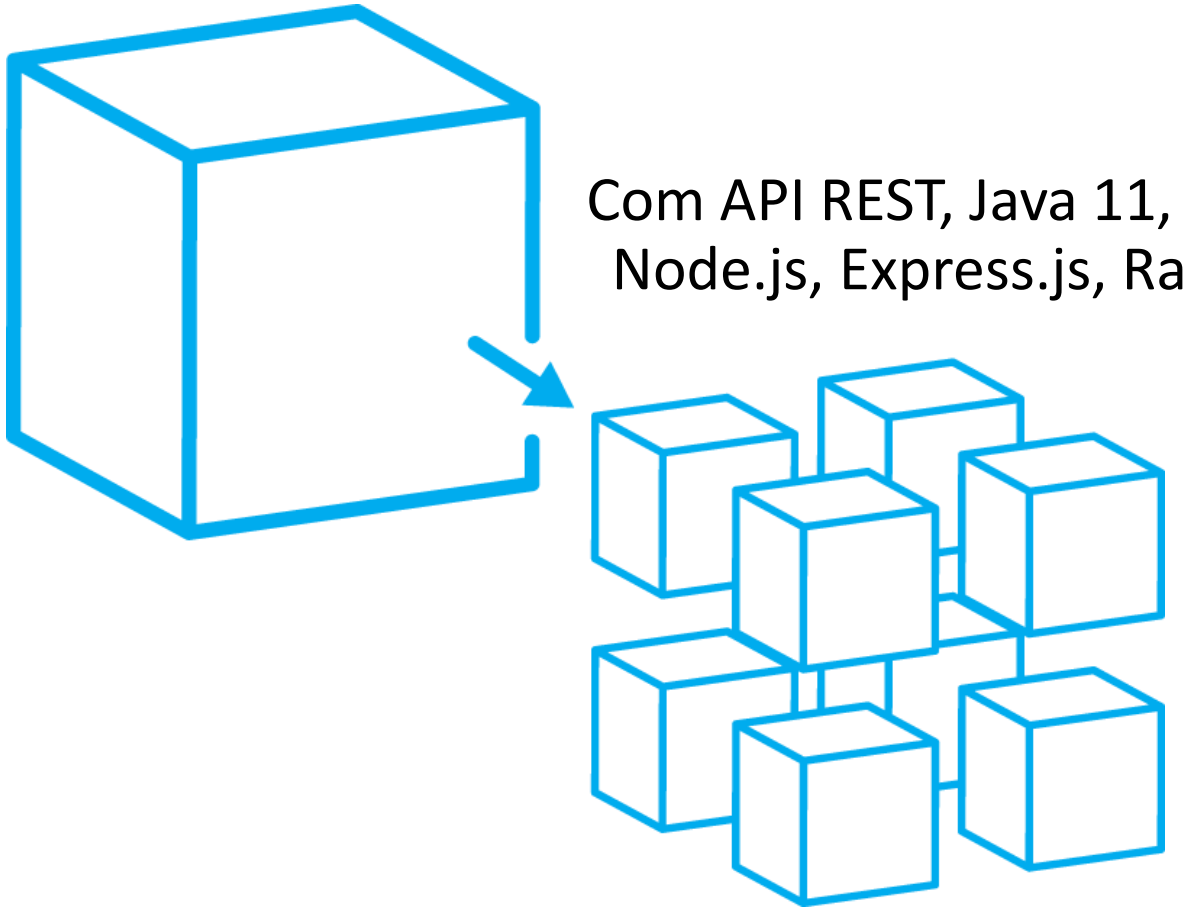
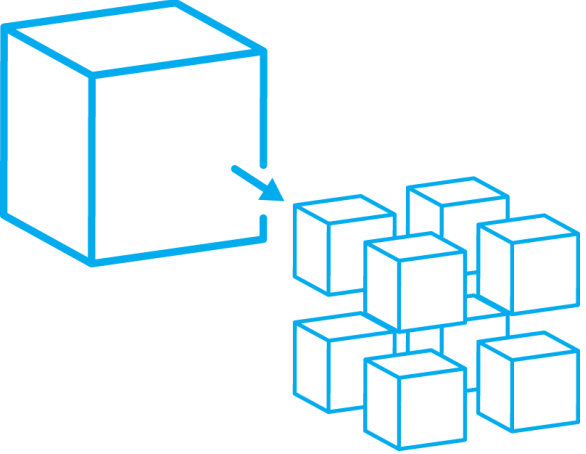
Comunicação entre Microsserviços



Com API REST, Java 11, Spring Boot, Javascript ES6, Node.js, Express.js, RabbitMQ, Docker e Heroku.

Victor Hugo Negrisoli
Desenvolvedor Back-End Sênior



Comunicação síncrona

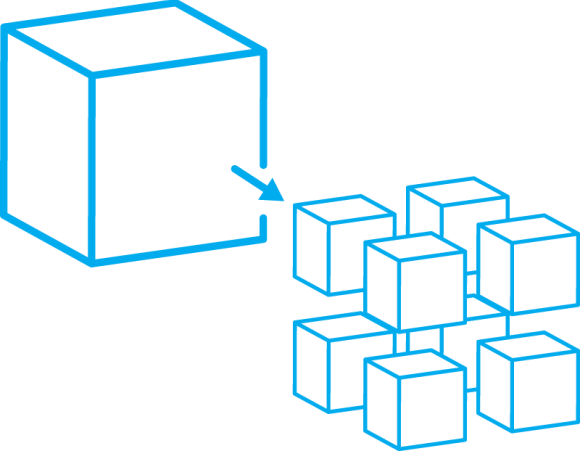
Chamadas a APIs são síncronas, porém, o processo executado pela API pode ser assíncrono. Exemplo:

https://sistema-exemplo.com.br/api/processar-vendas

Você pode programar essa API para esperar para dar a resposta assim que todas as vendas forem processadas, ou você pode devolver uma resposta como “Processo iniciado.”, e deixar o processamento executando em background.

O processamento da API é assíncrono, pode terminar daqui a 10, 15, 20 minutos, etc.

Mas a requisição HTTP não é, você faz a requisição e tem algum tipo de resposta, seja ela 200, 400, 500, etc.



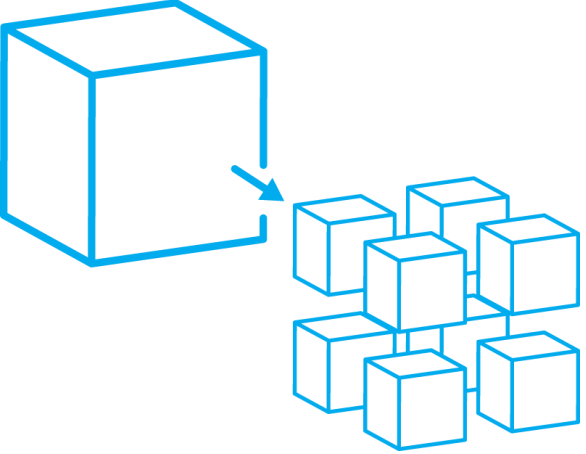
Métodos HTTP

Não há como falar de comunicação síncrona sem falar de métodos HTTP, pois eles serão nossa principal forma de comunicar nossas APIs de maneira síncrona!

Os métodos HTTP descrevem a ação a ser executada para determinado recurso. Eles definem se será um recurso para recuperar uma informação, salvar, atualizar, remover, alterar apenas alguns parâmetros, entre outras operações.

Os métodos HTTP mais conhecidos e utilizados são:

GET, POST, PUT, DELETE, OPTIONS e PATCH



Métodos HTTP

GET – recuperar um recurso

HEAD – recuperar um recurso porém sem o corpo da resposta

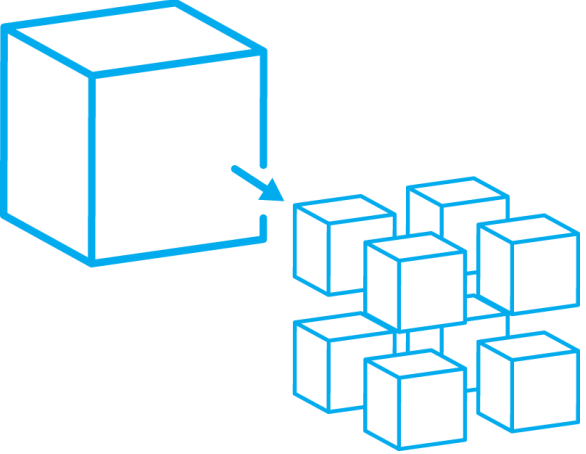
POST – salvar um recurso

PUT – atualizar um recurso completamente

DELETE – remover um recurso

OPTIONS – descrever a comunicação com um recurso de destino

PATCH – atualizar parcialmente um recurso



Métodos HTTP

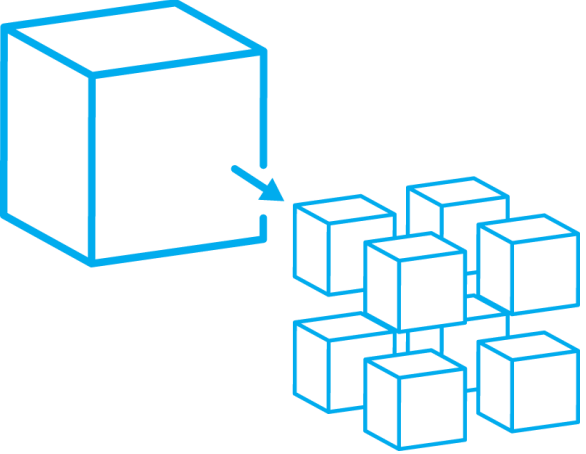
Métodos HTTP também podem ser:

Seguro (Safe) – não altera o estado do servidor, operações apenas de leitura: GET, HEAD, OPTIONS

Idempotentes – métodos que não surtem efeitos diferentes caso as requisições sejam feitas várias vezes de maneira idêntica: GET, HEAD, PUT, DELETE

Obs.: Todo método seguro é idempotente, mas nem todo método idempotente é seguro, exemplo, o PUT e o DELETE.

Cacheable – uma resposta é cacheada quando pode ser armazenada e recuperada posteriormente, evitando uma nova requisição e consumindo recursos do servidor.



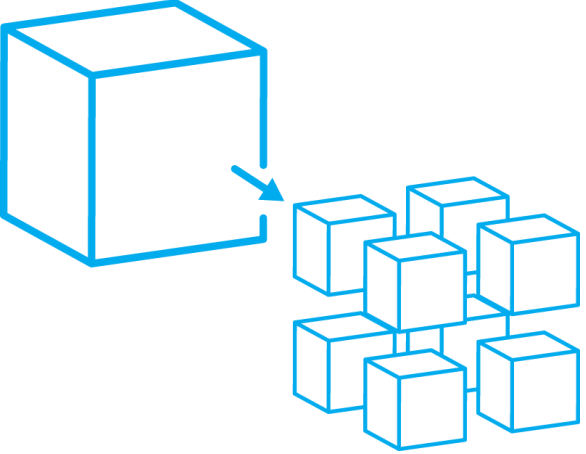
Status HTTP

As requisições HTTP também retornam um código que representa estado da requisição. Os mais utilizados:

Faixa 200 – OK, recurso processado com sucesso.

Faixa 400 – Erro, recurso enviado de maneira incorreta ao servidor, gerando alguma validação.

Faixa 500 – Erro interno, o recurso encontrou algum problema no código do servidor que não conseguiu processar a requisição.



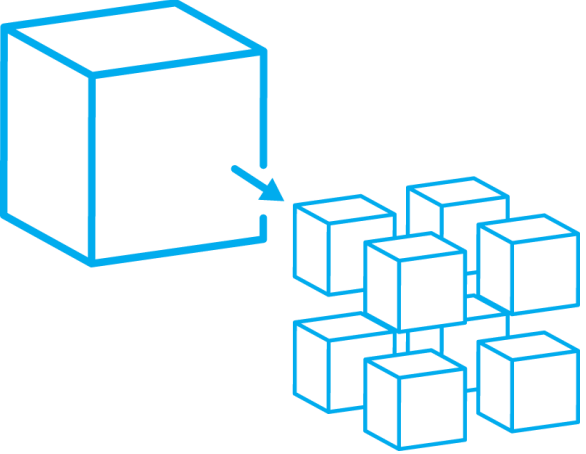
Status HTTP – Faixa 200

Os status que mais utilizamos atualmente:

200 – OK (sucesso)

201 – CREATED (criado)

202 – ACCEPTED (aceito, requisição recebido, porém nenhuma ação foi tomada sobre ela)



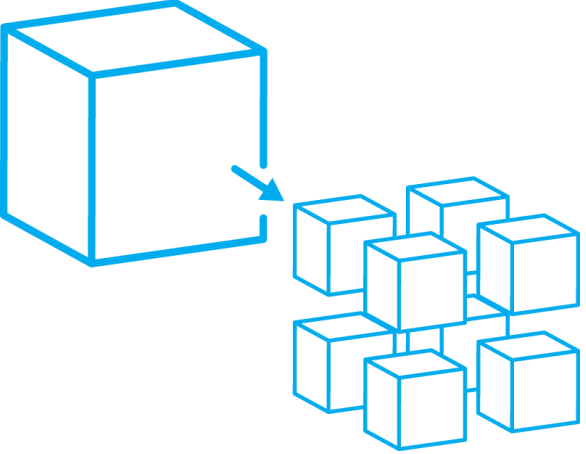
Status HTTP – Faixa 400

400 – BAD REQUEST (requisição inválida, ex: CPF inválido)

401 – UNAUTHORIZED (sem autorização, não estando autenticado)

403 – FORBIDDEN (proibido, possui permissão porém não pode visualizar o recurso)

404 – NOT FOUND (recurso não encontrado no servidor)

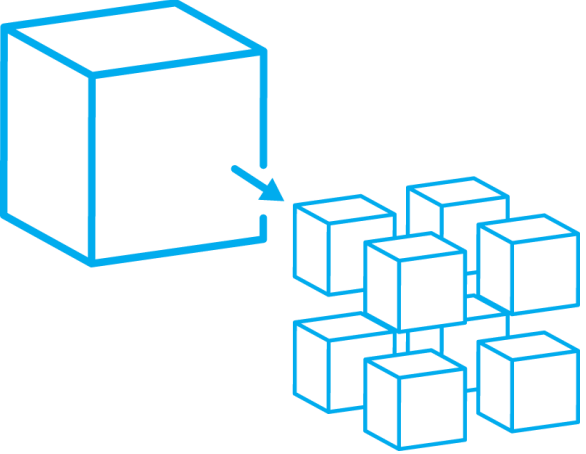


Status HTTP – Faixa 400

405 – METHOD NOT ALLOWED (método não permitido, ex: enviar método POST em uma requisição GET)

415 – UNSUPPORTED MEDIA TYPE (Mídia não suportada, geralmente um dado informado em um formato inválido)

429 – TOO MANY REQUESTS (muitas requisições feitas ao mesmo recurso, ex: site do Enem)



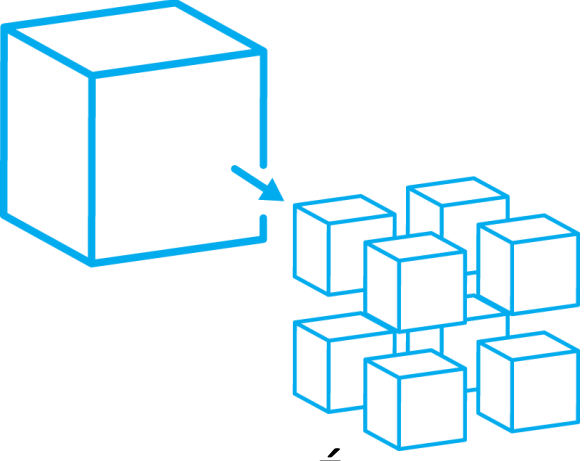
Status HTTP

500 – INTERNAL SERVER ERROR (erro interno no servidor, algo deu errado no back-end)

502 – BAD GATEWAY (servidor intermediário que recebeu uma resposta inválida de outro serviço)

503 – SERVICE UNAVALIABLE (servidor não está pronto para lidar com a requisição, sobrecarregado ou em manutenção)

504 – GATEWAY TIMEOUT (servidor não recebe a resposta de um gateway)

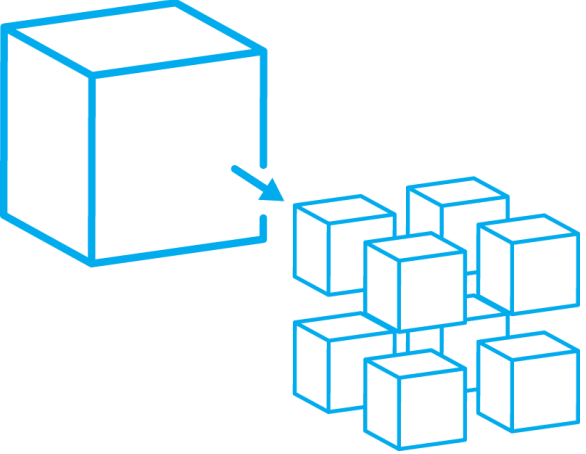


Comunicação Assíncrona

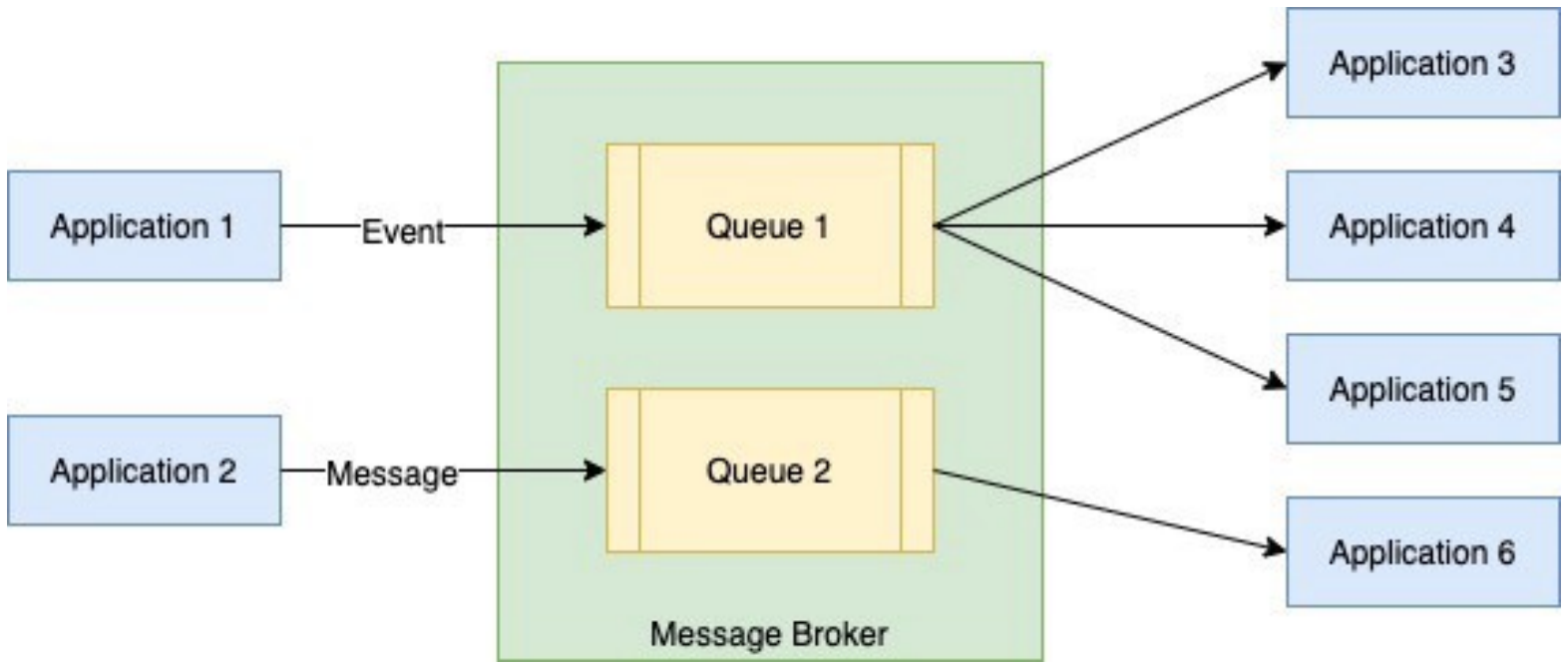
É uma comunicação em que há um agente emissor e um receptor, com a diferença que o receptor não irá receber a mensagem quando o emissor emiti-la, sua recepção será atemporal, ou seja, não se sabe quando irá receber.

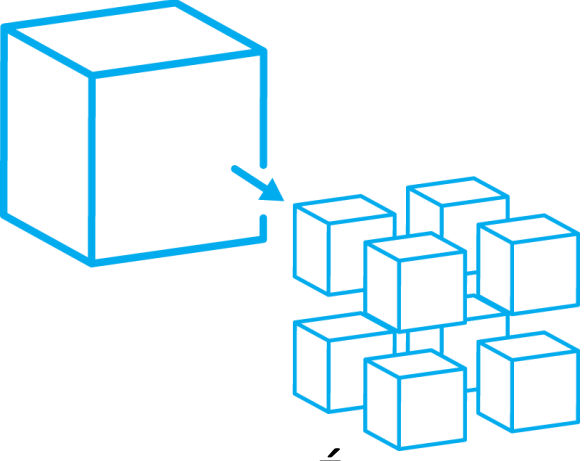
A melhor maneira de exemplificar é com uma fila de mensagens, ao qual um emissor apenas envia uma informação a uma fila, e algum outro agente receptor responsável por apenas escutar as mensagens recebidas dessa fila irá processar a mensagem sequencialmente conforme forem sendo recebidas.

Hoje o protocolo que melhor implementa as filas de mensagens é o protocolo AMQP – Advanced Message Queue Protocol, ou protocolo avançado de enfileiramento de mensagens.



Comunicação Assíncrona



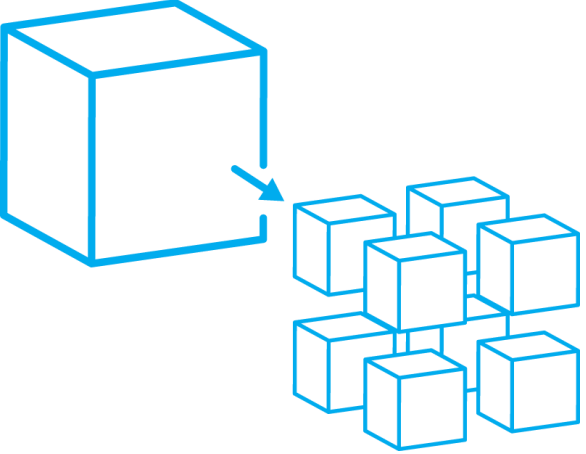


Comunicação Assíncrona

É uma comunicação em que há um agente emissor e um receptor, com a diferença que o receptor não irá receber a mensagem quando o emissor emiti-la, sua recepção será atemporal, ou seja, não se sabe quando irá receber.

A melhor maneira de exemplificar é com uma fila de mensagens, ao qual um emissor apenas envia uma informação a uma fila, e algum outro agente receptor responsável por apenas escutar as mensagens recebidas dessa fila irá processar a mensagem sequencialmente conforme forem sendo recebidas.

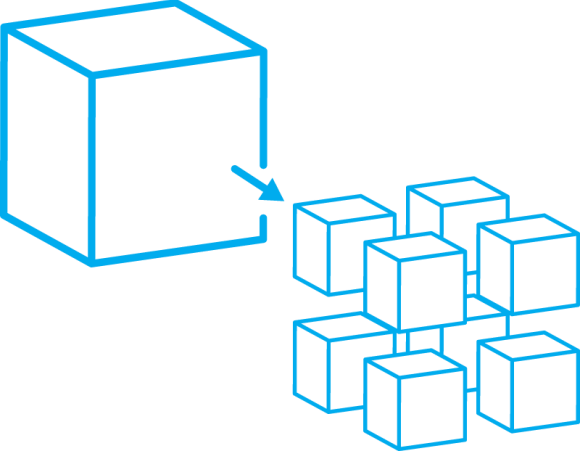
Hoje o protocolo que melhor implementa as filas de mensagens é o protocolo AMQP – Advanced Message Queue Protocol, ou protocolo avançado de enfileiramento de mensagens.



Comunicação Assíncrona

Hoje, os message brokers mais conhecidos e utilizados são:

- Apache Kafka
- RabbitMQ
- Azure Scheduler
- IBM MQ
- Apache ActiveMQ
- AmazonMQ
- Google Cloud Pub/Sub

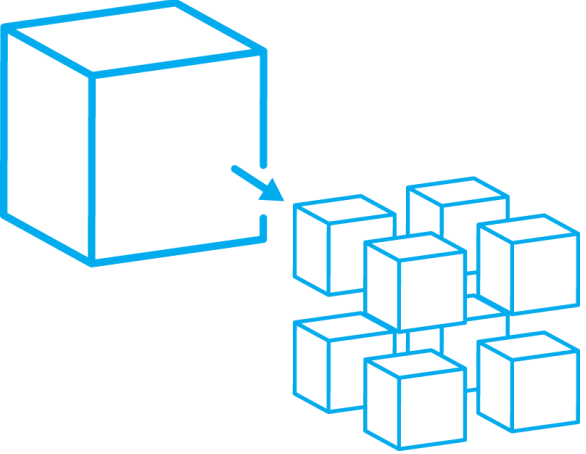


RabbitMQ

Neste curso, utilizaremos o RabbitMQ. É muito fácil de configurar e inicializar uma aplicação com ele.

O RabbitMQ tem alguns conceitos interessantes, como os tipos de Exchange, ou seja, regras de roteamento das mensagens. Nós temos:

- Direct Exchange
- Fanout Exchange
- Topic Exchange (esta que iremos utilizar)
- Headers Exchange

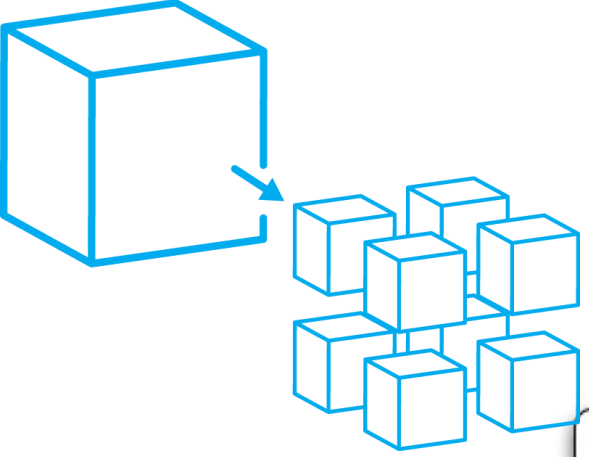


RabbitMQ – Direct Exchange

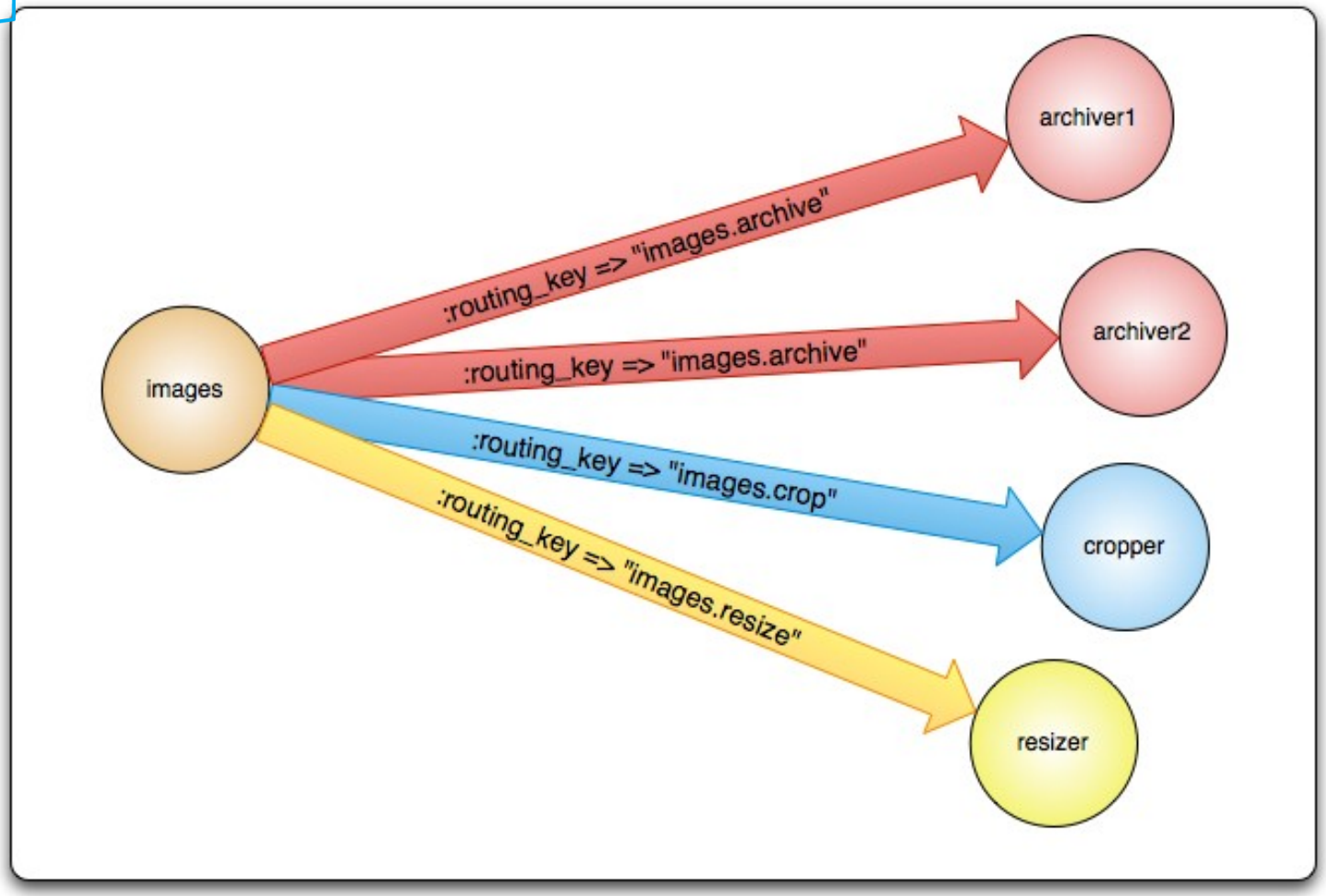
Uma fila é vinculada a uma Direct Exchange baseada em sua routing key.

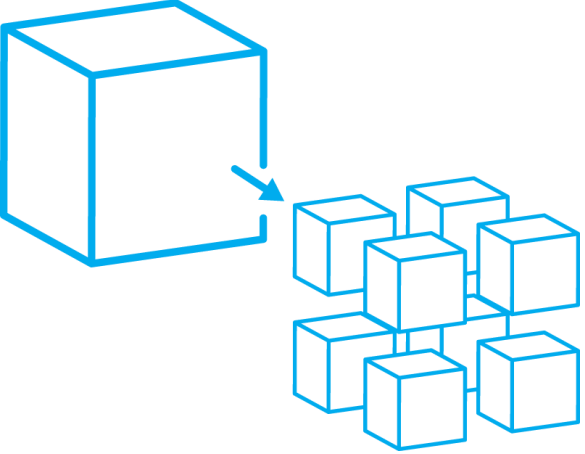
A routing key é um atributo da mensagem utilizada pela Exchange escolhida para decidir para qual rota a mensagem será enviada.

- Uma fila é vinculada à Direct Exchange pela routing key (K).
- Quando uma nova mensagem com uma routing key (R) é enviada a uma Direct Exchange, a Exchange irá rotear apenas se R = K.



RabbitMQ – Direct Exchange

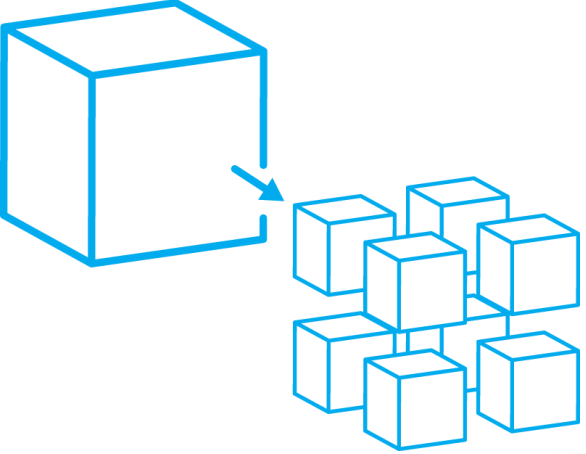




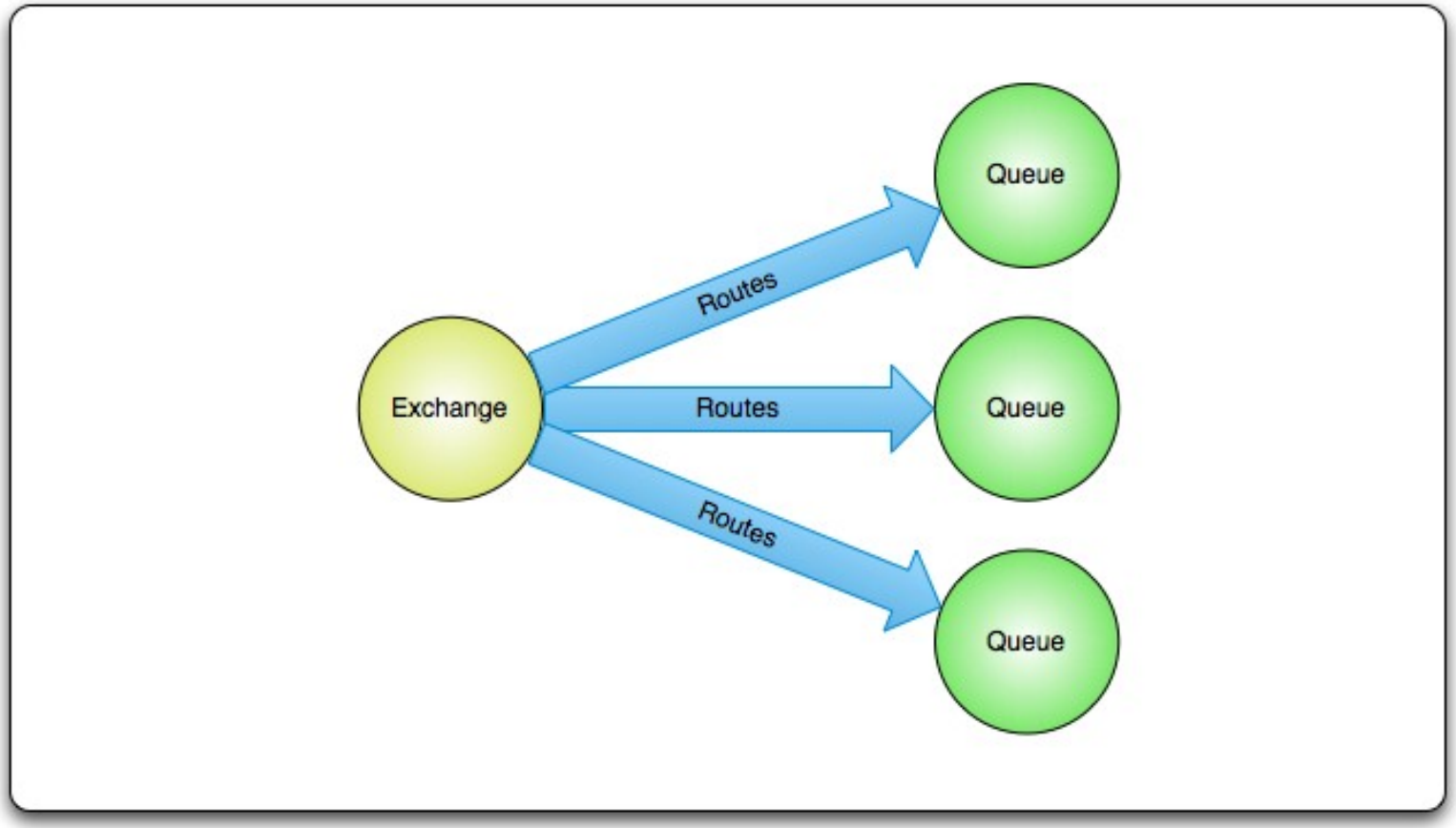
RabbitMQ – Fanout Exchange

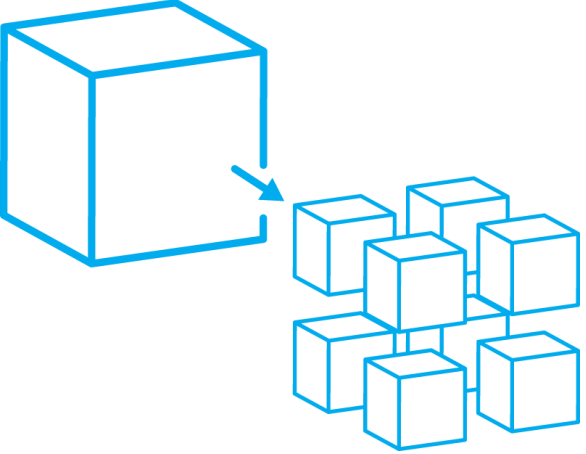
A Fanout Exchange roteia mensagens para todas as filas vinculadas a ela e a routing key é ignorada. Se N filas são vinculadas a uma Fanout Exchange, quando uma nova mensagem é publicada, é feita uma cópia dessa mensagem, que é entregue a todas as N filas vinculadas à Fanout Exchange.

- Jogos MMO utilizam bastante Fanout Exchanges.
- Sites de esportes
- Grupos de chats
- Sistemas distribuídos



RabbitMQ – Fanout Exchange



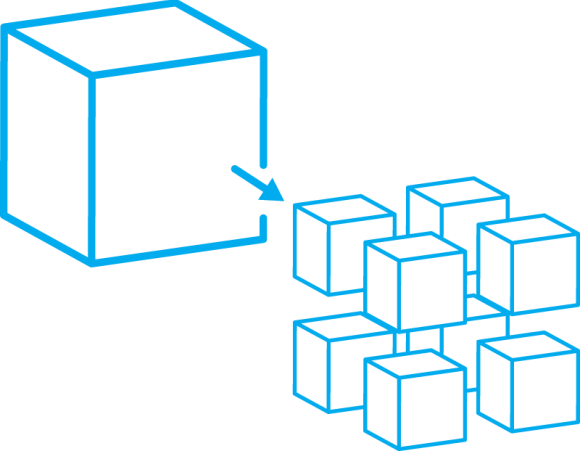


RabbitMQ – Topic Exchange

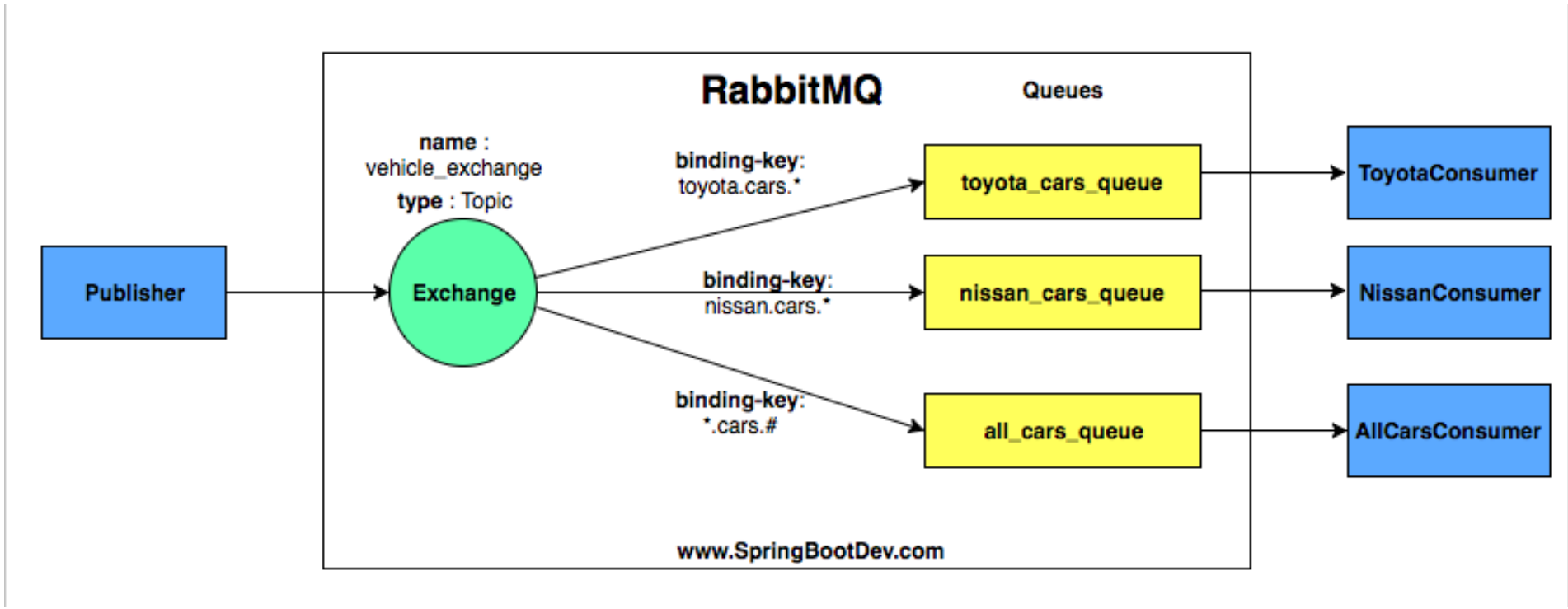
A Topic Exchange roteia mensagens para uma ou mais filas baseadas em uma correspondência entre a routing key e o padrão utilizado para vincular uma fila à Exchange.

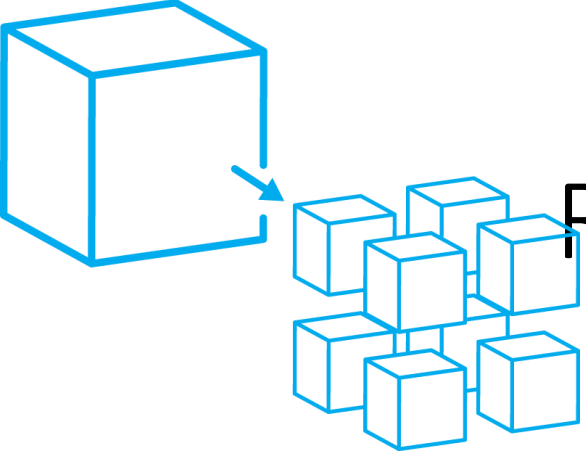
É muito utilizada para implementar vários padrões de publisheres e subscribers.

Topic Exchanges são muito utilizados para roteamento multicast de mensagens.



RabbitMQ – Topic Exchange





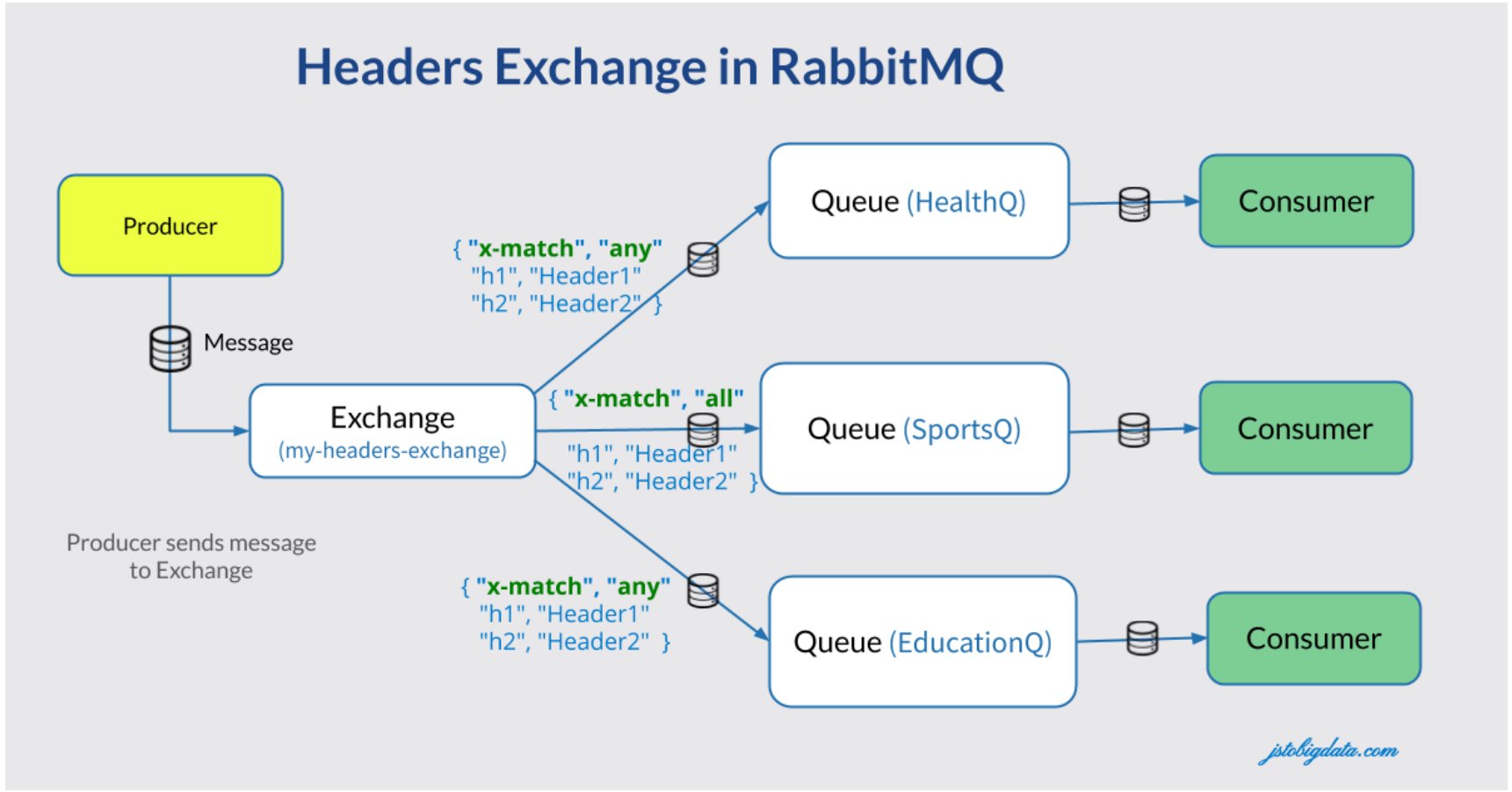
RabbitMQ – Headers Exchange

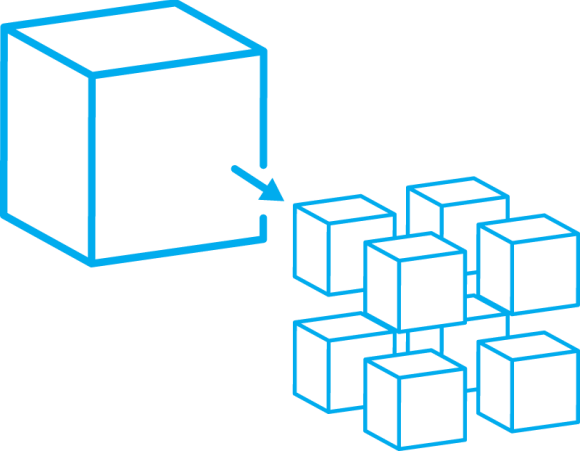
A Header Exchange é projetada para roteamento em vários atributos que são mais facilmente expressos como cabeçalhos (headers) da mensagem ao invés do uso de uma routing key.

A Header Exchange ignora a routing key.

Em vez disso, os atributos usados para roteamento são obtidos do atributo headers. Uma mensagem é considerada correspondente se o valor do header for igual ao valor especificado no vínculo.

RabbitMQ – Headers Exchange

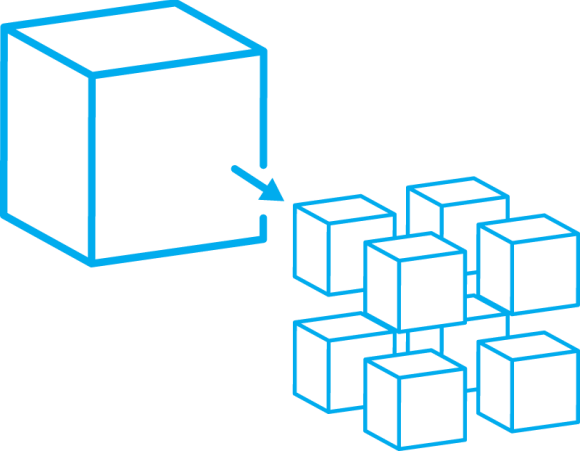




RabbitMQ

E os exchanges possuem alguns atributos:

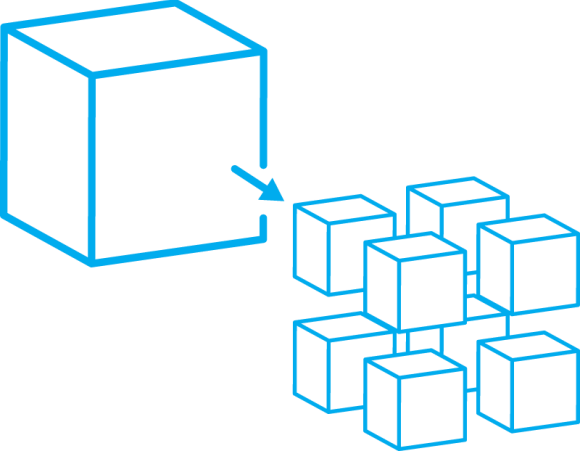
- Nome
- Durabilidade (se irão continuar existindo quando o broker reiniciar)
- Auto-delete (Exchange é deletado quando a sua última fila é desvinculada dele)
- Argumentos (são opcionais, para plugins e features específicas)



RabbitMQ

As filas, assim como os exchanges, possuem atributos, como:

- Nome
- Durabilidade (se irão sobreviver a um restart do broker)
- Exclusivas (usadas apenas para uma conexão, e removidas quando a conexão é fechada)
- Auto-delete (é removida quando seu último consumer se desvincula dela)
- Argumentos (mesma lógica das exchanges)



Referências e links utilizados

- Mozilla Developer – Métodos de requisição HTTP
- RabbitMQ – AMQP Concepts
- Microsoft – Comunicação em uma arquitetura de microsserviço
- DZone - Patterns for Microservices — Sync vs. Async