Modelos, Métodos e Técnicas de Engenharia de Software

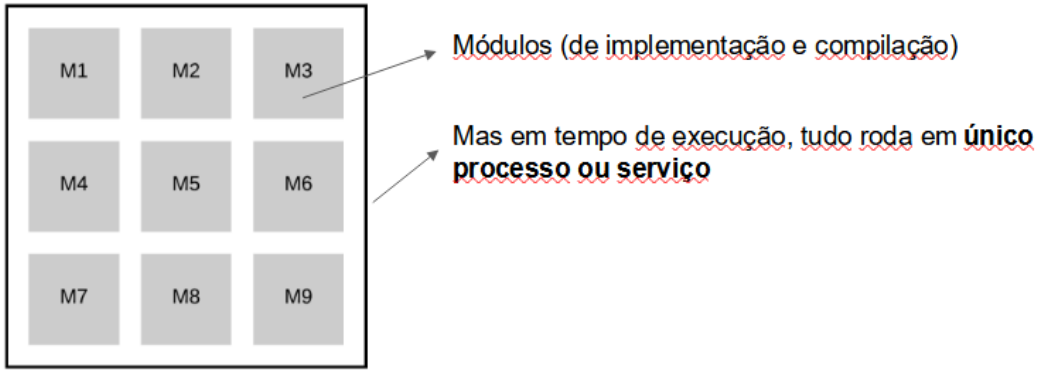
Prof.: Sônia Aparecida Santana

una

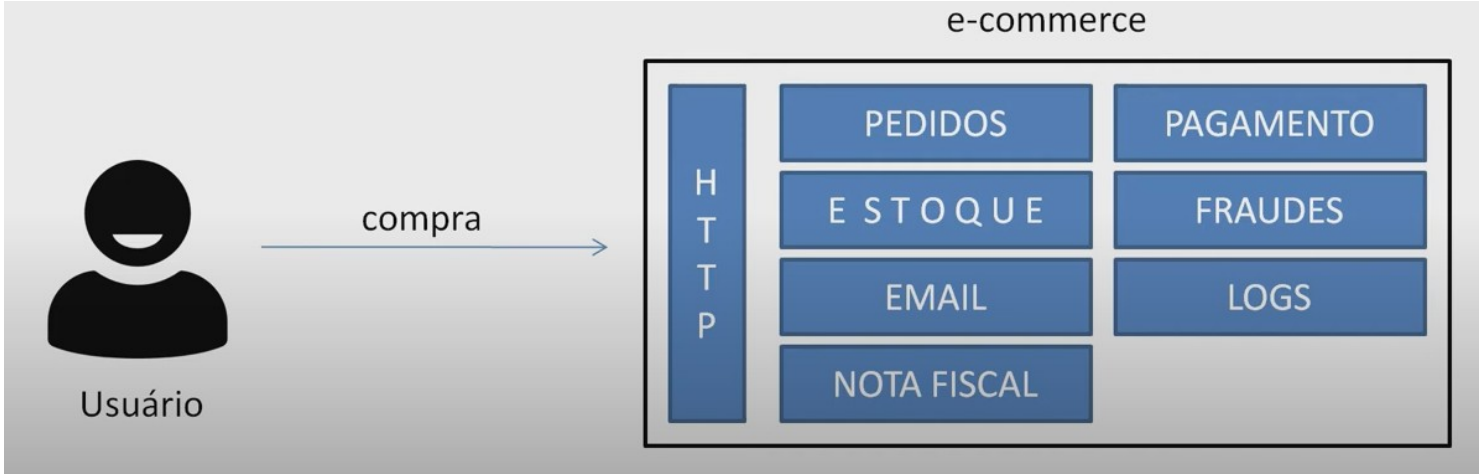
Monolitos

Monolitos

- Monolitos: em tempo de execução, sistema é um único processo (processo aqui é processo de sistema operacional)

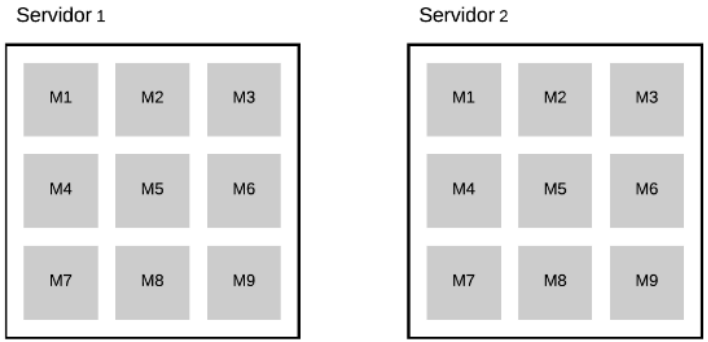


Monolitos: exemplo



Problemas com Monolito

- Escalabilidade
  - Deve-se escalar o monolito inteiro, mesmo quando o gargalo de desempenho está em um único módulo



Problemas com Monolito

- Processo de release é lento, centralizado e burocrático
- Times não tem poder para colocar módulos em produção
  - Motivo: mudanças em um módulo podem impactar módulos que já estejam funcionando
- Acabam existindo:
  - Datas pré-definidas para release
  - Processo de "homologação"

Problemas com Monolito

- Riscos de adicionar novas features em um código existente (principalmente, se monolítico)

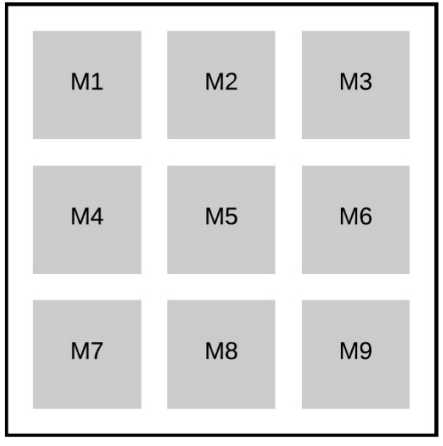
Microsserviços

Microsserviços

- Módulos (ou conjuntos de módulos) viram processos independentes em tempo de execução
- Esses módulos são menores do que de um monolito
  - Daí o nome microsserviço

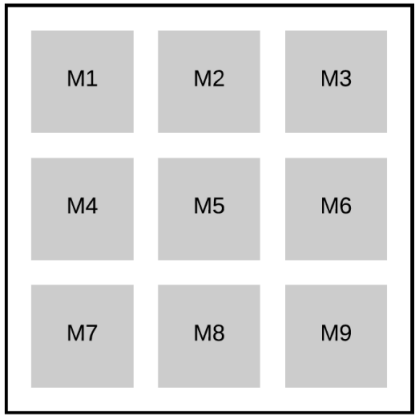
Microsserviços

Arquitetura
Monolítica

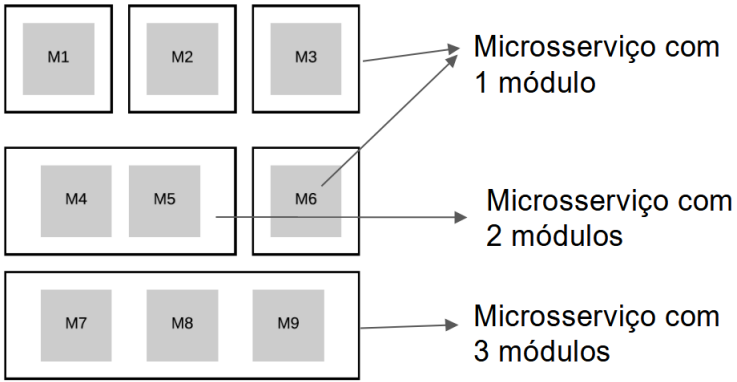


Microsserviços

Arquitetura Monolítica



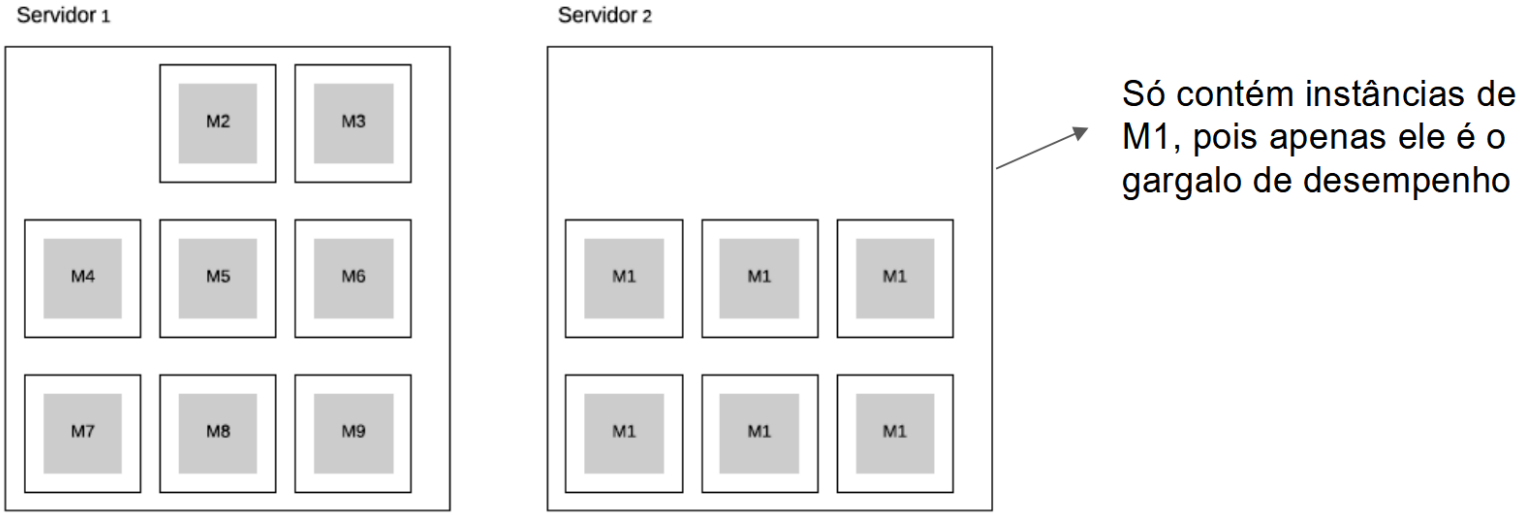
Arquitetura baseada em Microsserviços



microsserviço = processo (run-time, sistema operacional)

Microsserviços: Vantagens

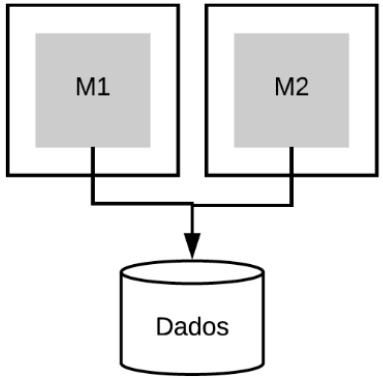
- Pode-se escalar apenas o módulo com problema de desempenho



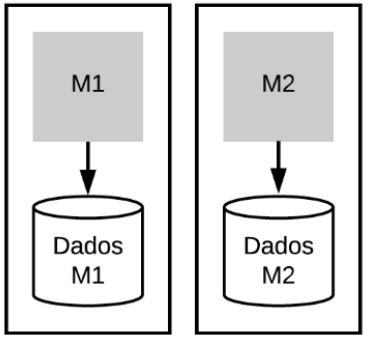
Microsserviços: Vantagens

- Times ganham autonomia para colocar microsserviços em produção
- Processo = espaço de endereçamento próprio
- Chances de interferências entre processos são menores
- Tecnologias diferentes
- Falhas parciais.
  - Exemplo: apenas o sistema de recomendação pode ficar fora do ar

Gerenciamento de Dados com Microsserviços



Arquitetura que não é recomendada.
Motivo: aumenta acoplamento entre M1 e M2



Arquitetura recomendada. Motivo: não existe acoplamento de dados entre M1 e M2. Logo, M1 e M2 podem evoluir de modo independente.
Se M1 precisar usar serviços de M2 (ou vice-versa), isso deve ocorrer via interfaces

Quando não Usar Microsserviços

- Arquitetura com microsserviços é mais complexa
  - Sistema distribuído (gerenciar centenas de processos)
  - Latência (comunicação é via rede)
  - Transações distribuídas

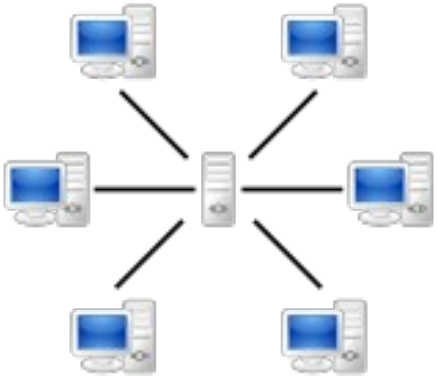
Recomendações

- Começar com monolitos e pensar em microsserviços quando:
  - O monolito apresentar problemas de desempenho
  - O monolito estiver atrasando o processo de release
  - Migração pode ser gradativa (microsserviços gradativamente extraídos do monolito)

Outros Padrões Arquiteturais

Cliente Servidor

- Muito comum em serviços básicos de redes
  - Exemplos:
    - Serviço de impressão
    - Serviço de arquivos
    - Serviço Web



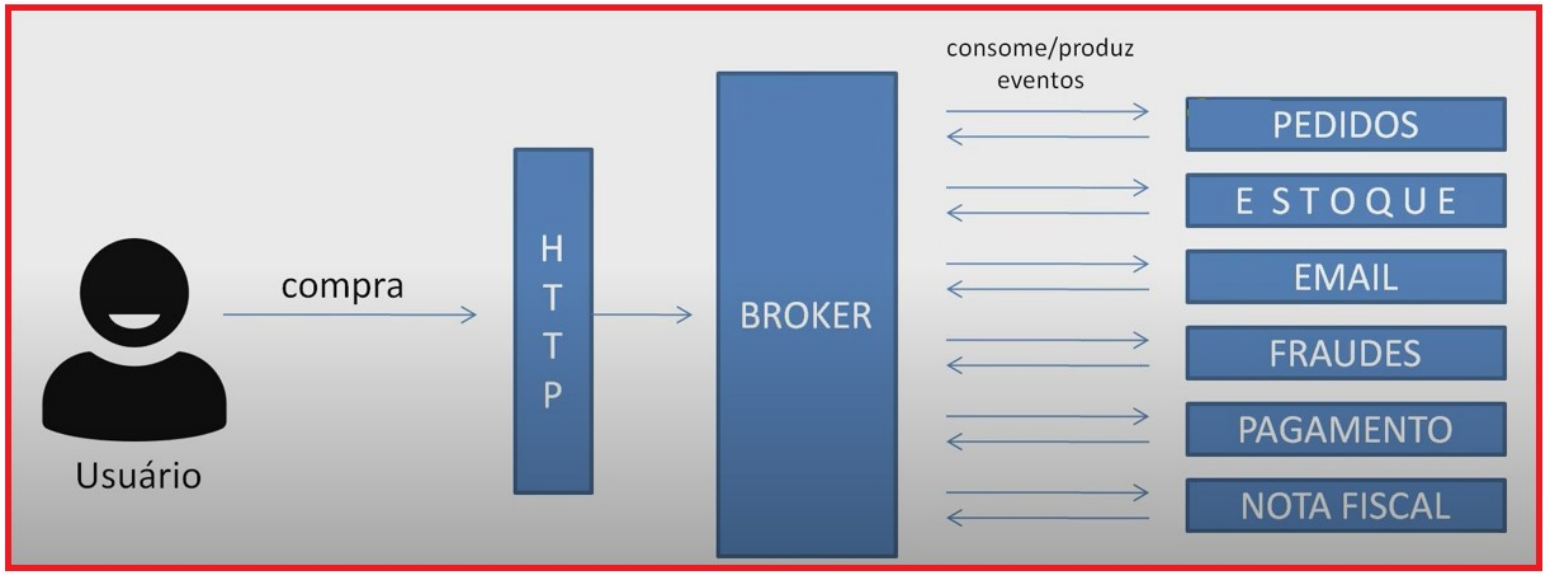
Publish/Subscribe

- Em arquiteturas publish/subscribe, as mensagens são chamadas de eventos. Os componentes da arquitetura são chamados de publicadores (publishers) e assinantes (subscribers) de eventos.
- Publicadores produzem eventos e os publicam no serviço de publish/subscribe, que normalmente executa em uma máquina separada. Assinantes devem previamente assinar eventos de seu interesse

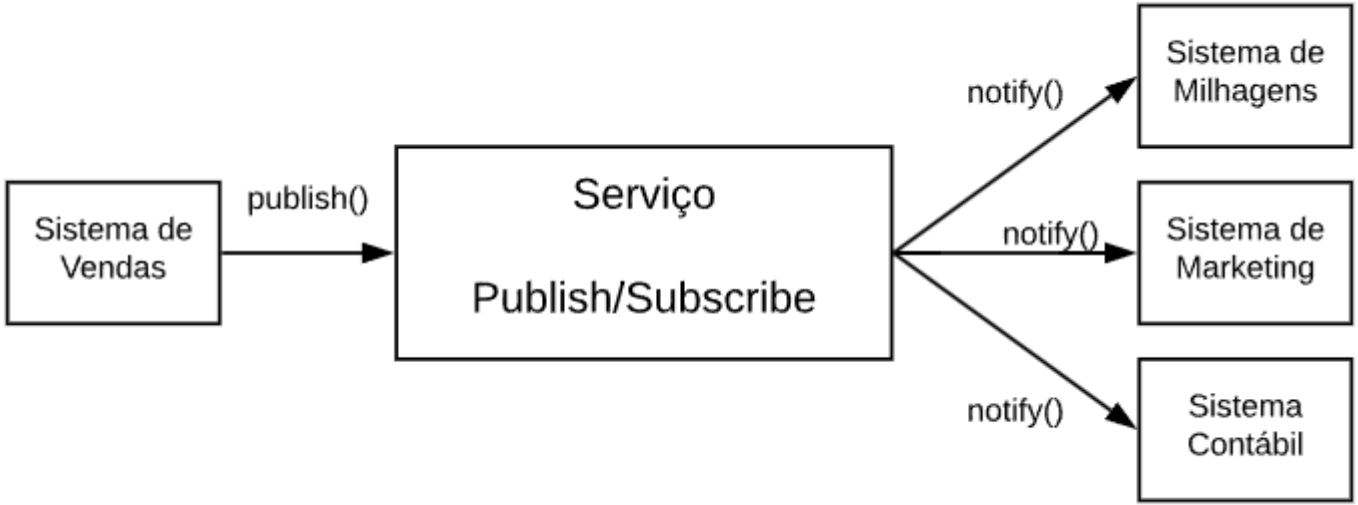
Publish/Subscribe

- Arquiteturas publish/subscribe são, às vezes, chamadas de arquiteturas orientadas a eventos.
- O serviço de publish/subscribe, às vezes, é chamado também de broker de eventos, pois ele funciona como um barramento por onde devem trafegar todos os eventos

Publish Subscribe



Publish Subscribe: exemplo



Publish Subscribe: exemplo

- Comunicação em grupo, pois o mesmo evento é assinado por vários sistemas;
- Desacoplamento no espaço, pois o sistema de vendas não tem conhecimento dos sistemas interessados nos eventos que ele publica;
- Desacoplamento no tempo, pois o sistema de publish/subscribe reenvia os eventos caso os sistemas assinantes estejam fora do ar;
- Notificação assíncrona, pois os assinantes são notificados assim que um evento ocorre; isto é, eles não precisam consultar periodicamente o sistema publish/subscribe sobre a ocorrência dos eventos de interesse.

Orientação a Mensagens

- Assim como ocorre quando se usa filas de mensagens, arquiteturas publish/subscribe também oferecem desacoplamento no espaço e no tempo.
- No entanto, existem duas diferenças principais entre publish/subscribe e sistemas baseados em filas de mensagens

Publish Subscribe x Orientação a Mensagens

- Em publish/subscribe, um evento gera notificações em todos os seus assinantes. Por outro lado, em filas de mensagens, as mensagens são sempre consumidas — isto é, retiradas da fila — por um único servidor.
- Portanto, em publish/subscribe temos um estilo de comunicação de 1 para n, também conhecido como comunicação em grupo.
- Já em filas de mensagens, a comunicação é 1 para 1, também chamada de comunicação ponto-a-ponto.

Publish Subscribe x Orientação a Mensagens

- Em publish/subscribe, os assinantes são notificados assincronamente. Primeiro, eles assinam certos eventos e, então, continuam seu processamento. Quando o evento de interesse ocorre, eles são notificados por meio da execução de um determinado método.
- Por outro lado, quando se usa uma fila de mensagens, os servidores — isto é, os consumidores das mensagens — têm que puxar (pull) as mensagens da fila

Anti Padrões Arquiteturais

- Modelo que não deve ser seguido
- Revela um sistema com sérios problemas arquiteturais

Big Ball of Mud

- Um módulo pode usar praticamente qualquer outro módulo do sistema; ou seja, siste "