# Problemas e Soluções

* Vimos que os microserviços vieram para resolver os problemas e limitações dos monolitos e das aplicações SOA, mas como elas resolvem exatamente?

## Single Technology Platform

* Nos monolitos
  + acabamos por usar apenas uma tecnologia para todos os problemas, e ficamos limitados
  + utilizamos tecnologias nem sempre úteis para resolver determinado problema
  + Upgrade é um problema, pois há necessidade de atualizar a aplicação por completo, e muitas vezes se tornando inviável
* O atributo Governança Distribuída resolve esse problema
  + Cada serviço pode utilizar sua própria plataforma de desenvolvimento

## Delpoys Inflexíveis

* Em monolitos
  + Fazer deploys do app inteiro pois não é possível entregar apenas uma parte dele
  + Testes rigorosos, já que o deploy envolve a entrega de toda a base de código
* O atributo Componentização via Serviços resolve este problema
  + Baixo acoplamento entre serviços
  + Deploys independentes
* O atributo Gerenciamento de Dados Descentralizados também contribui, já que não há compartilhamento de bases de dados entre serviços
  + Diminuiu ainda mais o acoplamento entre serviços

## Recursos Computacionais Ineficientes

* Em monolitos
  + Recursos compartilhados entre todos os componentes
  + Não é possível aumentar os recursos apenas de um componente
* O atributo Componentização via Serviços resolve este problema
  + É possível alocar recursos independentes para cada componente

## Grande e Complexo

* Monolito
  + Codebase enorme e complexa
  + Cada peq mudança pode afetar muiltiplos componentes, necessário testes de larga escala
  + Difícil de manter, e pode tornar o software obsoleto
* O atributo Componentização via Serviços resolve este problema
  + Cada serviço possui seu próprio código
  + Fácil de manter
* O atributo Gerenciamento de Dados Descentralizados resolve esse problema
  + Isolamento de dados de diferentes contextos
  + Diminui a complexidade e facilita a manutenção
* O atributo Organizado ao redor das Capacidades de Negócio
  + Pois temos apenas um time responsável por todos os aspectos desse microserviços: API, Database, Lógic, UI

## ESB Complicado e Caro

* ESB é um dos principais componentes do SOA
* Caro e difícil de manter
* O atributo Smart Endpoint and Dumb Pipes resolve este problema
  + Cada serviço possui seu próprio meio de comunicação, e geralmente é usado o padrão REST
  + Os microserviços não precisam de nenhum intermediário para se comunicarem, eles se comunicam diretamente entre eles
  + Conforme surgiram outros padrões como GraphQL e gRPC, o ideal é utilizar Gateways ou Service Discovery como entradas

## Falta de ferramentas

* SOA (Service Oriented Architecture)
  + Para o SOA ser eficiente, ciclos de desenvolvimento curtos seriam necessários
  + Permitir testes e desenvolvimento rápidos
  + Não existia ferramentas para dar suporte a essas necessidades
  + Portanto, não foi possível economizar tempo utilizando o SOA
* O atributo Automação de Infraestrutura resolve esse problema
  + AAutomatização de testes e de deployments
  + Permite ciclos curtos de desenvolvimento
  + Tornam a arquitetura eficiente e efetiva.