# Arquitetura de Micro serviços

* Muitos problemas com monolitos e SOA
* Precisava de uma nova arquitetura
* Precisava ser modular e simples
* Termo ‘micro serviços’ apareceu primeiro em 2011 e em 2014 Martin Fowler e James Lewis publicaram um artigo
* Existem 9 características principais nesse artigo:
  + Componentização via Serviços
  + Organizado em torno das Capacidades do Negócio
  + Produtos, não Projetos
  + Endpoints Inteligentes e Dumb Pipes
  + Governança Descentralizada
  + Gerenciamento de Dados Descentralizado
  + Automação da Infraestrutura
  + Desenhado para Falhas
  + Design Evolucionário

## Componentização via Serviços

* Design modular é SEMPRE boa ideia
* Podemos alcançar modularidade utilizando:
  + Bibliotecas – chamadas diretamente dentro do mesmo processo
  + Serviços – chamadas feitas por uma Web API, RPC
* Em Micro Serviços implementamos modularidade utilizando Serviços, e não Bibliotecas
* Bibliotecas podem e devem ser utilizadas, mas neste caso não representarão Componentização
* Motivação de usar Serviços para Componentização:
  + Componentes possuem deploys independentes
  + Utilizam recursos próprios
  + Interface de comunicação bem definida e de fácil entendimento

## Organizado em torno das Capacidades do Negócio

* Projetos tradicionais possuem times especializados por cada componente do monolito, conforme imagem abaixo. A comunicação entre diferentes times, com diferentes gestões e diferentes culturas é sempre lenta e burocrática.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

* Em Micro Serviços cada time é responsável por todos os aspectos daquele Serviço, dessa forma o time tem uma visão completa do sistema e consegue tomar melhores decisões para realizar alterações em cada uma dessas camadas

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

* Como saber os limites do Serviço e o que ele deve fazer? Onde termina um Serviço e começa outro?
  + Mapear as capacidades do negócio. Por exemplo um e-commerce teria um serviço de Gerenciamento, um de Pagamentos, um de Inventário. E cada Serviço seria mantido por um time diferente.
* Motivação de se Organizar ao redor das Capacidades do Negócio
  + Rápido desenvolvimento
  + Limites bem definidos

## Produtos, não Projetos

* Em projetos tradicionais
  + o objetivo é entregar código funcionando
  + Sem relacionamento ou proximidade com o cliente
  + Geralmente, após terminar, o time se muda para um novo projeto
  + Por esses motivos o time não pensa como o cliente, e acaba não entregando o melhor produto
* Com Micro Serviços
  + O objetivo é entregar um produto funcionando
  + Suporte constante e proximidade com o cliente
  + O time que desenvolveu é responsável pelo Micro Serviço após sua entrega
  + Pode ser passado para outro time, mas envolve passagem de conhecimento de forma organizada e responsável
* Motivação para Produtos, Não Projetos
  + Aumentar satisfação do cliente
  + Mudar a mentalidade dos desenvolvedores, pensando como o cliente e a longo prazo

## Endpoints Inteligentes e Dumb Pipes

* Para os projetos SOA, o ESB e o WS-\* Protocol eram dois mecanismos muito complicados, e difíceis de manter
* Nos micro serviços a ideia é utilizar o mecanismo de comunicação mais simples possível (dumb pipes) com protocolos simples, ou seja, são as REST API com os protocolos HTTP
* Observações importantes
  + Conexões diretas entre serviços não é uma boa ideia
  + A solução foi utilizar Service Discovery e Gateway
  + Além de HTTP, mais protocolos surgiram como o graphQL e gRPC, porém não são tão simples e não seguem tanto a recomendação ‘dumb pipes’
* Motivação para Endpoints Inteligentes e Dumb Pipes :
  + Acelerar desenvolvimento
  + Fácil de manter

## Governança Descentralizada

* Em projetos tradicionais, há padrões para tudo. Padrões para plataforma, banco de dados, loggging e formatos.... Não tem espaço para os times tomarem decisões
* Em micro serviços, o time decide quais padrões para plataforma, banco de dados e logging eles querem seguir, sendo cada time totalmente responsável por seus serviços
* Isto é possibilitado pelo baixo acoplamento em micro serviços
* Multi plataforma significa que a arquitetura é Poliglota
* Motivação
  + Possibilita escolher a tecnologia mas adequada para resolver determinado problema

## Gerenciamento de Dados Descentralizado

* Sistemas tradicionais possuem um banco de dados centralizado

Uma imagem contendo Forma

Descrição gerada automaticamente

* Em micro serviços, cada serviço possui sua própria base de dados

Diagrama

Descrição gerada automaticamente

* Este item é bem controverso, e envolve muita discussão. E algumas vezes compartilhar a base de dados é o único jeito, ou seja, em alguns casos vai ser impossível seguir com bases de dados independentes por micro serviço e que podem gerar problemas:
  + Duplicação de dados
  + Transações distribuídas
* Motivação
  + Possibilita utilizar bancos de dados específicos para cada solução
  + Encoraja o isolamento do serviço

## Automação de Infraestrutura

* O paradigma SOA, sofreu de falta de deste tipo de ferramenta de automatização
* Ajuda no aceleramento do deploy
  + Automatização de testes
  + Automatização de Deploy
* No caso de Micro Seviços, a automatização de testes e deploy é crucial, já que testar e deployar diversos sistemas toma muito tempo. Sem a automatização, fica inviável a utilização desta arquitetura
* Curtos ciclos de entrega demandam muito tempo com testes e deploys
* Ferramentas utilizadas
  + Azure DevOps
  + GitLab
  + Jenkins
* Motivação
  + Curtos ciclos de entrega

## Desenhado para Falhas

* Em micro serviços há muito trafego de rede e muitos processos, portanto, muita coisa pode dar errado, e os serviços precisam estar preparados para falhas
* O código deve considerar falhas e tratá-las
* Necessário logar e monitorar extensivamente a aplicação
* Logging
  + Capturar exceções
  + Fazer Retentativas
  + Logar as exceções, no caso das exceções não tratadas
* Monitoração
  + Recursos Memoria e CPU
  + Verificação da saúde da aplicação
  + Ferramentas: New Relic, Kubernetes, Azure Monitor
* Motivação
  + Aumenta a confiança do sistema

## Desenho Evolucionário

* Transação para microserviços deve ser gradual
* Começar com pequenos sistemas, evoluindo cada um separadamente

## Resumo

* Não levar todas os atributos acima à risca
* Adotar o que funciona para meu problema
* As tecnologias mudam rapidamente, assim como as abordagens acima podem ter mais ou menos relevância conforme o tempo passa
* Alguns atributos importantes
  + Componentização
  + Organizar ao redor da capacidade de negócio
  + Governança descentralizada
  + Gerenciamento de dados descentralizado (quando possível)
  + Automação de Infraestrutura