



Instituto Infnet

# Aula 6 - Arquitetura de Microsserviços e Mobile

## **Aula 6**

- Comunicações entre Microserviços.

## **Ementa proposta Aula 6**

- Estudar segurança em microserviços;
- Estudar resiliência em microserviços;
- Estudar escalabilidade de microserviços.



# Dúvidas sobre o projeto de disciplina?



# Exemplo Prático de Revisão

Sistema de simulador de provas de concurso

- <https://github.com/engRenanTorres/engenhariaDeConcursos/tree/main>  
<https://www.engenhariadeconcursos.com.br/>
- Tecnologias:  
React.js -> Nest.js -> MariaDB
- Para o exemplo LocalStack AWS:
- <https://www.localstack.cloud/>



# Comunicação entre Microserviços

Inter-process vs In-process

- Desempenho;
- Mudança de interfaces;
- Tratamento de erros.



# Comunicação entre Microserviços

## Inter-process vs In-process

- Desempenho;
- Mudança de interfaces

“Ao fazer uma alteração incompatível com versões anteriores em uma interface de microserviço, precisamos fazer uma implantação em etapas com os consumidores, certificando-nos de que eles sejam atualizados para usar a nova interface”;
- Tratamento de erros;




# Comunicação entre Microserviços

## Inter-process vs In-process


- Desempenho;
- Mudança de interfaces;
- Tratamento de erros

“...(em Microserviços) erros que estão fora do seu controle . As redes atingem o tempo limite. Os microserviços downstream podem ficar temporariamente indisponíveis. As redes são desconectadas, os contêineres são mortos devido ao consumo de muita memória e, em situações extremas, partes do seu data center podem pegar fogo”;



# Comunicação entre Microserviços

## Inter-process vs In-process

- Tratamento de erros:
    - Falha de colisão  
“Estava tudo bem até o servidor cair. Reinicie!”;
    - Falha de omissão  
“Você enviou algo, mas não obtive resposta.”;
    - Falha de tempo  
“Algo aconteceu tarde, ou cedo, demais (você não recebeu a tempo)!;
    - Falha de resposta  
“Você recebeu uma resposta, mas ela parece errada.”;
    - Falha de arbitrária  
“quando algo deu errado, mas os participantes não conseguem concordar se a falha ocorreu (ou por quê)”.
- 



# Comunicação entre Microserviços

## Inter-process vs In-process

- Tratamento de erros:
  - Falha de colisão  
“Estava tudo bem até o servidor cair. Reinicie!”;
  - Falha de omissão  
“Você enviou algo, mas não obtive resposta.”;
  - Falha de tempo  
“Algo aconteceu tarde, ou cedo, demais (você não recebeu a tempo)!;
  - Falha de resposta  
“Você recebeu uma resposta, mas ela parece errada.”;
  - Falha de arbitrária  
“quando algo deu errado, mas os participantes não conseguem concordar se a falha ocorreu (ou por quê)”.
- **Importante usar um protocolo como o HTTP!**

# Tecnologias p/ Comunicação de Microserviços

“E em um mundo onde temos muitas opções e pouco tempo, a coisa óbvia a fazer é simplesmente ignorar as coisas.” - Seth Godin



# Tecnologias p/ Comunicação de Microserviços

Problemas comuns:

- Escolher as tecnologias que lhe são familiares (sem avaliar os trade-offs);
- Escolher sempre as mais recentes (sem avaliar os trade-offs).



# Estilos p/ Comunicação de Microsserviços

Primeiro analisamos o estilo de comunicação mais adequado, para depois avaliar a tecnologia que será usada.



# Comunicação entre Microserviços

Entender os mecanismos de comunicações existentes é fundamental para escolher os microserviços. Entre eles temos:

- Comunicação síncrona bloqueante vs Comunicação assíncrona sem bloqueio;
- Solicitação-resposta vs orientada a eventos.



# Estilos p/ Comunicação de Microserviços

Estilos:

- Bloqueio síncrono

Um microserviço faz uma chamada para outro microserviço e bloqueia a operação aguardando a resposta.

- Não bloqueante assíncrono

O microserviço que emite uma chamada é capaz de continuar o processamento, independentemente de a chamada ser recebida ou não.



# Estilos p/ Comunicação de Microserviços

Estilos:

- Solicitação-resposta

Um microserviço envia uma solicitação a outro microserviço pedindo que algo seja feito. Ele espera receber uma resposta informando-o do resultado.

- Orientado por eventos

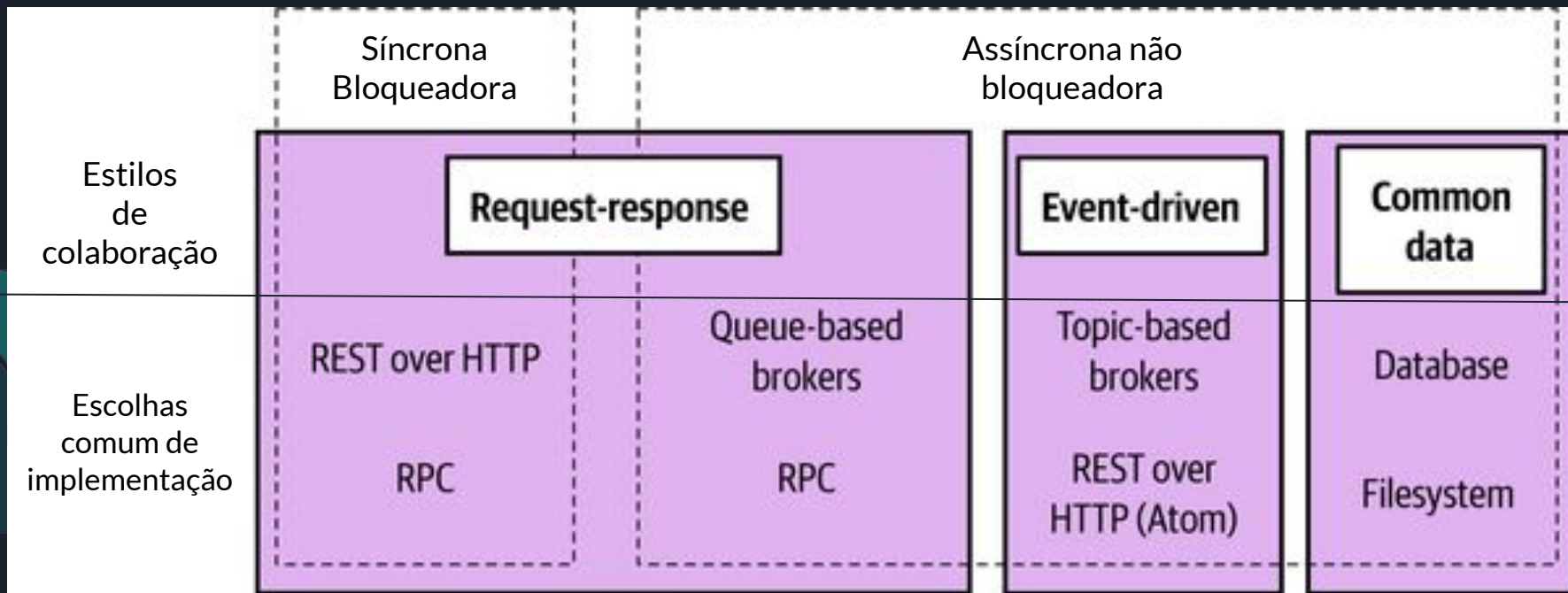
Microserviços emitem eventos, que outros microserviços consomem e reagem de acordo. O microserviço que emite o evento não tem conhecimento de quais microserviços, se houver, consomem os eventos que ele emite.

- Dados comuns

Não frequentemente vistos como um estilo de comunicação, os microserviços colaboram por meio de alguma fonte de dados compartilhada.

# Estilos p/ Comunicação de Microsserviços

Estilos:





# Estilos p/ Comunicação de Microserviços

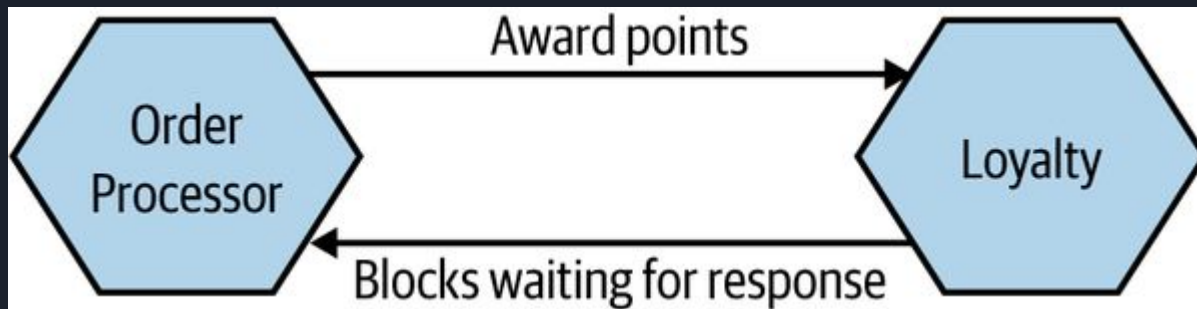
Estilos:

- O padrão em microserviços é combinar os estilos;
- É comum que um único microserviço implemente mais de uma forma de colaboração



# Padrão: Síncrono bloqueante

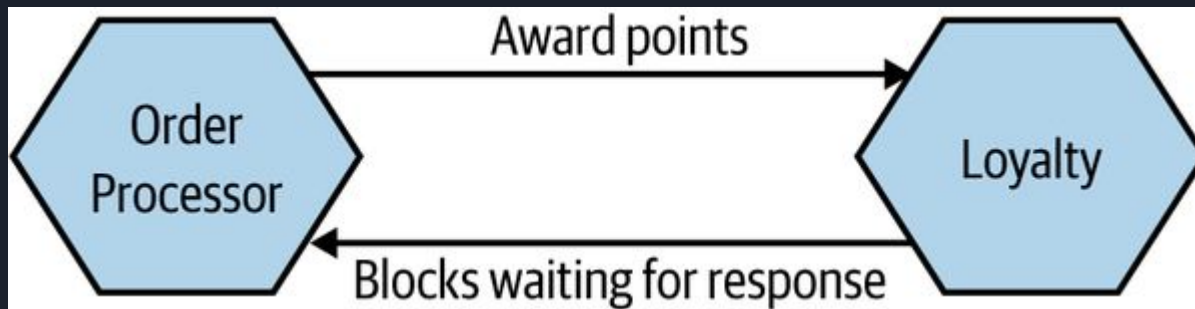
- Normalmente, uma chamada de bloqueio síncrona é aquela que está esperando por uma resposta do processo downstream.



# Padrão: Síncrono bloqueante

Vantagens:

- Simples;
- Familiar / Padrão na aprendizagem.



# Padrão: Síncrono bloqueante

Desvantagens:

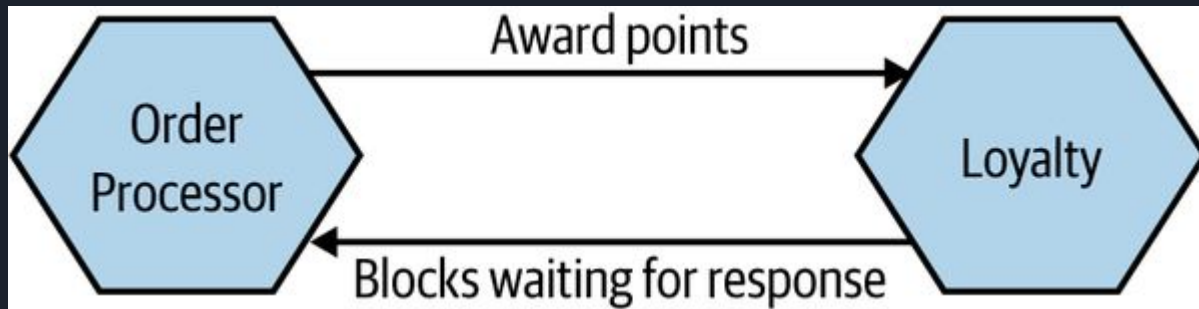
- Acoplamento Temporal;



# Padrão: Síncrono bloqueante

Onde usar:

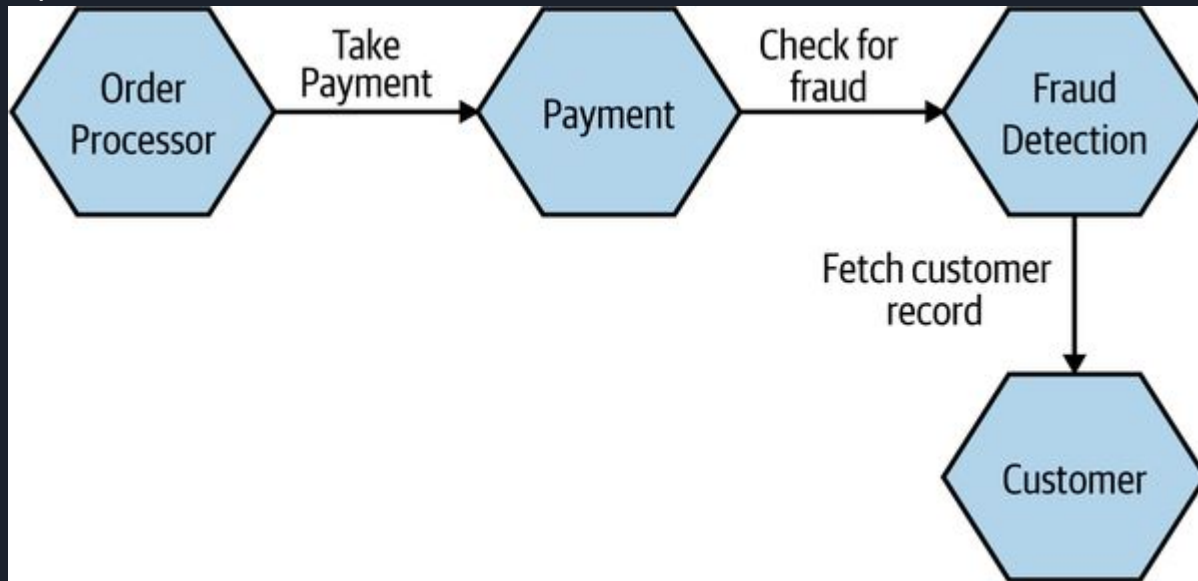
- Arquiteturas simples de microsserviços;
- Situações em que o um microsserviço precisa saber se a chamada funcionou;
- Cadeias de chamadas curtas.



# Padrão: Síncrono bloqueante

Onde não usar:

- Cadeias de chamadas longas - (gera conexões abertas em cascata e por longos períodos).



# Padrão: Assíncrono Não Bloqueante

O ato de enviar uma chamada pela rede não bloqueia o microserviço que emite a chamada. Ele é capaz de continuar com qualquer outro processamento sem ter que esperar por uma resposta.



# Padrão: Assíncrono Não Bloqueante

Estilos mais comuns:

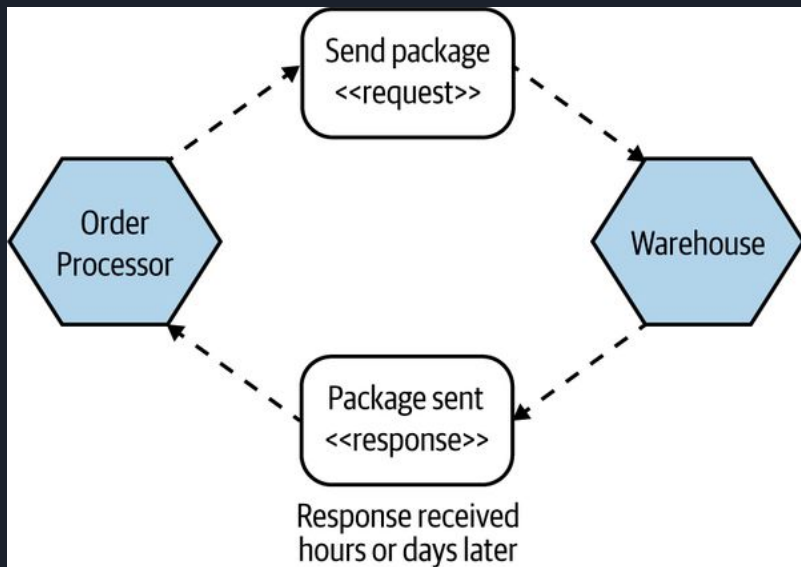
- Comunicação através de dados comuns  
“O microserviço upstream altera alguns dados comuns, que um ou mais microserviços posteriormente utilizam.”
- Solicitação-resposta  
“Um microserviço envia uma solicitação a outro microserviço pedindo que ele faça algo. Quando a operação solicitada é concluída, seja com sucesso ou não, o microserviço upstream recebe a resposta. Especificamente, qualquer instância do microserviço upstream deve ser capaz de manipular a resposta.”
- Interação orientada a eventos  
“Um microserviço transmite um evento, que pode ser pensado como uma declaração factual sobre algo que aconteceu. Outros microserviços podem ouvir os eventos em que estão interessados e reagir de acordo.”



# Padrão: Assíncrono Não Bloqueante

Vantagens:

- Desacoplados temporal;
- Os microsserviços que recebem a chamada não precisam estar acessíveis ao mesmo tempo em que a chamada é feita.

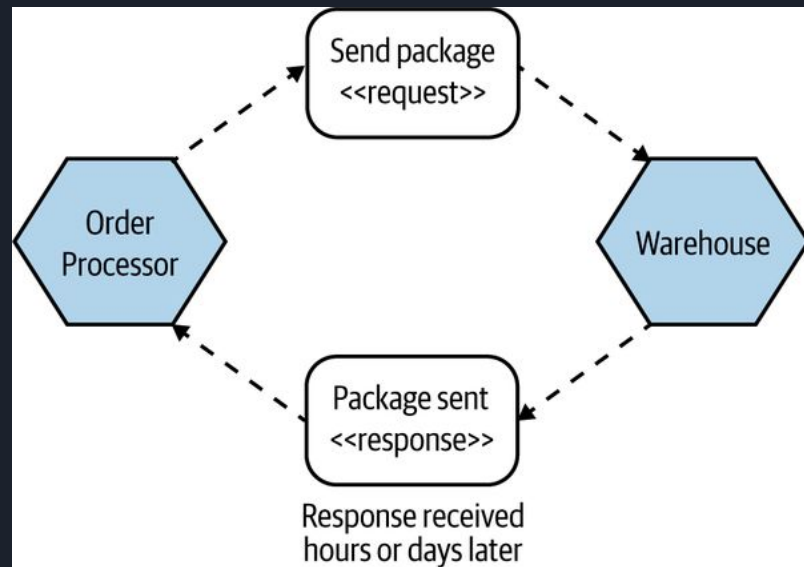


Exemplo assíncrono de comunicação de solicitação-resposta

# Padrão: Assíncrono Não Bloqueante

Desvantagens:

- Nível de complexidade;
- Gama de escolhas;
- Falta de familiaridade.



# Async/Await não é “Assíncrono Não Bloqueante”

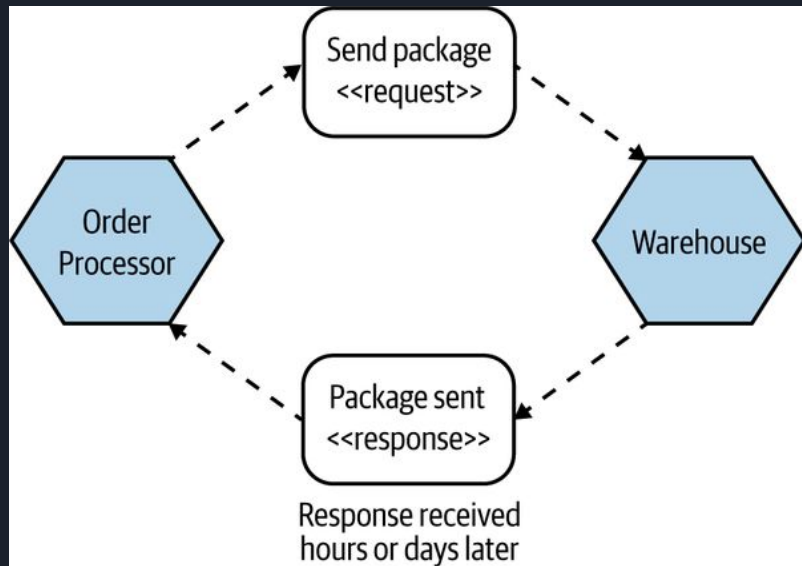
- Async/Await é potencialmente assíncrona, mais em um estilo síncrono e bloqueador.

```
async function f() {  
  
  let eurToGbp = new Promise((resolve, reject) => {  
    //code to fetch latest exchange rate between EUR and GBP  
    //exchange = taxa de câmbio  
    ...  
  });  
  
  var latestRate = await eurToGbp; //1  
  process(latestRate); //2  
}
```

# Padrão: Assíncrono Não Bloqueante

Onde usar:

- Dependerá do tipo de chamada Assíncrona;
- Cadeias de chamadas longas;
- Processos de longa duração;

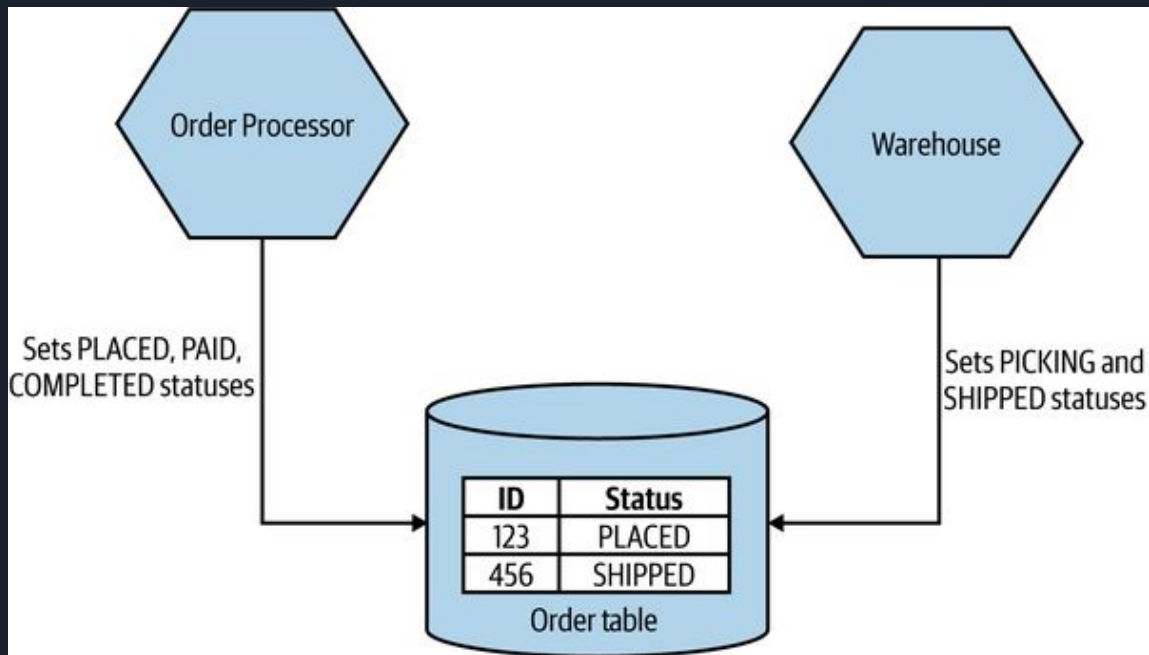


# Padrão: Assíncrono Não Bloqueante

Comunicação por meio de dados comuns

Inserir dados em um local para ser usado por outros microserviços depois

- Casos de uso: Data lakes e Data warehouse;



# Padrão: Assíncrono Não Bloqueante

Comunicação por meio de dados comuns

Vantagens:

- Simples;
- Tecnologias conhecidas;
- Interoperabilidade entre sistemas até com aplicações mais antigas;
- Suporte alto volume de dados (Sem preocupação com latência).



# Padrão: Assíncrono Não Bloqueante

Comunicação por meio de dados comuns

Desvantagens:

- Exige pesquisa sobre mecanismo de verificação de novos dados;
- Acoplamento comum - Alterações na estrutura podem quebrar os sistemas.



# Padrão: Assíncrono Não Bloqueante

Comunicação por meio de dados comuns

Onde usar:

- Sistemas que precisam de interoperabilidade;
- Compartilhar grandes dados (onde não há preocupação com latência).

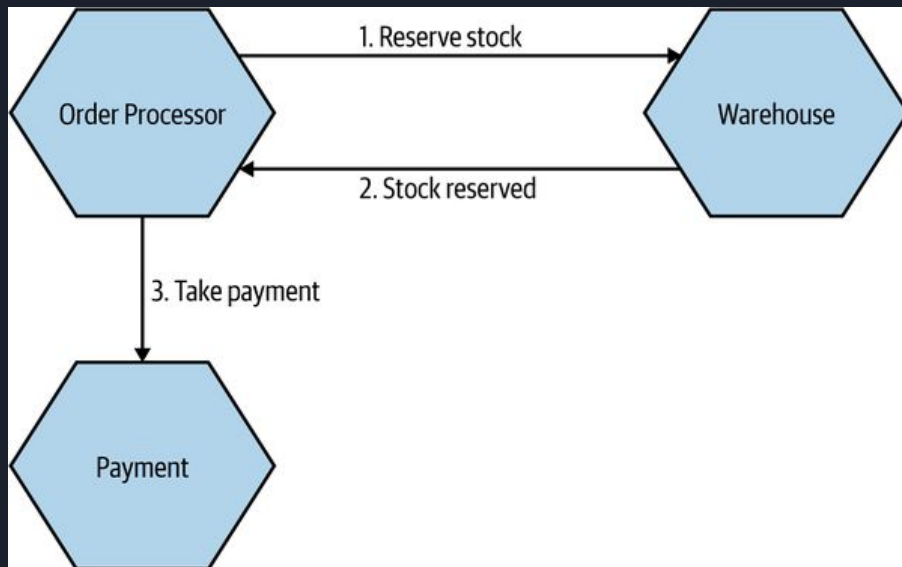




# Padrão: Assíncrono Não Bloqueante

Comunicação por solicitação-resposta

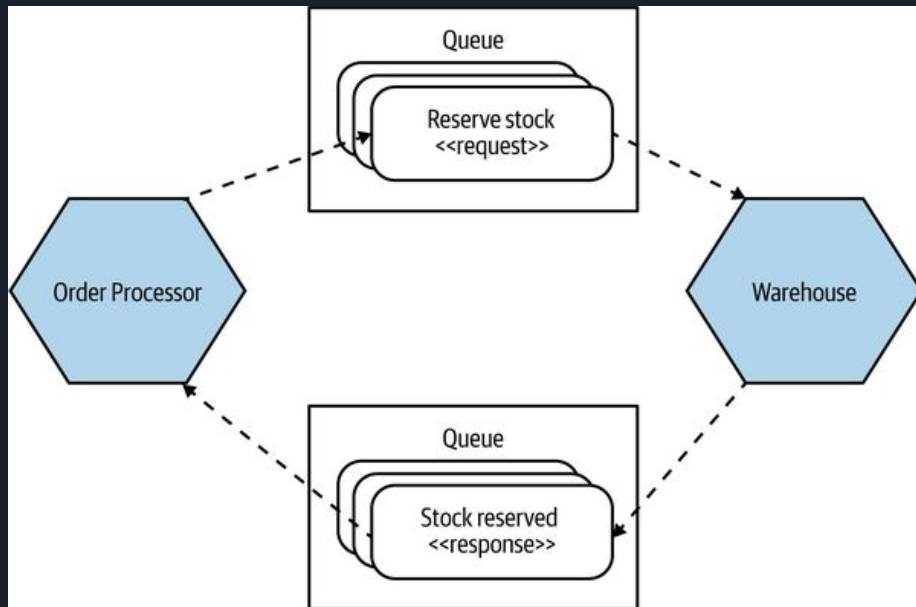
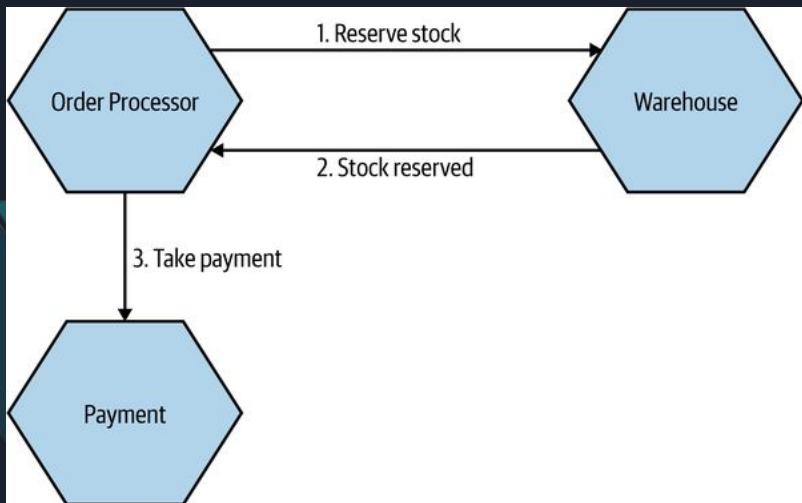
um microserviço envia uma solicitação a um serviço downstream solicitando que ele faça algo e espera receber uma resposta com o resultado da solicitação.



# Padrão: Assíncrono Não Bloqueante

Comunicação por solicitação-resposta

Diferença entre solicitação-resposta síncrona bloqueante e assíncrona não bloqueante.



# Padrão: Assíncrono Não Bloqueante

Comunicação por solicitação-resposta

Onde usar:

- Qualquer situação onde precisa de uma resposta sem manter conexão aberta;
- Situações em que o um microserviço precisa saber se a chamada funcionou;



# Padrão: Comunicação Orientada a Eventos

Em vez de um microserviço pedir para outro microserviço fazer algo, um microserviço emite eventos que podem ou não ser recebidos por outros microserviços.

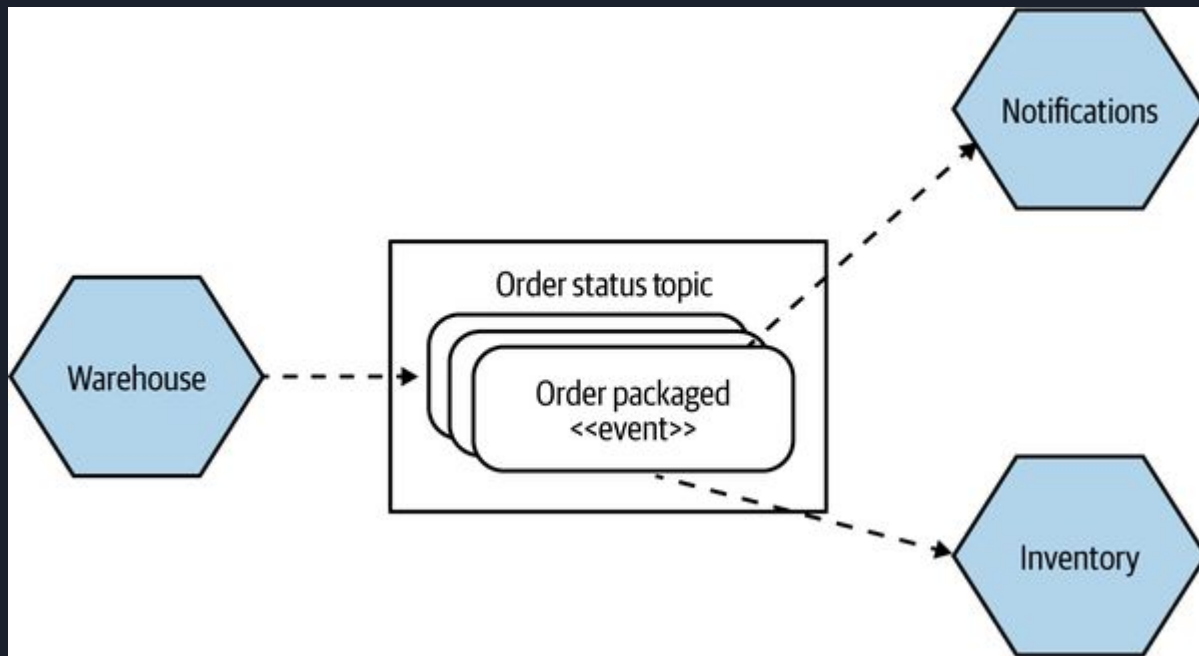
...

Um evento é uma declaração sobre algo que ocorreu, quase sempre algo que aconteceu dentro do mundo do microserviço que está emitindo o evento. O microserviço que emite o evento não tem conhecimento da intenção de outros microserviços de usar o evento e, de fato, pode nem estar ciente de que outros microserviços existem.




# Padrão: Comunicação Orientada a Eventos

Em vez de um microserviço pedir para outro microserviço fazer algo, um microserviço



# EVENTOS VS MENSAGENS

- Um evento é um fato — uma declaração de que algo aconteceu;
  - Uma mensagem é algo que enviamos por um mecanismo de comunicação assíncrono;
  - Com colaboração orientada a eventos, queremos transmitir esse evento, e uma maneira típica de implementar esse mecanismo de transmissão seria colocar o evento em uma mensagem. A mensagem é o meio; o evento é o payload;
  - Ao enviar uma solicitação como carga útil de uma mensagem, estaríamos implementando uma forma de solicitação-resposta assíncrona
- 

# Padrão: Comunicação Orientada a Eventos

Implementação:

- Precisa definir uma maneira dos produtores emitirem eventos;
- Precisa definir uma maneira dos consumidores emitirem eventos.

Ex.:

- RabbitMQ (Gerenciador de filas - Lidam com assinaturas);
- Atom (HTTP - compatível com REST)



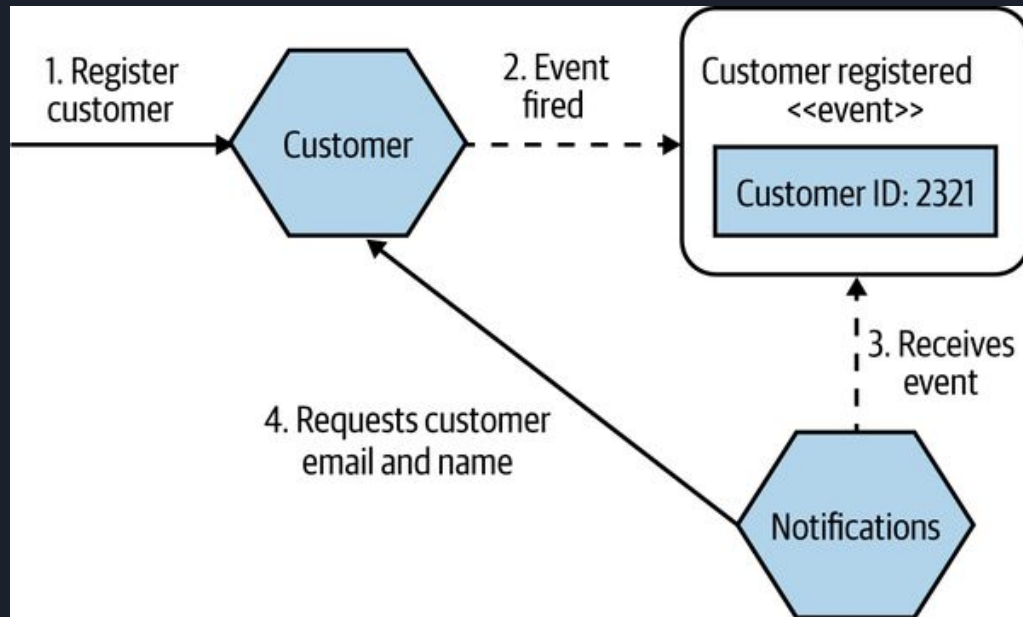
# Padrão: Comunicação Orientada a Eventos

Tipos de composições dos eventos:

- Eventos com apenas identificador

Desvantagem:

- Acoplamento domínio;
- Possibilidade de muitas chamadas retornando ao mesmo tempo.





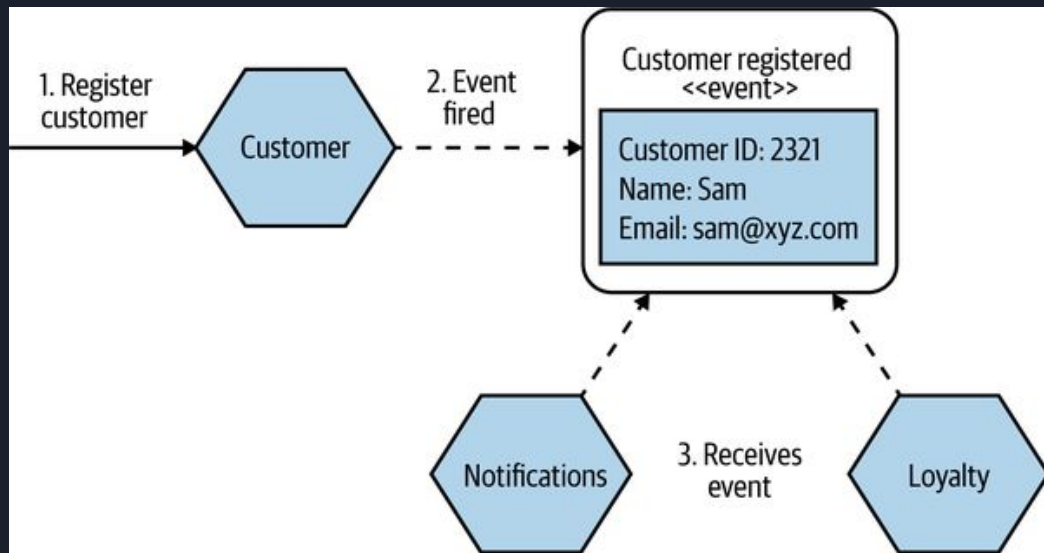
# Padrão: Comunicação Orientada a Eventos

Tipos de composições dos eventos:

- Eventos totalmente detalhados

Desvantagem:

- Preocupações sobre o tamanho;
- Baixa ocultação de informações;
- Os dados transmitidos se tornam contratos e podem quebrar outras aplicações ao serem alterados.



# Resumo dos Temas vistos





Instituto Infnet

# Aula 6 - Arquitetura de Microsserviços e Mobile