# Desinging Microservies Architecture

* O design tem que ser metódico
* Não se pode correr com o desenvolvimento
* Planejar mais, codificar menos
* Planejamento é crítico para o sucesso

## Passos Importantes

* Entender os Requisitos do Sistema proposto
* Entender os Requisitos não Funcionais
* Mapear os Componentes
* Selecionar o Stack de tecnologias
* Desenhar a arquitetura
* Documento a arquitetura
* Dar suporte ao time

Abaixo vamos falar um pouco sobre os passos mais críticos.

## Mapeando os Componentes

* Este é o passo mais importante de todos
* Determina como o sistema vai ficar a longo prazo
* É algo feito apenas uma vez, e é difícil ser modificado futuramente, pois este mapeamento que vai nortear toda a arquitetura e desenvolvimento
* O que fazer neste passo:
  + Definir os variados componentes do sistema
  + Lembrar que Componentes são Serviços
* O que levar em conta:
  + Requisitos de Negócio
  + Autonomia Funcional
  + Entidades de Dados
  + Autonomia dos Dados
* Requisito de Negócio
  + Requisitos ao redor das capacidades específicas do negócio
  + Por exemplo: Gerenciador de Ordens
    - Add, Remove, Update, Calculate Amount (of ORDERS)
* Autonomia Funcional
  + O Sistema deve realizar todas as funções relacionadas a determinado Requisito de Negócio, e não deve realizar funções de outros requisitos, que devem ser executados por outros Serviços.
  + Por exemplo: Gerenciador de Ordens
    - Recuperar as ordens feitas na última semana (deve estar incluída neste Serviço)
    - Pegar todas as ordens feitas por usuários de 34-45 anos (não deve estar incluída neste Serviço, pois utilizado dados que não fazem parte deste sistema)
  + Porém é comum se deparar com áreas cinzentas, que será necessário trabalhar com dados duplicados, ou que dependem de comunicação com outros serviços
* Entidades de Dados
  + Serviço é criado ao redor das entidades de dados
  + Por exemplo: ordens, itens
  + Dados devem ser relacionados apenas pelo ID
    - Exemplo: Ordem guarda apenas o ID do Customer
* Autonomia dos Dados
  + Os dados devem ser atômicos (menores unidades possível)
  + Serviço não pode depender de dados de outros serviços para funcionar

Exemplo:

Forma

Descrição gerada automaticamente

Problema 1:

* Temos que pegar a quantidade de ordens de uma lista de customers. Como executar, se temos um Serviço para Ordens e outro para Customers.
* Solução 1:
  + Duplicar dados, por exemplo, salvando os dados das ordens no lado do Customer

Diagrama, Forma

Descrição gerada automaticamente

* Solução 2:
  + Um serviço realizar uma consulta para o outro, mas pode onerar a rede dependendo da volumetria

Forma

Descrição gerada automaticamente

* Solução 3
  + Criar um terceiro Serviço de agregação

Forma, Polígono

Descrição gerada automaticamente

Problema 2:

* Retornar uma lista de todas as ordens no sistema
* Problema é o grande volume de dados
* Serviços não são desenhados para este cenário
* Entender bem o por que da necessidade de fazer essa query
* Talvez um “Report engine” seria o mecanismo ideal para este cenário

Continuando o Mapeamento

* Cross-cutting services, serviços que fornecem :
  + Logging
  + Caching
  + Gerenciamento de usuários
* Eles devem fazer parte do mapeamento

## Definindo os Padrões de Comunicação

* Comunicação entre serviços são críticos
* Importante escolher o padrão correto
* Principais padrões:
  + 1-1 Sync
  + 1-1 Async
  + Pub-Sub/Event Driven
* 1-1 Syns
  + Um serviço chama outro serviço, e aguarda pela resposta
  + Quando o primeiro serviço precisa da resposta para continuar o processamento
  + Acaba criando Conexões Diretas entre serviços

Interface gráfica do usuário, Aplicativo, Teams

Descrição gerada automaticamente

Diagrama

Descrição gerada automaticamente

Problema abaixo relacionado com conexões diretas. A alteração em 1 serviços acaba impactando em múltiplos outros serviços.

Diagrama

Descrição gerada automaticamente

Para diminuir o acoplamento mostrado acima, pode-se utilizar o Service Discovery ou Gateway:

Diagrama

Descrição gerada automaticamente

Diagrama

Descrição gerada automaticamente

Service Discovery é mais simples de implementar, porém o Gateway tem mais funcionalidades como monitoramento, autenticação e autorização, e melhor rastreabilidade.

* 1-1 Async
  + Serviço não espera pela resposta – Fire and Forget pattern
  + Este padrão é utilizado principalmente qnd um serviço quer passar uma mensagem para o outro
  + O primeiro serviço não se preocupa/ não importa se houve erro

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente com confiança média

Uma imagem contendo relógio, desenho

Descrição gerada automaticamente

* Pub-Sub / Event Driver
  + Qnd um serviço quer notificar outros serviços
  + O serviço não tem dieia de qtos serviços estão ouvindo a mensagem
  + Não aguarda pelo resposta
  + Usado principalmente qnd o serviço que notificar sobre um evento importante

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Diagrama

Descrição gerada automaticamente

## Definindo o Stack de Tecnologia