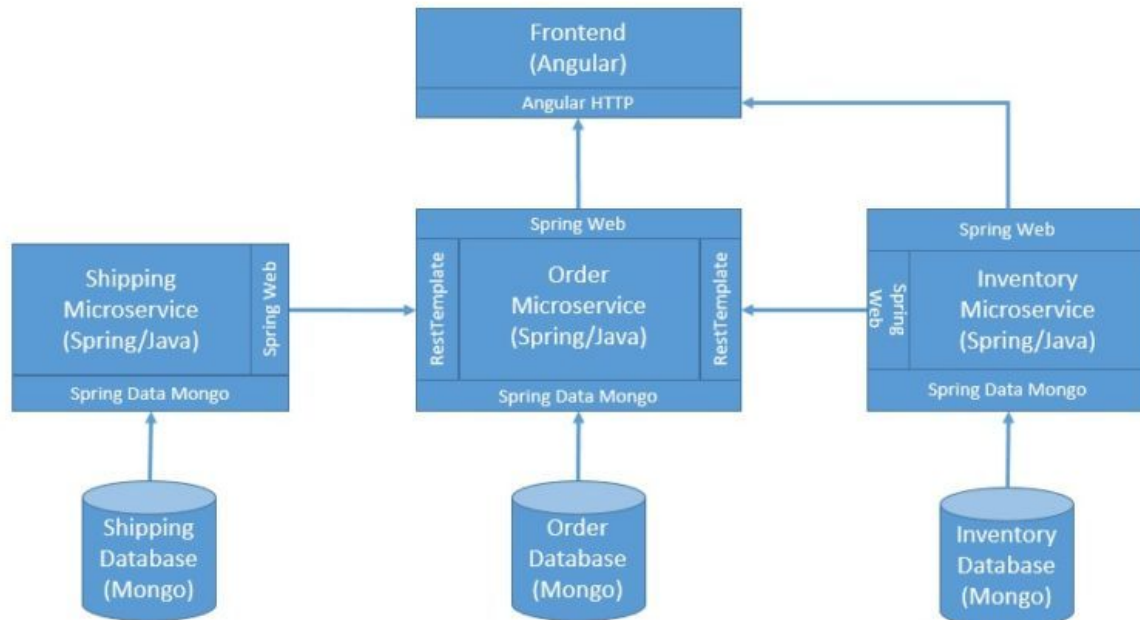


# Sistemas reativos anotações de estudos.

Sistema usado como exemplo:



No ano de 2013, **uma equipe de desenvolvedores, liderada por Jonas Boner, se reuniu para definir um conjunto de princípios básicos** em um documento conhecido como [Manifesto Reativo](#). Isto é o que lançou as bases para um estilo de arquitetura para criar Sistemas Reativos. Desde então, este manifesto atraiu muito interesse da comunidade de desenvolvedores.

Basicamente, este documento prescreve **a receita para um sistema reativo ser flexível, fracamente acoplado e escalável**. Isso torna esses sistemas fáceis de desenvolver, tolerantes a falhas e, o mais importante, altamente responsivos, a base para experiências incríveis do usuário.

Então, qual é essa receita secreta? Bem, dificilmente é segredo! O manifesto define as características ou princípios fundamentais de um sistema reativo:

- *Responsivo* : um sistema reativo deve fornecer um tempo de resposta rápido e consistente e, portanto, uma qualidade de serviço consistente.
- *Resiliente* : um sistema reativo deve permanecer responsivo em caso de falhas aleatórias por meio de replicação e isolamento
- *Elástico* : esse sistema deve permanecer responsivo sob cargas de trabalho imprevisíveis por meio de escalabilidade econômica
- *Orientado a mensagens* : deve contar com a passagem de mensagens assíncronas entre os componentes do sistema

Esses princípios parecem simples e sensatos, mas nem sempre são mais fáceis de implementar em uma arquitetura corporativa complexa. Neste tutorial, desenvolveremos um sistema de amostra em Java com esses princípios em mente!

**Contrapressão:**

**A contrapressão em sistemas de software é a capacidade de sobrecarregar a comunicação de tráfego** . Em outras palavras, os emissores de informações sobrecarregam os consumidores com dados que eles não são capazes de processar.

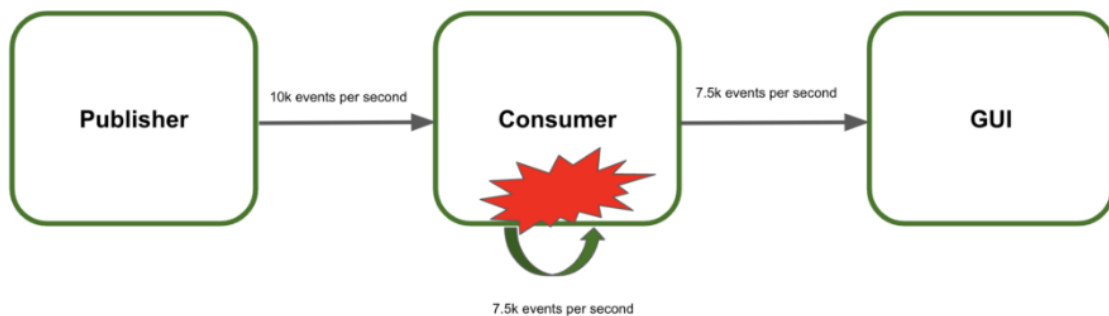
Eventualmente, as pessoas também aplicam esse termo como o mecanismo para controlar e lidar com isso. São as ações de proteção tomadas pelos sistemas para controlar as forças a jusante.

## 2.1. O que é contrapressão?

Em fluxos reativos, a contrapressão também define como regular a transmissão dos elementos do fluxo . Em outras palavras, controle quantos elementos o destinatário pode consumir.

Vamos usar um exemplo para descrever claramente o que é:

- O sistema contém três serviços: o Publicador, o Consumidor e a Interface Gráfica do Usuário (GUI).
- O Publicador envia 10.000 eventos por segundo para o Consumidor
- O consumidor os processa e envia o resultado para a GUI
- A GUI exibe os resultados para os usuários
- O consumidor só pode lidar com 7500 eventos por segundo

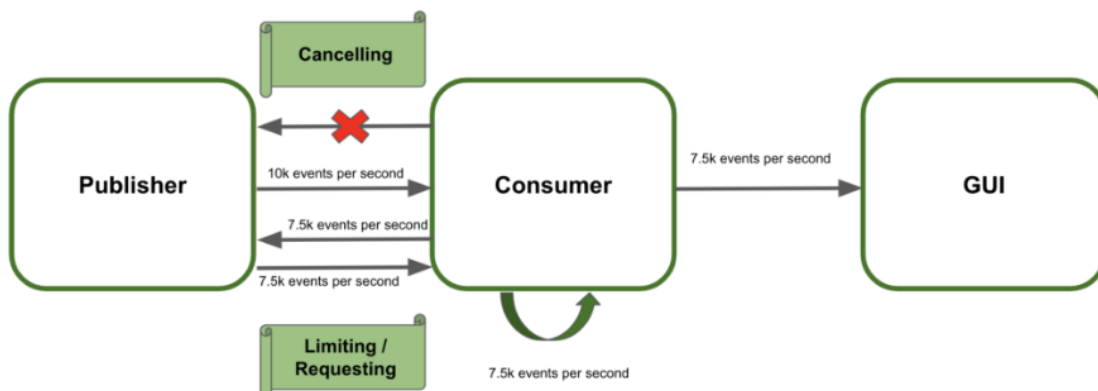


Nessa velocidade, o consumidor não consegue gerenciar os eventos ( contrapressão) . Consequentemente, o sistema entraria em colapso e os usuários não veriam os resultados.

## 2.3. Controlando a contrapressão

Vamos nos concentrar em controlar os eventos emitidos pelo editor. Basicamente, existem três estratégias a seguir:

- **Envie novos eventos somente quando o assinante os solicitar** . Esta é uma estratégia pull para reunir elementos na solicitação do emissor
- **Limitando o número de eventos a serem recebidos no lado do cliente** . Trabalhando como uma estratégia de push limitado, o publisher só pode enviar uma quantidade máxima de itens para o cliente de uma só vez
- **Cancelamento do streaming de dados quando o consumidor não puder processar mais eventos** . Nesse caso, o receptor pode abortar a transmissão a qualquer momento e assinar o fluxo novamente mais tarde.



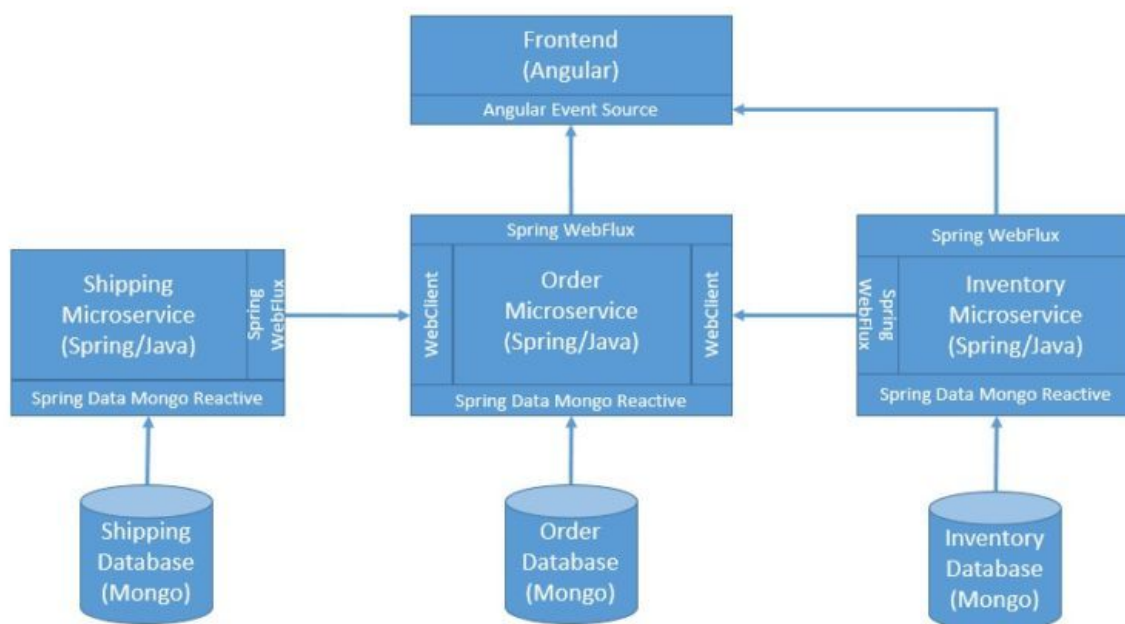
**Programação reativa:**

**A programação reativa é um paradigma de programação em que o foco está no desenvolvimento de componentes assíncronos e sem bloqueio.**

Bloquear chamadas em qualquer programa geralmente **resulta em recursos críticos apenas esperando que as coisas aconteçam**. Isso inclui chamadas de banco de dados, chamadas para serviços da Web e chamadas de sistema de arquivos. Se pudermos liberar os threads de execução dessa espera e fornecer um mecanismo para retornar assim que os resultados estiverem disponíveis, isso resultará em uma utilização de recursos muito melhor.

Isso é o que a adoção do paradigma de programação reativa faz por nós.

Embora seja possível mudar para uma biblioteca reativa para muitas dessas chamadas, pode não ser possível para tudo. Para nós, felizmente, o Spring torna muito mais fácil usar programação reativa com MongoDB e APIs REST:



[Spring Data Mongo](#) tem suporte para acesso reativo por meio do MongoDB Reactive Streams Java Driver. Ele fornece *ReactiveMongoTemplate* e *ReactiveMongoRepository*, ambos com ampla funcionalidade de mapeamento.

O [Spring WebFlux](#) fornece a estrutura da Web de pilha reativa para Spring, permitindo código sem bloqueio e contrapressão de fluxos reativos. Ele aproveita o Reactor como sua biblioteca reativa. Além disso, ele fornece o *WebClient* para executar solicitações HTTP com contrapressão de fluxos reativos. Ele usa o Reactor Netty como a biblioteca cliente HTTP.

