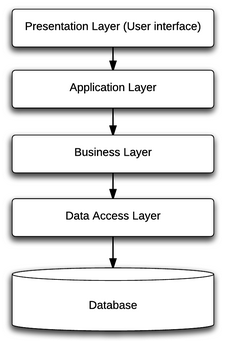
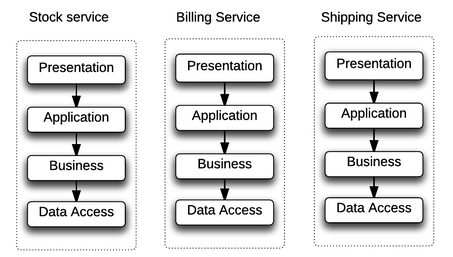
Os Microsserviços prometem uma maneira melhor de gerar um impacto nos negócios de forma sustentável. Em vez de uma **única unidade monolítica,** as aplicações construídas usando microsserviços são compostas de serviços fracamente acoplados e autônomos.

A figura abaixo mostra uma típica arquitetura em camadas de uma aplicação monolítica:



Ao construir serviços que fazem bem uma coisa, você pode evitar a inércia e a entropia de grandes aplicações. Mesmo em aplicativos existentes, você pode progressivamente extrair funcionalidades em serviços independentes para tornar todo o seu sistema mais sustentável.  
  
Quando começamos a trabalhar com microsserviços, percebemos rapidamente que construir serviços pequenos e auto-suficientes é apenas uma parte da execução de uma aplicação de negócio critica.

Abaixo temos uma figura que mostra como podemos dividir uma aplicação em microserviços:



Mas o que é afinal uma aplicação com Microserviço ?

Um aplicativo de microsserviço é uma coleção de serviços autônomos, cada um dos quais  
realizando bem uma única tarefa, que trabalham juntos para realizar operações mais complexas.   
  
Em vez de um único sistema complexo, você constrói e gerencia um conjunto de serviços relativamente simples que podem interagir de formas complexas. Esses serviços colaboram entre si por meio de protocolos de mensagens agnósticos em tecnologia, ponto a ponto ou de forma assíncrona.

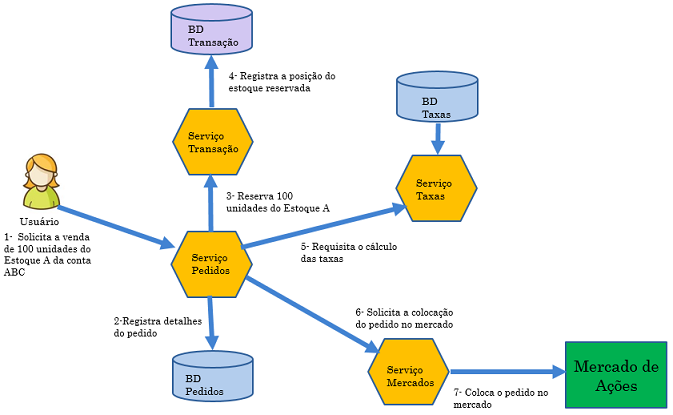
Isto pode parecer uma ideia simples, mas tem implicações notáveis para reduzir o atrito no  
desenvolvimento de sistemas complexos. A prática clássica de engenharia de software advoga  
coesão e acoplamento fraco como propriedades desejáveis de um sistema bem projetado. Sistemas que  
possuem essas propriedades serão mais fáceis de manter e mais maleáveis diante da mudança.

A coesão é o grau em que os elementos de um determinado módulo permanecem juntos, enquanto  
acoplamento é o grau em que um elemento sabe sobre o funcionamento interno de outro.  
  
O Princípio de Responsabilidade Única - SRP de *Robert C. Martin* é uma maneira útil de considerar a coesão:*Agrupe as coisas que mudam pelas mesmas razões. Separe as coisas que mudam por razoes diferentes.*  
  
Em um aplicativo monolítico, você tenta projetar para essas propriedades em uma classe, módulo,  
ou nível de biblioteca.  Em um aplicativo de microsserviço, você deseja, em vez disso, obter essas propriedades no nível de unidades de funcionalidade independentemente implantáveis.

Um único microsserviço deve ser altamente coeso: deve ser responsável por uma única responsabilidade dentro de uma aplicação. Da mesma forma, quanto menos cada serviço souber sobre o funcionamento interno de outros serviços, mais fácil será fazer alterações em um serviço sem forçar alterações nos demais.

Para ter uma ideia melhor de como um aplicativo de microsserviço funciona, vamos consider alguns dos recursos de um sistema hipotético de investimento on-line com as seguintes funcionalidades:  
  
- Abrir uma conta;  
- Depositar e retirar dinheiro;  
- Fazer pedidos para comprar ou vender posições em produtos fnanceiros *(por exemplo, ações);*  
- Modelar o risco e fazer previsões fnanceiras;  
  
Vamos explorar o processo de venda de ações:  
  
1 - Um cliente cria uma ordem para vender algumas ações de um estoque de sua conta;  
2 - Esta posição é reservada na sua conta, por isso não pode ser vendida várias vezes;  
3 - Como custa dinheiro fazer um pedido no mercado - uma taxa será cobrada da conta do cliente;  
4 - O sistema precisa comunicar esse pedido ao mercado de ações apropriado;

A figura abaixo mostra o fluxo de comunicação através de microserviços em uma aplicação que permite o usuário vender posições de suas ações no mercado:



Nesta figura podemos identificar dois atributos fundamentais dos microserviços:

1. Cada microserviço pode ser implantado de forma independente;
2. Um microserviço possui apenas uma responsabilidade e é substituível;

A ideia de que os microsserviços são responsáveis por coordenar ações em um sistema é a diferença crucial entre essa abordagem e as arquiteturas orientadas a serviços tradicionais (SOAs). 

Esses tipos de sistemas (SOAs) costumavam usar barramentos de serviço corporativo (ESBs) ou padrões de orquestração mais complexos para externalizar a orquestração de mensagens e processos dos próprios aplicativos. Nesse modelo, os serviços freqüentemente careciam de coesão, com a lógica de negócios sendo cada vez mais acrescida ao barramento de serviço, ao invés dos próprios serviços.

É interessante pensar em como o desacoplamento das funcionalidades no sistema de investimento on-line o ajuda a ser mais flexível face à mudança de requisitos.   
  
Imagine os seguintes cenários:  
  
1- Você precisa alterar como as taxas são calculadas; *(alterar um requisito existente)*Você poderia fazer e liberar essas mudanças no serviço de taxas, sem qualquer alteração nos demais serviços.   
  
2- Quando um pedido for feito, você precisa alertar a equipe de risco; *(temos aqui um novo requisito)*  
Seria fácil criar um novo microsserviço para realizar essa operação com base em um evento gerado pelo serviço de pedidos sem ter que alterar o resto do sistema;

Para concluir esta parte do artigo podemos *pensar que o estilo de arquitetura de microsserviços é uma abordagem que desenvolve um aplicativo único como uma suite de pequenos serviços, cada um executando seu próprio processo e se comunicando através de mecanismos leves, muitas vezes em uma API com recursos HTTP.*

*Esses serviços são construídos em torno de capacidades de negócios e funcionam através de mecanismos de deploy independentes totalmente automatizados. Há o mínimo possível de gerenciamento centralizado desses serviços, que podem ser escritos em diferentes linguagens de programação e utilizam diferentes tecnologias de armazenamento de dados.  
(citação integral de :*[*https://martinfowler.com/articles/microservices.html*](https://martinfowler.com/articles/microservices.html)*acessado em outubro/2018)*

Desta forma ,a arquitetura de microservices tem muitas vantagens, por exemplo, serviços individuais são mais fáceis de entender e podem ser desenvolvidos e implantados de forma independente. Adotar novas tecnologias e frameworks torna-se mais fácil, pois a adoção pode ser aplicada em um serviço de cada vez.

Mas nem tudo são flores, pois as aplicações com muitos microsserviços tornam-se mais complexas de gerenciar pois são constituídas por mais elementos, e, para ser utilizada de forma eficaz, a arquitetura de microservices exige um alto nível de automação, como um PaaS, além disso, para desenvolver microsserviços, é necessário lidar com alguns problemas complexos de gerenciamento de dados distribuídos.