Microservices

Decomposição de aplicações para Implantação e Escalabilidade

Como se originou o estilo.

- No início dos anos 80 com a introdução da primeira principal tecnologia de distribuição de sistemas: Chamadas de Procedimento Remoto (RPC), a ideia básica era fazer chamadas remotas transparentes para os desenvolvedores.
- Seria possível desenvolver sistemas maiores entre máquinas que poderiam evitar os problemas de escalabilidade e processamento e de memória que afetavam os sistemas da época.

Contexto

Após o avanço da capacidade de armazenamento e processamento, as primeiras tecnologias de distribuição de sistemas deixaram importantes lições:

- Simplesmente porque algo pode ser distribuído, isso não significa que deve ser distribuído.
- Tentar fazer uma chamada distribuída agir como uma chamada local sempre termina mal.
- Sempre que possível, seus programas e seus ambientes de tempo de execução devem ser totalmente autocontidos.

Na definição de Martin Fowler essas três observações são a base para o que ele chama de microservices.

Definição e filosofia

- Em 2011 durante um evento para arquitetos de software, o termo Microservices foi então usado para descrever um estilo de arquitetura que muitos já estavam experimentando na época.
- O estilo arquitetural de microservices desenvolve aplicações complexas a partir de pequenas e individuais aplicações que se comunicam através de APIs.
- Apesar de não existir uma definição precisa desse estilo arquitetural, certamente à características comuns em torno das capacidades de negócios, controle descentralizado de linguagens e dados, etc.

Problemas Resolvidos

Dependência de tecnologia

Aplicações Grandes

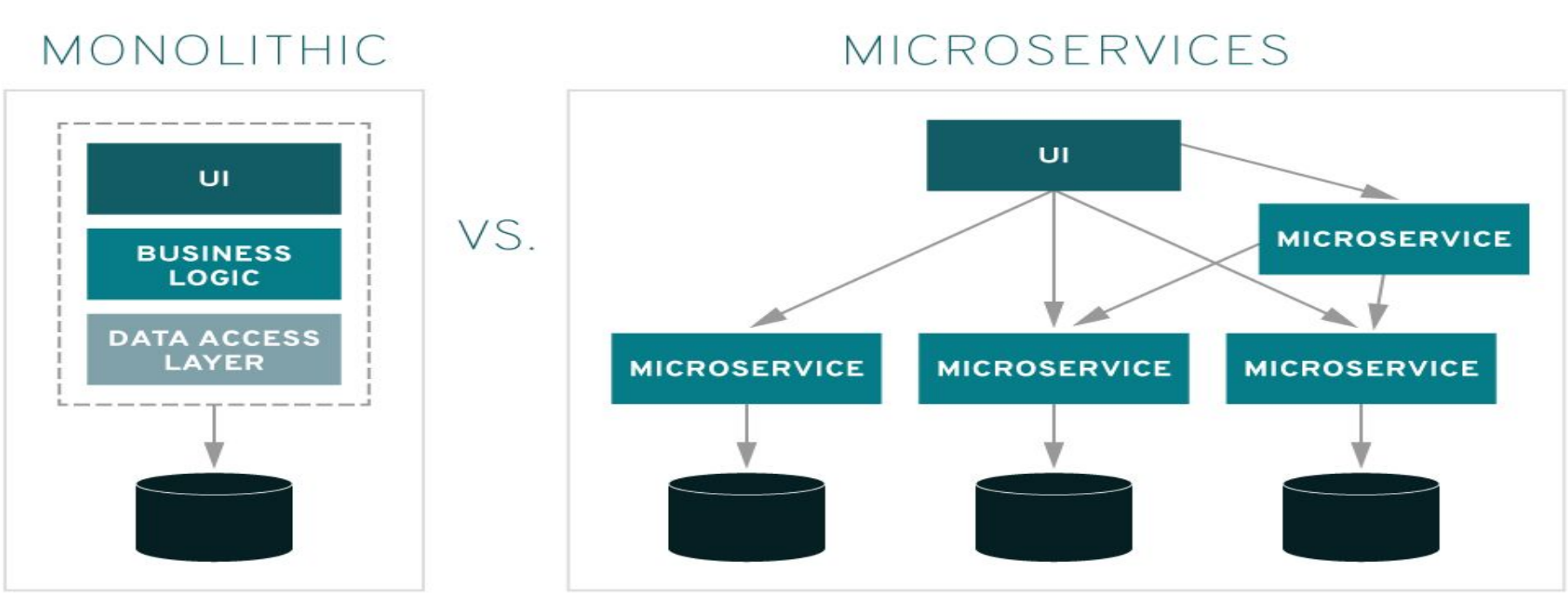
- Que precisam de alta taxa de velocidade e de liberação.

Aplicações Complexas

- Que precisam ser altamente escaláveis e dimensionáveis.

Organizações com Equipes pequenas de Desenvolvimento

A solução apresentada pelo estilo



Suas Vantagens

- Escalabilidade Seletiva
- Resiliência
- Divisão dos times
- Capacidade de composição
- Manutenção mais eficiente
- A diversificação de tecnologia

Suas Desvantagens

Complexidade Adicional (SD).

- Primeiramente, os desenvolvedores devem lidar com a complexidade adicional de desenvolvimento de sistemas distribuídos.

Latência

- Serviços que chamam outros serviços.

Testes

- Escrever testes automatizados para vários serviços é um desafio.

Dependências entre serviços

- A implantação de funcionalidades que abrangem vários serviços requer uma coordenação cuidadosa entre as várias equipes de desenvolvimento.
- É preciso criar um plano ordenado de implantações de serviços com base nas dependências entre serviços.

3 Exemplos de Utilização/Aplicação do Estilo

Referências

Fontes:

- https://www.ibm.com/developerworks/br/cloud/library/cl-evolution-microservices-patterns/index.html
- http://www.pedromendes.com.br/2016/01/02/microservicos/
- https://www.thoughtworks.com/pt/insights/blog/microservices-nutshell
- https://docs.microsoft.com/pt-br/dotnet/architecture/microservices/architect-microservice-container-applications/data-sovereignty-per-microservice

Obrigado!