

Reúso em Arquitetura Microserviços

An illustration showing several stylized human figures from a top-down perspective, reaching out to touch or hold large, interlocking gears. The gears are colored yellow, teal, red, and dark blue. The figures are also in various colors, representing diversity. The entire scene is set against a light blue background.

Olá!

Ali de França Husseinat

Bruno Lanzoni Rossi

Tiago Marino Silva

Otávio Luis de Aguiar

Microserviços

Conceitual

“*Microserviços* – um novo nome nas populosas ruas da arquitetura de software.”

- Martin Fowler

O que são microsserviços?

- Novo estilo de desenvolvimento
- Abordagem arquitetônica de decomposição da aplicação
- “Suíte de serviços”

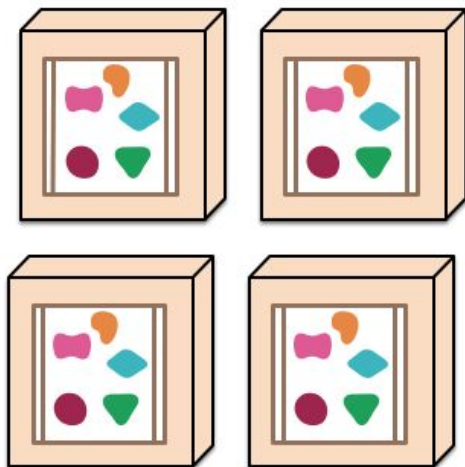
Em resumo: um microsserviço é uma pequena aplicação que executa uma única tarefa e o faz com eficiência.

Monolítica vs Microserviços

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



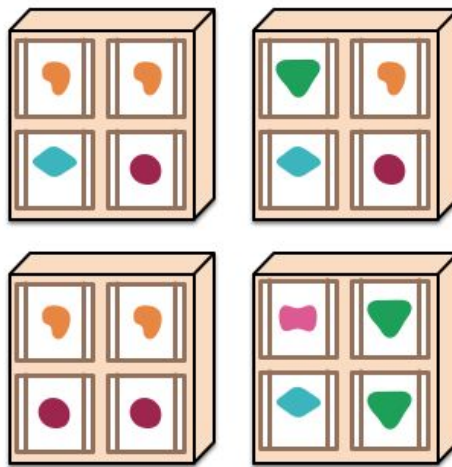
... e escala replicando a aplicação monolítica em vários servidores



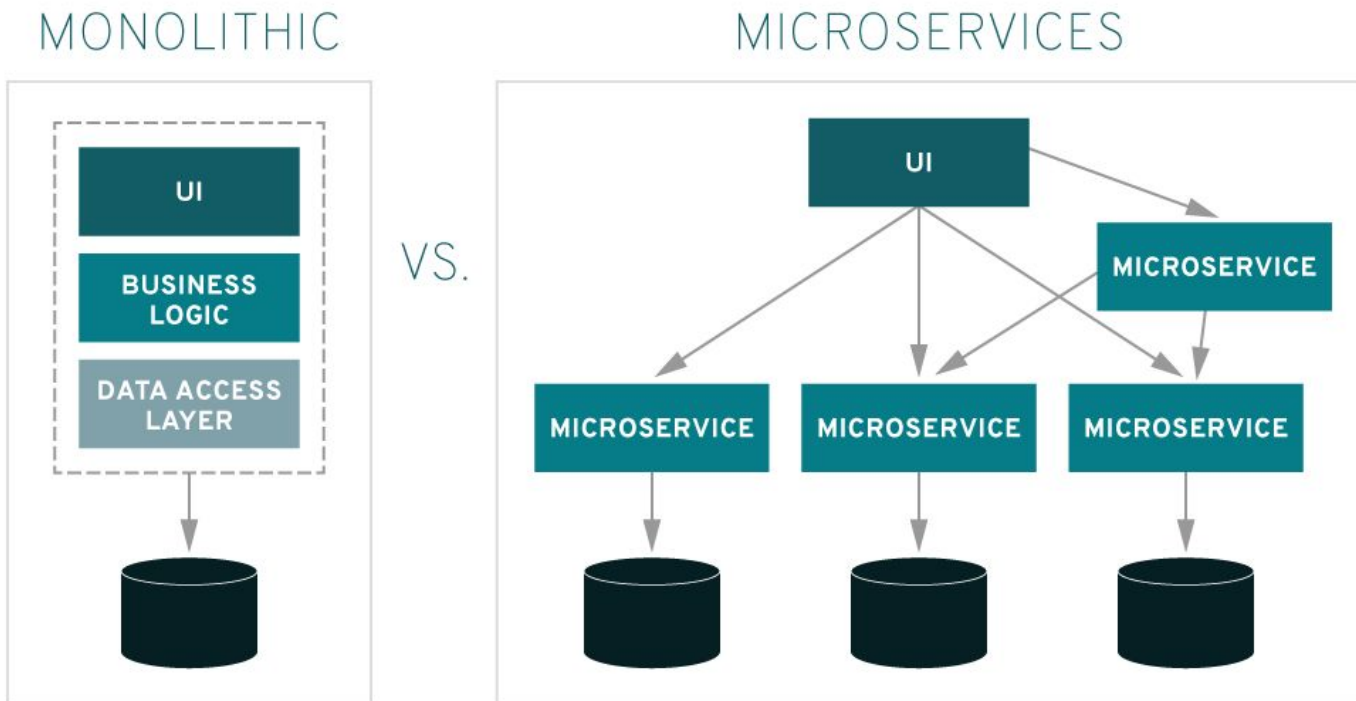
Uma arquitetura em microserviços põe cada elemento de uma funcionalidade em um serviço separado ...



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



Monolítica vs Microserviços





Propriedades dos microserviços

Características, princípios e
benefícios

Princípios

Alta coesão

Um único foco

Autonomia

Serviços independentes

Resiliência

Reagir a falhas inesperadas

Observável

Monitoramento em tempo real

Automatização

Integração contínua

Centrado no domínio do negócio

Contextos delimitados

Benefícios

Escala independente

Apenas o necessário

Atualizações independentes

Integração/entrega contínuas

Manutenção fácil

Fácil de entender

Liberdade tecnológica

Incentivo para inovação

Isolamento de falhas e resiliência

Falhas podem ser isoladas

Código reutilizável

Diversas possibilidades

Tópicos principais



Conceitual

Implementação
Moderna

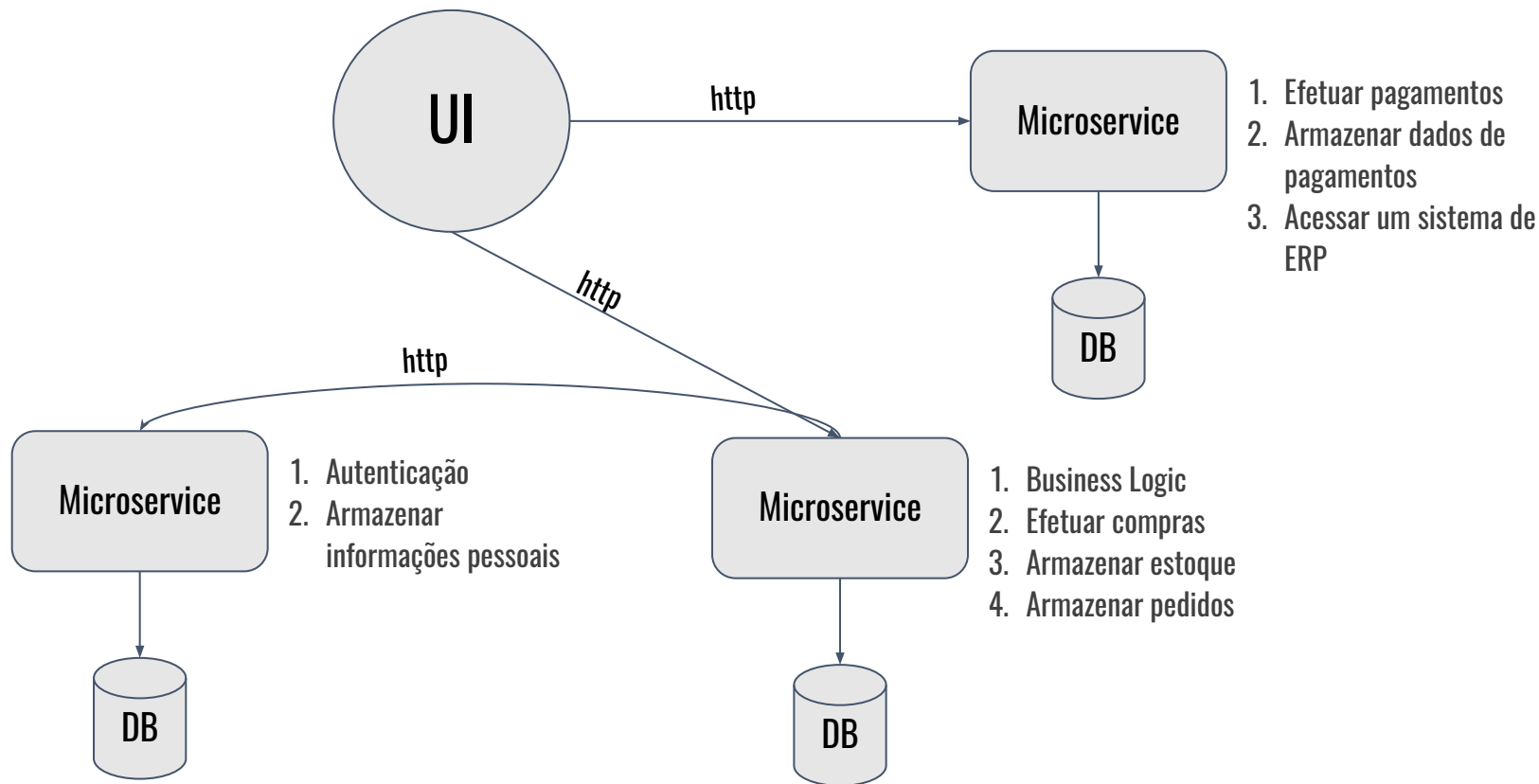
Reúso

Implementação Moderna

Como utilizar o padrão de microsserviços colabora para uma arquitetura de alta disponibilidade?

Alta disponibilidade

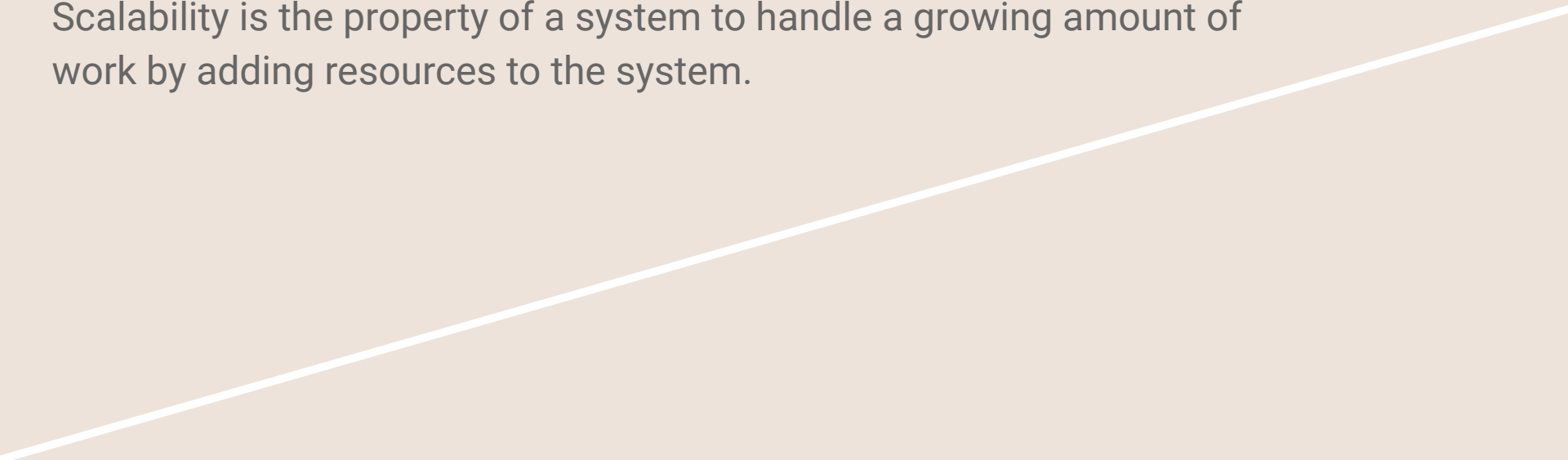
High availability (HA) is a characteristic of a system which aims to ensure an agreed level of operational performance, usually uptime, for a higher than normal period.



O que fazer quando temos
um aumento na carga do
sistema?

Escalabilidade

Scalability is the property of a system to handle a growing amount of work by adding resources to the system.



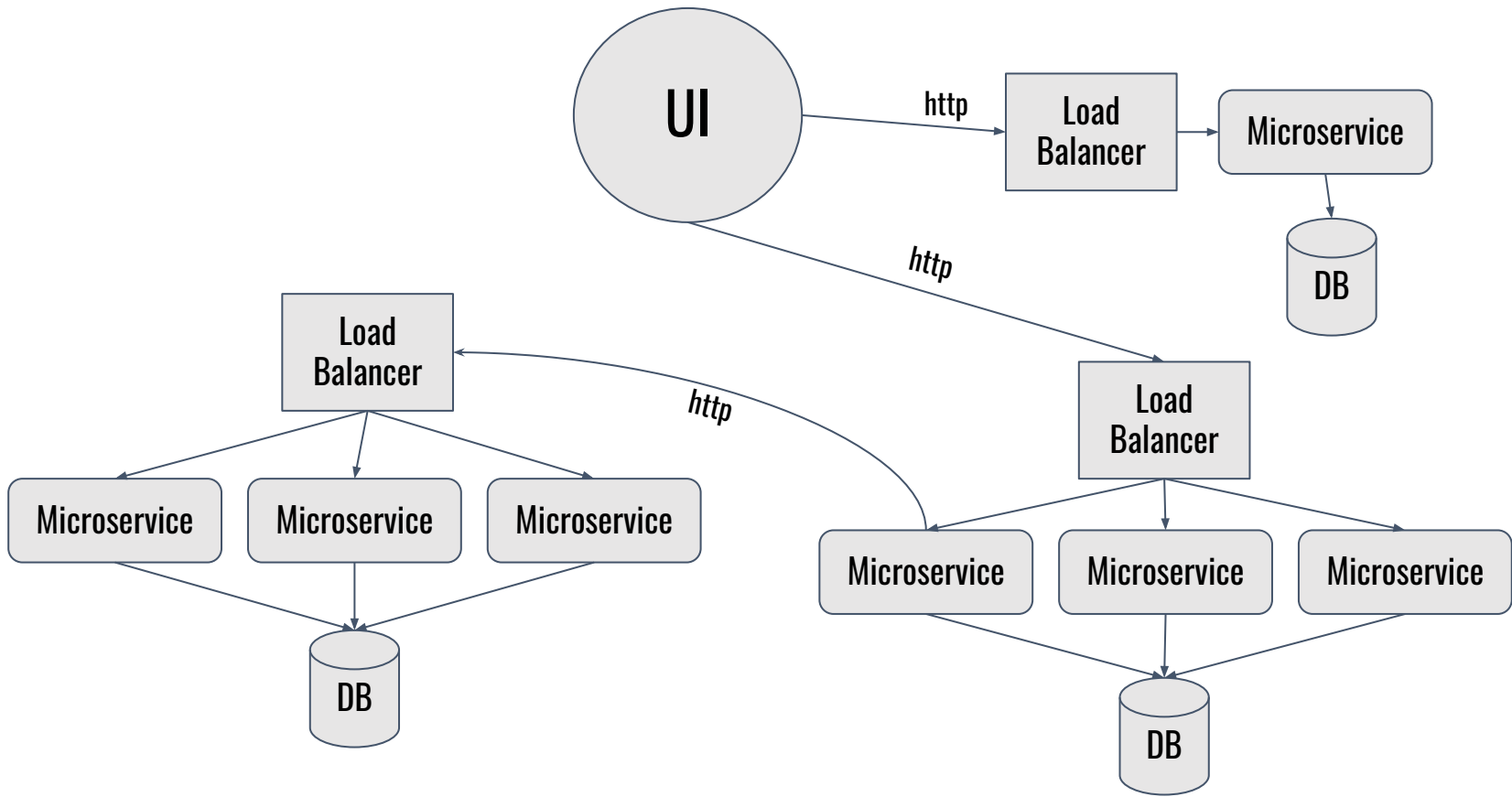
Porquê MSs escalam com mais facilidade?



Facilidade em
virtualização

Rapidez no
startup

Sistemas
altamente
desacoplados



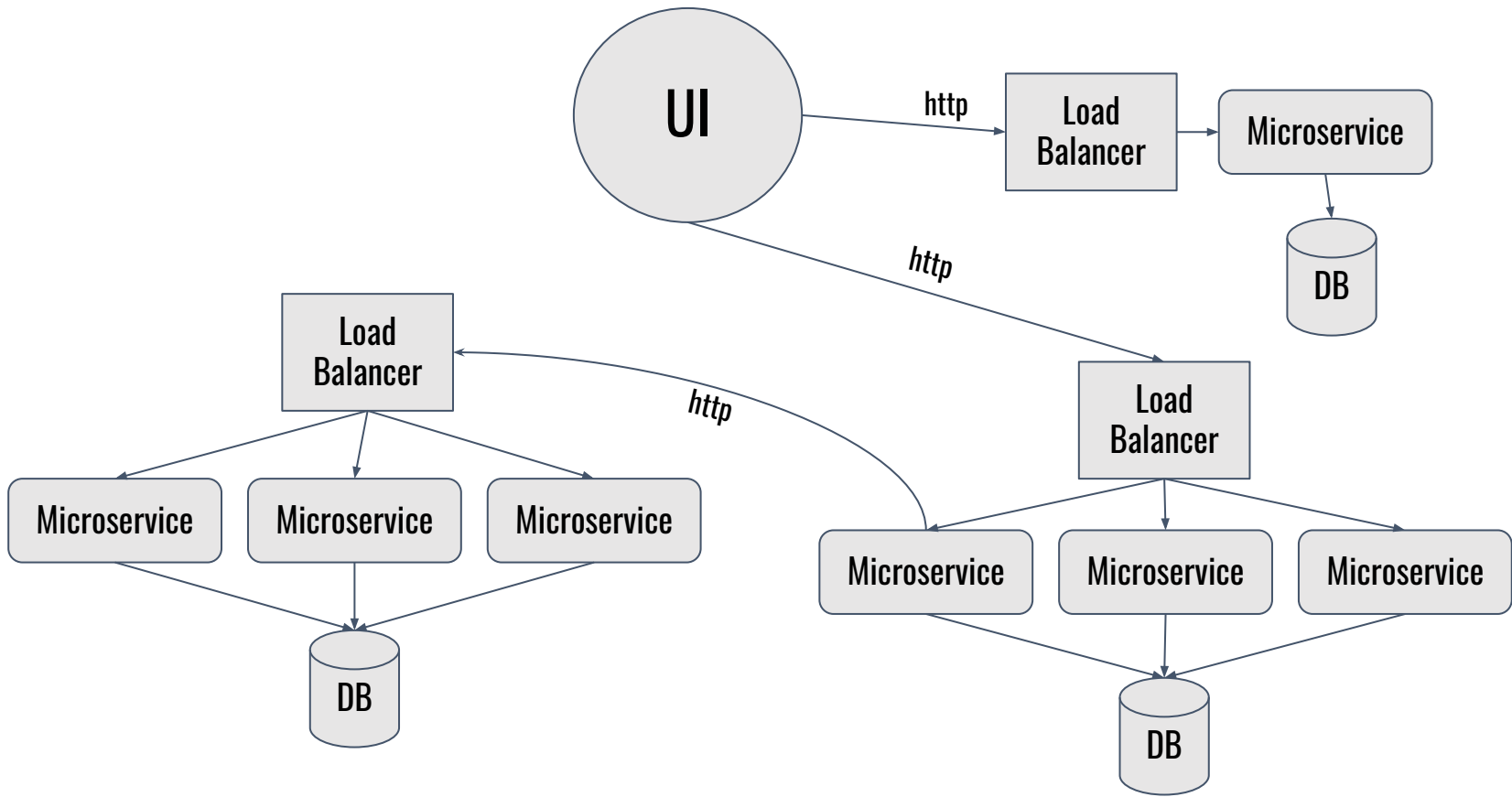


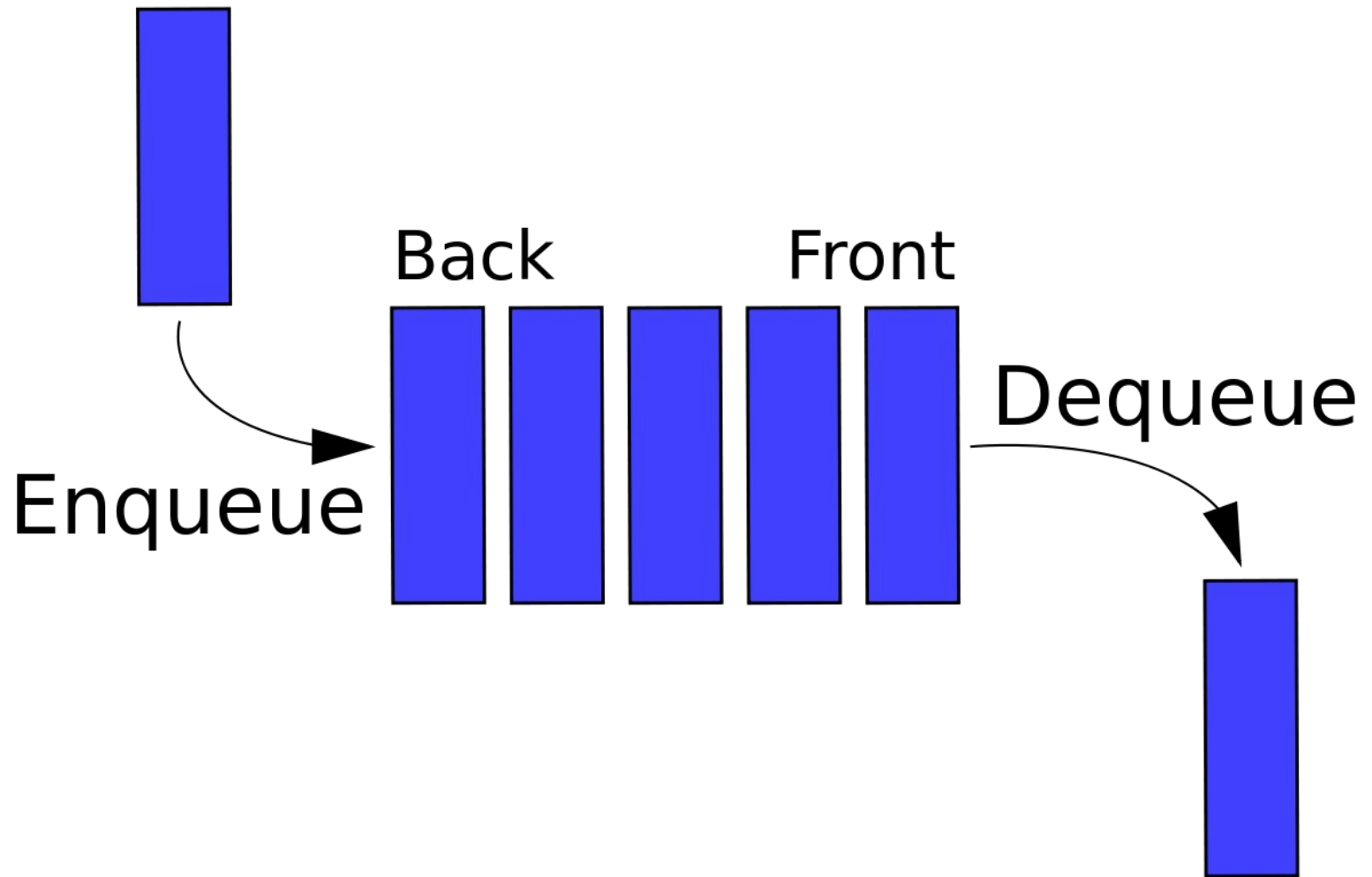
kubernetes

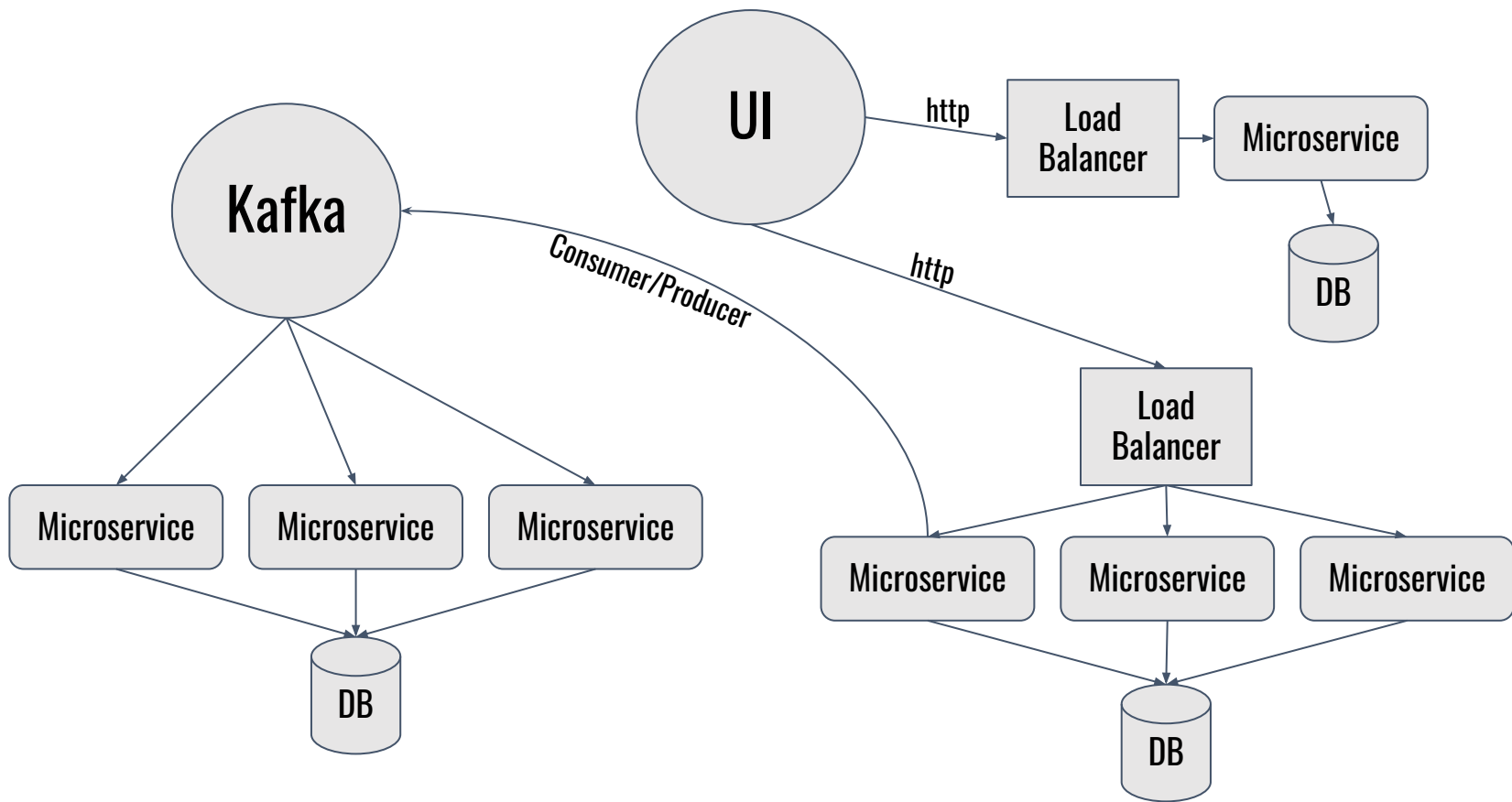


docker

O que fazer quando um dos
MSs está fora do ar? Toda
o ecossistema está
prejudicado?

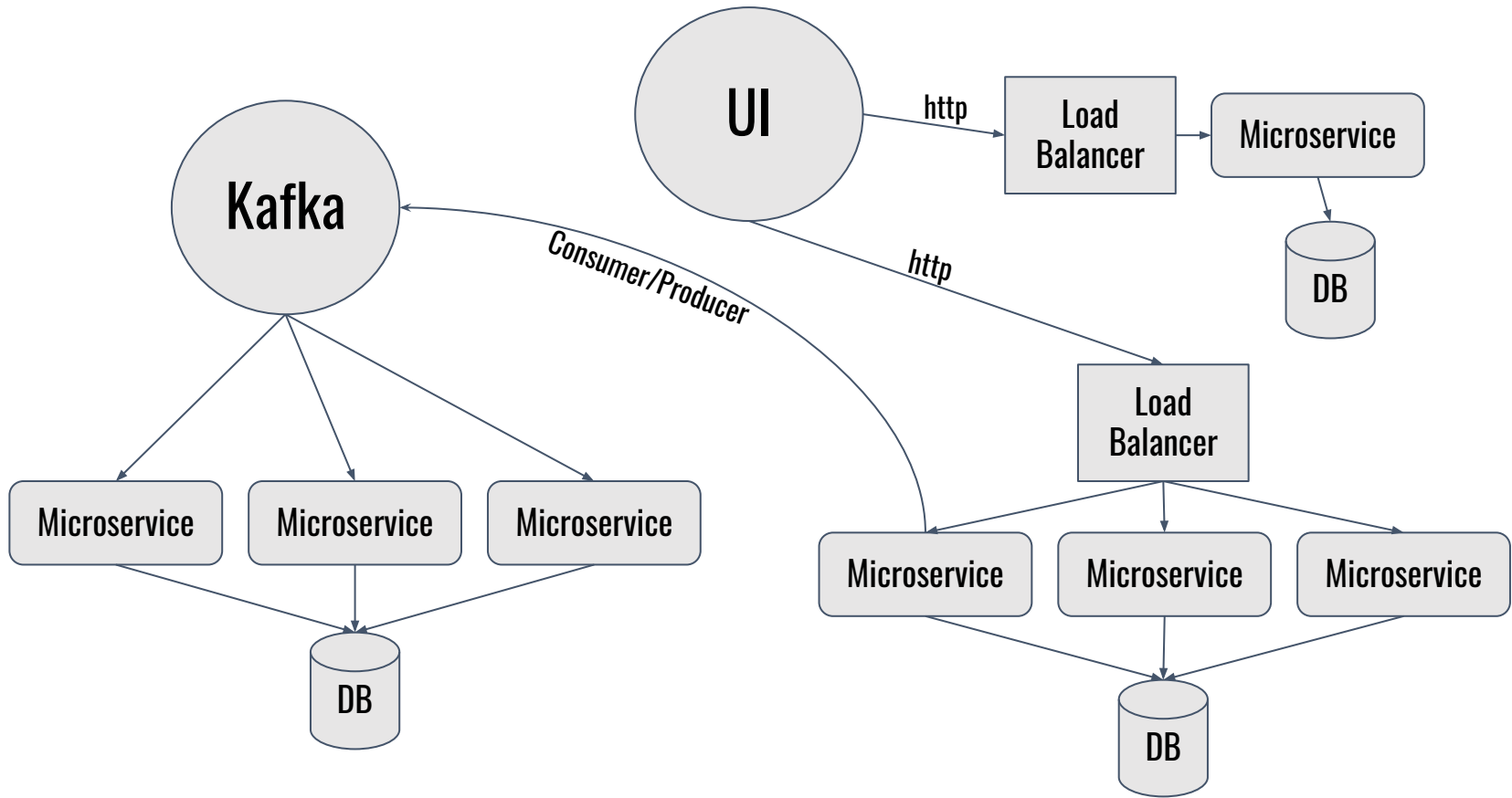


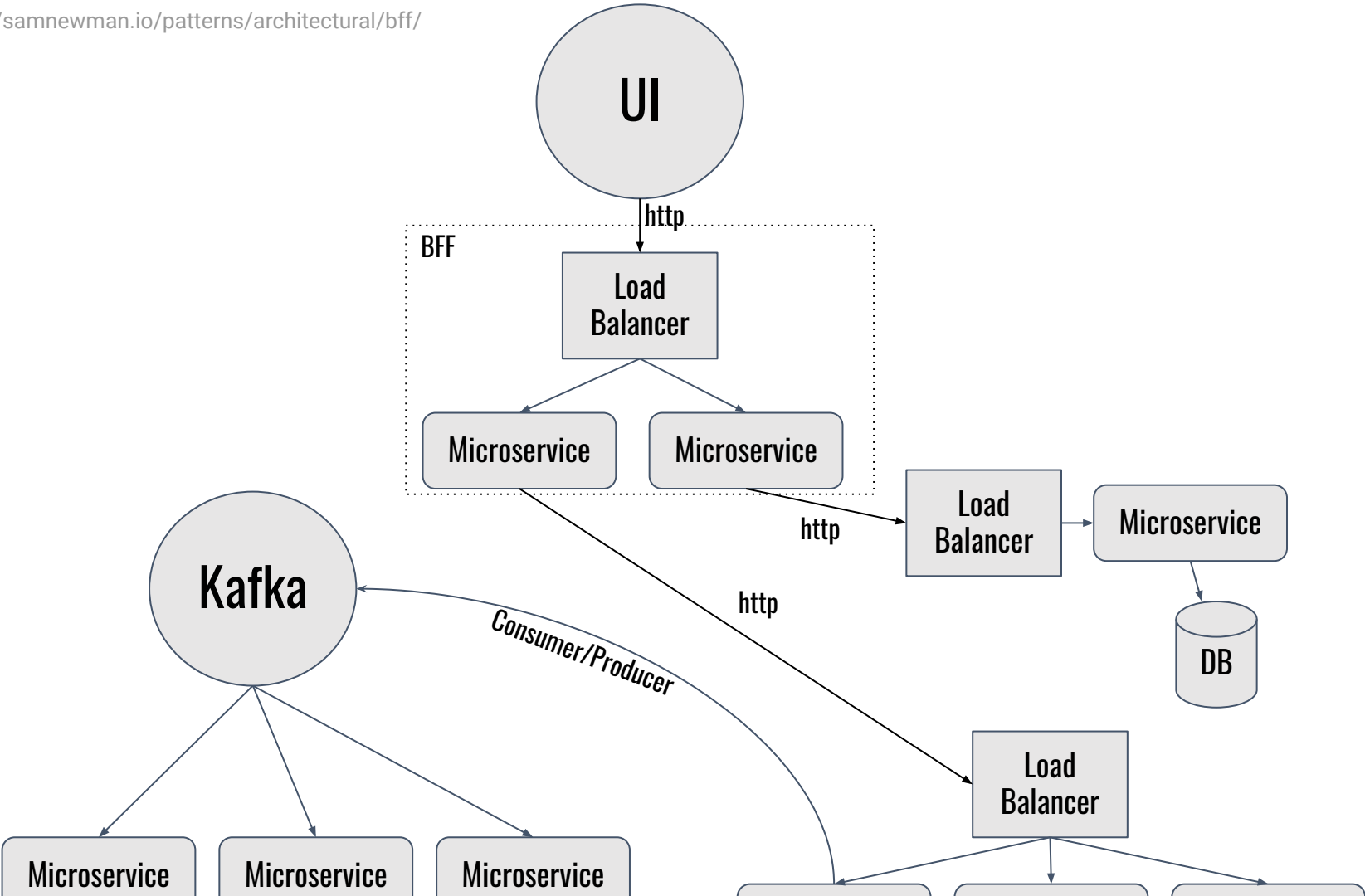


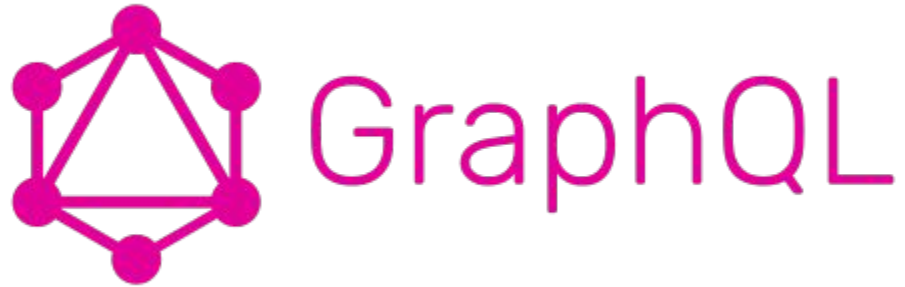




Como simplificar o acesso
do front end e centralizar
o acesso externo?







Describe your data

```
type Project {  
  name: String  
  tagline: String  
  contributors: [User]  
}
```

Ask for what you want

```
{  
  project(name: "GraphQL") {  
    tagline  
  }  
}
```

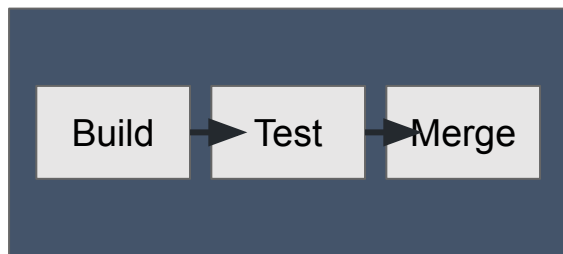
Get predictable results

```
{  
  "project": {  
    "tagline": "A query language for APIs"  
  }  
}
```

Como manter o código
consistente e subir a
aplicação com facilidade?

Deployment – CI/CD

Integração
Contínua



Entrega
Contínua

Automaticamente lança
para o repositório

Implantação
Contínua

Automaticamente lança
para produção



Branch build flow



Pull request build flow



Como subir uma nova
versão sem afetar a
disponibilidade do
sistema?

Rollout Deployment

State 0



State 1



State 2



Final State



Como monitorar os vários sistemas independentes e efêmeros que foram introduzidos nessa arquitetura?



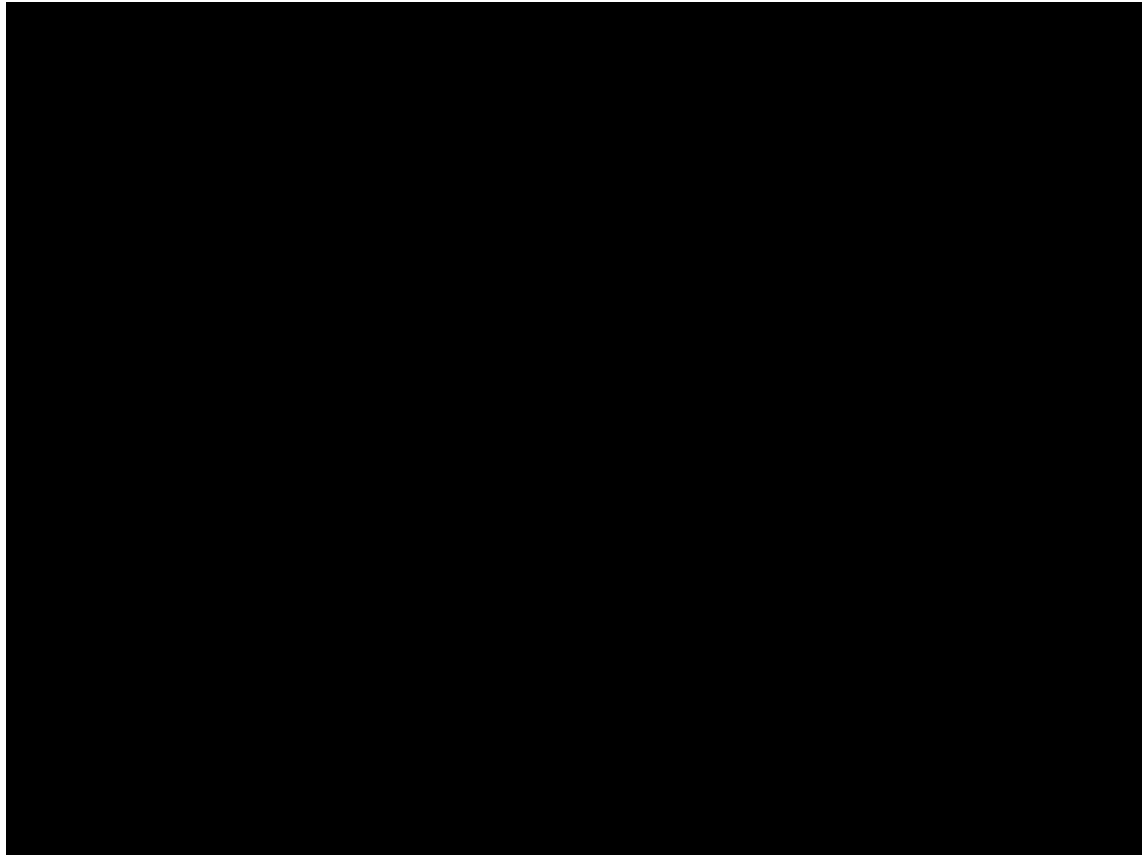
**Alertas
automáticos**

**Armazenamento
eficiente**

Métricas temporais

Query customizada

**Integração com
diversos sistemas**



Maturidade da equipe

Reúso de Código em Microserviços

- Como reutilizar código em uma arquitetura de microserviços?
 - Várias alternativas, cada uma com vantagens e desvantagens
 - Decisão complexa:
 - Diversas equipes, diversos microserviços
 - Como reutilizar mantendo a independência dos MS
 - Não existe alternativa “correta”

Duplicação de código entre MS

- Duplicação de código entre MS
 - Cópia manual
 - Managed copy-paste
 - <https://bit.dev/>
 - Múltiplas instâncias do componente em diferentes MS
 - Sincronização entre repositórios
 - Permite uso de diferentes estratégias de update
 - Uso errado pode quebrar outros MS
 - Independência entre MS
 - Requer mesma linguagem
 - Generalização do código

Bibliotecas compartilhadas

- Criação de bibliotecas compartilhadas
 - Códigos que não são alterados com frequência
 - Pouco código
 - Requer versionamento
 - Enumerations, utils, helper, base
 - Uso errado aumenta acoplamento entre MS
 - Requer mesma linguagem
- Exemplo (Java):

```
public interface CrudService<T extends BaseModel>  
  
public class CrudServiceJpaImpl<L extends JpaRepository, T extends BaseModel> implements CrudService<T>  
  
public abstract class BaseController<K extends CrudService, MODEL, DTO>
```

Reúso de Código como Microserviço

- Principal desvantagem das abordagens vistas até agora:
 - Uso da mesma linguagem para o código compartilhado
- Alternativa?
- Disponibilizar o código como microserviço
 - Permite uso de linguagens e tecnologias diferentes
 - Permite incorporar lógica de negócios, não apenas classes mais genéricas
 - Requer versionamento da API
 - Mantém independência
 - Mais um MS para ser administrado
 - Uso mais fácil em um ambiente com muitas equipes desenvolvendo

Bibliografia

- <https://martinfowler.com/articles/microservices.html>
- <https://medium.com/introducao-a-arquitetura-de-microservicos/introdu%C3%A7%C3%A3o-a-microservi%C3%A7os-25378269e6f>
- <https://www.amazon.com/Production-Ready-Microservices-Susan-Fowler/dp/1491965975>
- <https://www.amazon.com.br/Building-Microservices-Sam-Newman/dp/1491950358>

Obrigado!

Perguntas?

