Qual o tamanho adequado de um micro serviço?

Apresentação

HypeFlame

agibank

Rafael Salerno de Oliveira

- Solution Architect na HypeFlame/Agibank
- Mais de 15 anos em Desenvolvimento de Software
- Últimos 5 anos trabalhando com ferramentas de Devops/Cloud/ Arquitetura de Micro Services
- Entusiasta
  - TDD,DDD, linguagens funcionais
  - XP, métodos ágeis
  - Arquitetura Evolucionária

https://www.linkedin.com/in/rafael-salerno-de-oliveira-a8551b23/

HypeFlame

agibank

Qual o tamanho adequado de um micro serviço?

Aprendizado que tivemos reconstruindo a arquitetura de um banco.

- Core banking
- CRM
- Cadastro central de pessoas
- Infraestrutura de componentes (Kubernetes/AWS)
- Apenas .Net mais linguagens Java/Python/Scala/NodeJs
- Cultura

kubernetes

Cenário Atual 2021

HypeFlame

agibank

- Nós trabalhamos desde 2017 com cerca de 80% da arquitetura baseada em micro serviços.
- *Hoje temos cerca de 800 micro serviços*
- *200 Bancos de Dados (Relacionais e Não relacionais)*
- *Fazemos media de 1200 deploys por mês / média de 40 por dia*
- *Temos cerca de 35 time de desenvolvimento (Squads)*
- *Muito testes automatizado ….*
- *Muitas Soluções e muitos problemas ……*

Microservices

Como começamos

HypeFlame

agibank

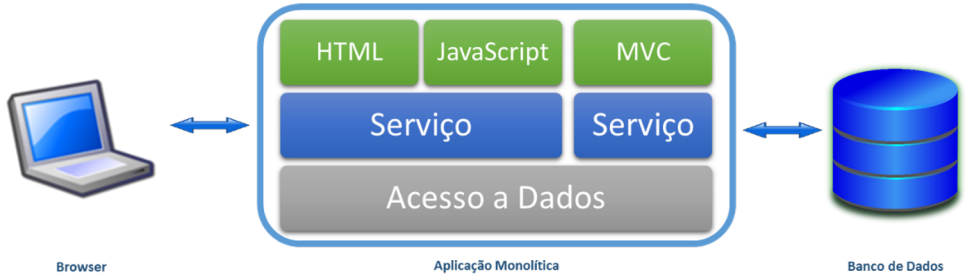
até 2016 ....

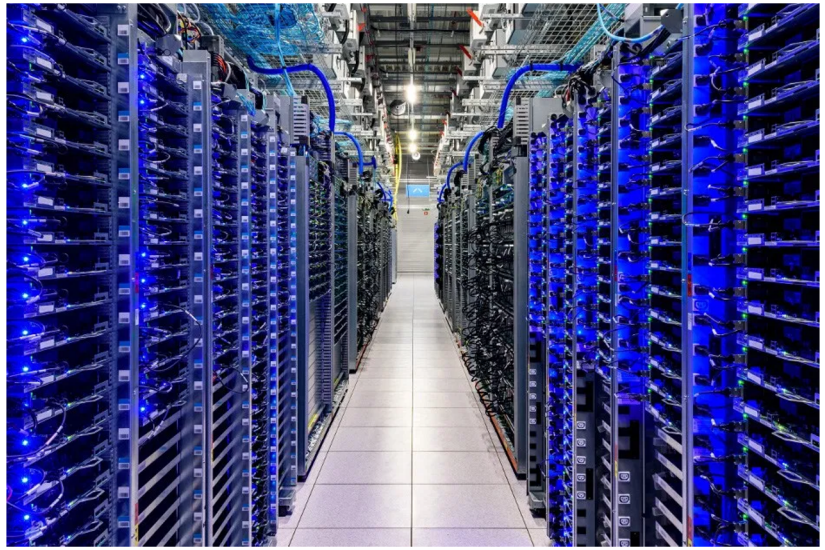
Tínhamos uma arquitetura monolítica

Cada deploy era um evento, com muita sorte 4 por mês

Poucos testes automatizados

Times alocados o final de semana todo testando e realizando ajustes





Quando tudo mudou

HypeFlame

agibank

Em 2017 ....

Com o objetivo de escalar agressivamente o Business.

Criamos um time de Arquitetura e trouxemos /apresentamos um novo padrão e novas tecnologias como:

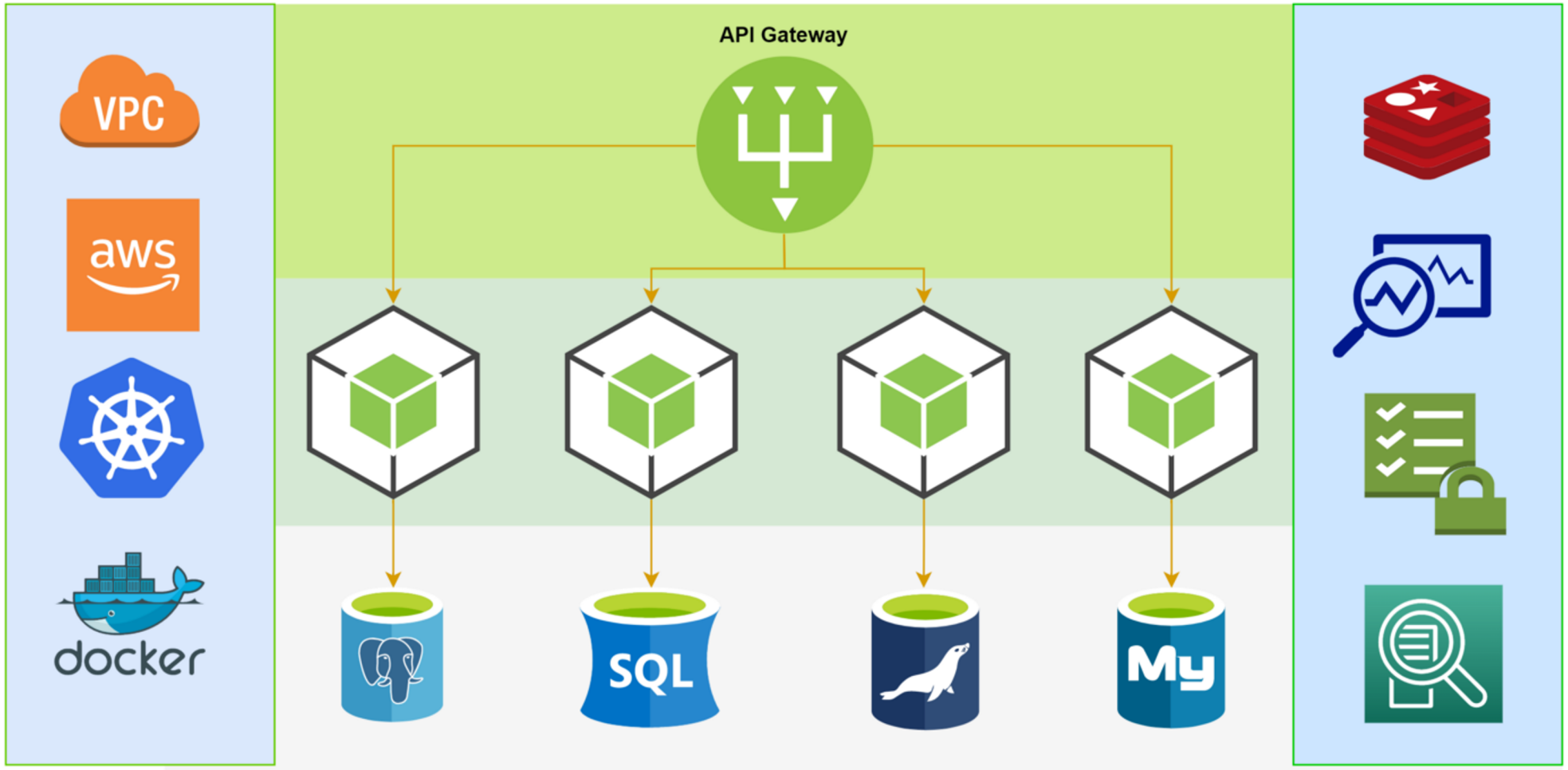
- Micro Serviços
- Kubernetes
- API Gateway
- Cloud
- CI/CD
- Monitoramento
- Treacebility
- Log Centralizado
- Kafka/Rabbitmq
- Diversidade de linguagens
- Testes Unitários / Contrato/ Outros (de forma obrigatória)
- Times organizados por Business

Porem ....

HypeFlame

agibank

Fomos tentados pelo purismo da arquitetura de micro serviços



Logo...

HypeFlame

agibank

500 micro serviços 300 bancos de dados em questão de menos de 1 ano

15 times, 20% da nova arquitetura

O tempo nos ensinou que alguns caminhos escolhidos na arquitetura de micro serviços não são tão bons assim, ao menos para a nossa realidade.

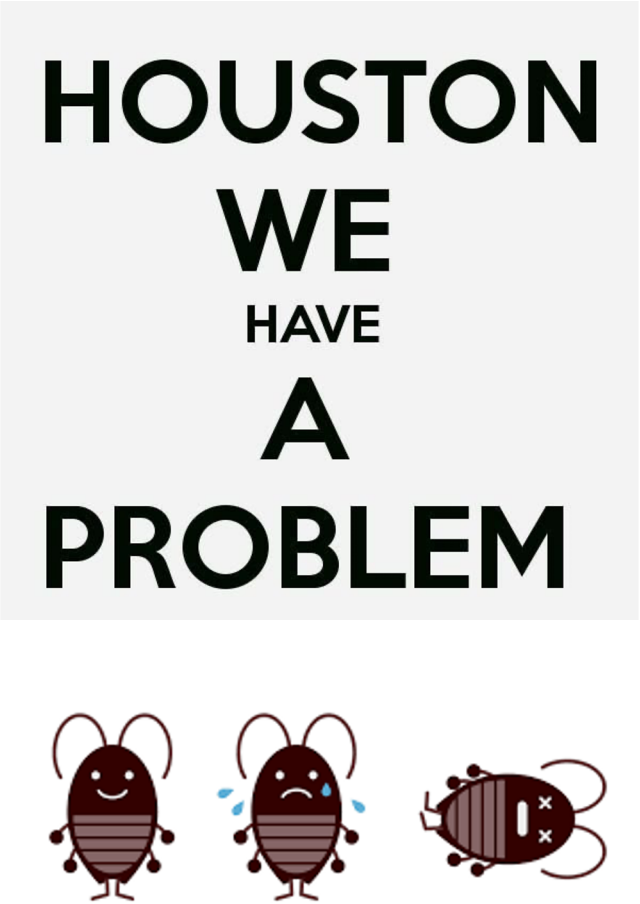
Problemas...

HypeFlame

agibank

Com o passar do tempo vieram alguns **problemas**, como:

- Custo
- Rastreabilidade/ Bugs complexos
- Escalabilidade
- Manutenibilidade
- Qualidade dos Testes



Custo

HypeFlame

agibank

- 500 Micro serviços
  - 1gb, 512mb, 256mb por serviço
  - 1 vCPU
  - Quantas maquinas precisamos?
- 1 Semana 1tb de logs
- alguns monolito (ferramentas de terceiros/outros mal projetados)
- Banco de Dados (muitos)
- Tudo isso na AWS
- Custo por manter isso ligado 24/7 passou a ser fixo
- Número de pessoas para manutenção precisou crescer arquitetos, Infraestrutura, Segurança, Desenvolvedores, etc...

OBS: Impacto de uma decisão de arquitetura

Rastreabilidade

HypeFlame

agibank

500 micro serviços
300 bancos de dados

Exigem maior maturidade para rastreabilidade
- Ferramentas
- Expertise

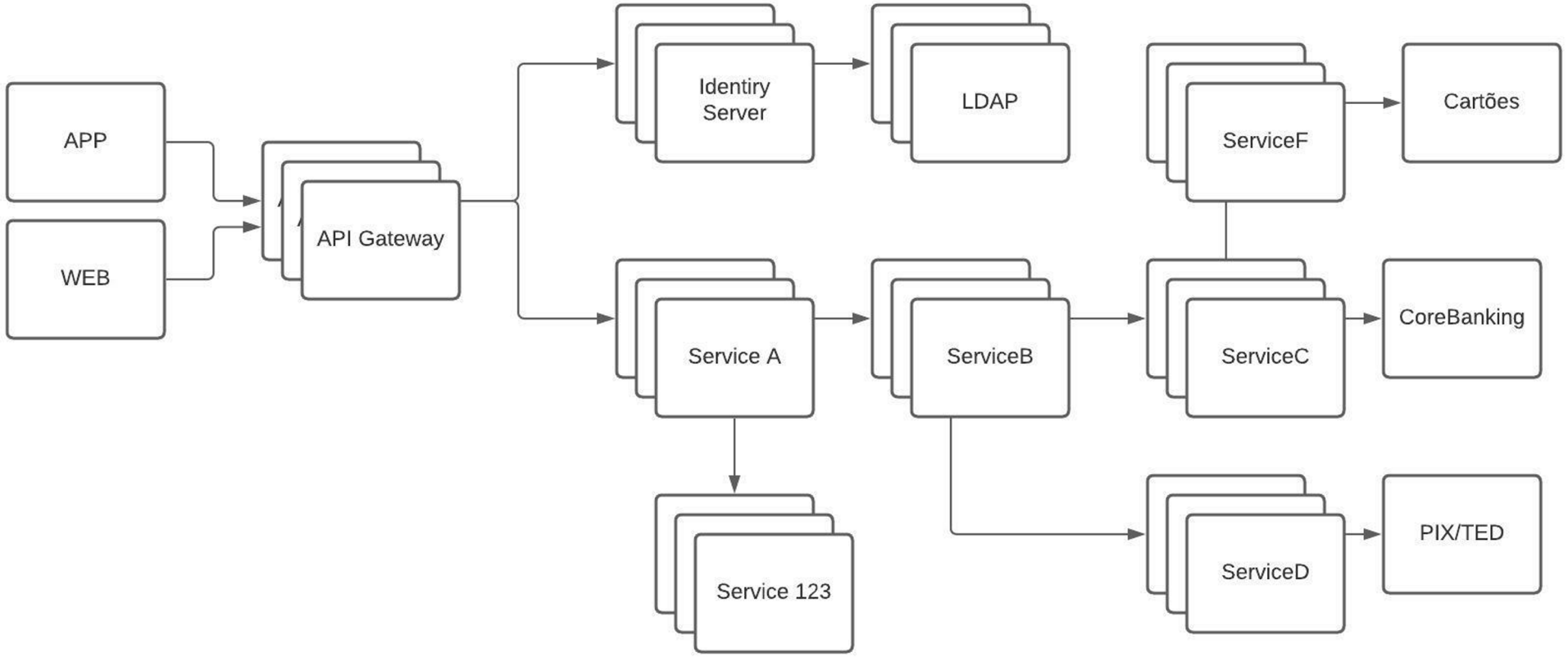
Em pouco tempo tínhamos poucas pessoas que conseguia fazer um throubleshooting completo

Rastreabilidade

HypeFlame

agibank

Imaginamos então um cenário onde o usuário fez login no seu aplicativo e essa única chamada, se desencadeou em 10 ou 20 apenas para uma operação.



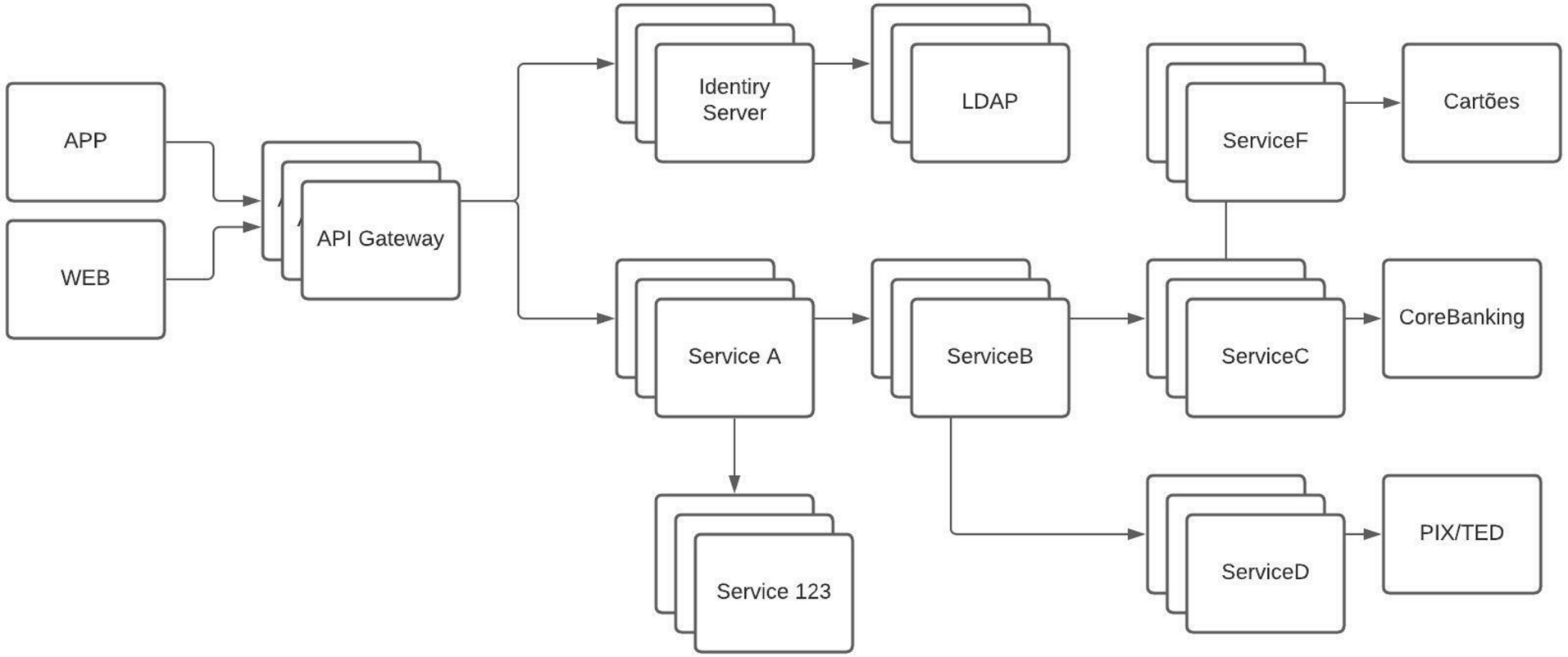
Quando tivermos um problema teremos 10, 20 possíveis lugares de pontos de falha.

Rastreabilidade

HypeFlame

agibank

Imaginamos então um cenário onde o usuário fez login no seu aplicativo e essa única chamada, se desencadeou em 10 ou 20 apenas para uma operação.



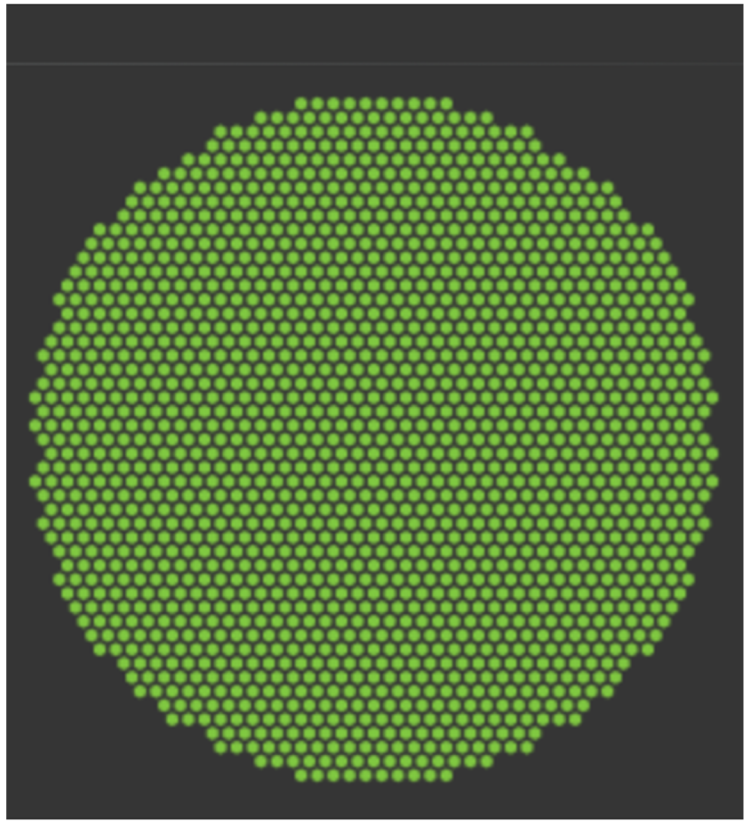
Quando tivermos um problema teremos 10, 20 possíveis lugares de pontos de falha.

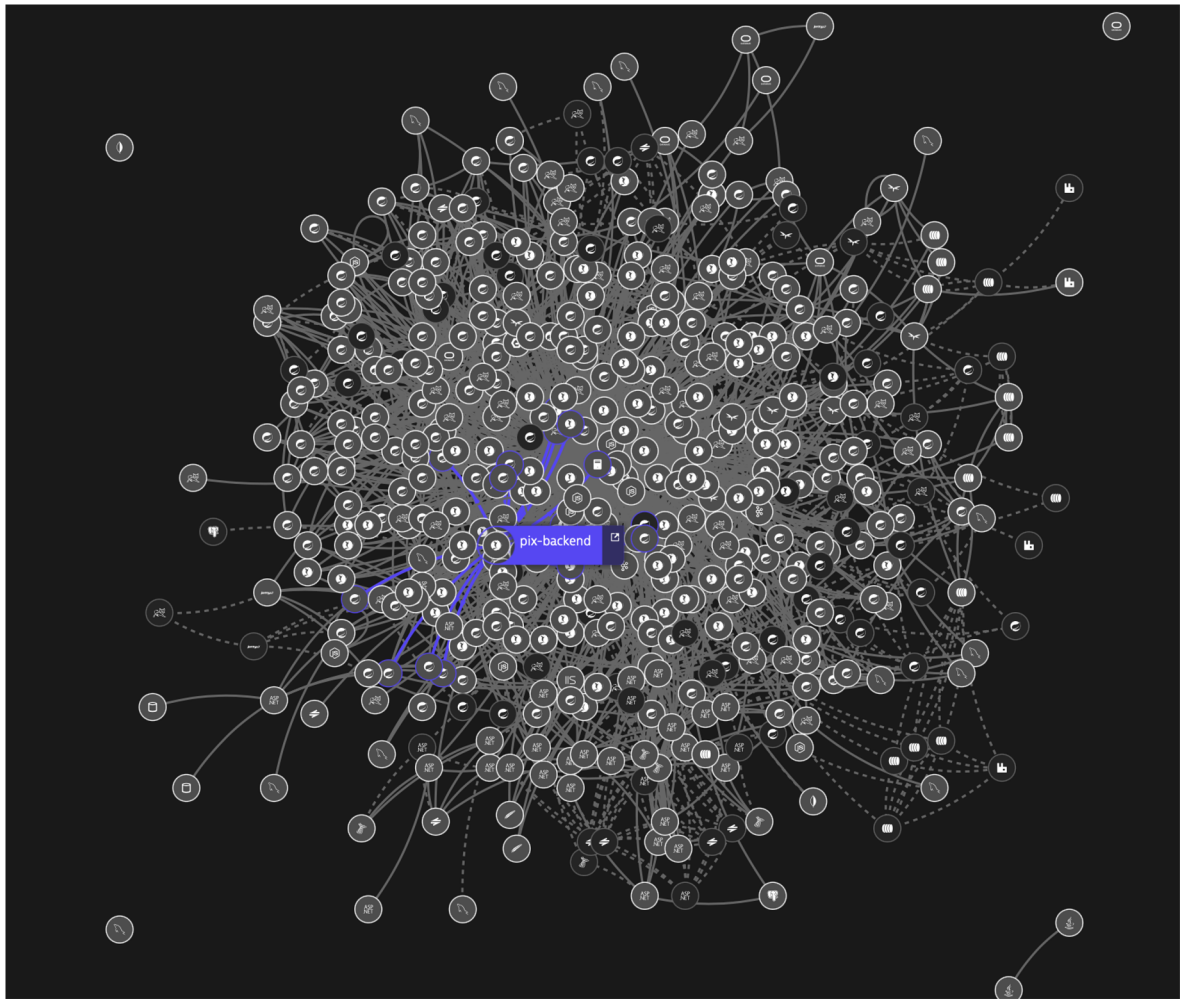
Rastreabilidade

HypeFlame

agibank

Service Health





Escalabilidade

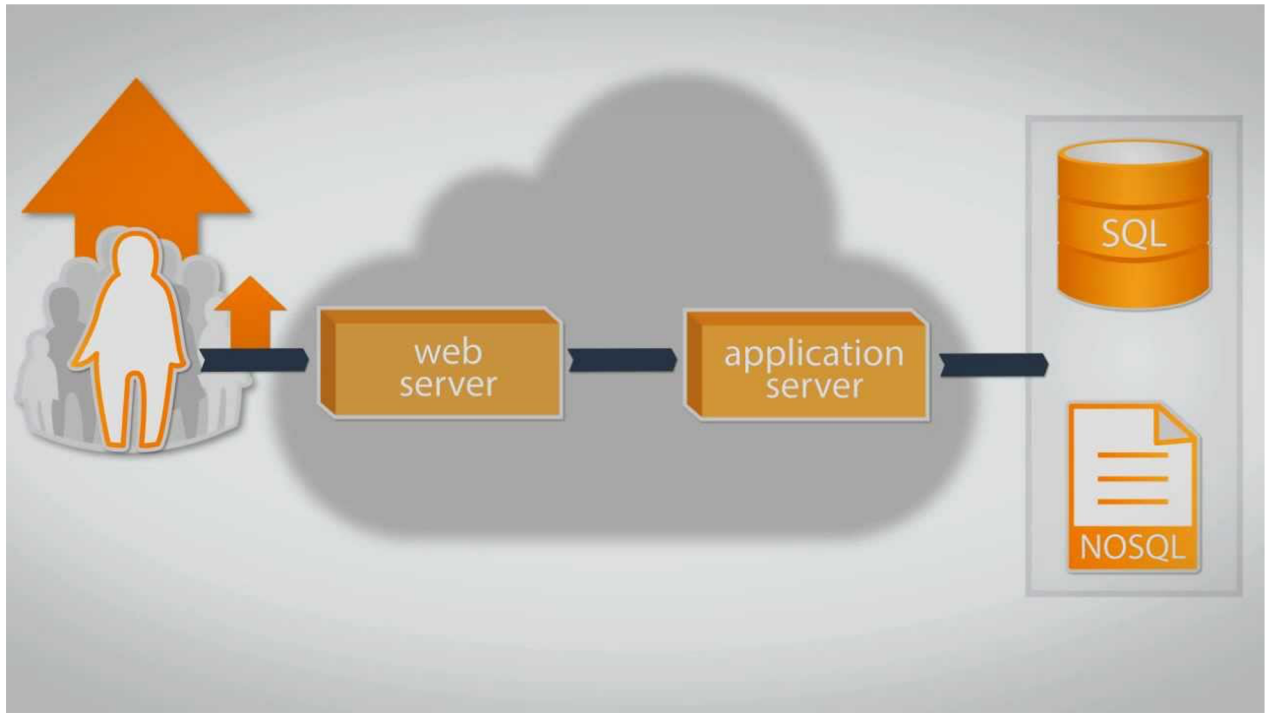
HypeFlame

agibank

Com esses mesmos:
- 500 micro serviços
- 300 bancos de dados

De nada adianta escalar alguns serviços e outros não

Fazer tunning de alguns bancos outros não



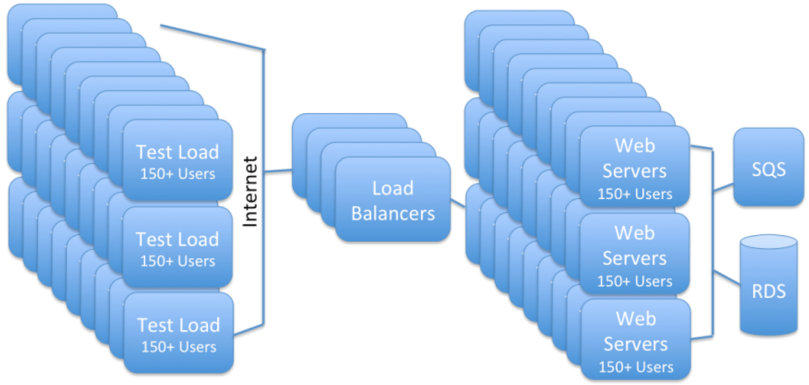
Escalabilidade

HypeFlame

agibank

Uma coisa era escalar Monolito outra é micro serviços

A escalabilidade em um ambiente tão segmentado e granular pode virar um problema.



Mesmo cenário:

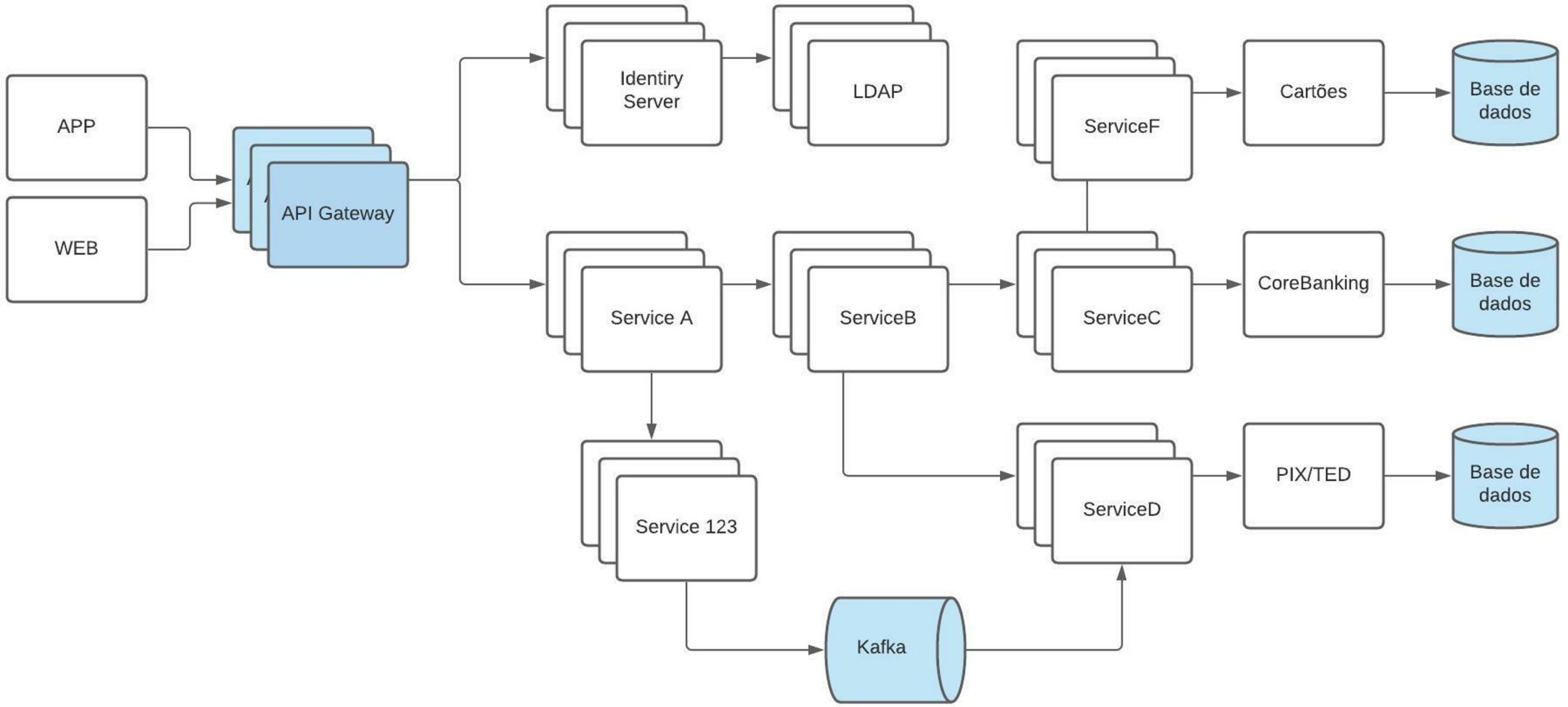
- Um usuário executando uma operação em seu aplicativo,
- Operação que **desencadeia 10 ou 20 chamadas** de micro serviços
  Maior parte deles deve subir **horizontalmente**
- **Bancos de dados** também deve estar preparados para esse **volume**, que pode **ser sazonal** ou **definitivo**.

O **pool de conexões** deve subir também para receber esses novos serviços que estão subindo.
Ou estar preparados para esses volume.

Escalabilidade

HypeFlame

agibank

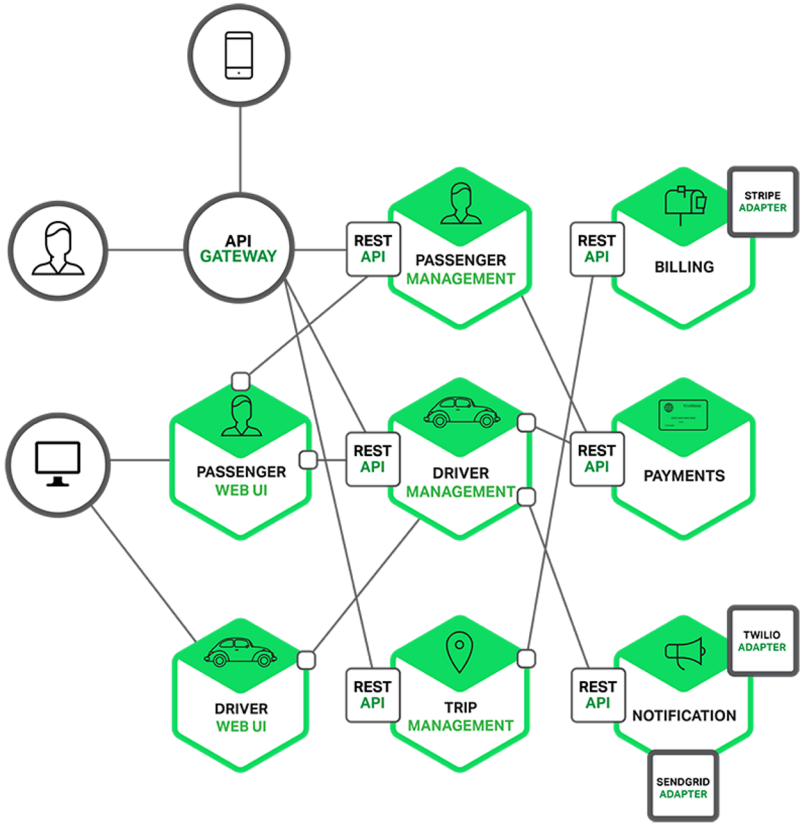


Manutenibilidade

HypeFlame

agibank

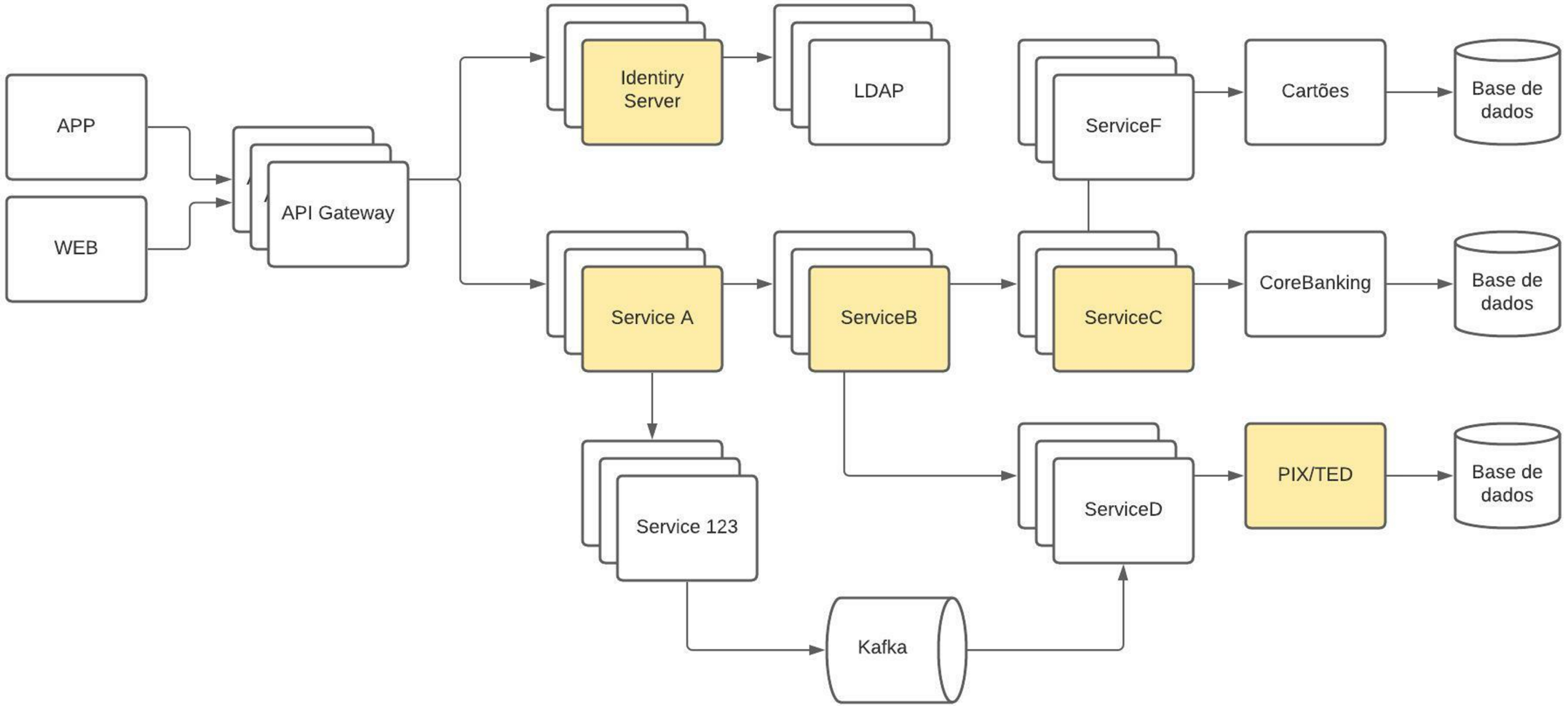
- Um time tinha 10 a 20 serviços, muito pequenos
- 1 Feature precisava commit em quase todos
- Mesmo dev fazia 10 commits em 10 serviços
- Monolito distribuido



Manutenibilidade

HypeFlame

agibank



Qualidade de Testes

HypeFlame

agibank

Apenas testes unitários com 90% de cobertura não é o suficiente

Precisa de um testes mais abrangente

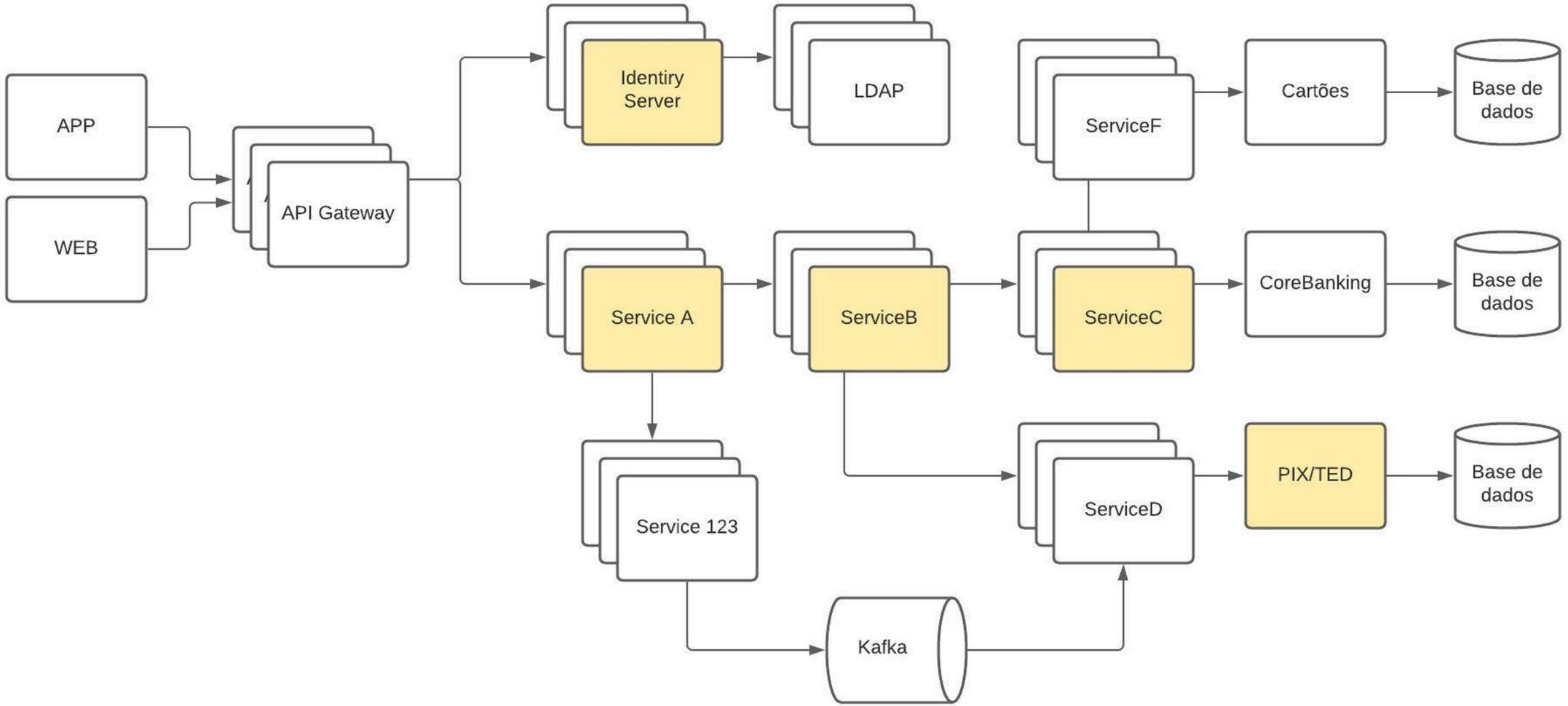
Alguém precisa testar com visão mais ampla

Performance de 1 serviço pode desencadear um problema generalizado, assim como um banco com performance

Qualidade dos Testes

HypeFlame

agibank



Então...

HypeFlame

agibank

Após algum tempo nesse cenário que passou a ser frequente, chegou a hora de parar e analisar o cenário atual.

A empresa crescendo, novos clientes, ações de marketing frequentes...

Seguir assim poderia ser cada vez mais problemático para um futuro sustentável

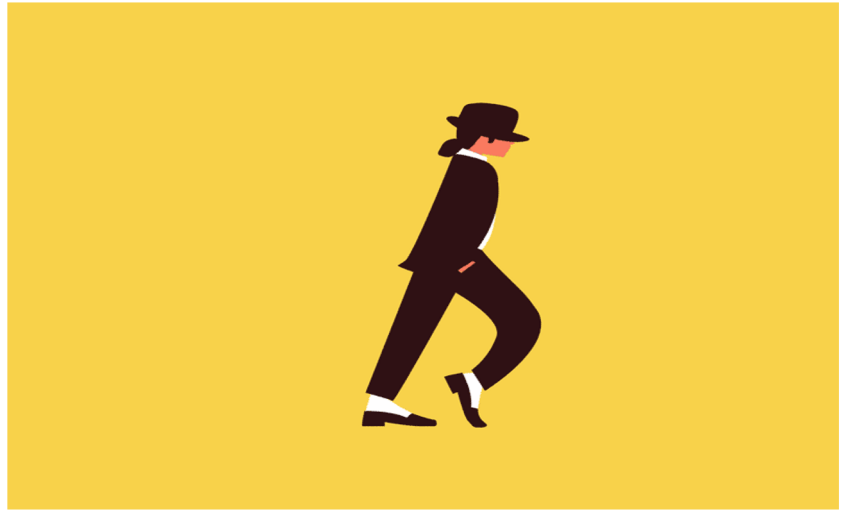
Então...

Esse pontos de problemas que nos fizeram pensar e dar alguns passos atrás

Continuar com arquitetura de micro serviços era uma premissa

Os benefícios dos micros serviços é uma questão indiscutível, já que favoreceu um ***crescimento de negócio***, ***técnico***, ***agilidade***, ***escalabilidade***, ***desacoplamento***, *entre outros.*

As entregas de demandas independência dos times era boa comparada ao passado

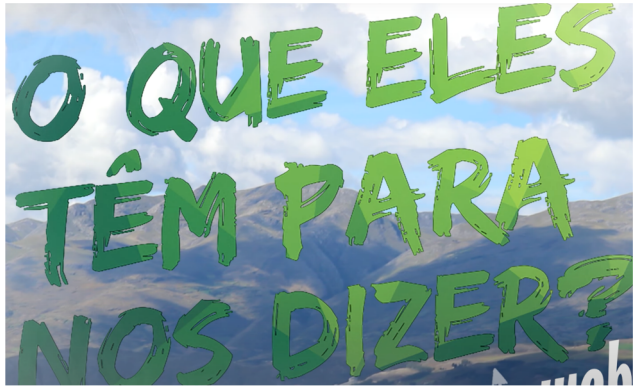


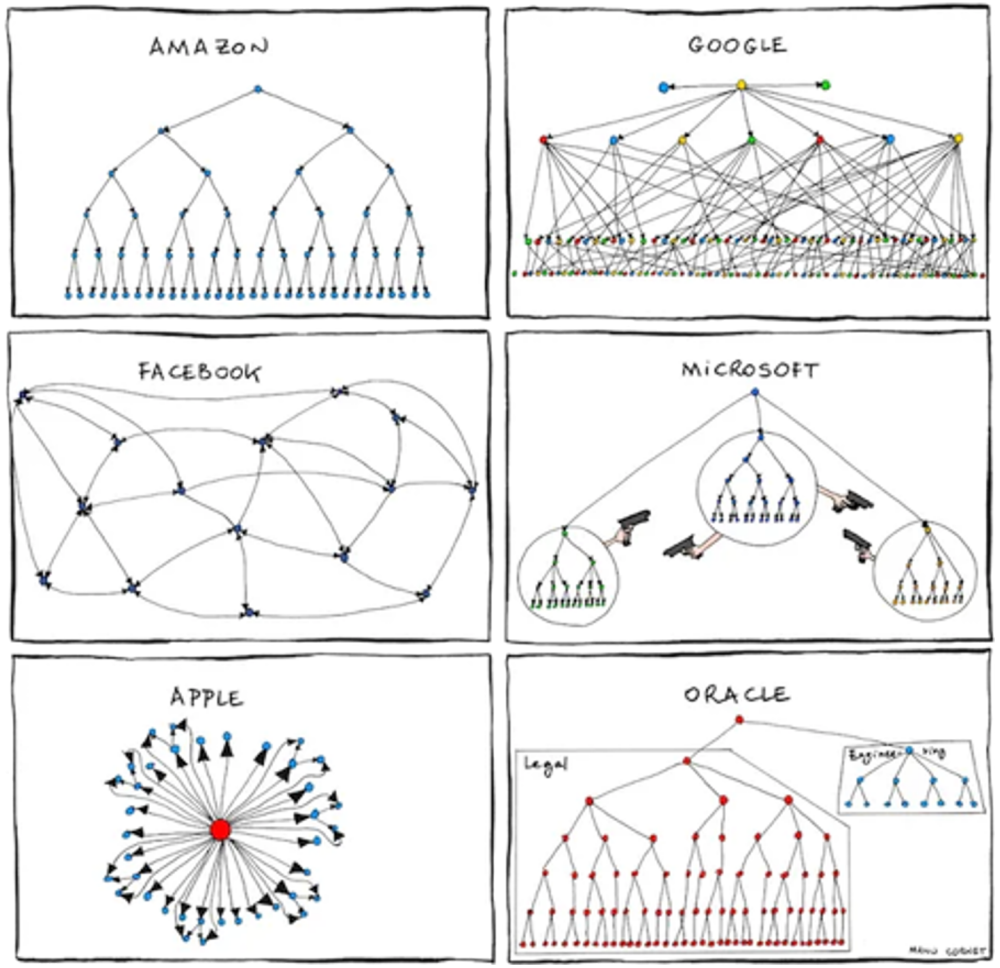
O Que a História da Software conta?

HypeFlame

agibank

Paramos então para avaliar o que algumas empresas estão fazendo nesse sentido, o que a história do mudo de software tem para nos contar.

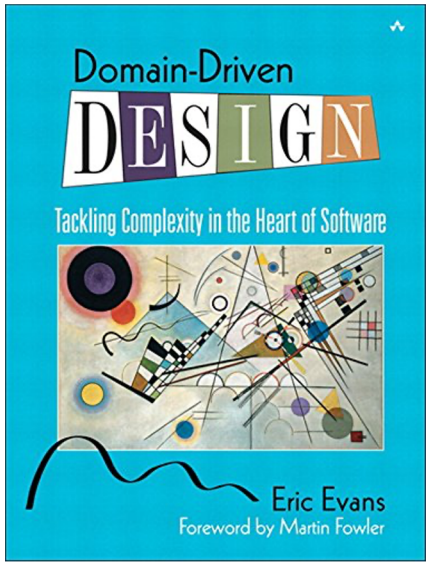


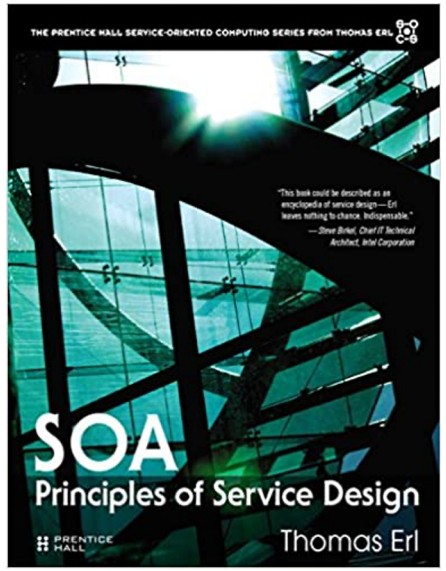


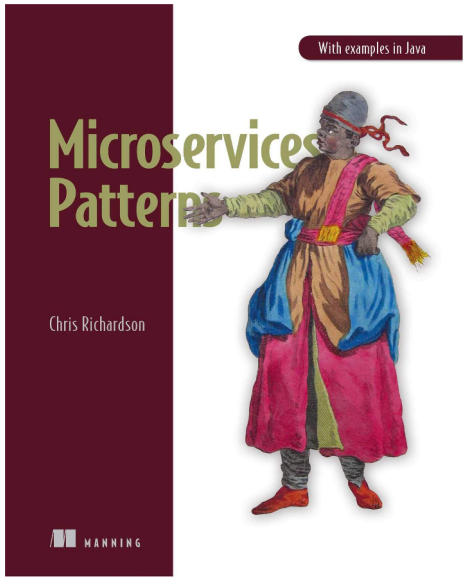
O Que a História da Software conta?

HypeFlame

agibank







MonolithFirst

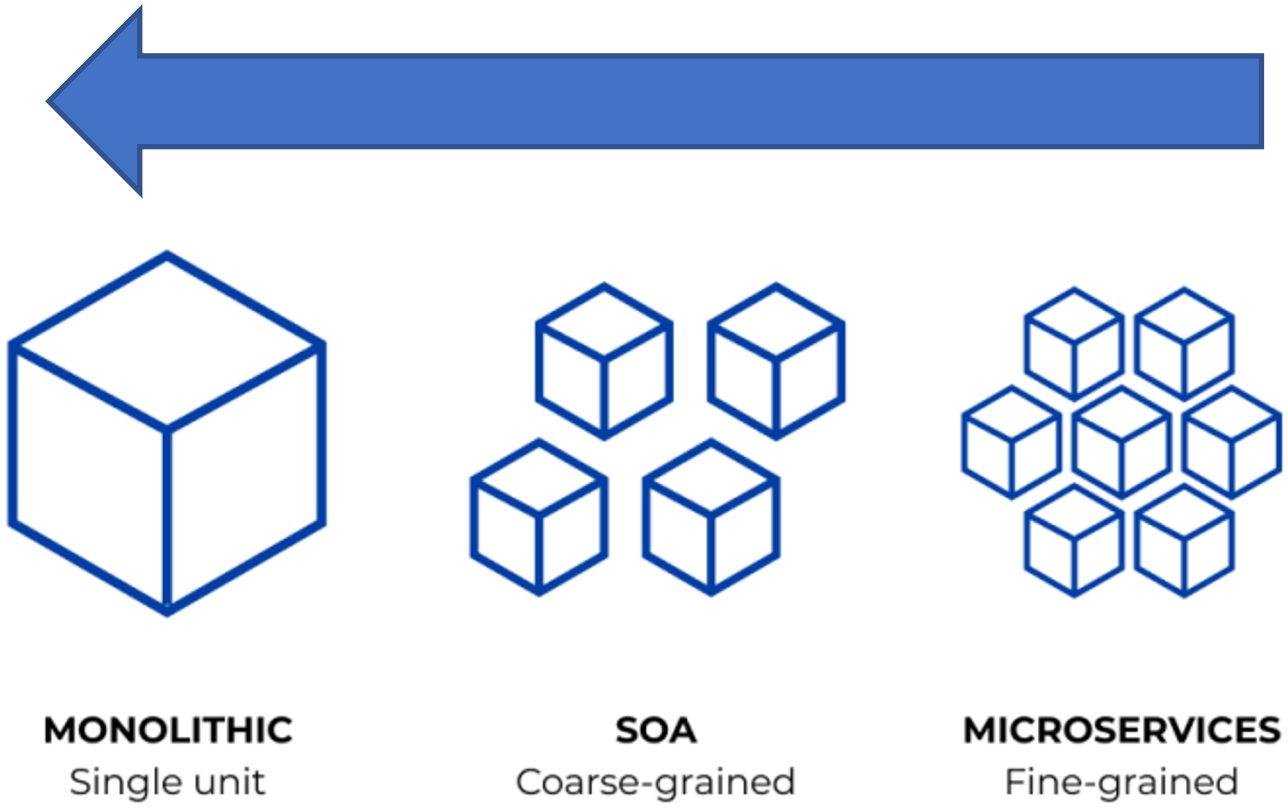
3 June 2015

Martin Fowler

EVOLUTIONARY DESIGN
MICROSERVICES

As I hear stories about teams using a microservices architecture, I've noticed a common pattern.

Passo Atrás



MONOLITHIC
Single unit

SOA
Coarse-grained

MICROSERVICES
Fine-grained

Como passamos a pensar...

HypeFlame

agibank

Daremos então esse “passo atrás” e **pensamos agora em domínio DDD**.

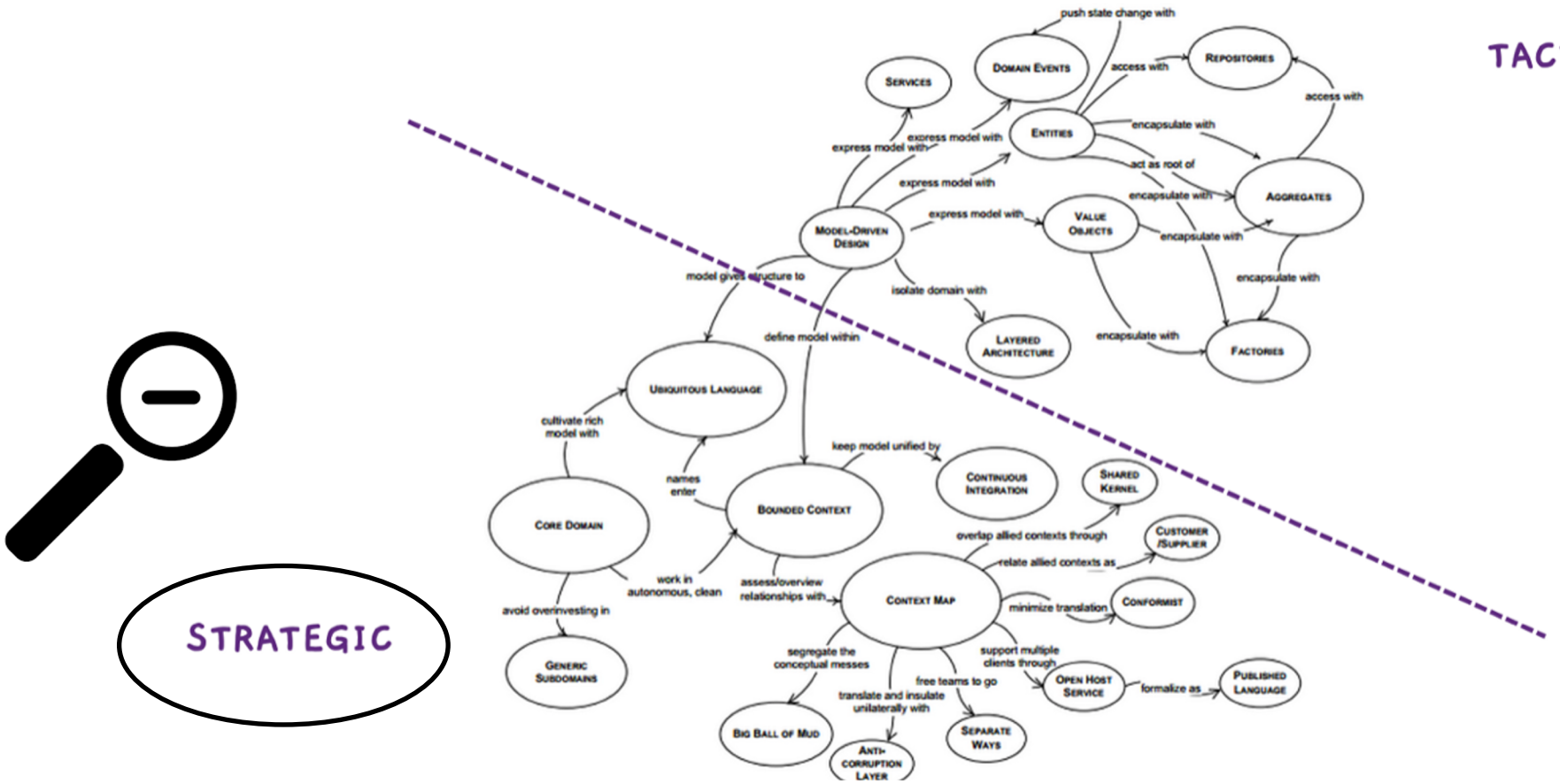
Palavra chave desse passo, pensando em domínio começamos a ver que antes de criar novos serviços, precisamos pensar:

- **A qual domínio ele pertence?**
- **Esse domínio que ele pertence está bem conceituado?**
- **Podemos agrupar domínios pequenos com o Core Domain?**

Como passamos a pensar...

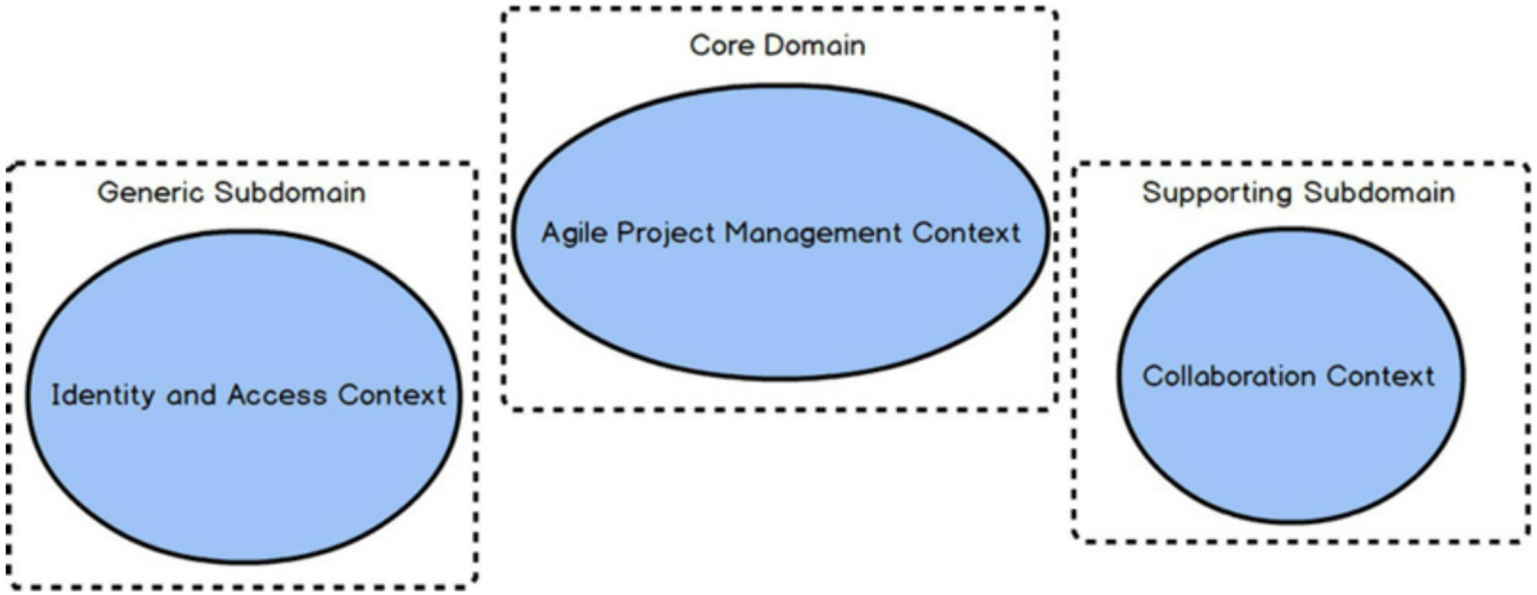
HypeFlame

agibank



Geralmente nos concentramos apenas no Tactical (mais próximas do código)

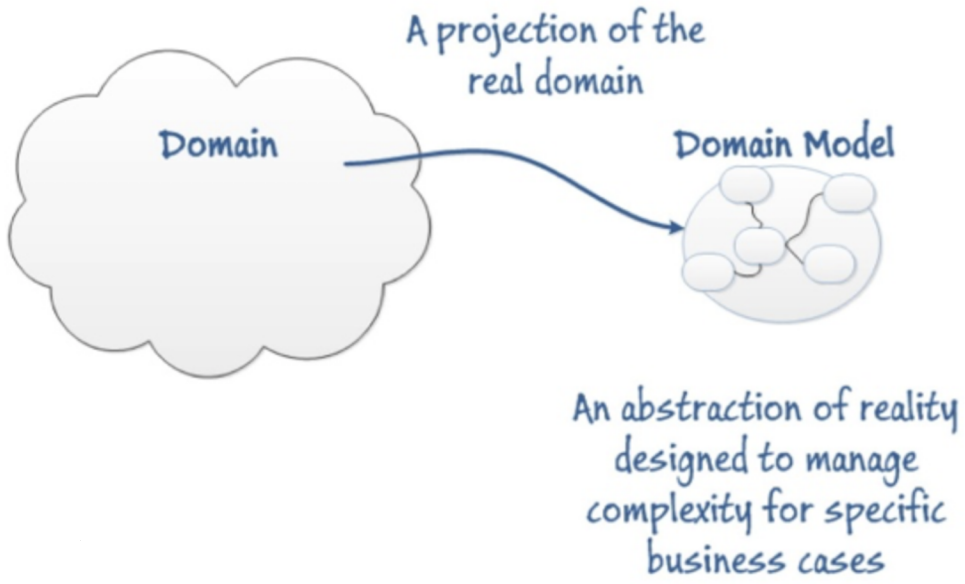
Subdomains : 3 types

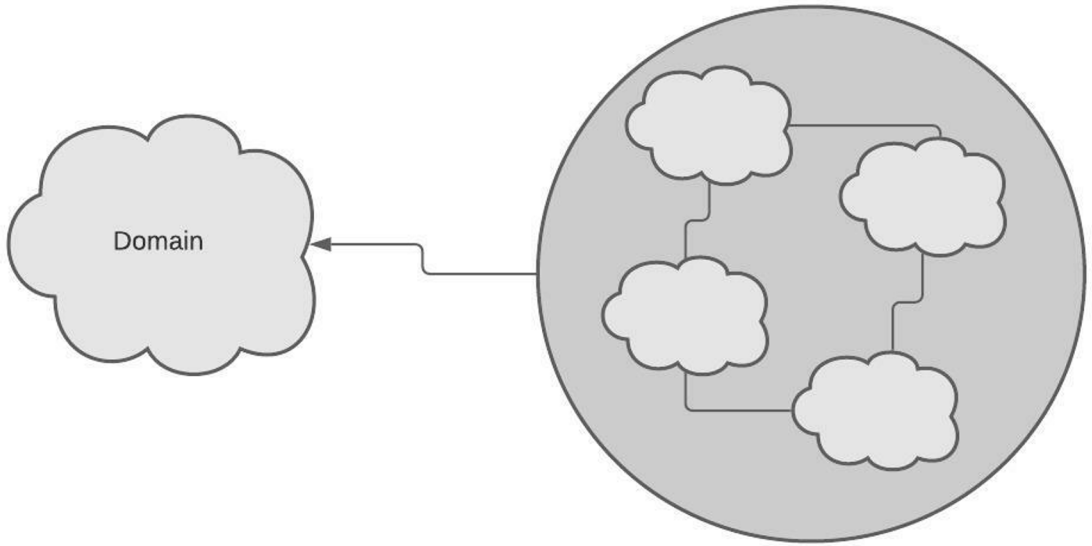


Como passamos a pensar...

HypeFlame

agibank





martinFowler.com

Refactoring Agile Architecture About

MonolithFirst

3 June 2015

Martin Fowler

EVOLUTIONARY DESIGN
MICROSERVICES

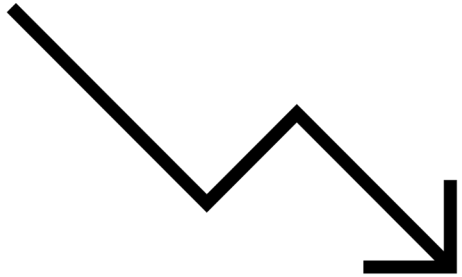
As I hear stories about teams using a microservices architecture, I've noticed a common pattern.

Definindo o domínio alguns problemas passaram a ser resolvidos.

A quantidade de micro serviços passou a **cair**, a manutenção passou a fazer mais **simples**, a escalabilidade passou a ser mais **objetiva**, quando ocorria erro eram menos lugares para se verificar logs.

Testes passaram agora a fazer mais **sentido**, um teste unitário de fato estava testando código de negócio.

Testes de performance e integrações passaram a ser mais **simples** de analisar.

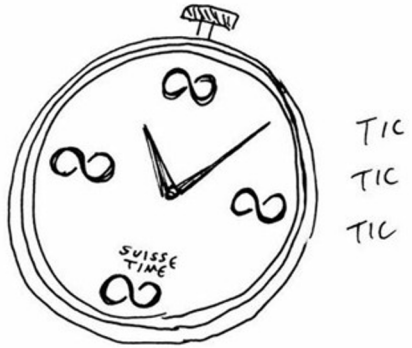


Passo Atrás

Se considerarmos que um serviço deveria iniciar como:

- **um monolito, porém modularizado**, com domínio bem definido
- Levando alguns conceitos de DDD como Bounded Contexts
- Design by Contract

os micro serviços parecem estar mais adequados para uma futura segmentação.



Isso leva tempo....

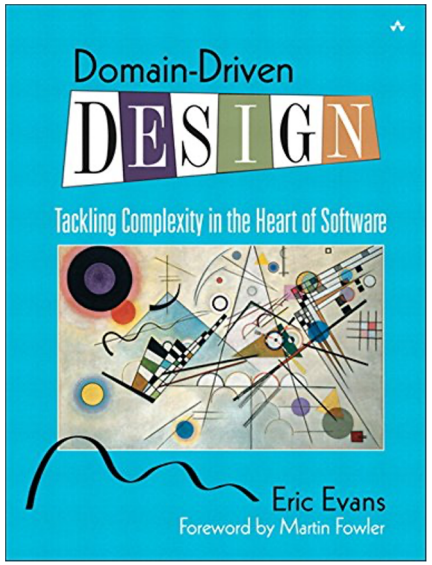
E qual o Tamanho ideal?

HypeFlame

agibank

O Mais certo é depende:
- da empresa
- do business

- Qual a maturidade que tem a empresa ? Ferramentas? Cultura? Profissionais?
- Qual o Business?



Mas o que podemos considerar e que nos traz perto da idealidade é o DDD.

Pensar no seu domínio, pois é ele que faz com que as coisas façam sentido e tragam o que realmente importa.

Motivations

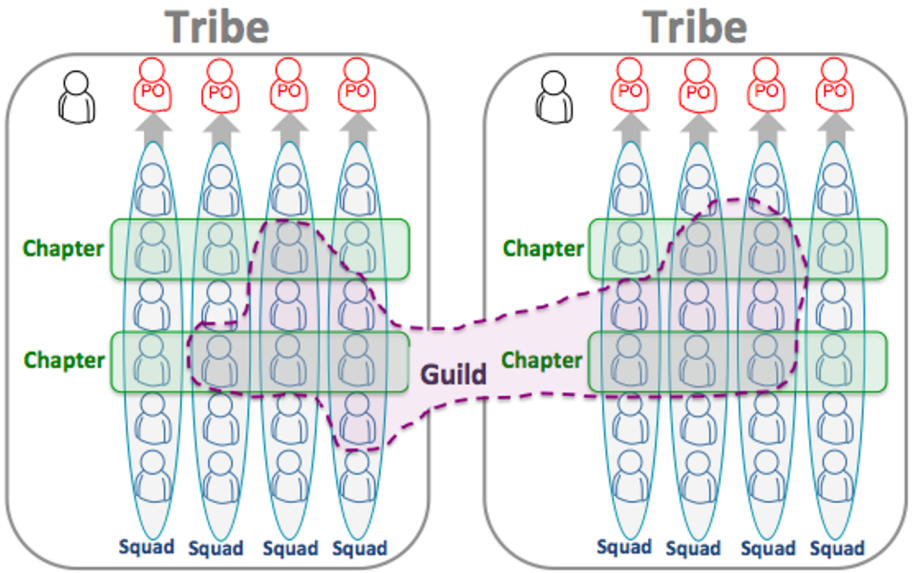
- +2000 Services
- Availability Risks
- Risky, expensive deployments
- Poor separation of concerns
- Inefficient execution

Uber Engineering

07/2020

https://eng.uber.com/microservice-architecture/

Cultura/Ferramentas



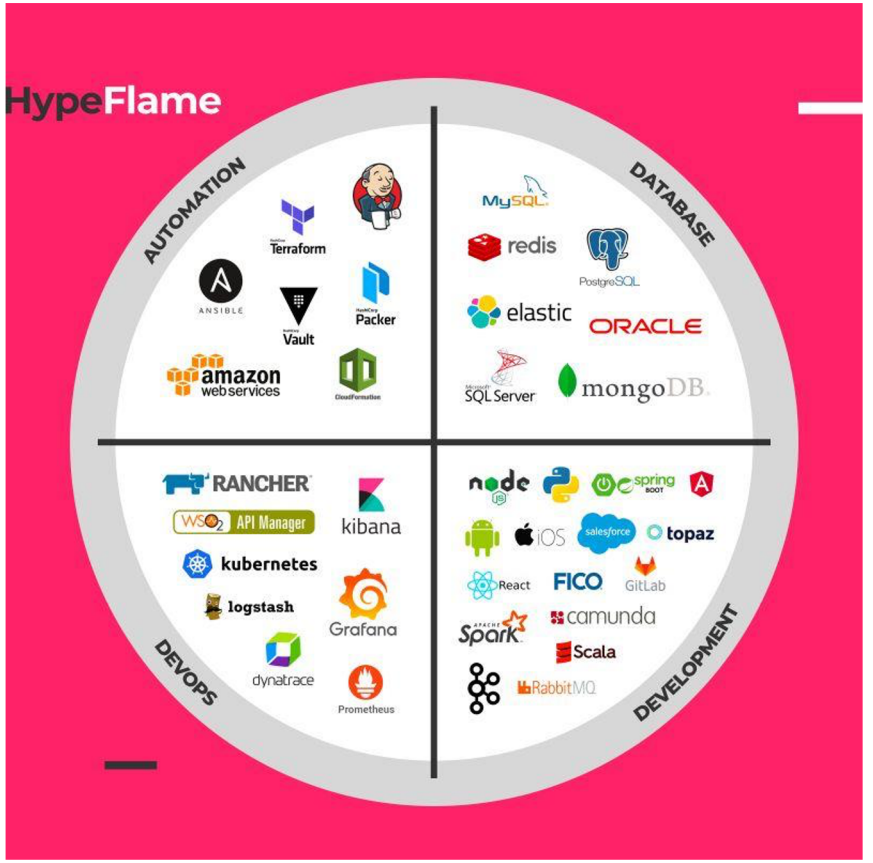
Istio

Prometheus

Grafana

dynatrace

Jaeger



elasticsearch

logstash

kibana

Kiali

Aprendizados

HypeFlame

agibank

Comesse pensando no Domínio/Core Domain
Micro serviço não é bala de prata/ Ele pode trazer mais problemas
Lembrar-se dos das ferramentas/disciplina que acompanham o micro serviço

- Log centralizado
- Monitoramento
- CI/CD
- Escalabilidade/Disponibilidade/Resiliência
- Rastreabilidade
- Manutenibilidade
- Observability
- Devops/Automação
- Testes muitos testes

Refletir frequentemente sobre onde estamos querendo chegar
Maturidade em todos os times

agibank

HypeFlame

Qual o tamanho adequado de um micro serviço?