Aplicações

Padrões de Infraestrutura

Confiabilidade

Circuit Breaker

Reliability

Problema

Como evitar que uma falha seja propagada para outros serviços?

Solução

Serviço remoto pode ser acessado por um **proxy** que **retenha e isole falhas**

Vantagens

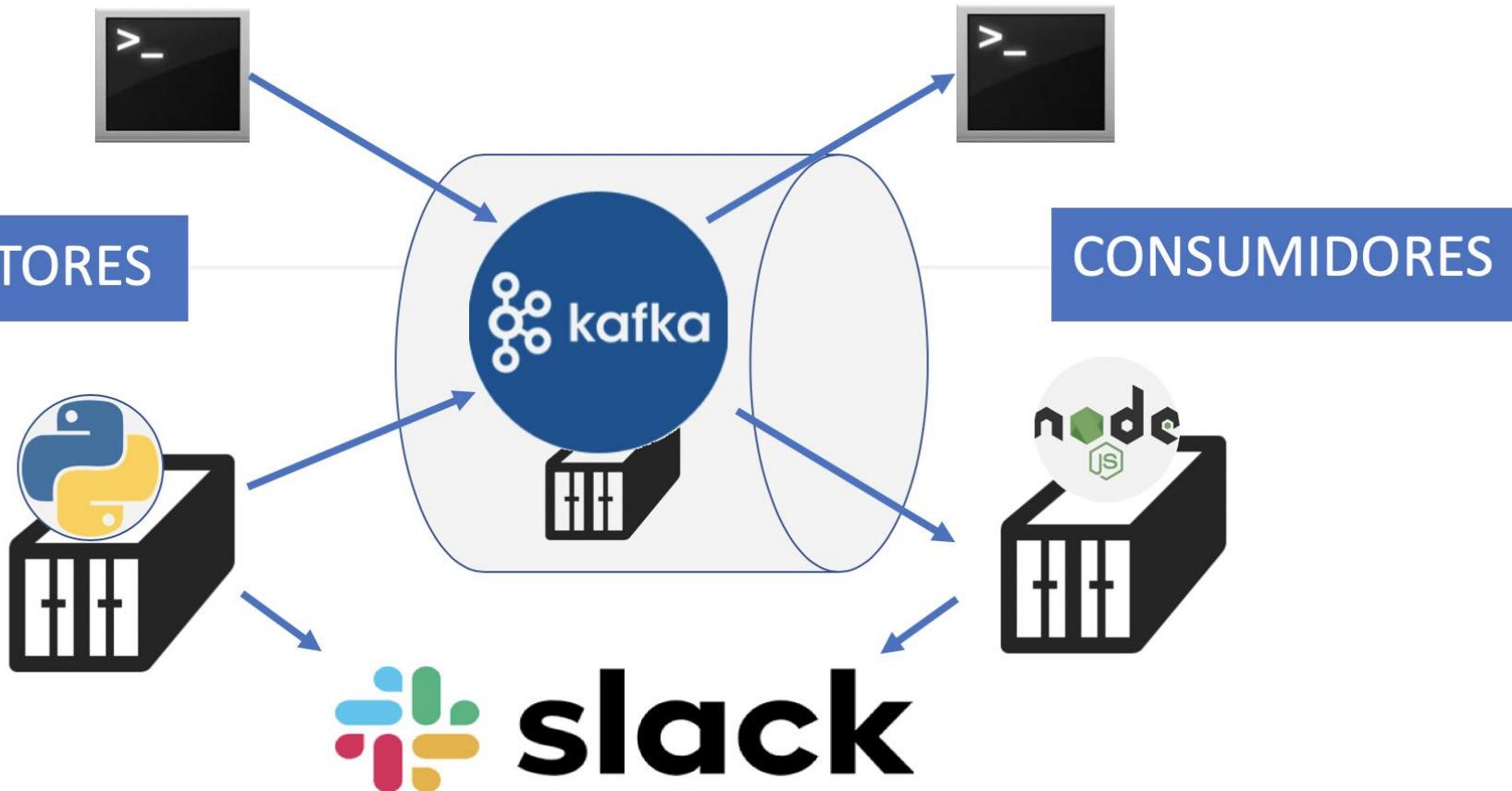
Mitigação de risco de falhas

Maior resiliência da solução

Circuit breaker: disjuntor para evitar falhas

LAB: TOPOLOGIA KAFKA

FIAP



```
mirror_mod = modifier_obj
# mirror object to mirror
mirror_mod.mirror_object =
    operation == "MIRROR_X":
        mirror_mod.use_x = True
        mirror_mod.use_y = False
        mirror_mod.use_z = False
    operation == "MIRROR_Y":
        mirror_mod.use_x = False
        mirror_mod.use_y = True
        mirror_mod.use_z = False
    operation == "MIRROR_Z":
        mirror_mod.use_x = False
        mirror_mod.use_y = False
        mirror_mod.use_z = True
```

```
#selection at the end -add
modifier_obj.select= 1
modifier_obj.select=1
context.scene.objects.active = modifier_obj
("Selected" + str(modifier_obj))
modifier_obj.select = 0
bpy.context.selected_objects = []
data.objects[one.name].select = 1
```

```
print("please select exactly one object")
- OPERATOR CLASSES -
```

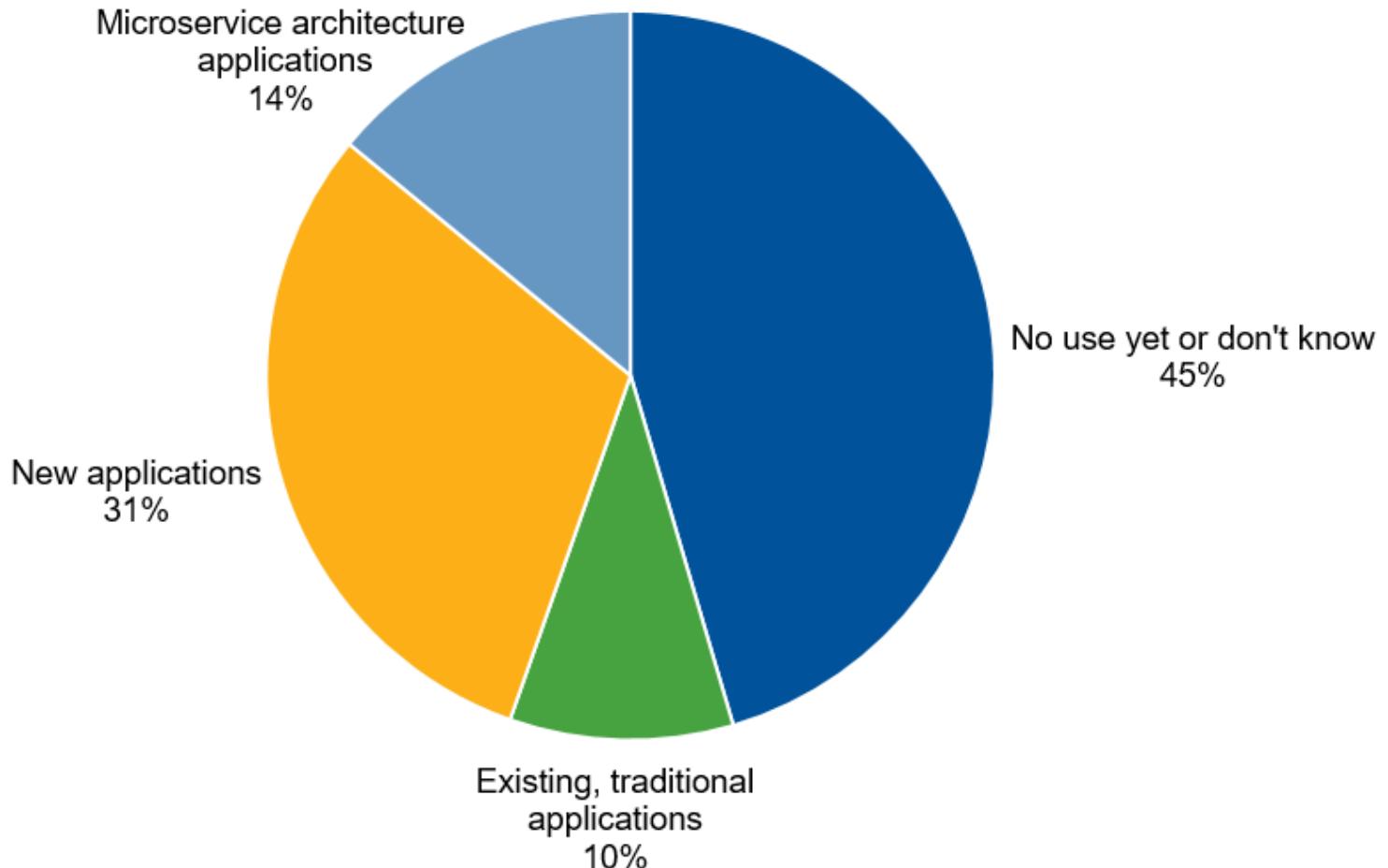
```
types.Operator):
    X mirror to the selected
    object.mirror_mirror_x"
    mirror X"
```

```
context):
    context.active_object is not None
```

Utilização de Contêineres

FIAP

Pesquisa realizada em 2017 sobre iniciativas utilizando tecnologias baseada em “Contêiner”:



Utilização de Contêineres

FIAP

“Containerização”

Registro de uma
imagem no Hub

Desenvolvimento
em máquina local

Implantação e
distribuição da
imagem

Execução
padronizada dos
aplicativos

Alta disponibilidade
em diversos nodes

Características:

Facilita as
práticas DevOps

Aplicações não
possuem dependências
do sistema

Para
dimensionamento
do aplicativo,
geram-se novos
executáveis (e não
criar hosts ou VMs)

Utilização de Microserviços em Docker

FIAP

Docker

Imagen portátil definida no Dockerfile

Permite reutilização de imagens prontas

Desnecessário instalar sistema operacional e demais dependências da aplicação

Imagen pode ser disponibilizada em um Registro (Docker Hub) para facilitar o deploy

Aplicação é executada da mesma forma independente do ambiente

Utilização de Microserviços em Docker

FIAP

Uso de contêiner está cada vez mais popular

Flexibilidade

Maioria das aplicações suportam **execução em contêiner**

Agilidade

Suportam updates com as aplicações em execução (**on the fly**)

Execução

Compartilham trechos do host para uma execução mais leve

Portabilidade

Possibilitam desenvolvimento local e **distribuição com automação**

Escalabilidade

Suportam o aumento horizontal para **distribuir carga** para novas réplicas

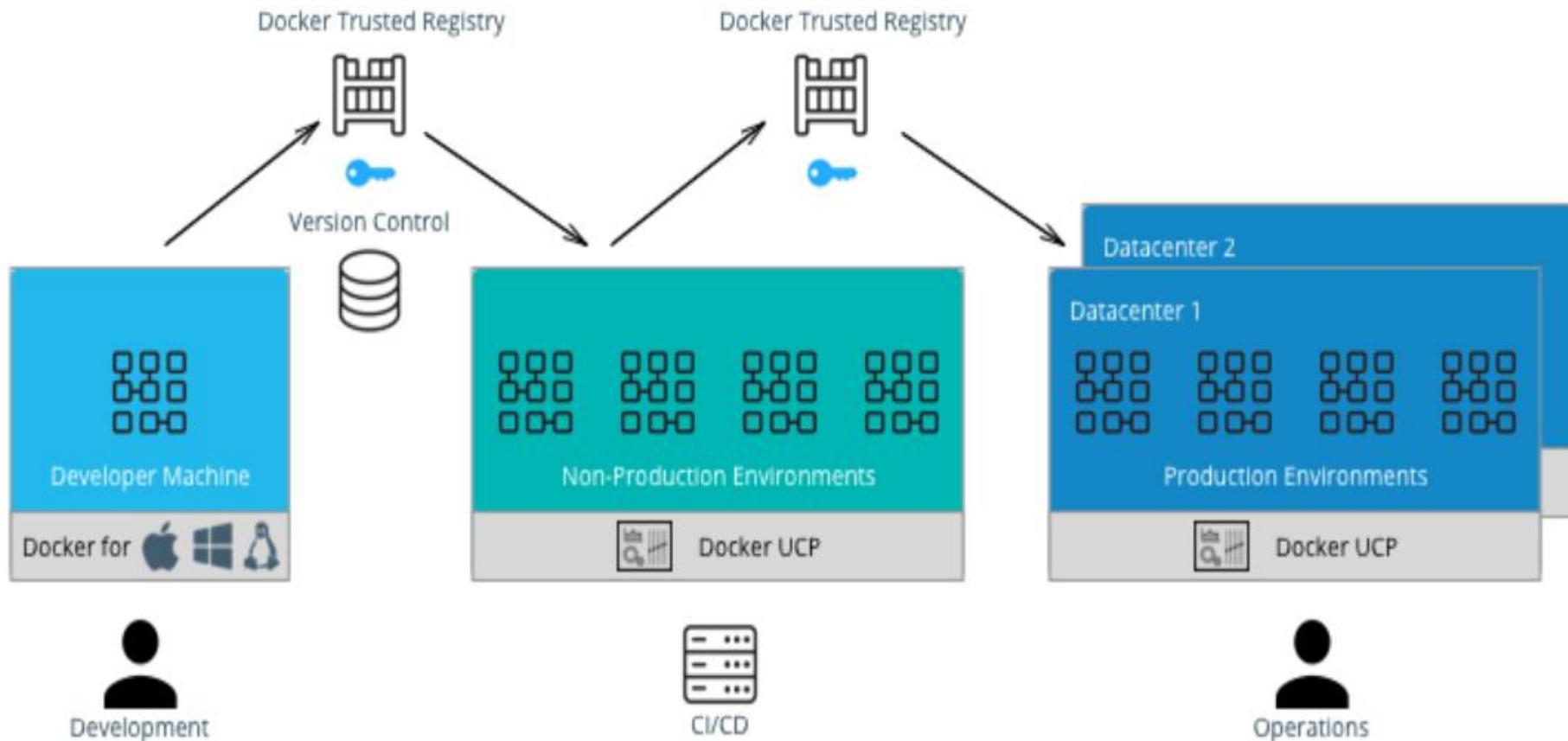
Orquestração

Agrupamento de **serviços relacionados interdependentes** (stacks)

Utilização de Microserviços em Docker

FIAP

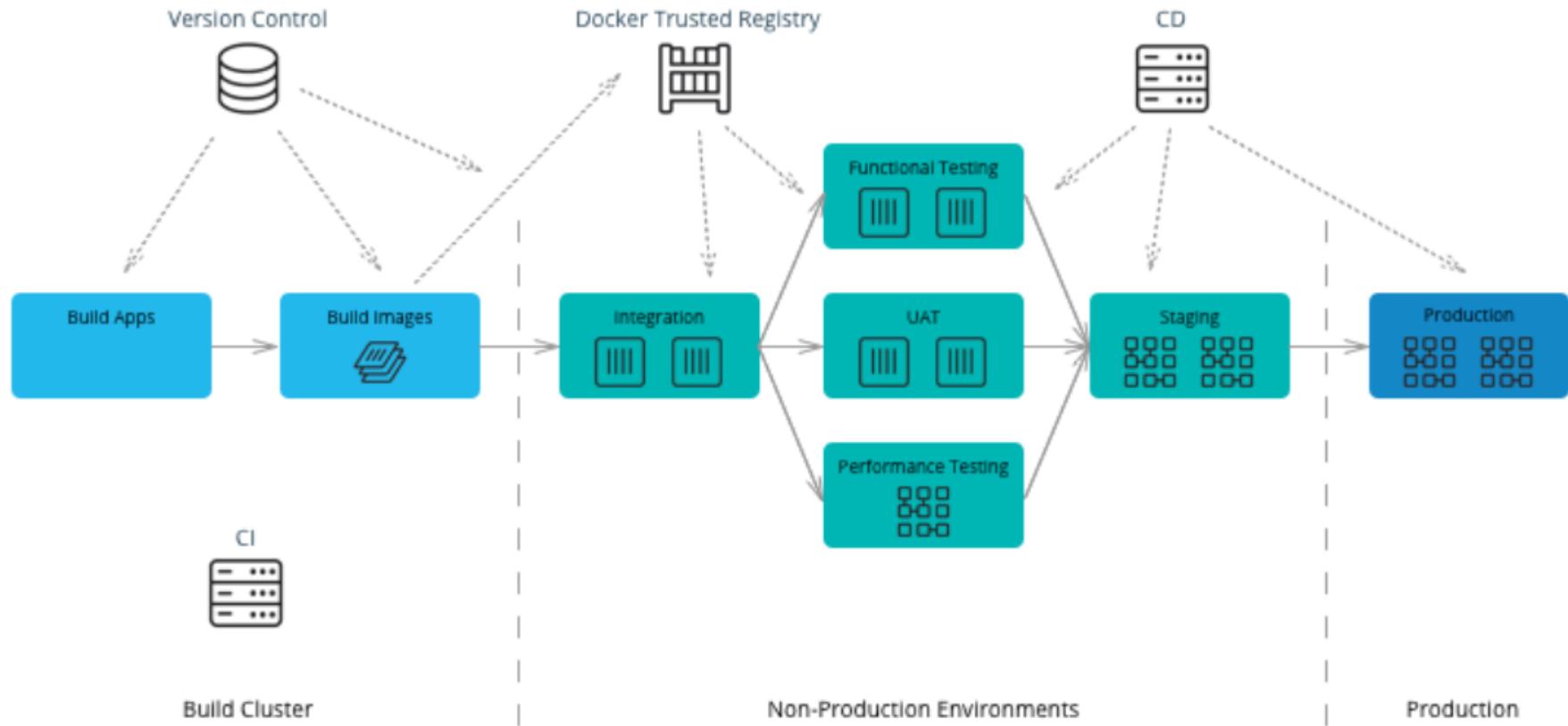
Exemplo de Fluxo de Implantação



Utilização de Microserviços em Docker

FIAP

Detalhes de um Fluxo CI/CD com Docker



Docker Universal Control Plane

Padrões de Arquitetura

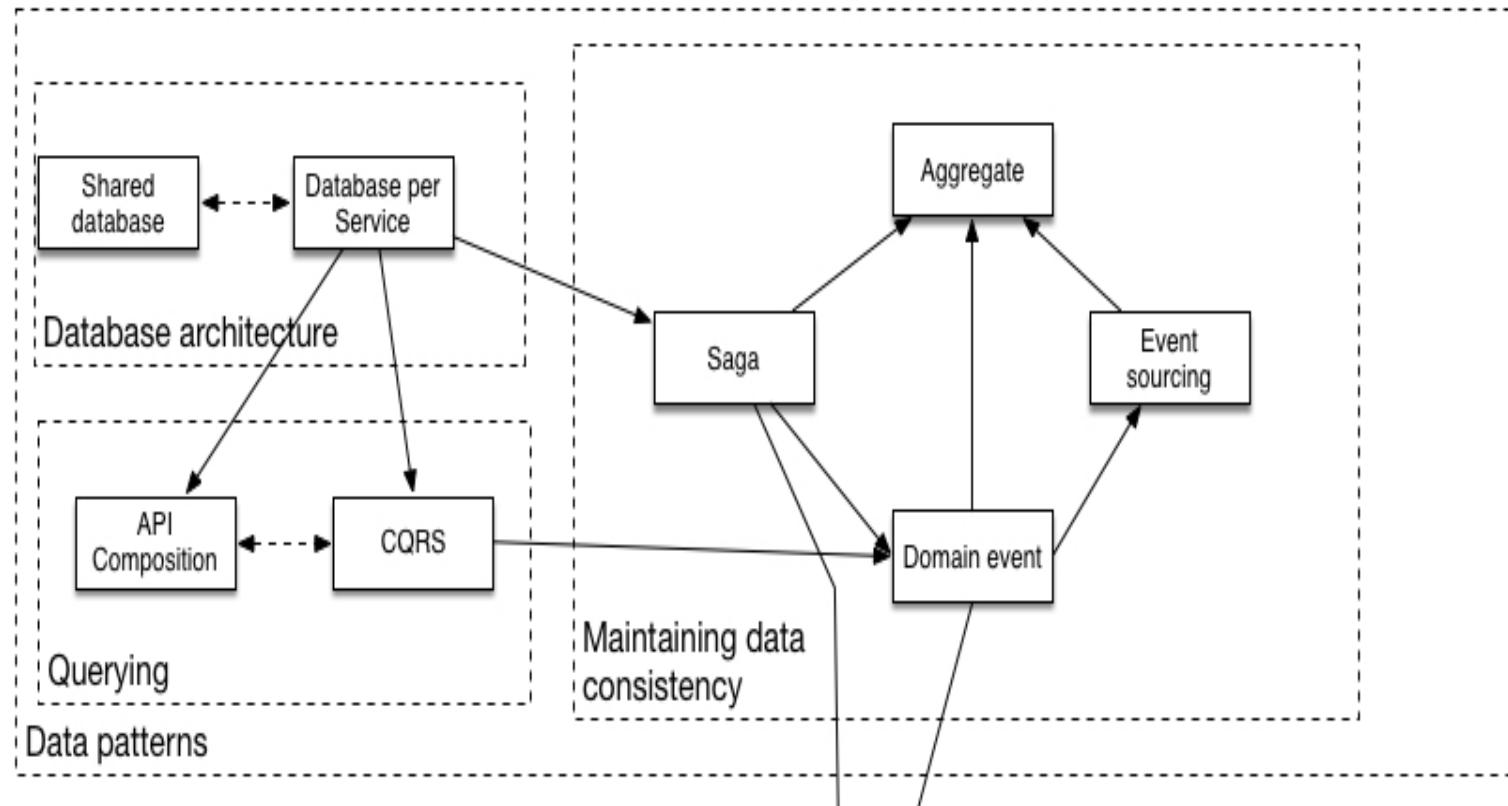
FIAP

Padrões de
Aplicativos

Infraestrutura de
Aplicações

Padrões de
Infraestrutura

Application
patterns



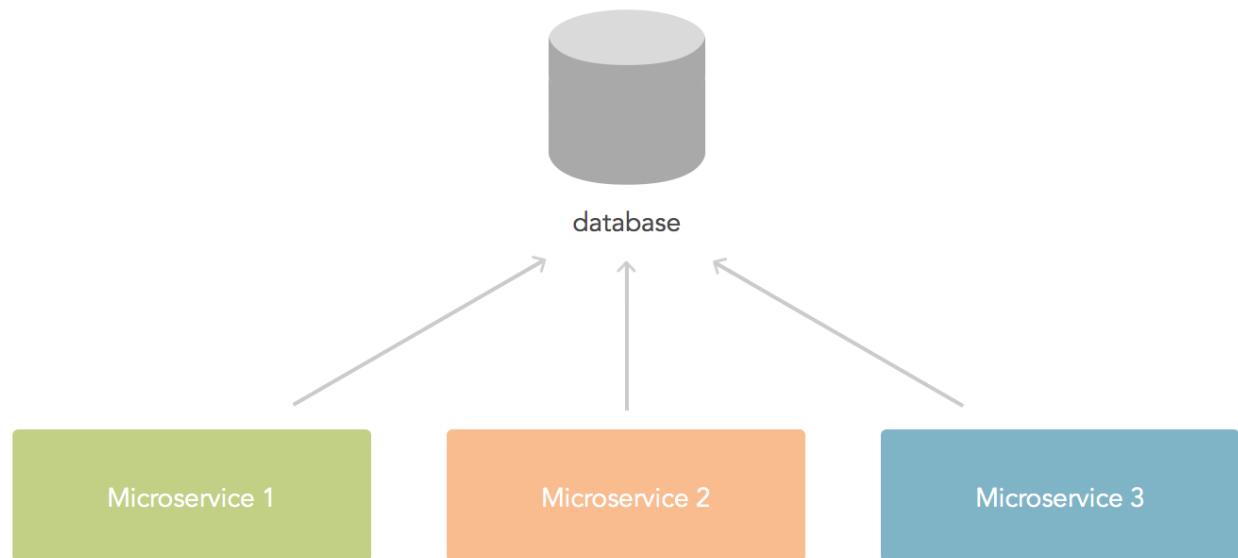
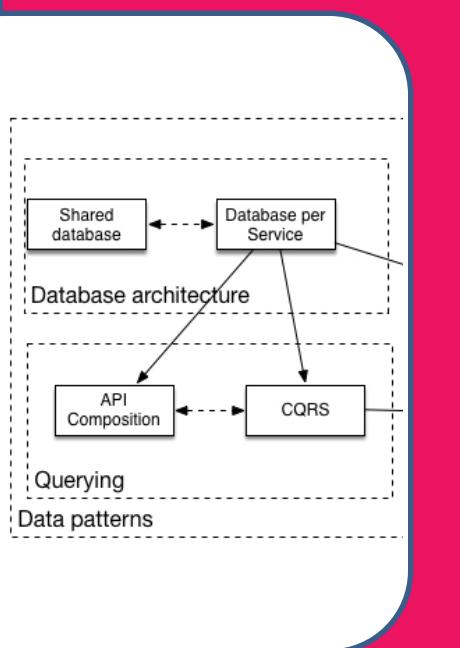
Padrões de Aplicativos

Padrões de Banco de Dados

Infraestrutura de Aplicações

Padrões de Infraestrutura

BD Compartilhado: forma mais comum de integração

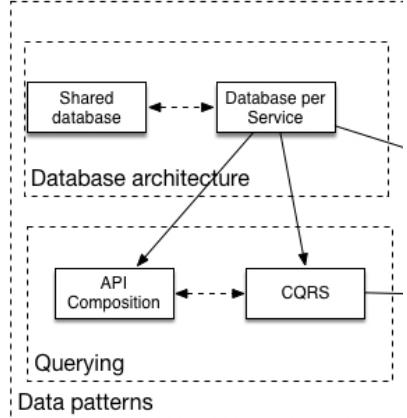


Padrões de Aplicativos

Infraestrutura de Aplicações

Padrões de Infraestrutura

Padrões de Banco de Dados



Banco de Dados compartilhado

Problema

Partes externas visualizam detalhes **internos** com uma **tecnologia específica**

Solução

Utilizar Banco de Dados por serviço **acessíveis** somente por meio de **API**

Desvantagens

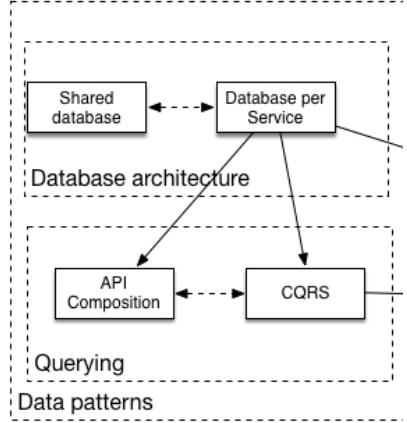
Alto acoplamento: mudanças no banco de dados podem **afetar diversos serviços**

Padrões de Aplicativos

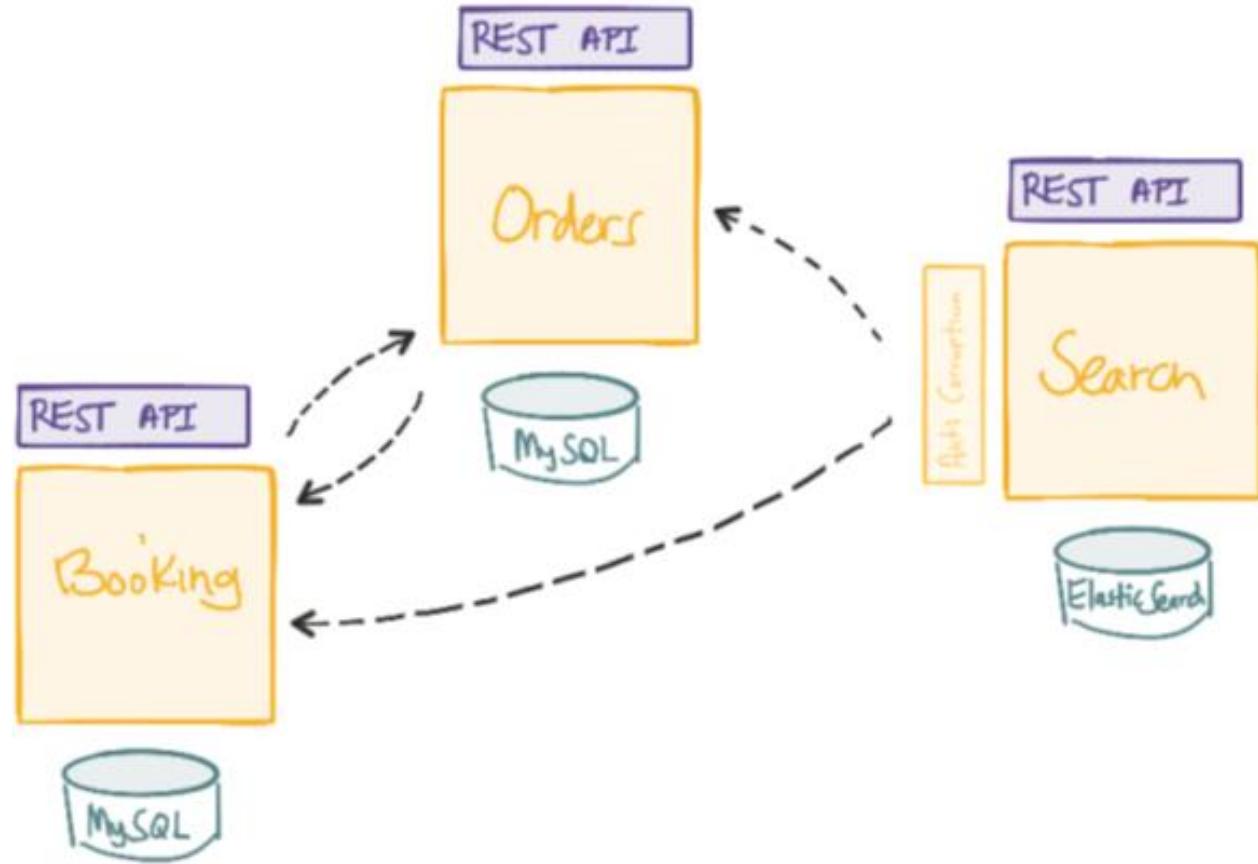
Infraestrutura de Aplicações

Padrões de Infraestrutura

Padrões de Banco de Dados



BD por Serviço: acesso aos dados por API



Padrões de Arquitetura

FIAP

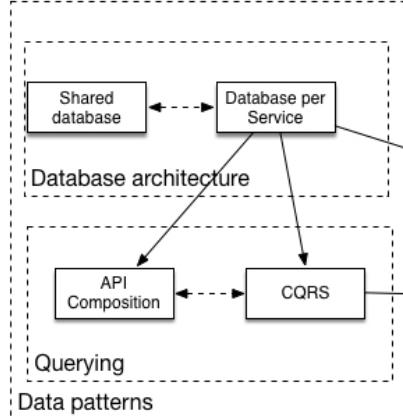
Padrões de Aplicativos

Padrões de Banco de Dados

Infraestrutura de Aplicações

Padrões de Infraestrutura

BD por Serviço: acesso aos dados por API



Problema

Compartilhar informações entre os serviços

Solução

Controle de acesso **privativos** às tabelas

Servidor dedicado para atender alta performance / tecnologias

Vantagens

Alterações **não afetam** serviço externos

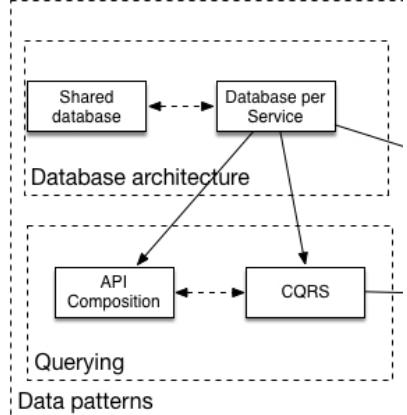
Serviço usa o **BD mais adequado** às suas necessidades

Padrões de Aplicativos

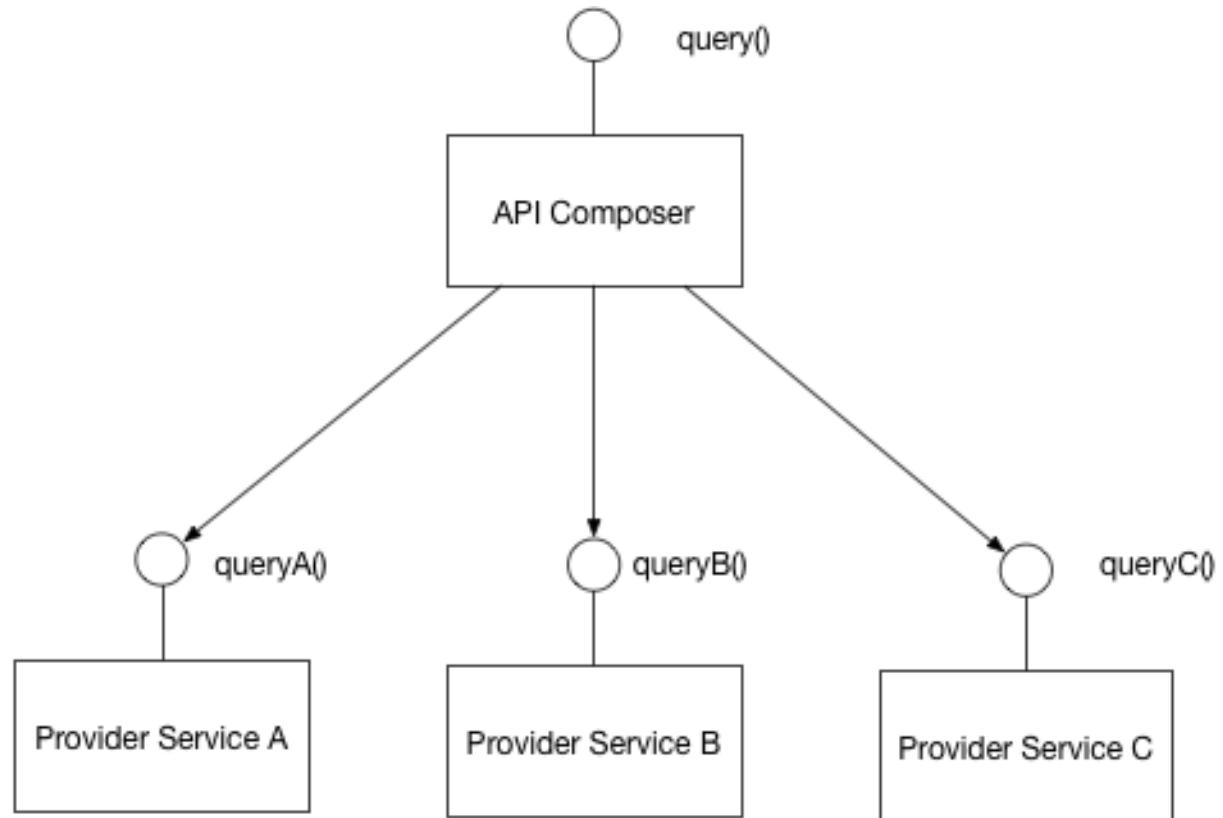
Infraestrutura de Aplicações

Padrões de Infraestrutura

Consultas aos Dados



Composição por API: agregar queries/joins

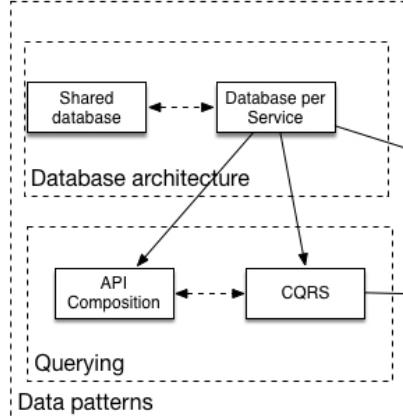


Padrões de Aplicativos

Infraestrutura de Aplicações

Padrões de Infraestrutura

Consultas aos Dados



Composição por API: agregar *queries/joins*

Problema

Como consultar informações de vários microserviços

Solução

Chamar **API** dos serviços que possuem os **dados** e realiza **junção dos resultados** em memória

Desvantagens

Impactar o desempenho com agregação de grandes conjuntos de dados

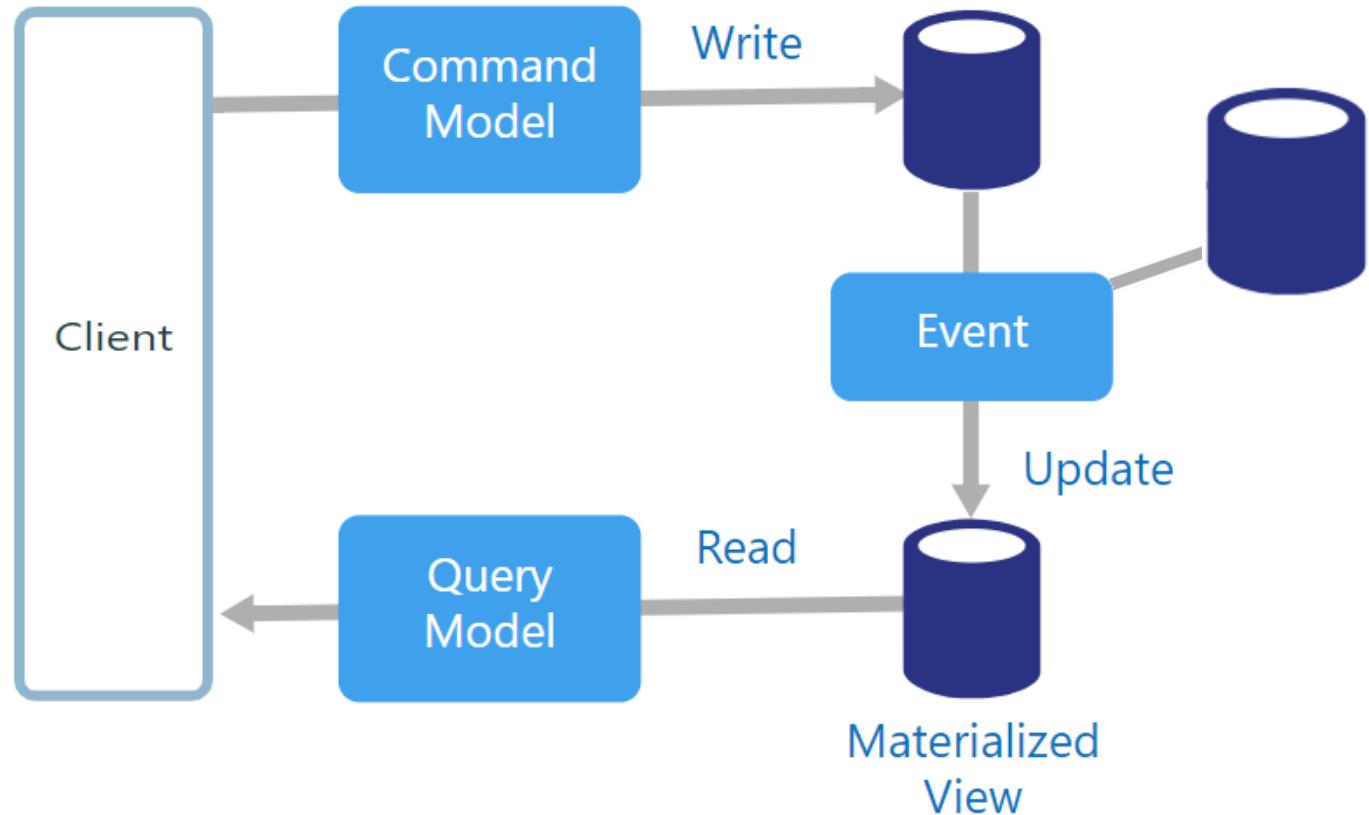
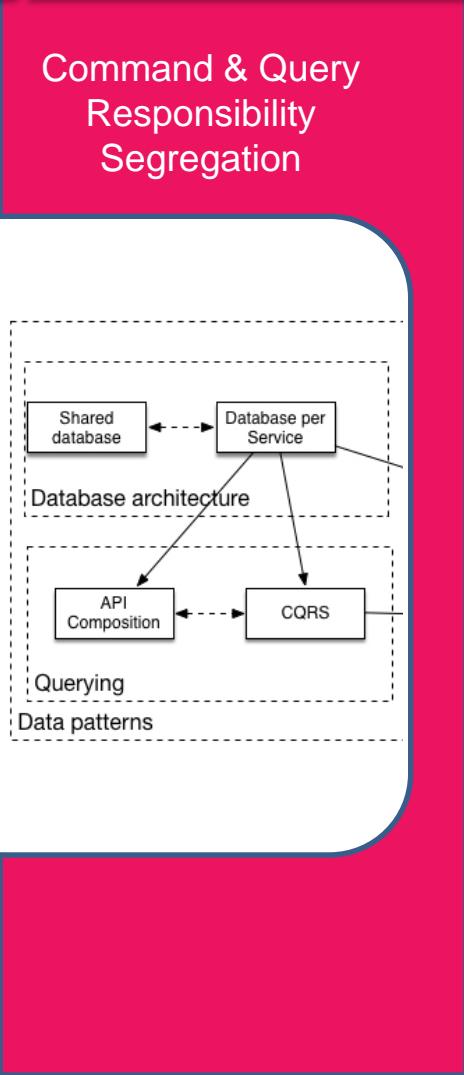
Padrões de Aplicativos

Command & Query
Responsibility
Segregation

Infraestrutura de Aplicações

Padrões de Infraestrutura

CQRS: Segregação de BD Consulta e Escrita



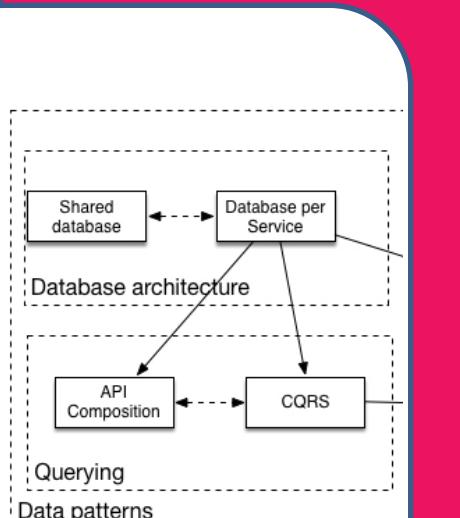
Padrões de Aplicativos

Command & Query
Responsibility
Segregation

Infraestrutura de Aplicações

Padrões de Infraestrutura

CQRS: Segregação de BD Consulta e Escrita



Problema

Como consultar informações de vários microserviços

Solução

Gerar visões em **BD de consulta** que materializam o conjunto de **dados necessários**

Vantagens

Menor contenção, com bases R/W independentes

Visões evitam operações complexas ao consultar

Desafios que devem ser atendidos

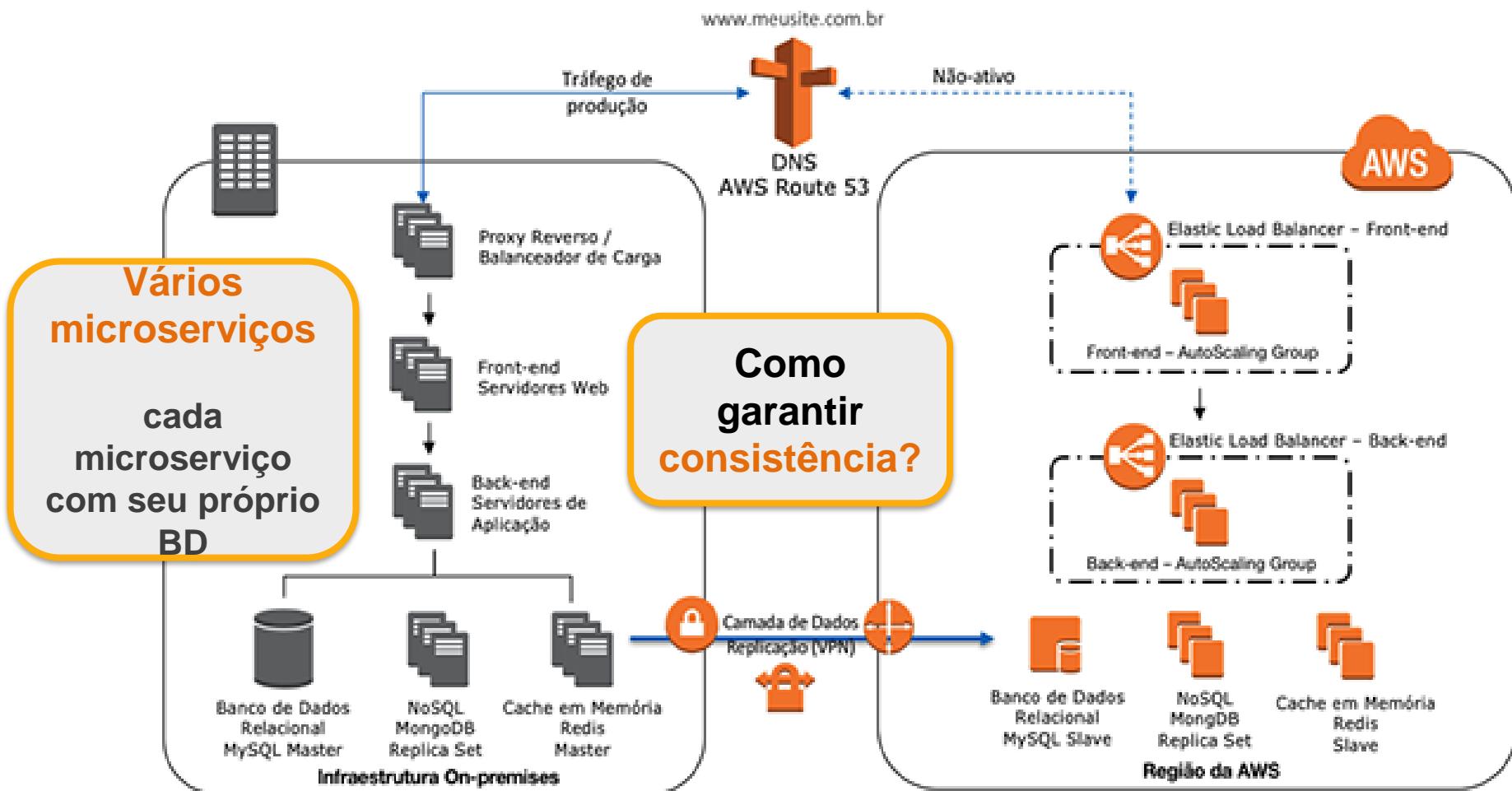
Obter alta disponibilidade de Microserviços

Resiliência em sistemas de missão crítica

Integração entre:
Microserviços
(Baixa Plataforma)
e
Mainframe
(Alta Plataforma)

Armazenar dezenas de terabytes de informação

Consistência no uso de múltiplos Banco de Dados



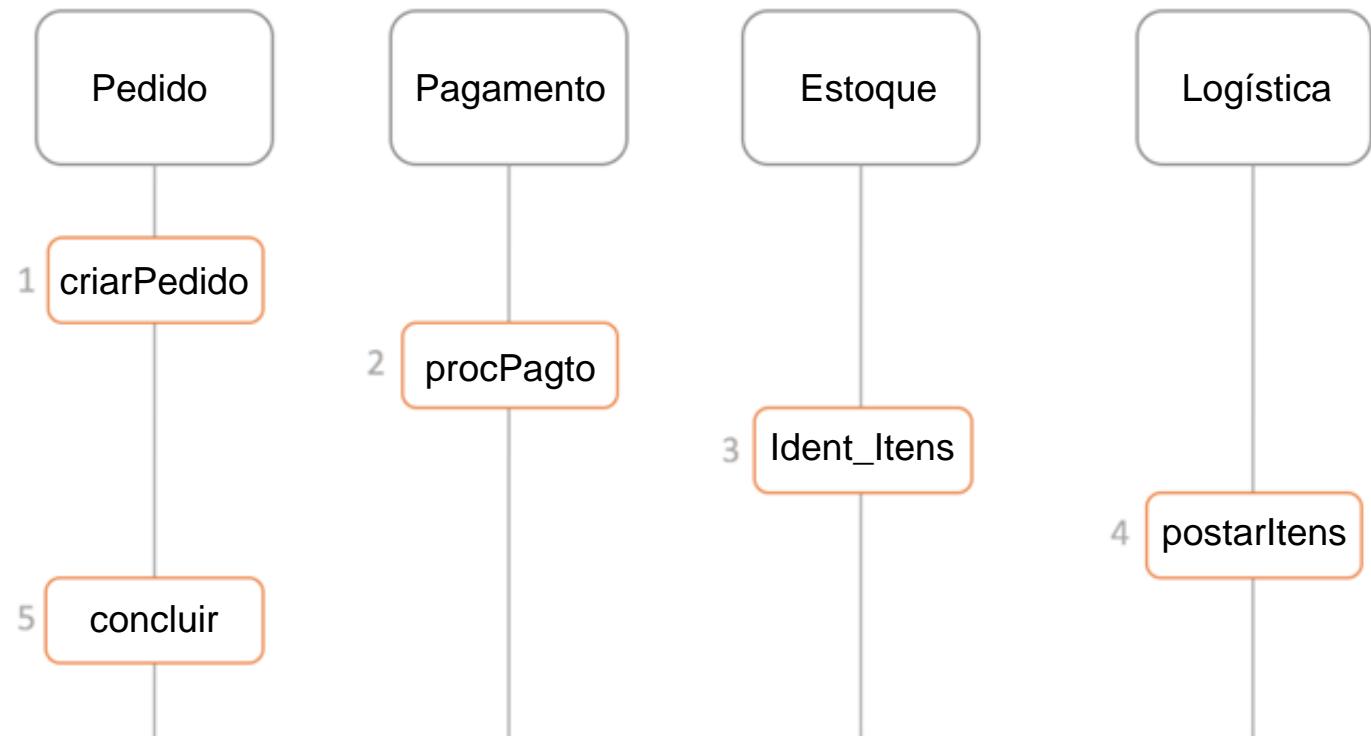
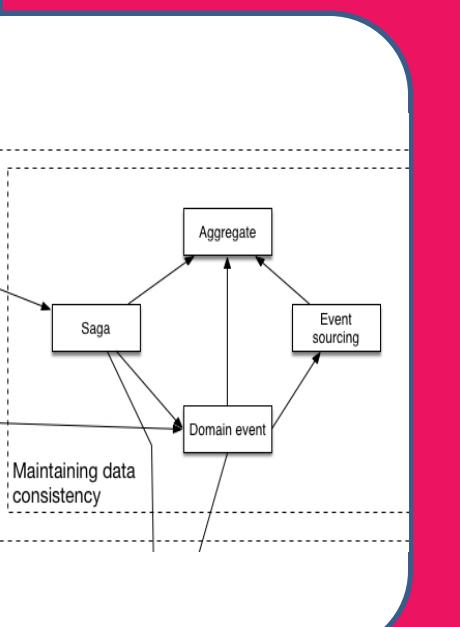
Padrões de Aplicativos

Consistência dos dados

Infraestrutura de Aplicações

Padrões de Infraestrutura

SAGA: Sequência de transações distribuídas

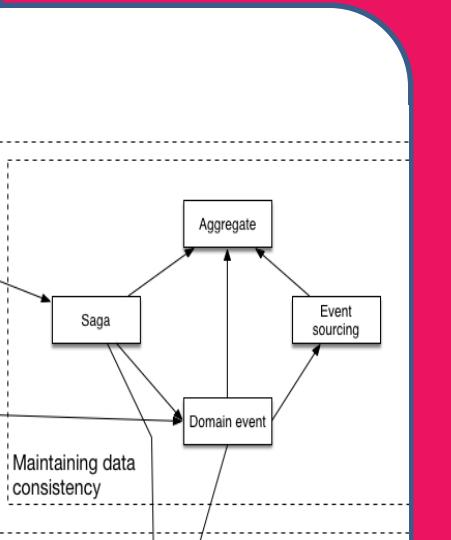


Padrões de Arquitetura

FIAP

Padrões de Aplicativos

Consistência dos dados



Infraestrutura de Aplicações

Padrões de Infraestrutura

SAGA: Orquestração versus Coreografia

Orquestração

há um responsável central para guiar e conduzir o processo



Coreografia

cada parte do sistema realiza seu trabalho reagindo uns aos outros

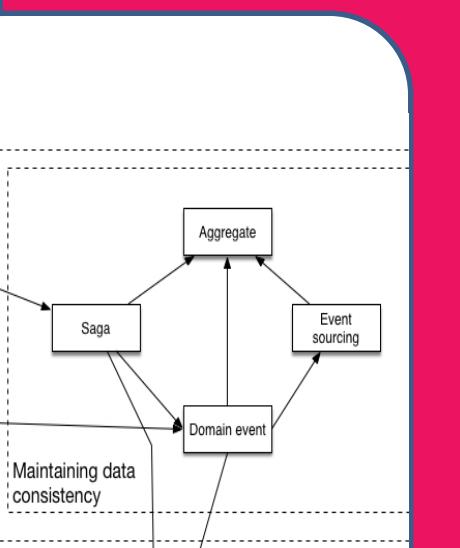


Padrões de Aplicativos

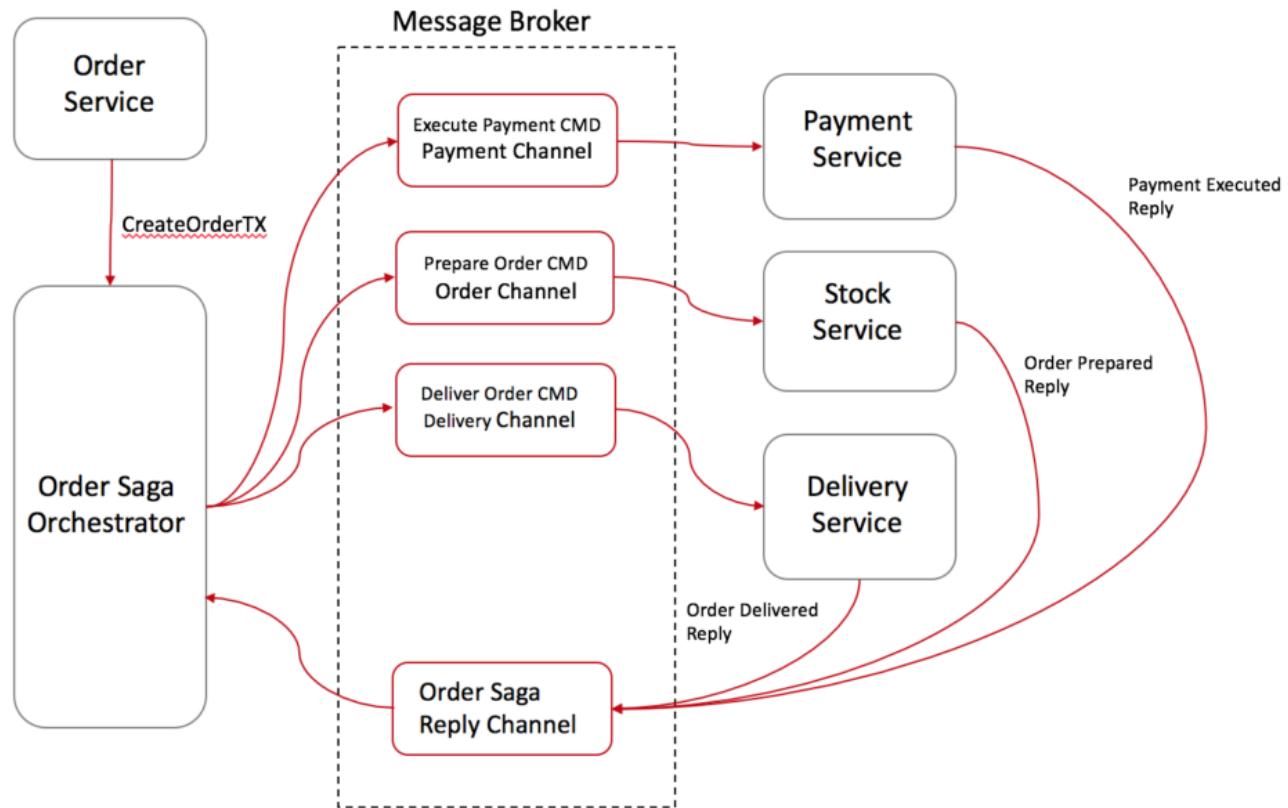
Infraestrutura de Aplicações

Padrões de Infraestrutura

Consistência dos dados



SAGA: Orquestração de transações distribuídas

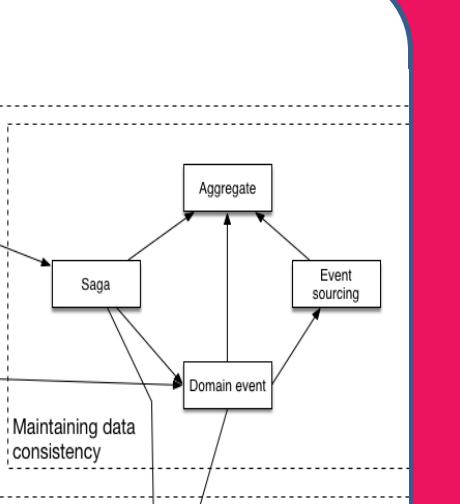


Padrões de Aplicativos

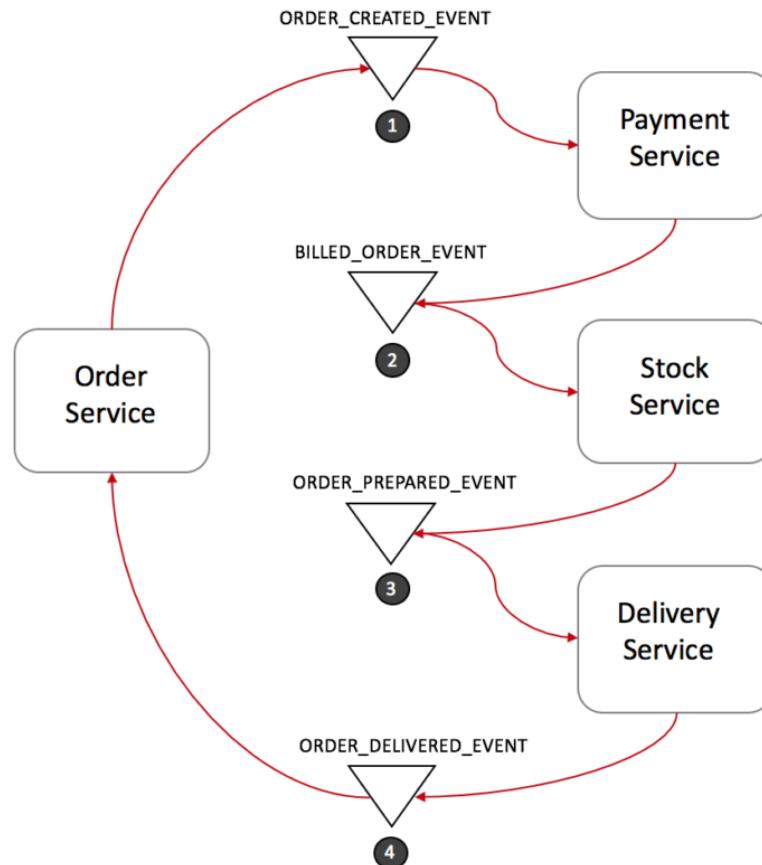
Infraestrutura de Aplicações

Padrões de Infraestrutura

Consistência dos dados



SAGA: Coreografia de transações distribuídas

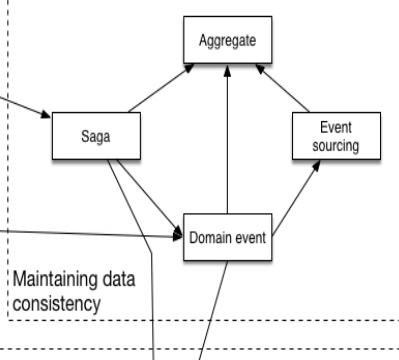


Padrões de Arquitetura

FIAP

Padrões de Aplicativos

Consistência dos dados



Infraestrutura de Aplicações

SAGA: Sequência de transações distribuídas

Problema

Como manter **consistência dos dados** nos serviços?

2-phase-commit não é uma opção

Padrões de Infraestrutura

Solução

Orquestração de eventos

Coreografia de eventos

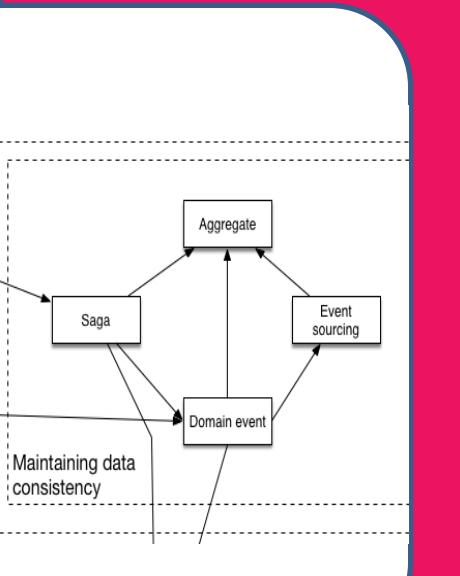
Vantagens

Fácil **adicionar novos participantes** com baixo acoplamento

Implementação por **Máquina de Estados**

Padrões de Aplicativos

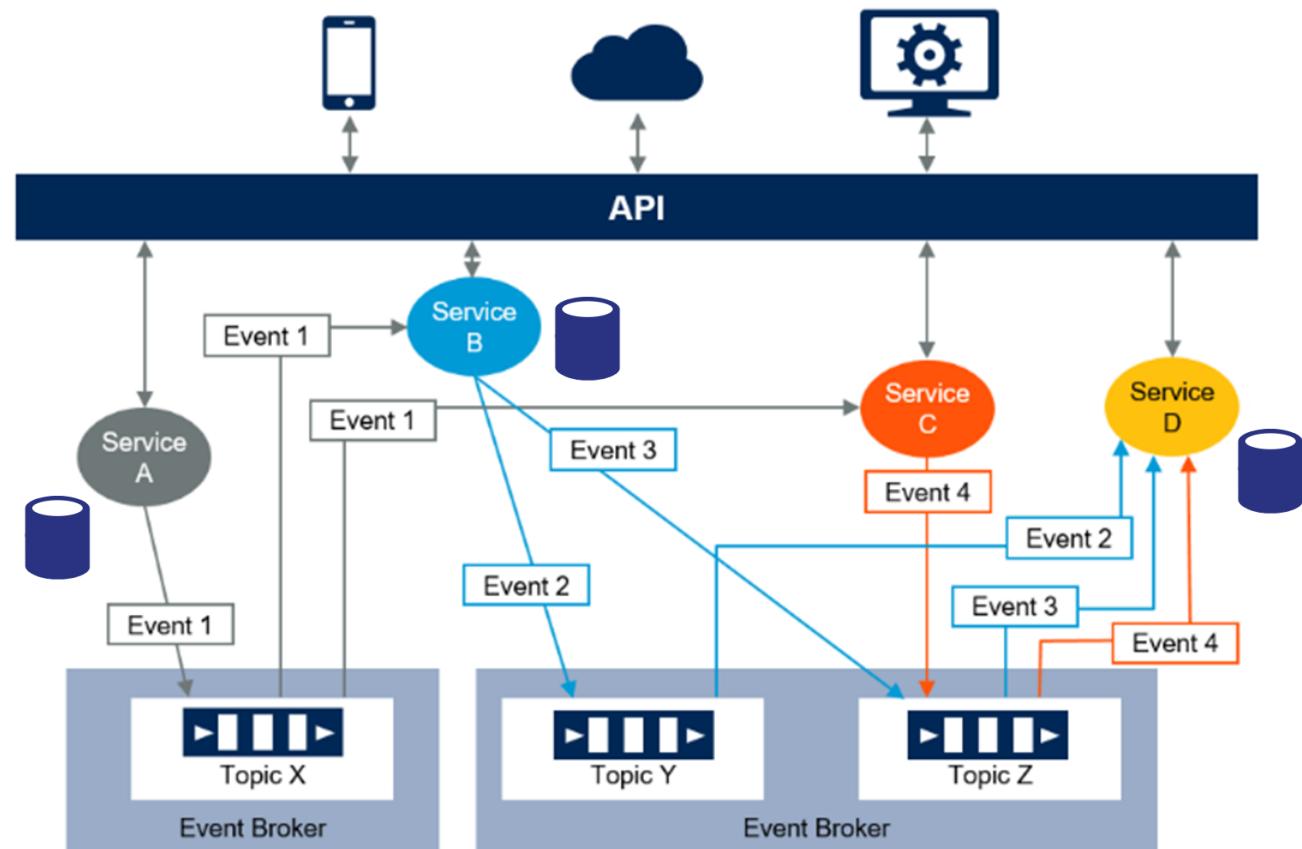
Consistência dos dados



Infraestrutura de Aplicações

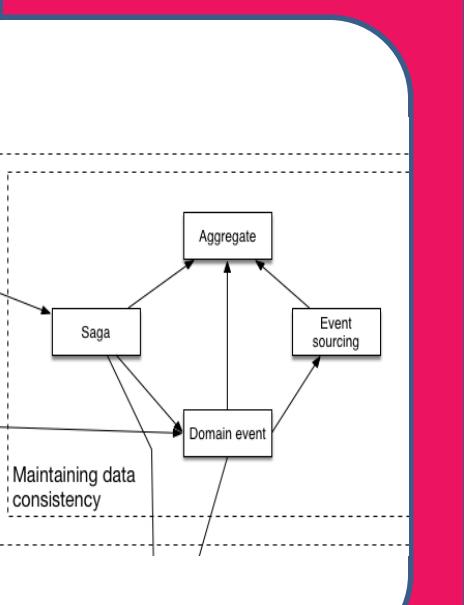
Padrões de Infraestrutura

Event sourcing: operações controladas por sequência de eventos



Padrões de Aplicativos

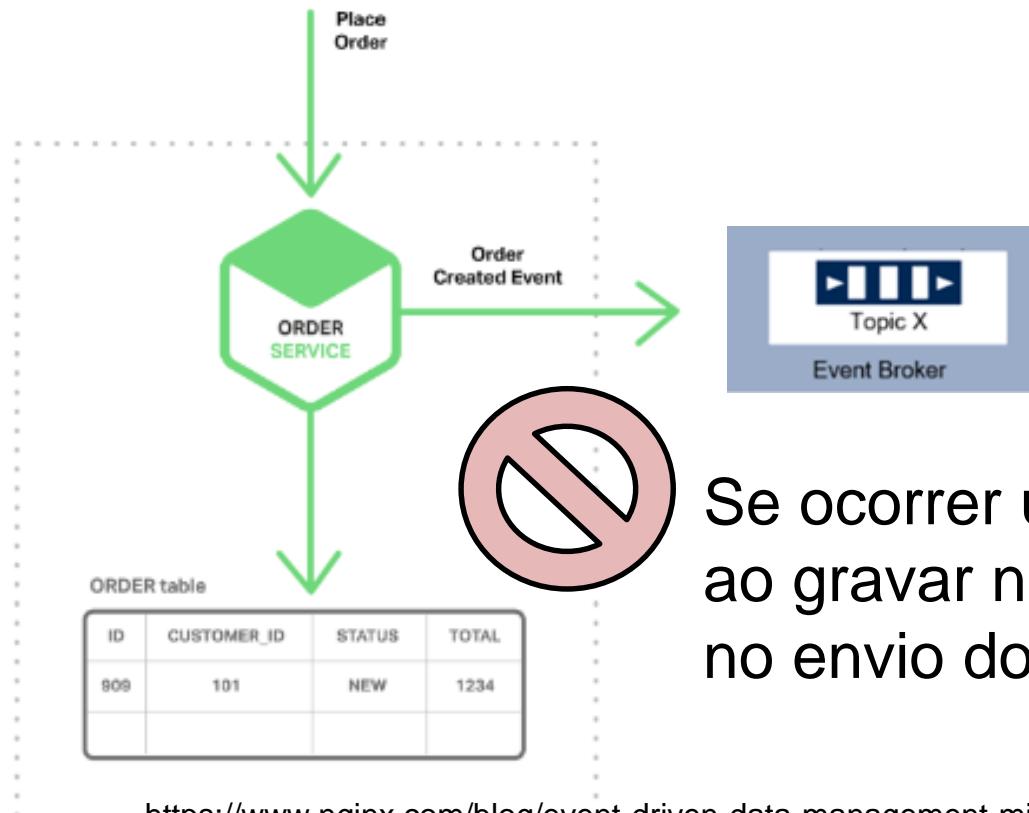
Consistência dos dados



Infraestrutura de Aplicações

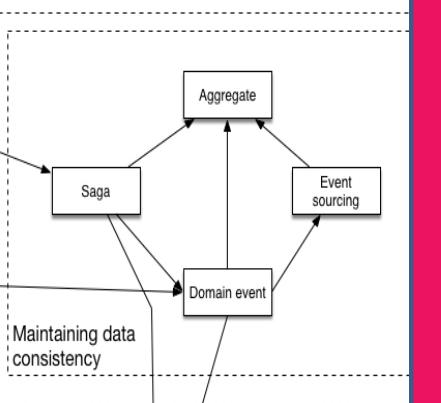
Padrões de Infraestrutura

Event sourcing: operações assíncronas



Padrões de Aplicativos

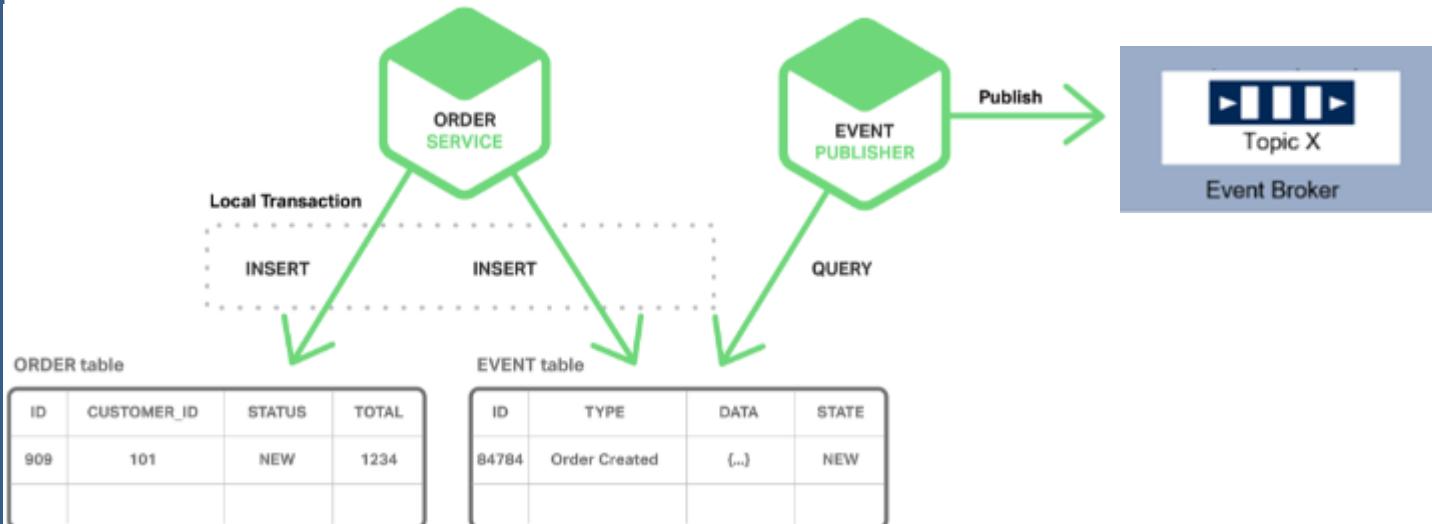
Consistência dos dados



Infraestrutura de Aplicações

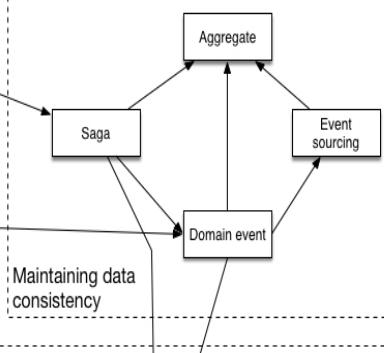
Padrões de Infraestrutura

Event sourcing: operações atômicas



Padrões de Aplicativos

Consistência dos dados



Infraestrutura de Aplicações

Padrões de Infraestrutura

Event sourcing: operações controladas por sequência de eventos

Problema

Como atualizar o BD e ao mesmo tempo (atômico) publicar mensagens / eventos?

Solução

Utilizar banco de dados de eventos em transações locais para garantir atomicidade

Vantagens

Publicação confiável de eventos com consistência de dados

Log de auditoria das alterações com consultas temporais

Vamos nos Divertir!

FIAP



O que vimos até aqui:

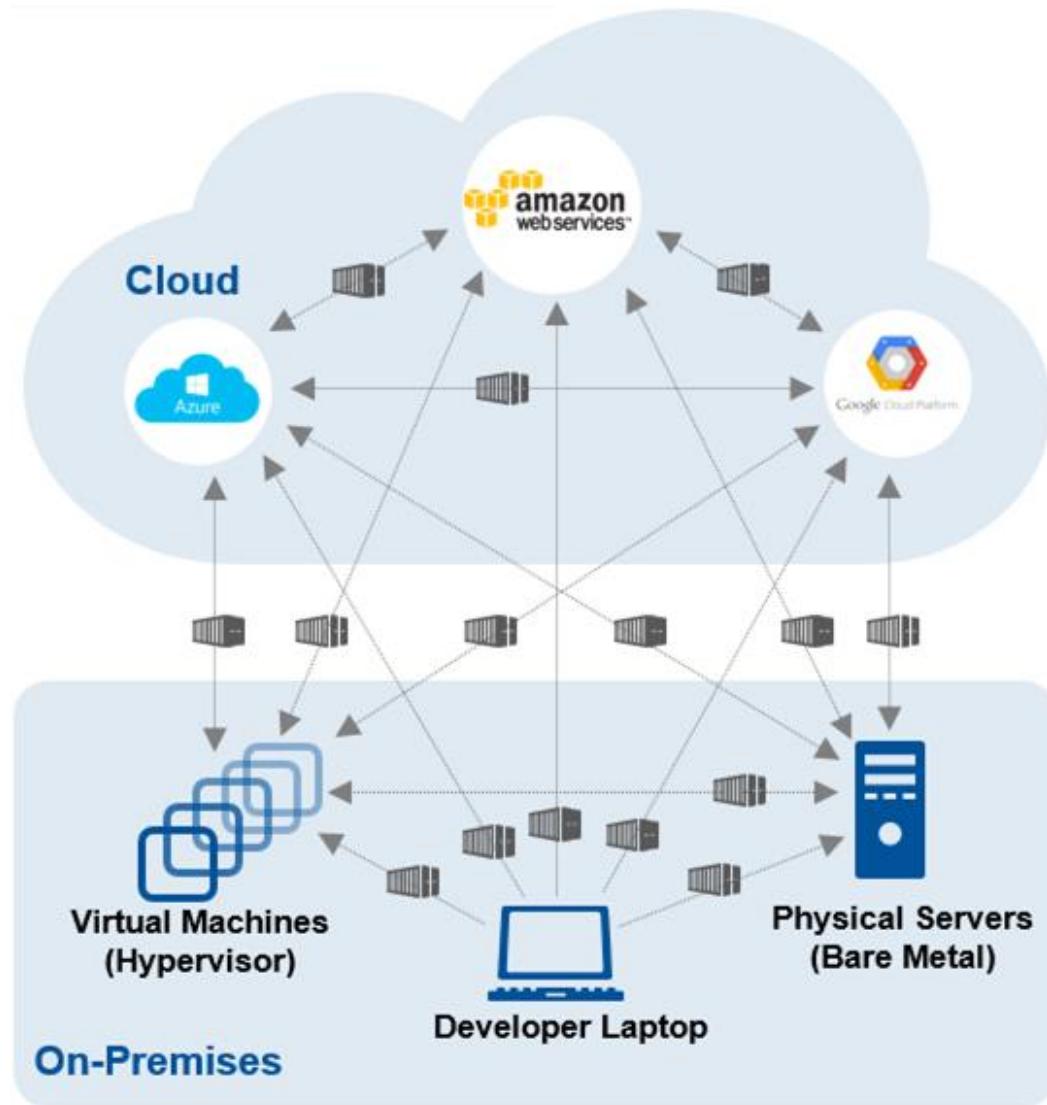
- Utilização de Microserviços alinhado aos Negócios
- Visão geral da Arquitetura de Microserviços
- Fluxo CI/CD com Docker
- Arquitetura de Projetos
 - Comunicação síncrona versus assíncrona
 - BD compartilhado versus BD por serviço
 - Queries e Consistências em BD
 - Configurações comuns, Segurança e Confiabilidade

MICROSERVIÇOS NO DIA A DIA

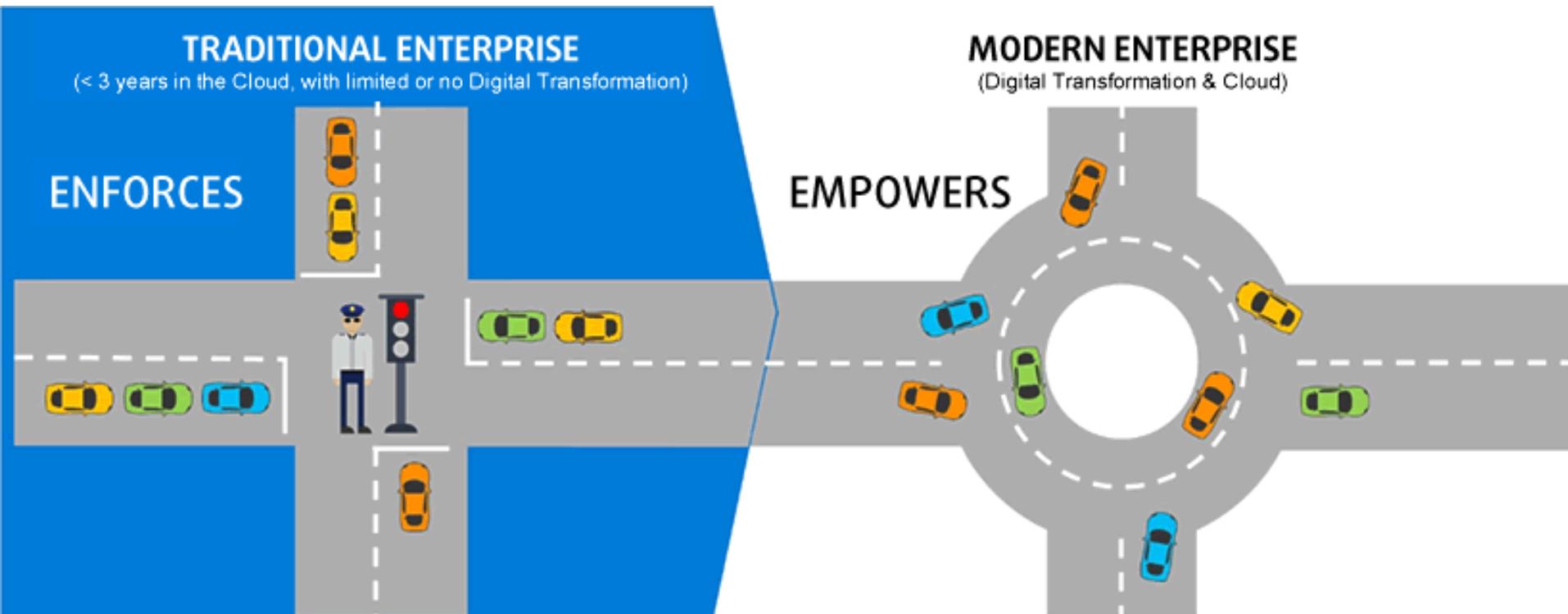
Microserviços no dia a dia

FIAP

Mobilidade entre diversos ambientes



Empresas precisam capacitar seus times para a inovação



Controle central
Imposição

Liberdade
Responsabilidade

Microserviços no dia a dia

FIAP

Difícil é escolher quais plataformas utilizar

Service Mesh



External Gateway

Runtime & Platform

Backing Services



Automation



Telemetry



Microserviços no dia a dia

FIAP

Algumas ferramentas...

Service Mesh



NETFLIX
OSS
Eureka/
Archaius

External Gateway



AWS
API Gateway

Runtime & Platform



docker



SWARM



kubernetes
Google



Spring
Boot



AWS Lambda

Backing Services



redis

Automation



Jenkins

Telemetry



elastic

Descobrir e comparar Stacks (Microframeworks Backends)

<https://stackshare.io/index/microframeworks>

1		ExpressJS <small>4.08K Stacks</small>	Sinatra inspired web development framework for node.js.	Visit Website
2		Flask <small>2.66K Stacks</small>	a microframework for Python based on Werkzeug, Jinja 2...	Visit Website
3		Django REST framework <small>567 Stacks</small>	Web APIs for Django	Visit Website
4		Sinatra <small>337 Stacks</small>	Classy web-development dressed in a DSL	Visit Website
5		hapi <small>205 Stacks</small>	Server Framework for Node.js	Visit Website
6		Sails.js <small>190 Stacks</small>	Realtime MVC Framework for Node.js	Visit Website
7		Koa <small>168 Stacks</small>	Next generation web framework for node.js	Visit Website
8		Slim <small>162 Stacks</small>	A PHP micro framework	Visit Website

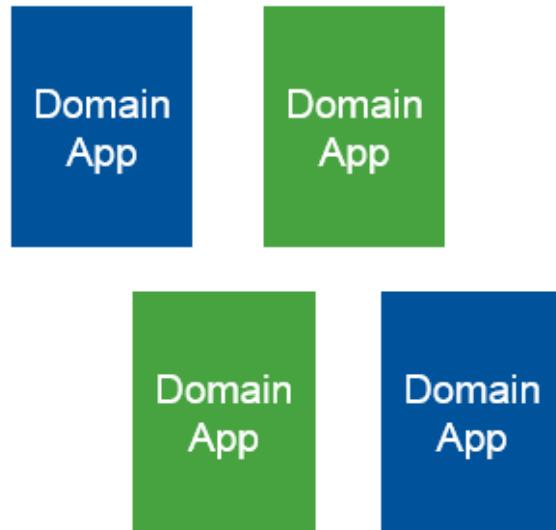
Flexibilidade: baixo acoplamento entre os componentes

Looser Coupling = More Options

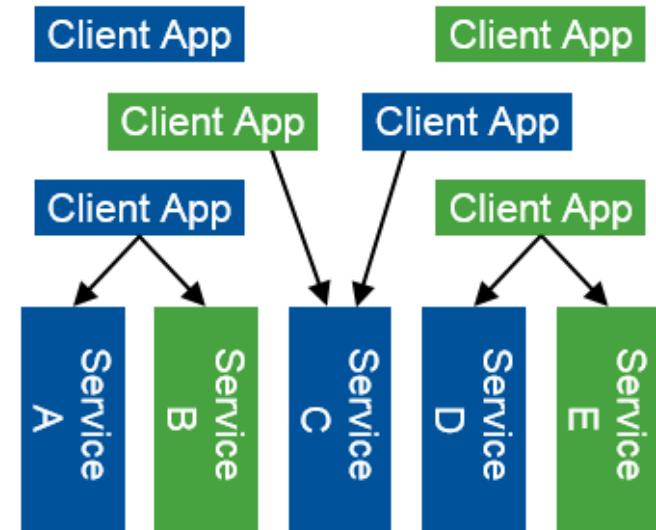
Monolithic Apps



Domain Apps



Loosely Coupled Client/Services



Definindo e priorizando a recuperação de falhas

Continuar a prestar serviços de TI visando a sobrevivência do negócio

Etapas necessárias para recuperar após um DESASTRE

Quais os serviços mais críticos?

Manter em funcionamento os serviços críticos alinhados com as prioridades de negócio (PCN)

Quais
microserviços
devem ser
oferecidos

Como os
clientes irão
consumir

Como monitorar
o seu uso

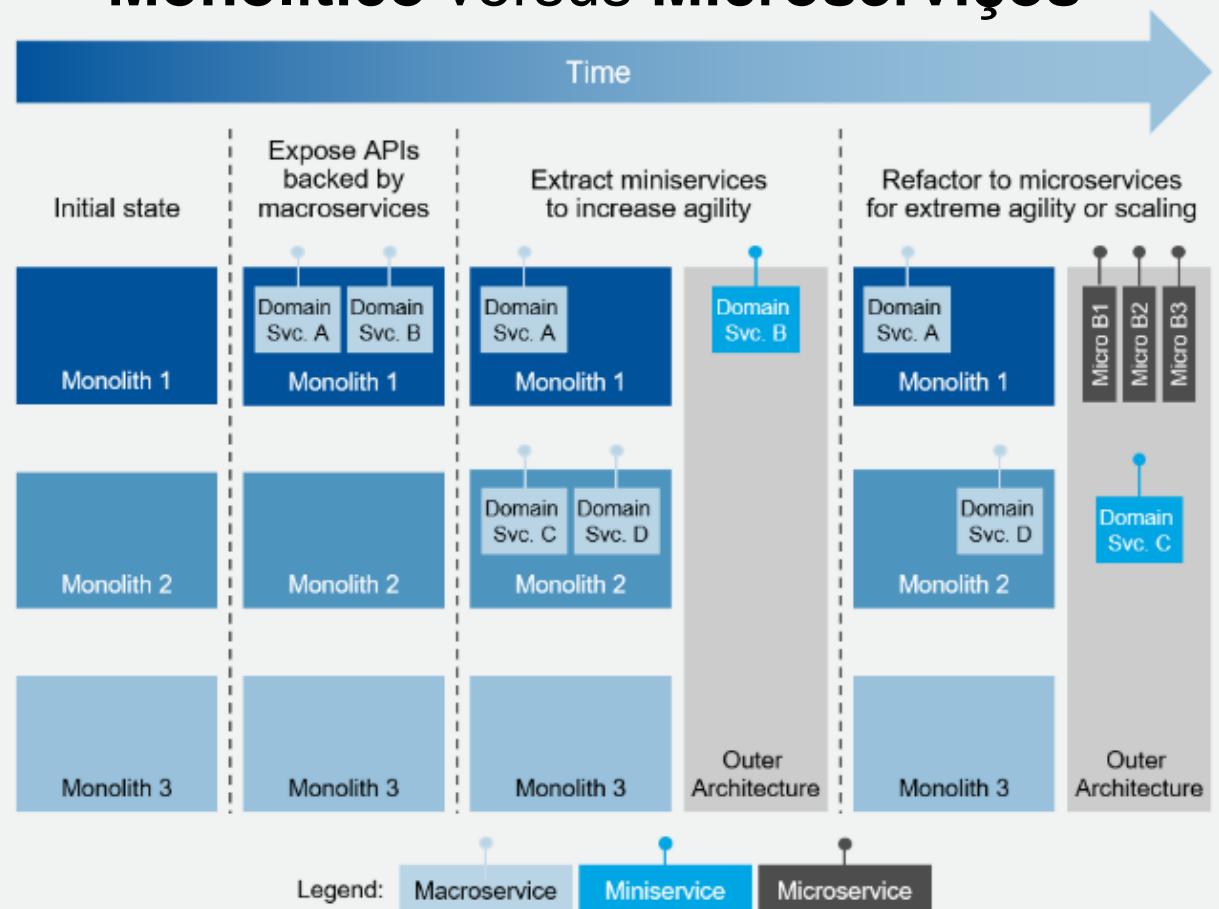
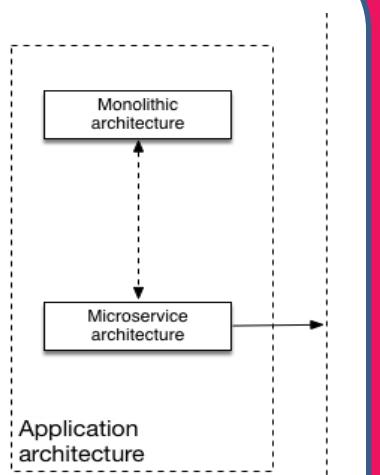
Como monetizar
a cobrança

Arquitetura de Aplicações

Infraestrutura de Aplicações

Arquitetura de Infraestrutura

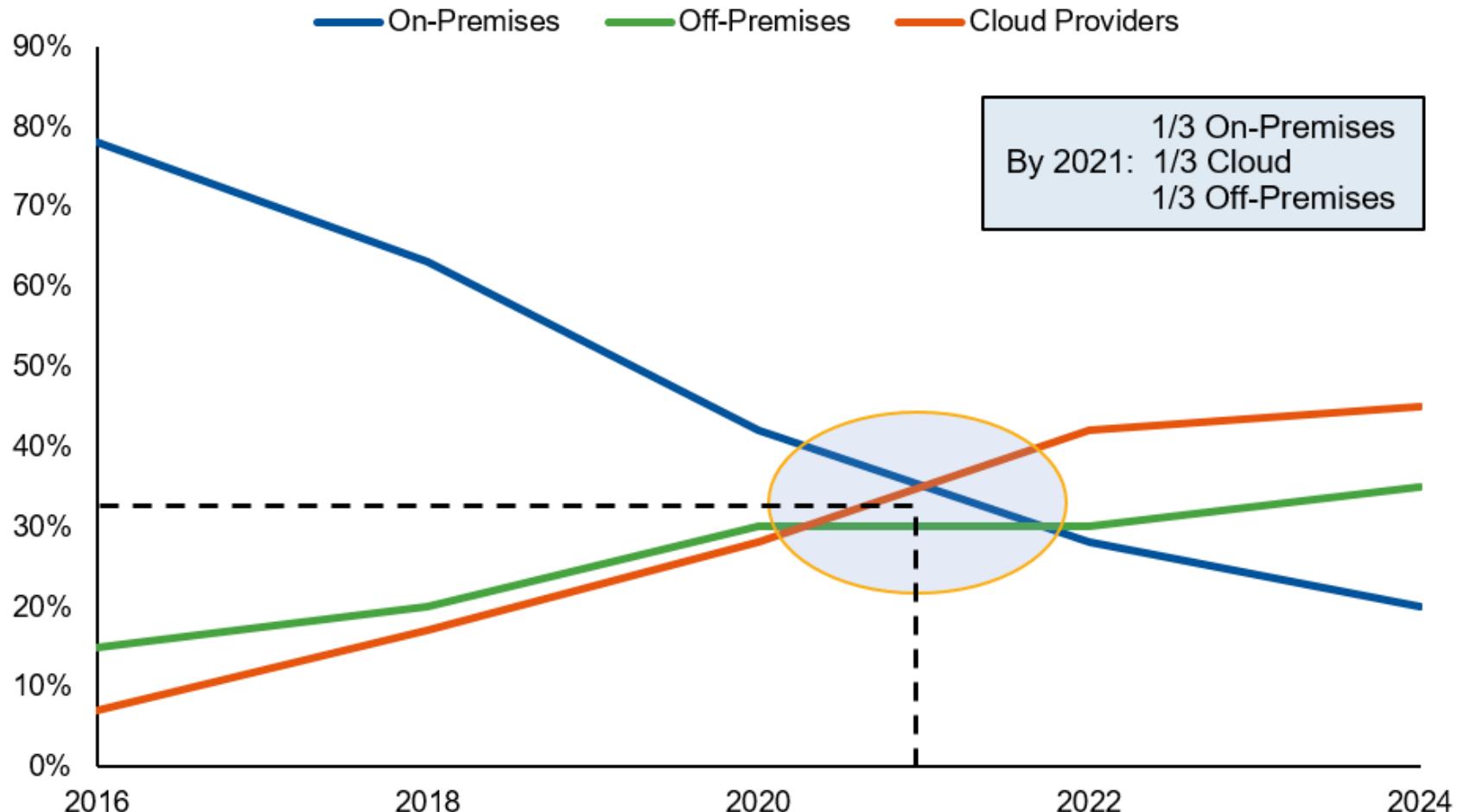
Monolítico versus Microserviços



Microserviços no dia a dia

FIAP

TI precisa adaptar seus processos para operar e gerenciar serviços em nuvem



Arquitetura escalável:

- Adição de recursos:

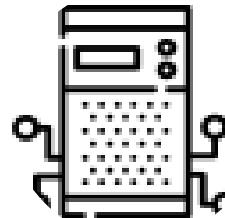


Verticalmente (scale up/down)

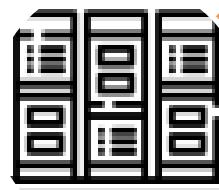


Horizontalmente (scale out/in)

- Elasticidade



Aumento automático para manter a qualidade de serviço



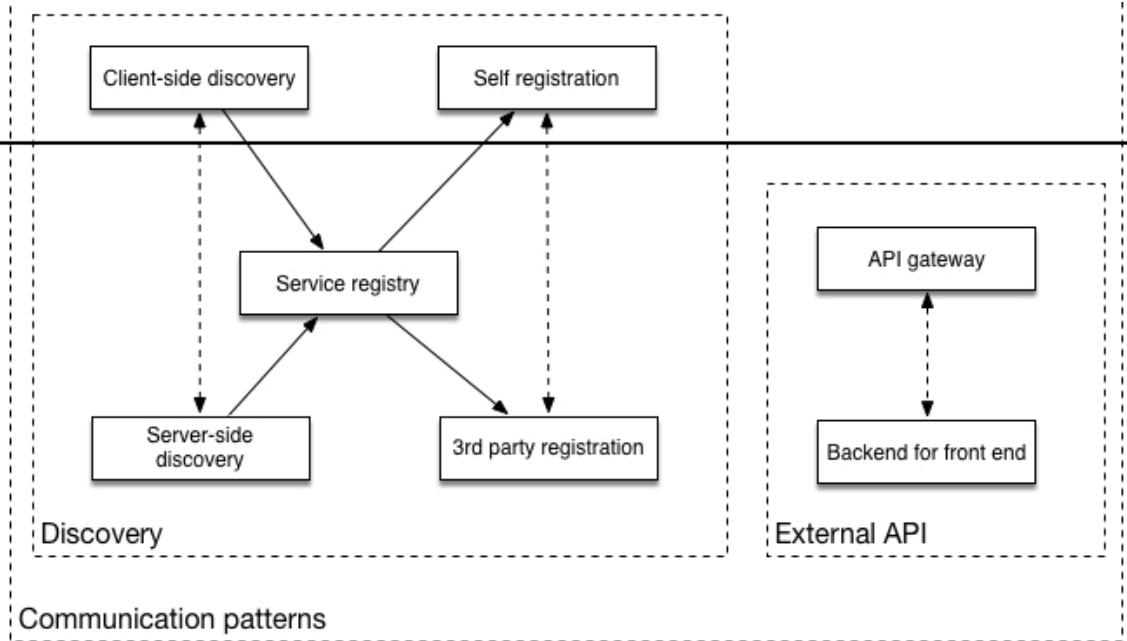
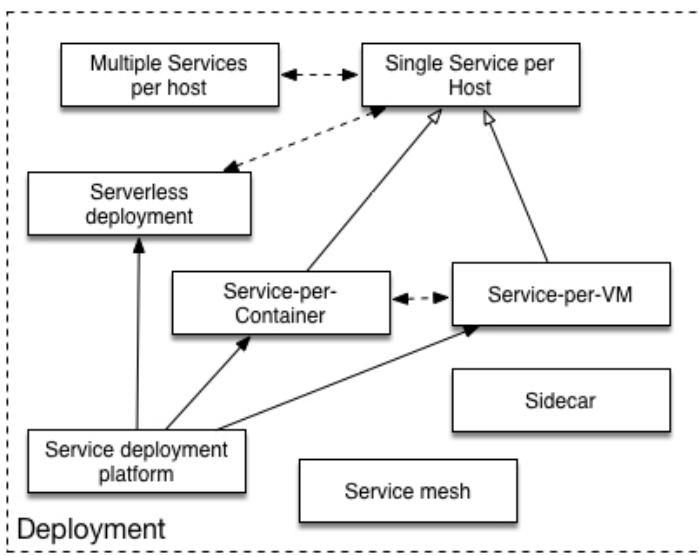
Remoção da capacidade excessiva quando a demanda diminuir, evitando gastos

Padrões de Arquitetura

FIAP

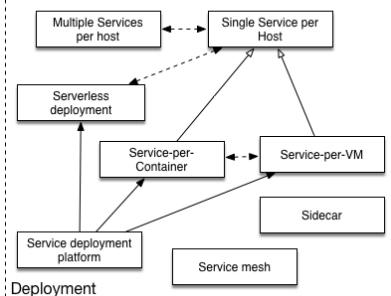
Padrões de Infraestrutura

Infrastructure patterns

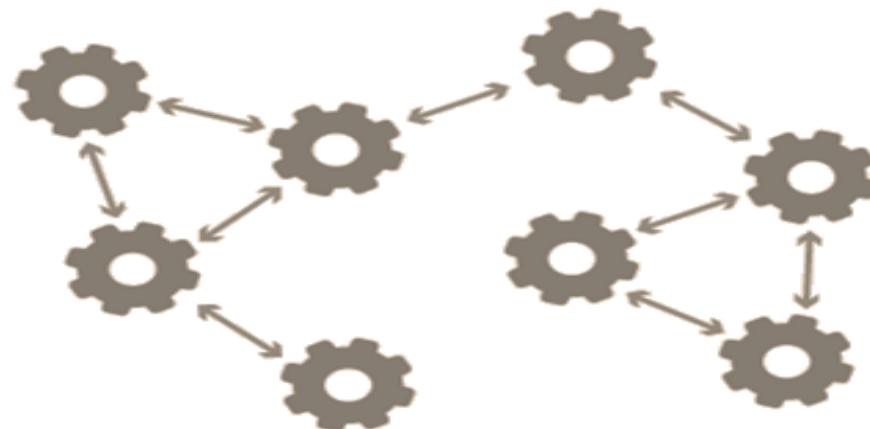


Implantação

Infrastructure patterns



Arquitetura Service Mesh (multi-conectada) e Side Car (controle das requisições)



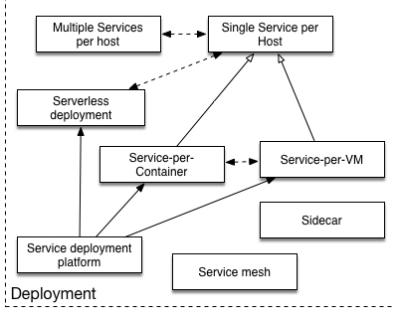
SERVICE MESH

Aplicações se
comunicam de forma
independente entre si

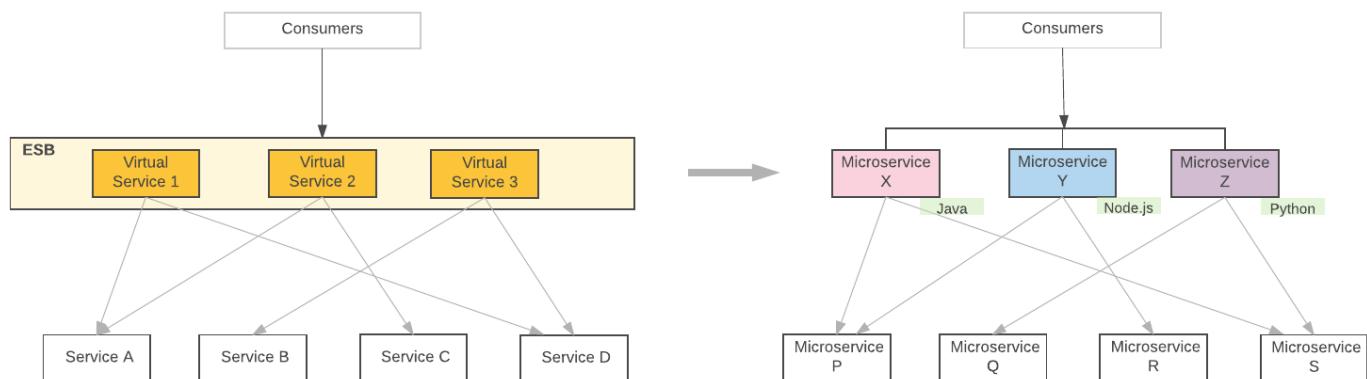
Padrões de Infraestrutura

Implantação

Infrastructure patterns



Arquitetura Service Mesh (multi-conectada) e Side Car (controle das requisições)



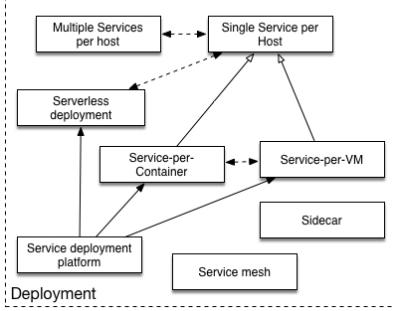
Exemplos:



Padrões de Infraestrutura

Implantação

Infrastructure patterns



Arquitetura Service Mesh (multi-conectada) e Side Car (controle das requisições)

Service Mesh

é uma **rede de microsserviços**

oferece: **segurança, descoberta, balanceamento e tolerância a falhas**

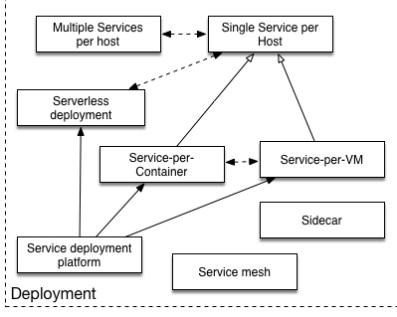
insere um **Side Car** junto da aplicação para controle das comunicações

funcionalidades de infraestrutura são usadas **fora do código da aplicação**

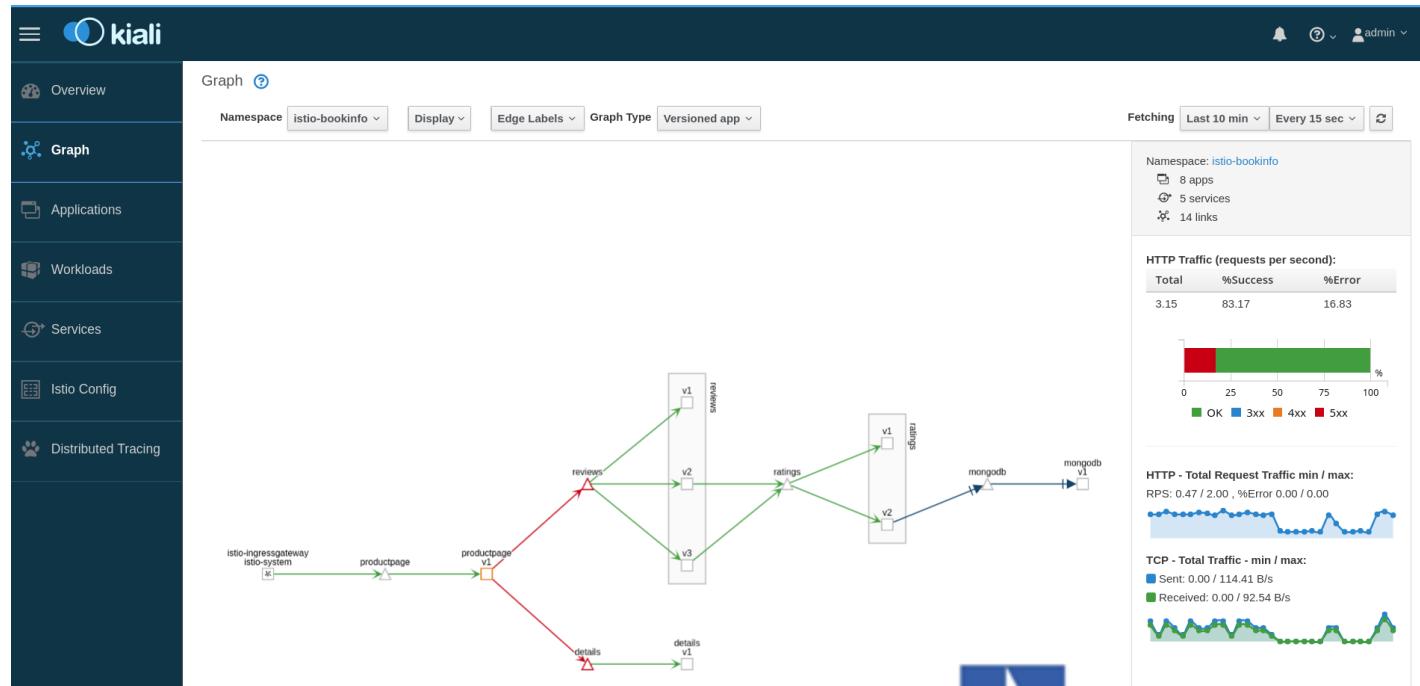
Implantação

Padrões de Infraestrutura

Infrastructure patterns



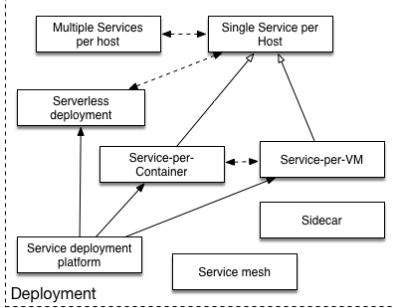
Arquitetura Service Mesh (multi-conectada) e Side Car (controle das requisições)



Padrões de Infraestrutura

Implantação

Infrastructure patterns



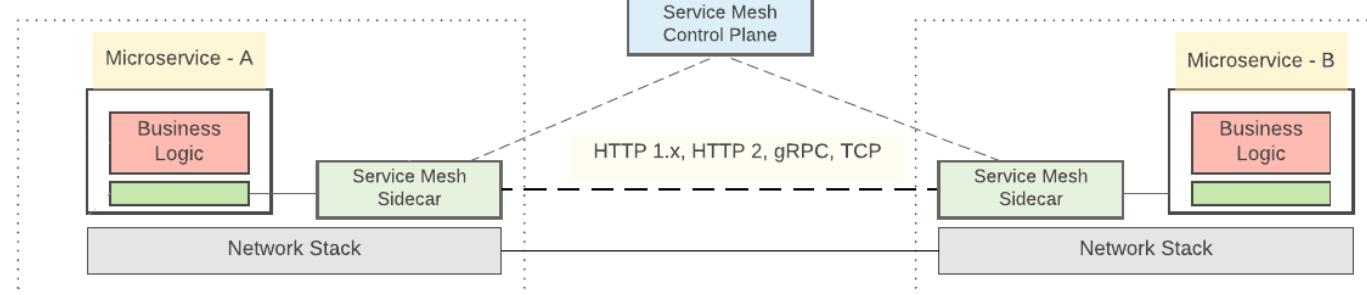
Arquitetura Service Mesh (multi-conectada) e Side Car (controle das requisições)

Monitoramento e Segurança

Balanceamento e failover

Descoberta de Serviço

Resiliência nas comunicações



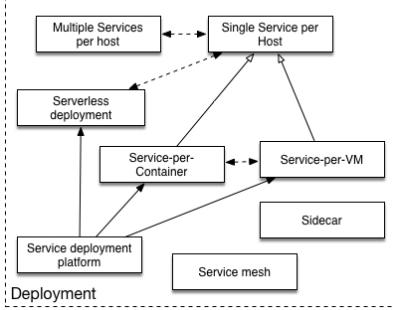
Padrões de Arquitetura

FIAP

Padrões de Infraestrutura

Implantação

Infrastructure patterns



Arquitetura Service Mesh (multi-conectada) e Side Car (controle das requisições)

Problema

Como lidar com **falhas** durante a comunicação dos serviços?

Solução

Utilizar uma plataforma de Service Mesh com **funções de controle**

Vantagens

Padrões prontos para problemas comuns

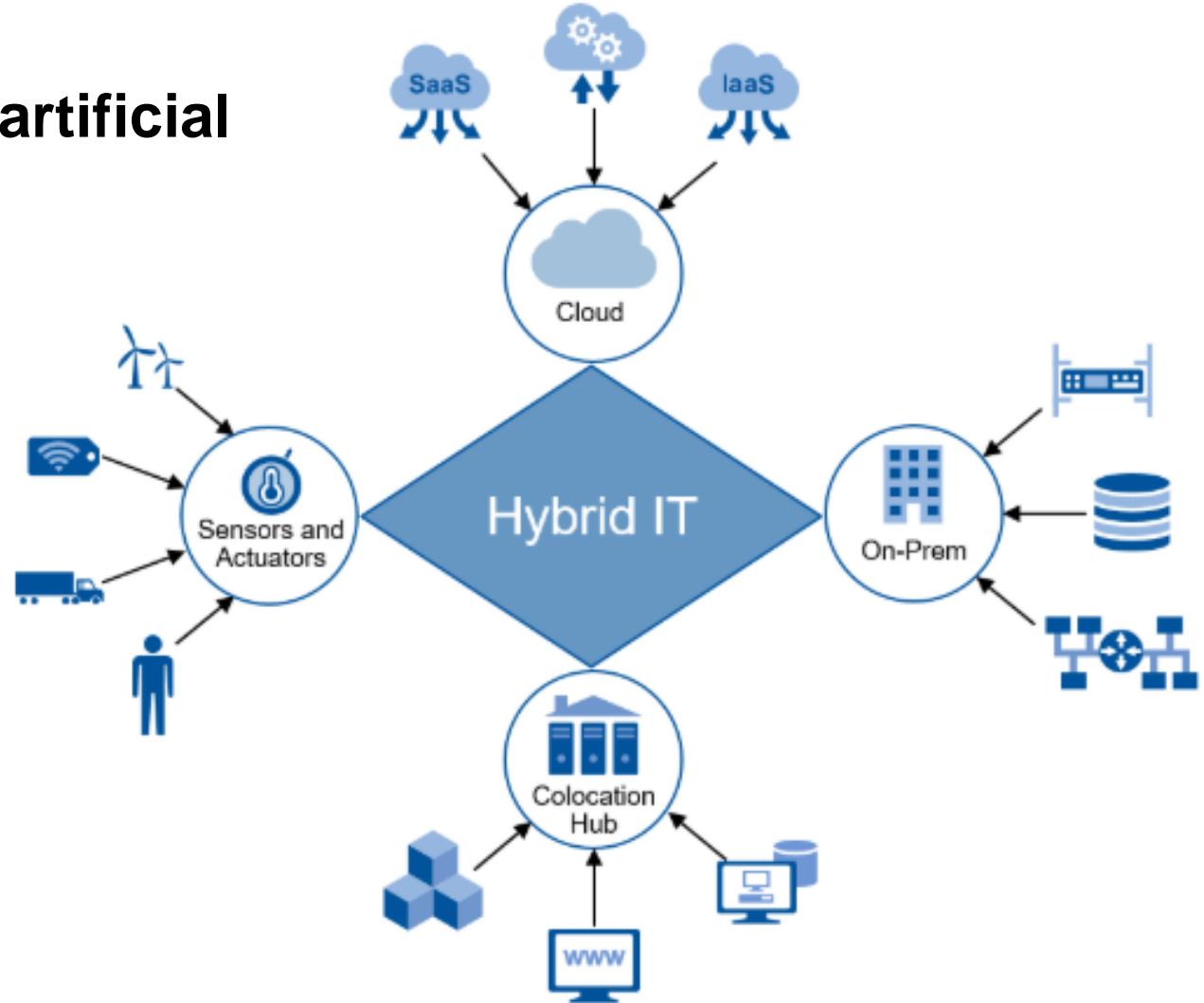
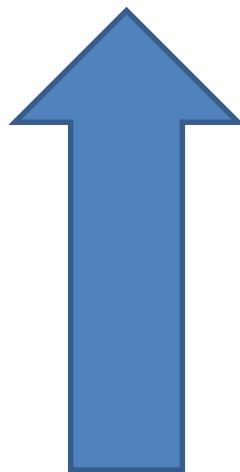
Liberar o foco para **requisitos de negócio**

Utilização de Microserviços

FIAP

Demandas crescentes :

- Internet das coisas
- Big Data
- Inteligência artificial

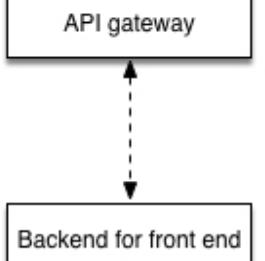


Padrões de Arquitetura

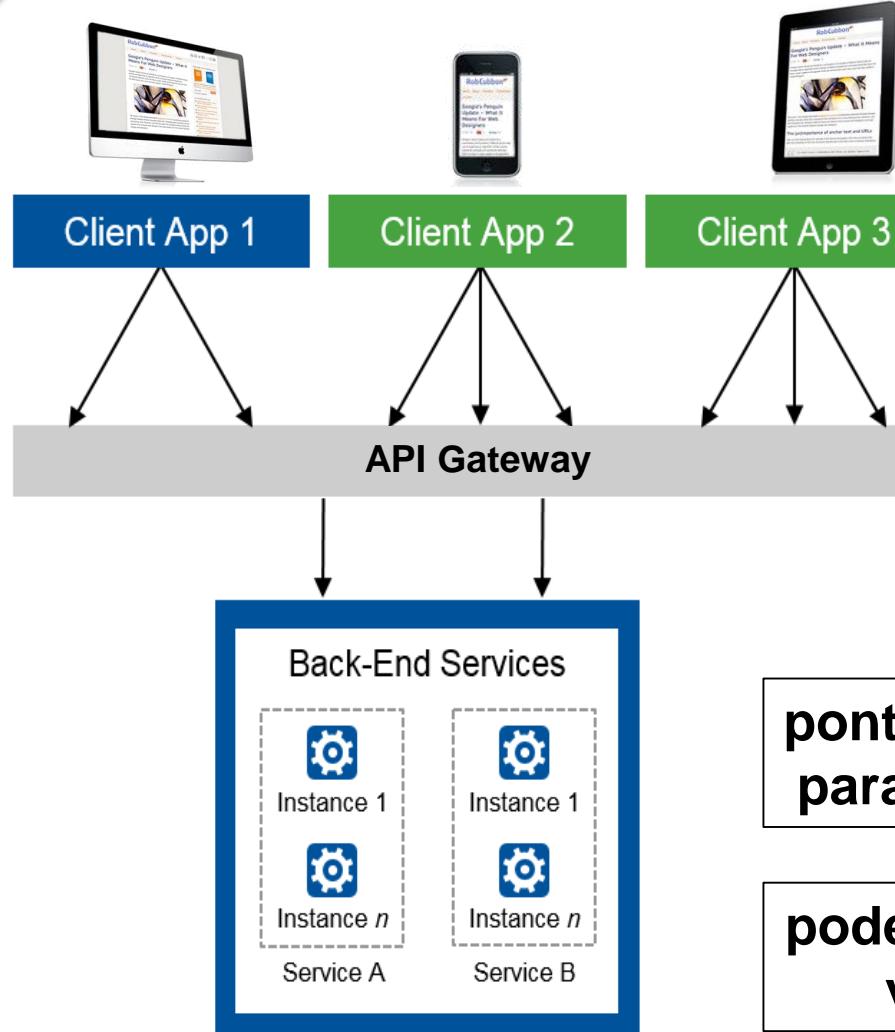
FIAP

Padrões de Infraestrutura

API Externa



API Gateway versus Backend For Front-end



ponto único de entrada
para todos os clientes

pode agregar dados de
vários serviços

Padrões de Arquitetura

FIAP

Padrões de Infraestrutura

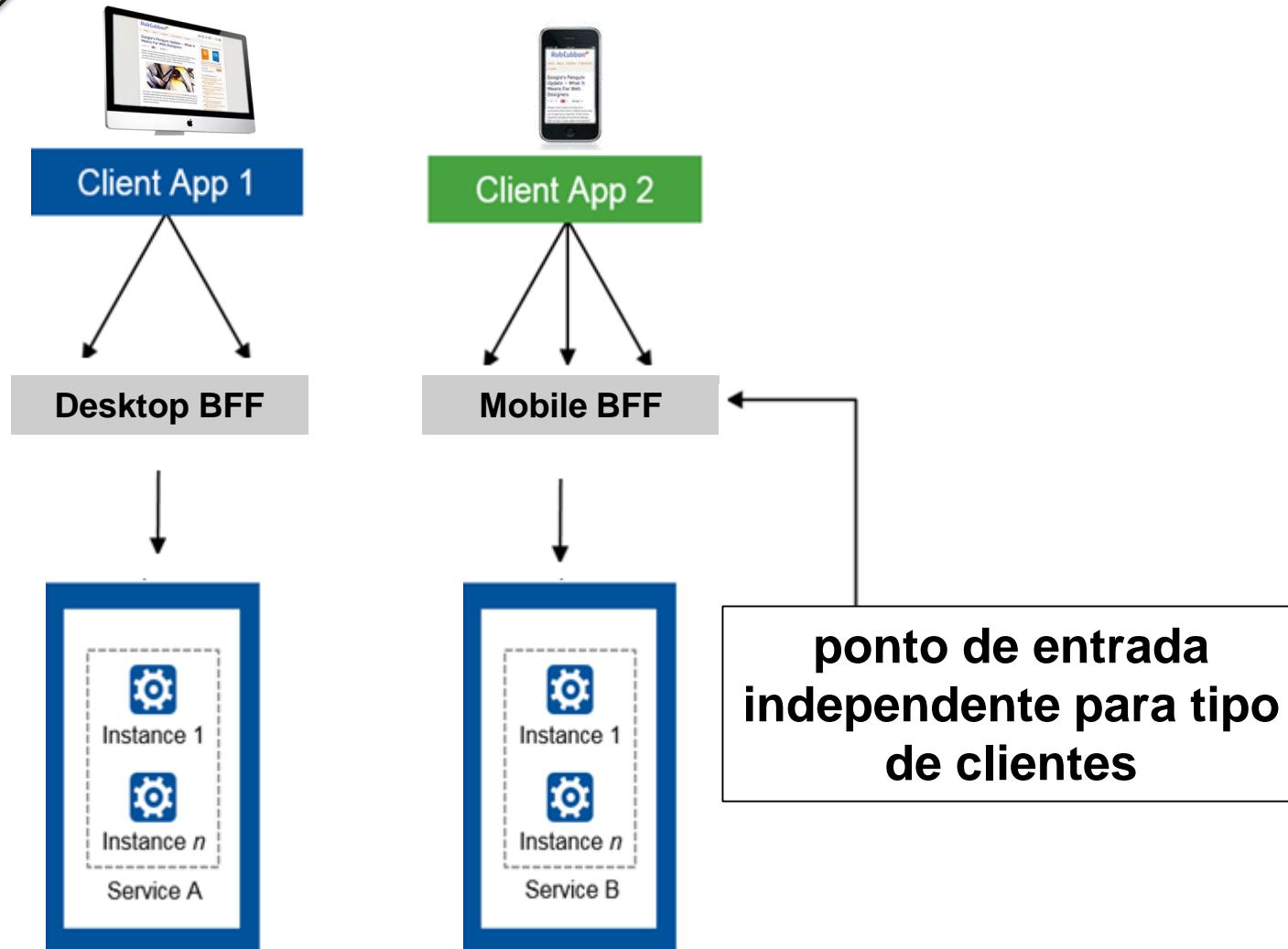
API Externa

API gateway

Backend for front end

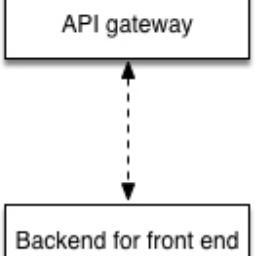
External API

API Gateway versus Backend For Front-end



Padrões de Infraestrutura

API Externa



API Gateway versus Backend For Front-end

Problema

Como os **clientes** de um serviço **podem** consumí-lo?

Solução

API Gateway agrega dados e transforma formatos

BFF fornece respostas para cada tipo de cliente

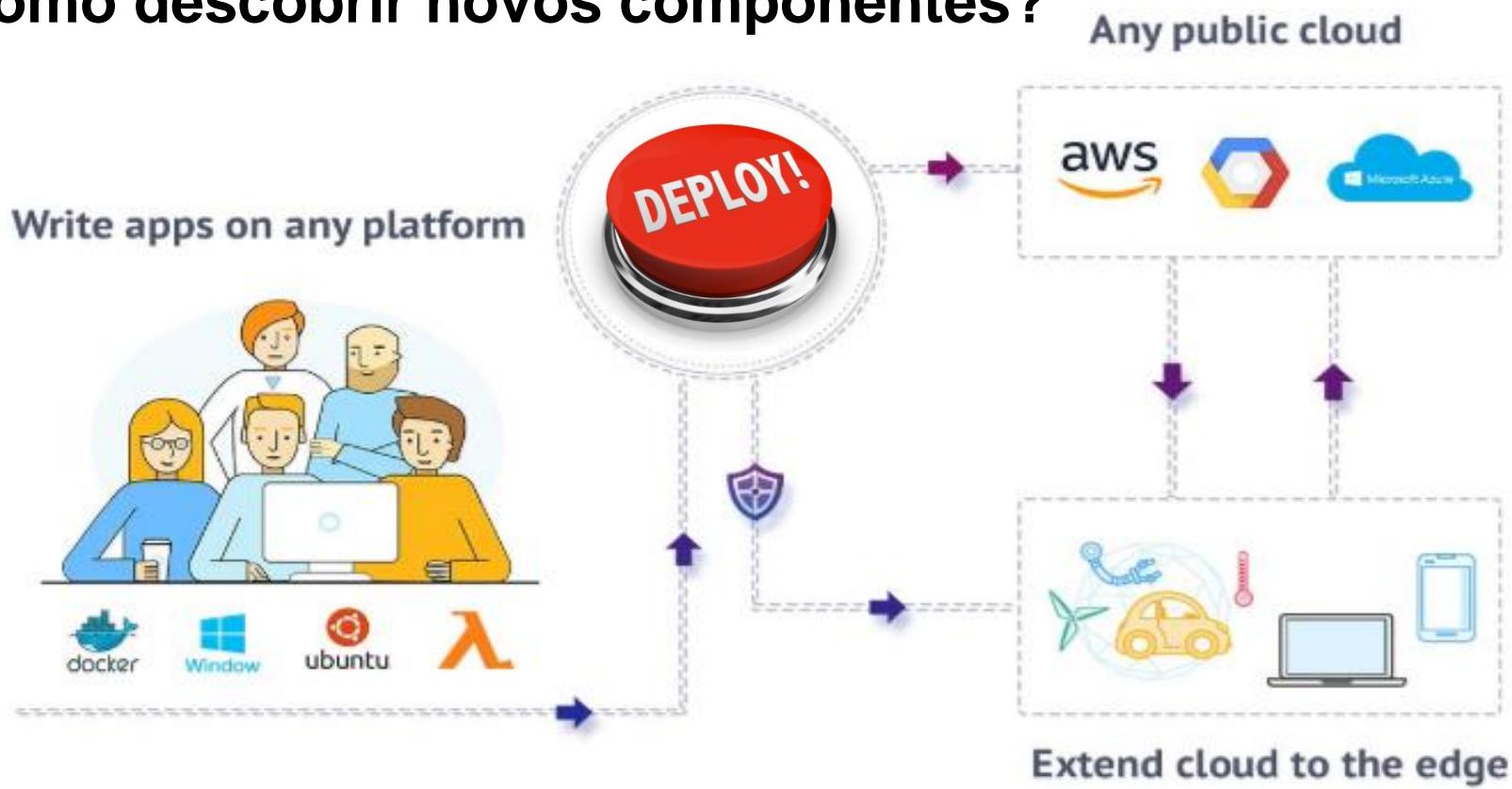
Vantagens

Abstração de configurações dinâmicas e localização

Centraliza controles de **segurança** e **auditoria**

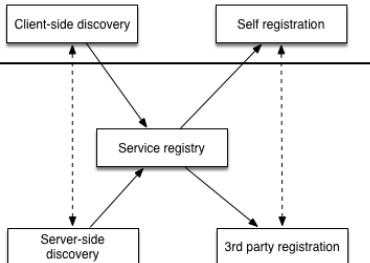
Contêineres permitem o *deploy* de Microserviços em diversos ambientes.

Como descobrir novos componentes?



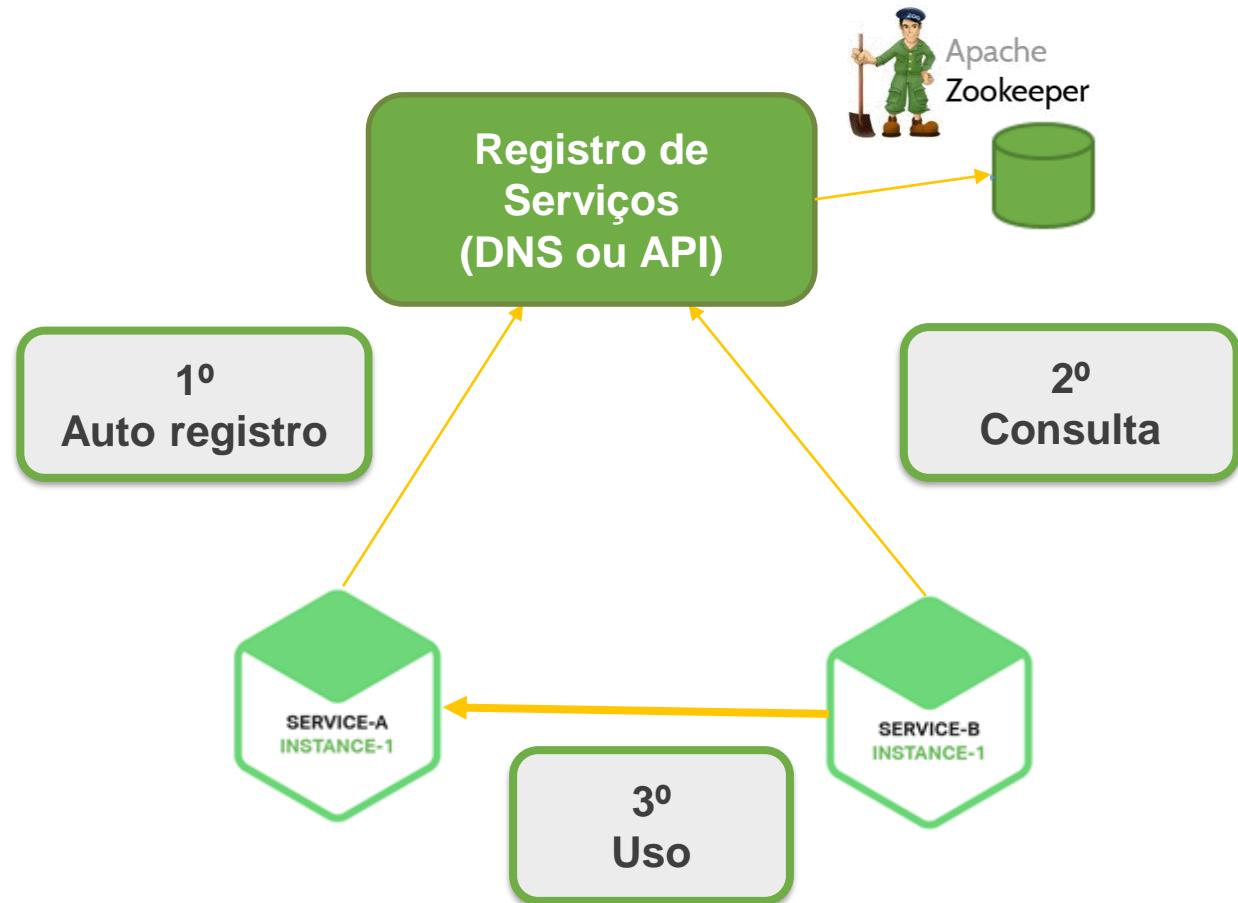
Padrões de Infraestrutura

Descoberta



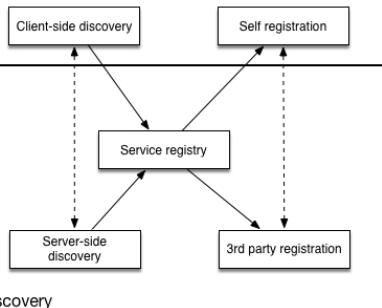
Descoberta de Serviços

Registro de Serviços (Service Registry)



Padrões de Infraestrutura

Descoberta



Descoberta de Serviços

Registro de Serviços (Service Registry)

Problema

Como um serviço pode ser **descoberto de forma dinâmica** em ambiente heterogêneo?

Solução

Utilizar um **Registro que mantém uma base** dos serviços associados

Vantagens

Descoberta dinâmica das instâncias do serviço

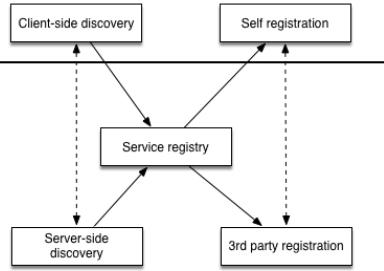
Agilizar o uso e descoberta de serviços

Padrões de Arquitetura

FIAP

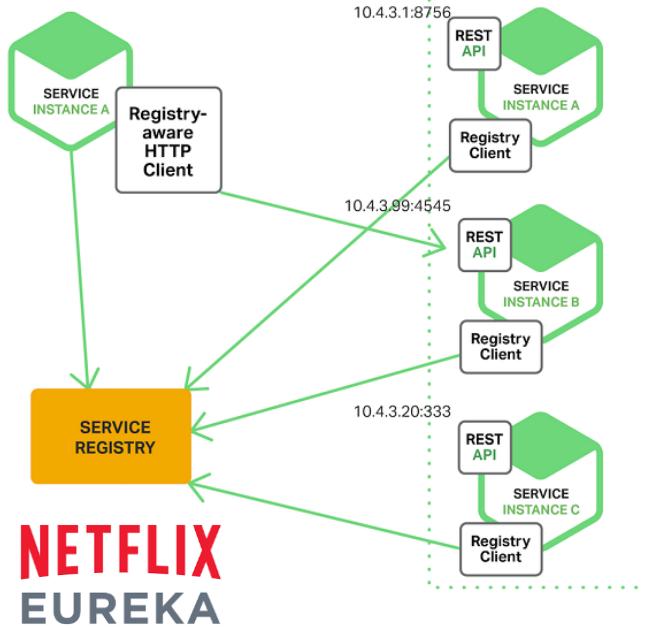
Padrões de Infraestrutura

Descoberta

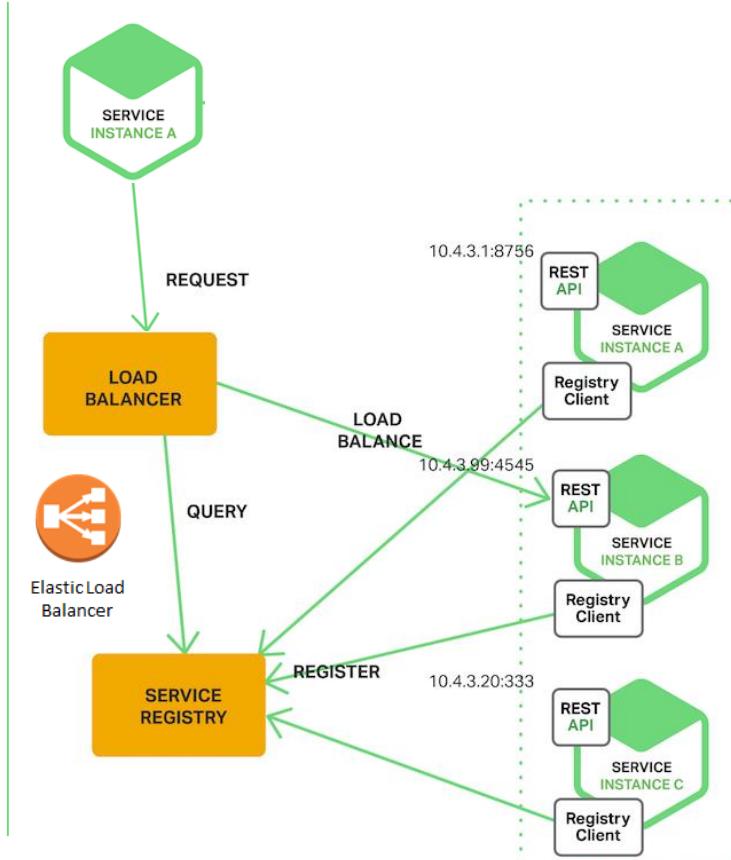


Descoberta de Serviços Client-side versus Server-side

NETFLIX
OSS

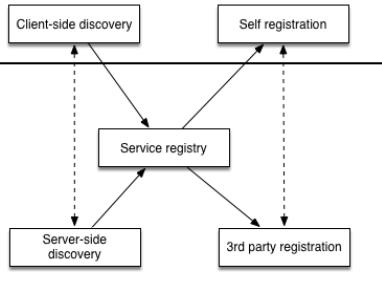


NETFLIX
EUREKA



Padrões de Infraestrutura

Descoberta



Descoberta de Serviços Client-side versus Server-side

Problema

Como o **cliente** de um serviço **descobre** sua **localização** para efetuar solicitações?

Solução

Cliente consulta um Registro para obter IP:porta

Servidor obtém **rotas** ao consultar um Registro

Vantagens

Padrões prontos para utilização

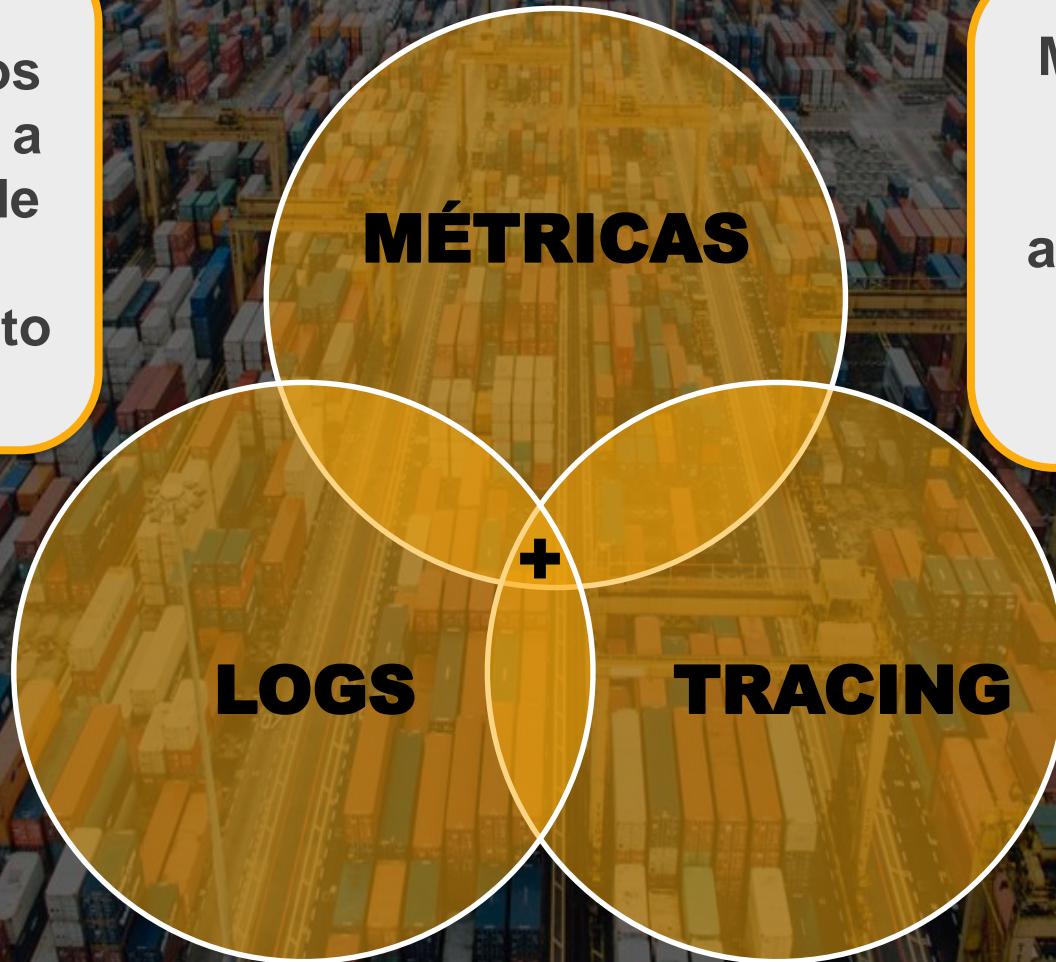
Liberar o foco para **desenvolver** o negócio

Monitoração de Microserviços

FIAP

Microserviços aumentaram a complexidade do gerenciamento

Monitoramento de serviços: requer coleta, armazenamento e análise de dados



*"If we have data, let's look at data.
If all we have are opinions, let's go with mine."*

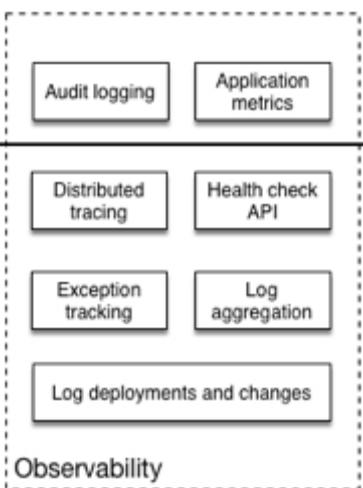
Jim Barksdale, Netscape CEO / AOL

Monitoração de Microserviços

FIAP

Arquitetura
Front-end

Monitoração



Arquitetura de
Aplicações

Arquitetura de
Contêineres

Arquitetura de
Infraestrutura

I

Agregação de log:
consolidar os diversos logs de aplicativos

II

Métricas de aplicativos:
instrumentam o código fonte para coletar estatísticas

III

Registro de auditoria:
atividades dos usuários são guardadas em um BD

IV

Rastreamento distribuído:
armazena horário de início e término de cada chamadas

V

Rastreamento de exceções:
registra todas as exceções, notificando os desenvolvedores

VI

API de verificação de integridade:
verificar funcionamento do serviço

VII

Log de implantações e alterações:
Rastrear deploys no ambiente (de produção)

Monitoração de Microserviços

FIAP

Arquitetura
Front-end

Arquitetura de
Aplicações

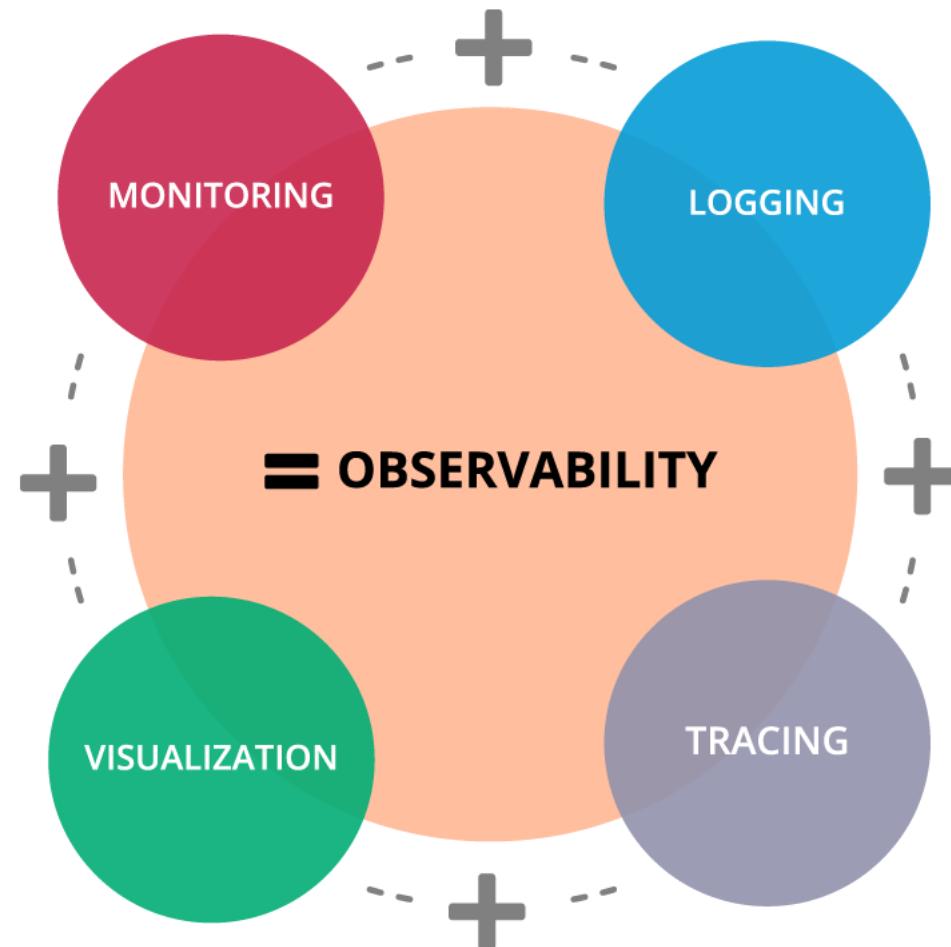
Arquitetura de
Contêineres

Arquitetura de
Infraestrutura

Monitoração

- Audit logging
- Application metrics
- Distributed tracing
- Health check API
- Exception tracking
- Log aggregation
- Log deployments and changes

Observability



LAB: 4

FIAP





Desenvolvimento
ágil trouxe
responsabilidades
compartilhadas

Quem será o
novo público
desses
relatórios?

Monitoração de Microserviços

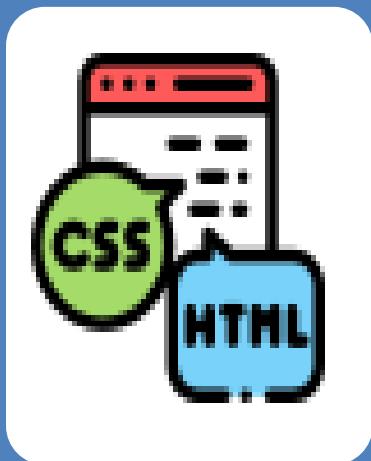
FIAP





Monitoração

- Saúde e desempenho de serviços críticos de negócios
- APIs consumidas por HTTP
- Dados sobre o uso podem gerar conhecimento para tomar decisão



DevOps

- trabalho colaborativo para entregar o serviço
- fornecimento rápido de valor
- automatiza ciclos de lançamento (CI/CD)

Monitoração de Microserviços

FIAP



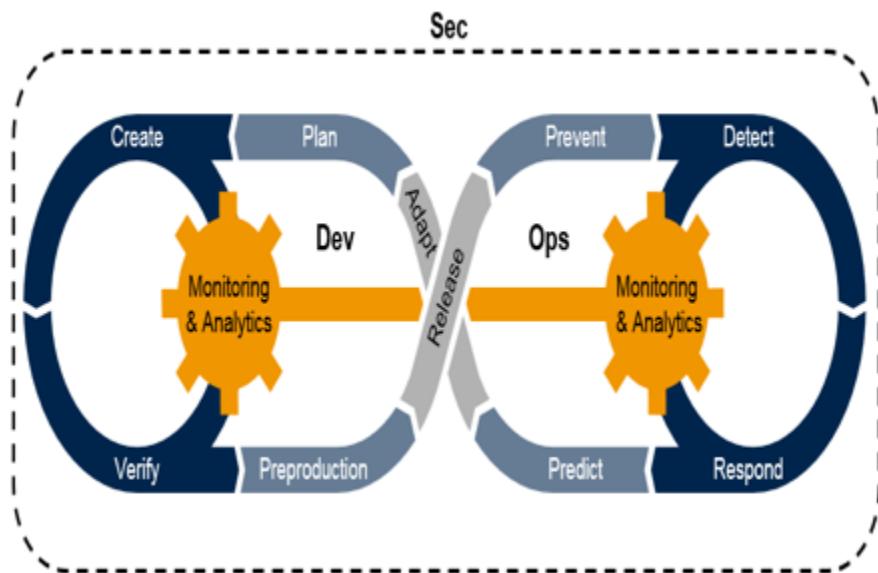
Negócios

- BizDevOps



Segurança

- DevSecOps



Métricas

registro de um valor em determinada data e hora

utilizam banco de dados de séries temporais

Logs

possuem formatos variados

convertidos de texto para um formato definido

consome muitos recursos

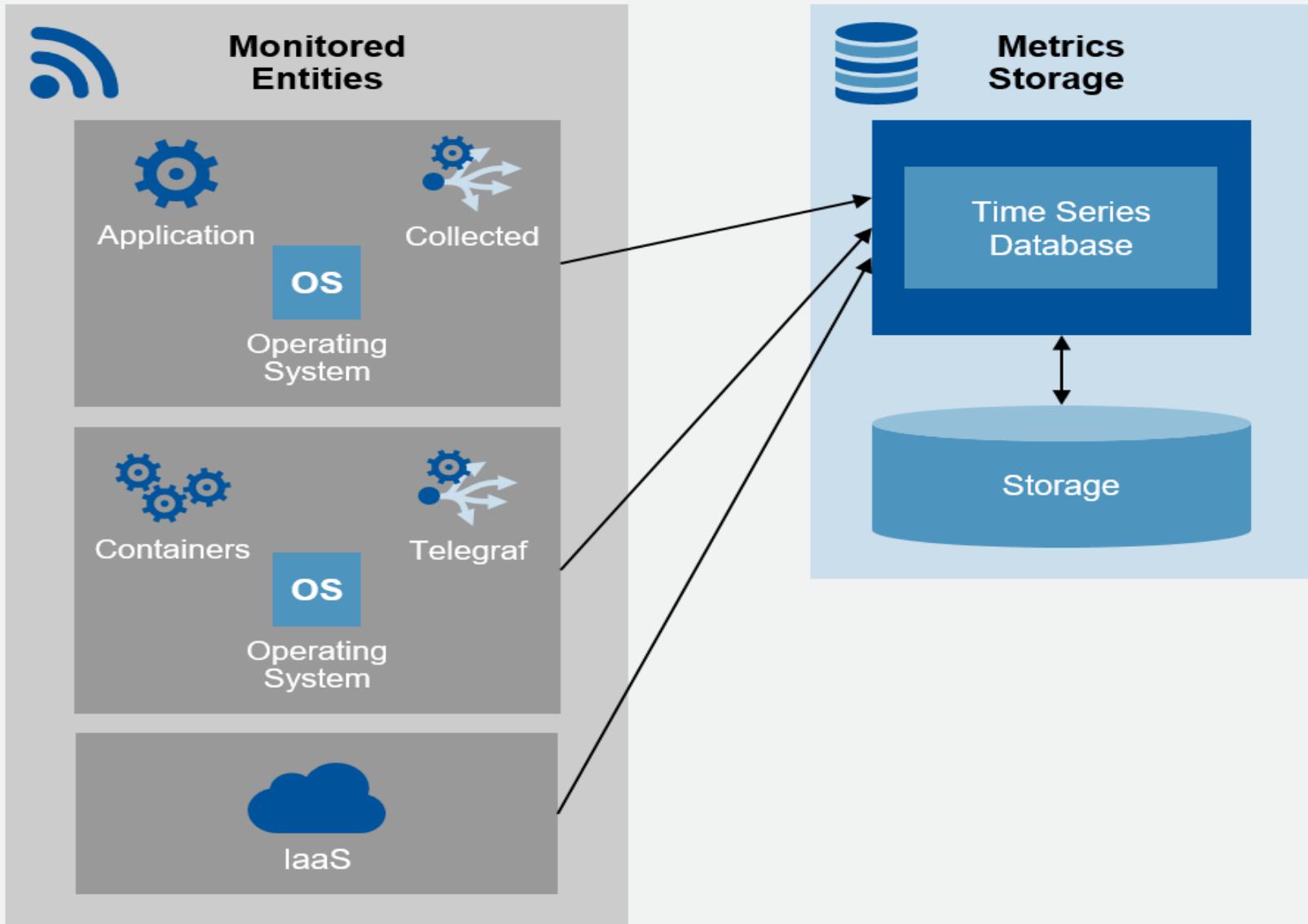
Tracing

instrumenta-se código fonte de aplicativos

avaliam anomalias de desempenho

BD de Séries Temporais

FIAP



Banco de dados Temporal

Representação do
passado, presente e
projeção de futuro

Gerenciamento temporal
controlado pelo BD

Requer mais espaço, pois
possui **crescimento**
acelerado

Banco de dados Relacional

Representação do estado
presente de um dado

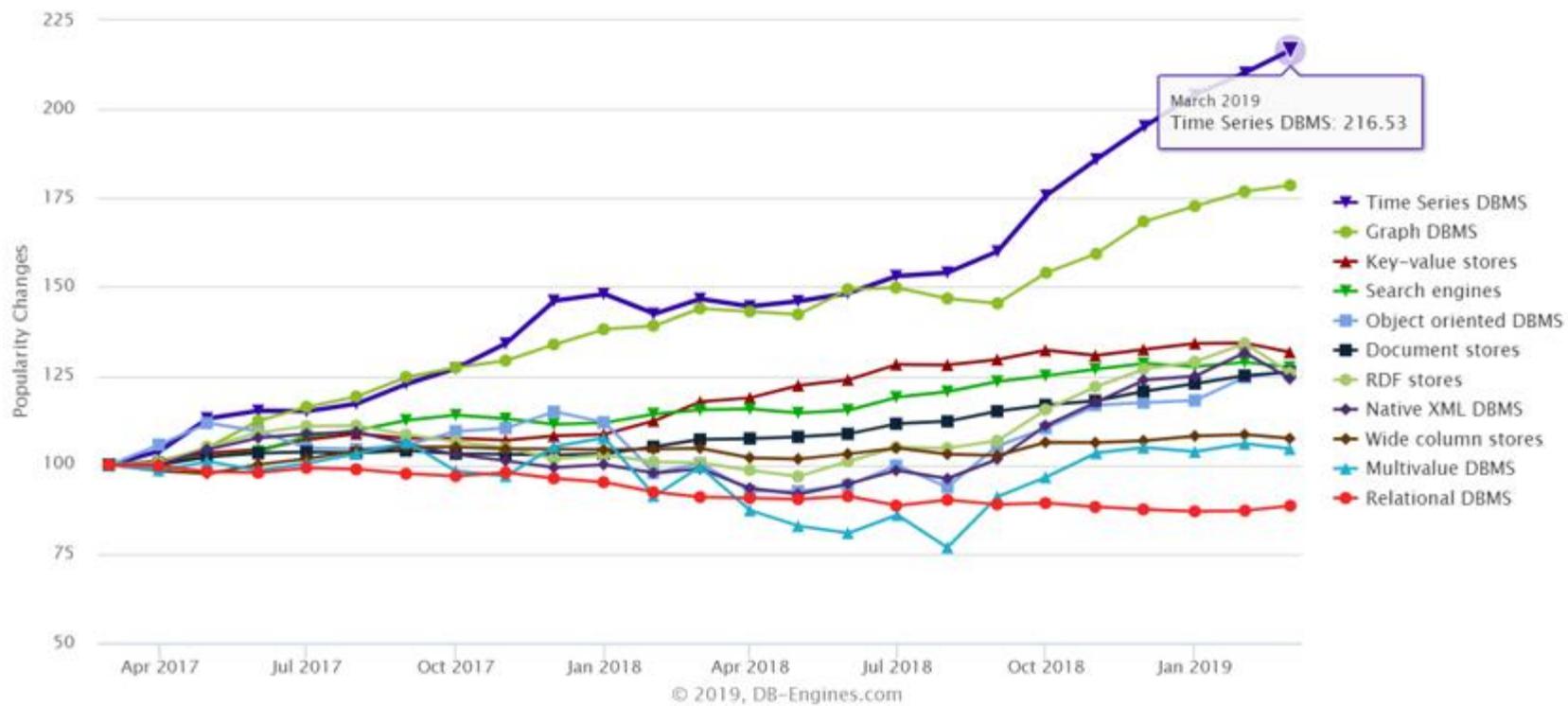
Gerenciamento temporal **a**
cargo da aplicação

Sobrescreve dados,
descartando os antigos,
otimizando espaço

BD de Séries Temporais

FIAP

Trend of the last 24 months



Time Series DBMS (Prometheus, Influxdb)

Relational DB (PostgreSQL, MySQL)

Search engines (Solr, Elasticsearch)

Document Stores (MongoDB, CouchDB)

Key/Value Stores (Redis, RocksDB)

Column Stores (HBase, Cassandra)

Graph DBMS (Neo4J, JanusGraph)

Monitoração de Microserviços

FIAP

Métricas

Logs
Tracing

Monitoramento
caixa preta

Monitoramento
caixa branca

Responder: "Está
funcionando?"

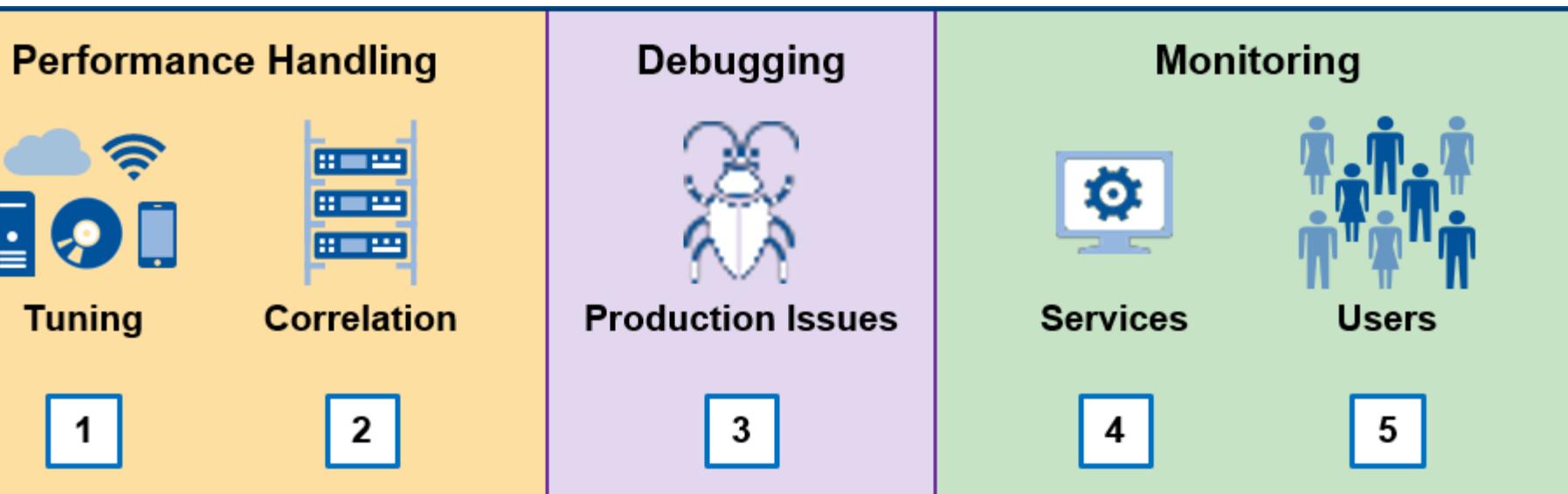
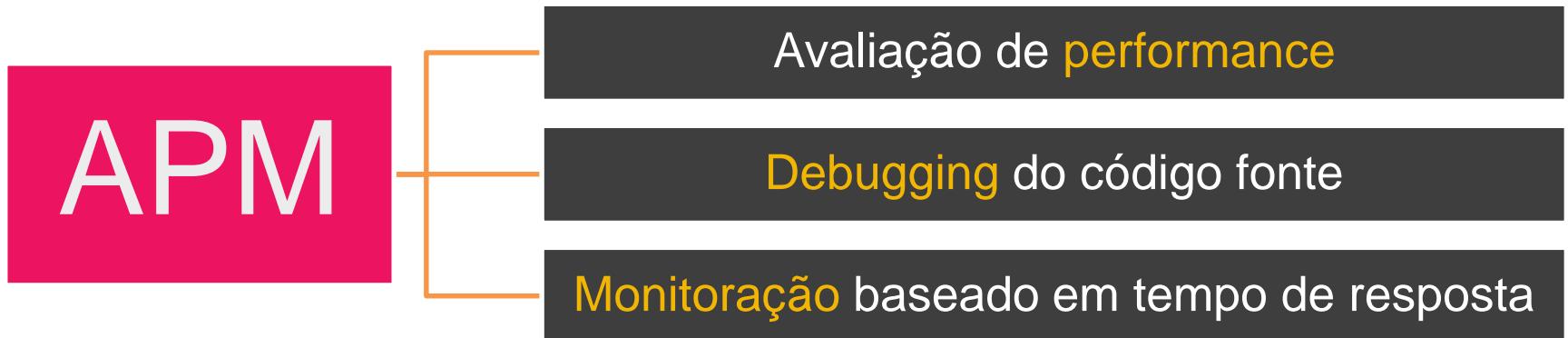
Avalia disponibilidade ou
desempenho

Responder: "Por que não
está funcionando?"

Detalhes sobre os
componentes do sistema

Monitoração de Microserviços

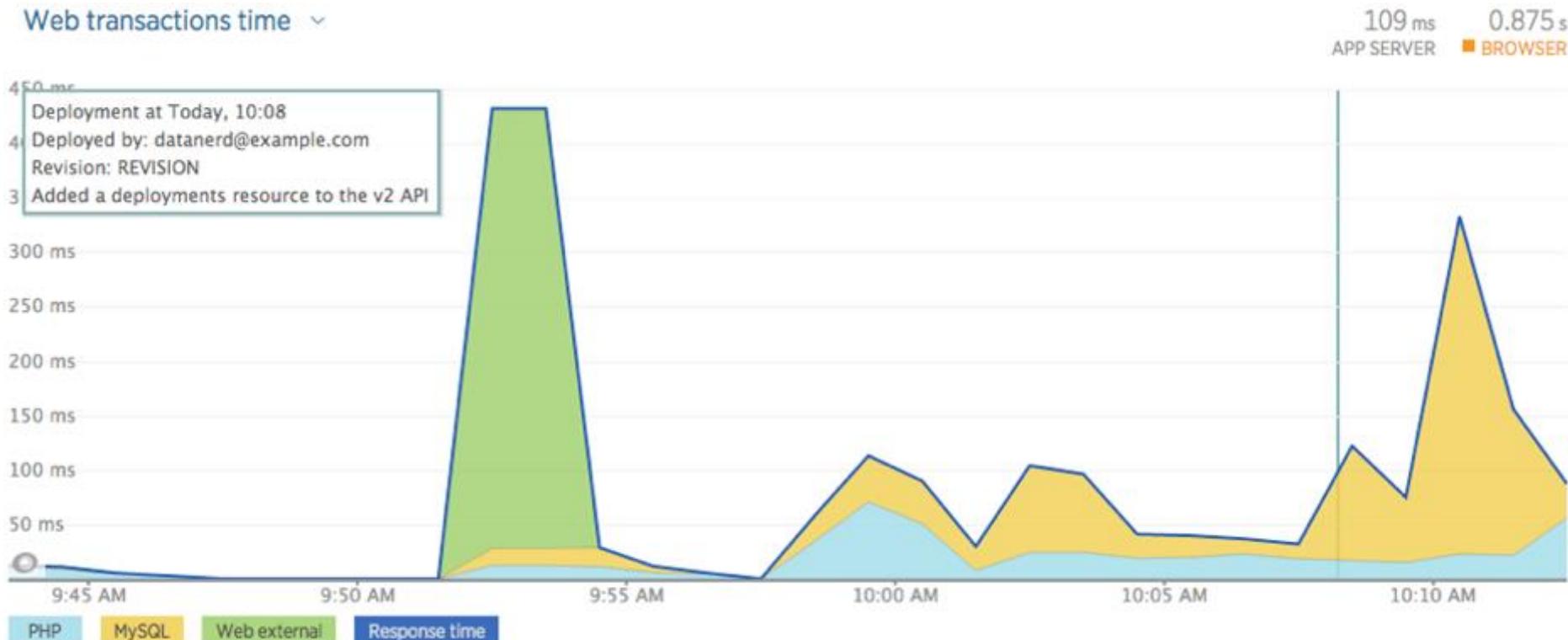
FIAP



APM

— Avaliação de performance

Onde está a causa de lentidão no serviço?



APM

— Avaliação de performance

BASELINE: Comparação DEV versus PROD

Production

Overall: 600 ms

Java object: 300 ms

SQL call: 250 ms

Development

Overall: 900 ms

Java object: 400 ms

SQL call: 350 ms

Correlation

$900/600 = 1.5$

$400/300 = 1.333$

$350/250 = 1.4$

A change of the development SQL call to 200 ms indicates an estimated change to 142 ms on production (200/1.4 differential).

APM

— Debugging do código fonte

Quais eram os parâmetros no momento da lentidão?

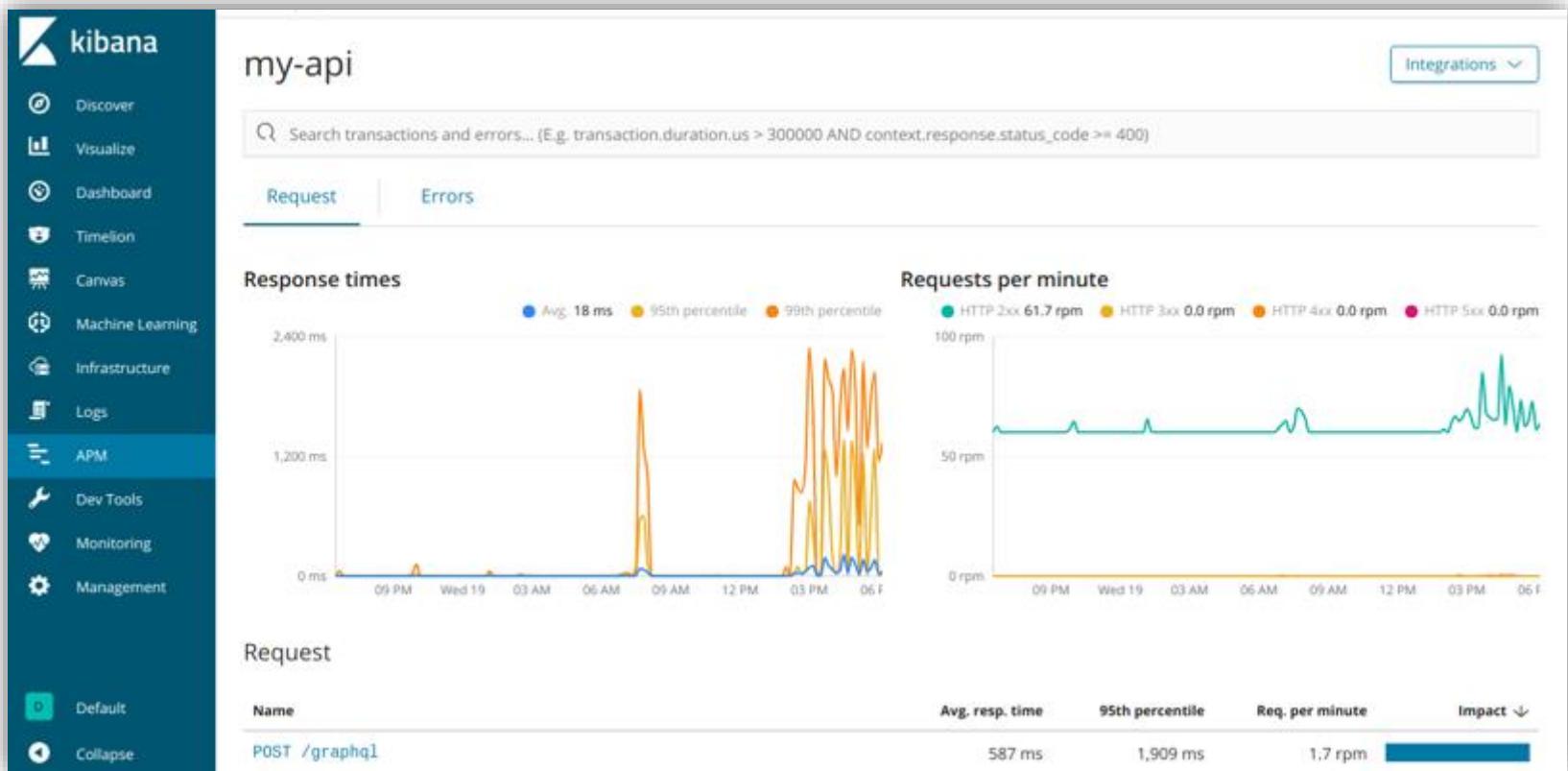
The screenshot shows the Windows Event Viewer interface for a Performance Event ID 6141. The window title is "Performance Event ID 6141 - Windows Internet Explorer". The "Actions" menu is open. The "Event properties" tab is selected. The "Similar events" tab is visible. The "Problem" dropdown shows "Problem" and the URL "/dinnernow/search.aspx". The "Group By" dropdown shows "None" and the "Include light events" checkbox is unchecked. The "From" and "To" fields show the date range from 10/6/2011 12:17:50 to 10/20/2011 12:17:50. An "Apply" button is present. Below the search controls, it says "First Event Date: 10/20/2011 11:23:20 AM" and "Last Event Date: 10/20/2011 12:17:50 PM". A table below lists the event details:

ID	Description	Duration	Computer	Status	Local Date
6141	/dinnernow/search.aspx? PostalCode=98101&MenuType=Breakfast&RestaurantCategory=48e4138c-b465- 440c-ad31-c5cc0b... [6,890 ms] slow at DinnerNow.WebUX.MenuSearchService.IMenuSearchService.GetRestaurantCategories () [3,552 ms]	6,890 ms.		New	10/20/2011 12:17:50 PM

APM

Monitoração

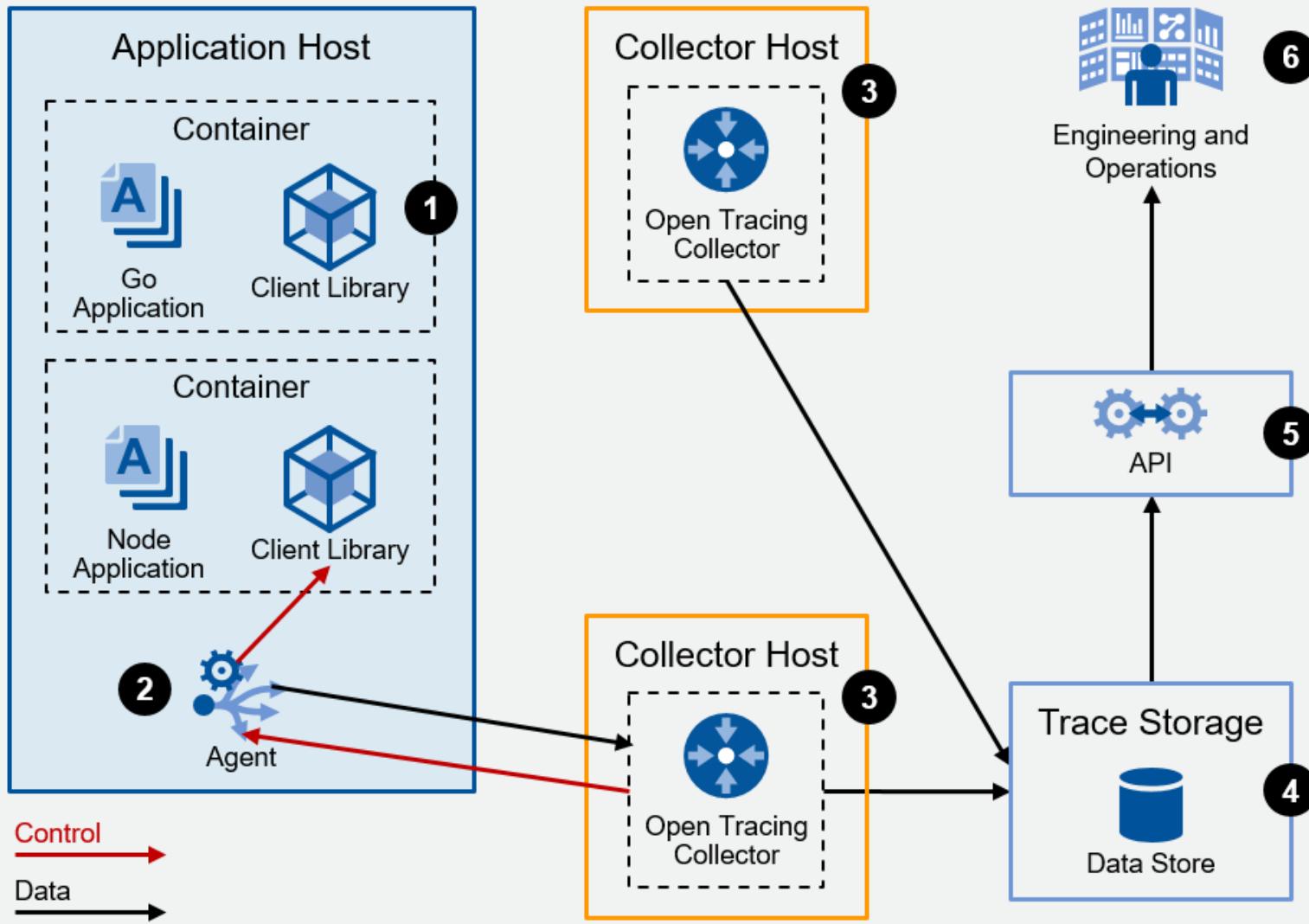
Tempo de carregamento das páginas



Arquitetura de uma solução APM

FIAP

Distributed Tracing Architecture



STACKS
Open-source

ELK (fornecido pela empresa Elastic)

Prometheus + Grafana

TICK (baseado no Influxdb)



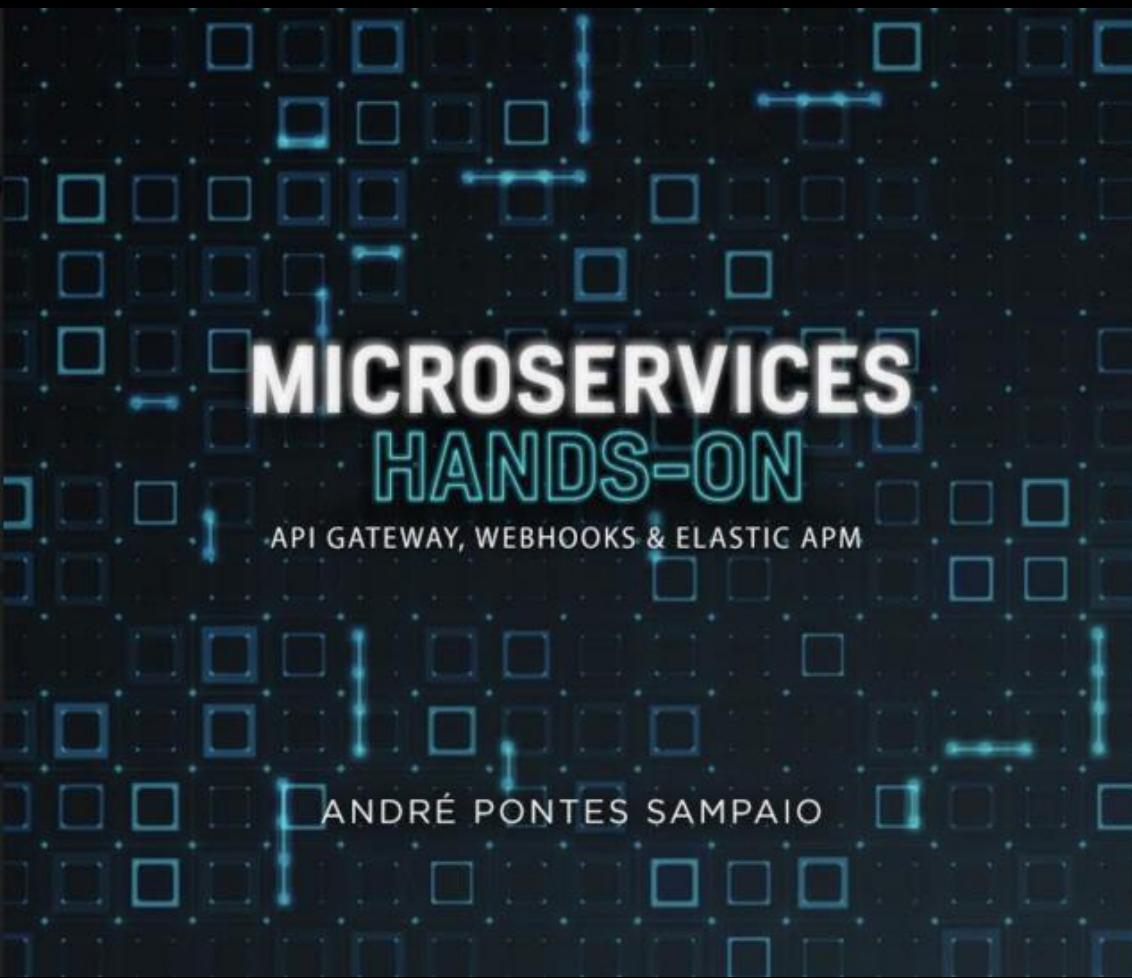
Vamos nos Divertir!

FIAP



O que vimos até aqui:

- Dividindo o Monolítico através de uma arquitetura escalável
- Exemplos de ferramentas que podem ser utilizadas no fornecimento de microserviços
- Monitoramento: coleta, armazenamento e análise
- Observabilidade: Métricas; Logs e Tracing (APM)



Copyright © 2019

Prof. André Pontes Sampaio

<https://www.linkedin.com/in/andre-pontes-sampaio/>

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).