NestJS e o package microservices

É fundamentalmente, um **subsistema integrado** do **Nest**, que utiliza uma **camada** diferente de **transporte**, que não é HTTP, para **viabilizar** a **comunicação** entre **aplicações** através da **rede**

Seu core são chamados de **TRANSPORTERS:**

São responsáveis por transmitir as mensagens entre diferentes instancias de microservices

A maioria destes transporters suportam nativamente os estilos de mensagem: request-response e event-based

O Nest abstrai os detalhes de implementação de cada transporter atrás de uma interface canônica

NestJS e o package microservices

Temos dois tipos de **Nest Transporters**:

- **Broker-based**: Redis, NATS, RabbitMQ, MQTT e Kafka
- **Point-to-point**: TCP e gRPC

Broker-Based

Nos permitem **desacoplar** vários **componentes** da **aplicação**. Cada **componente** somente precisa se **conectar** ao **broker**, e pode permanecer sem **necessidade** de **conhecer** a **existência**, **localização** ou **detalhes** da **implementação** de **outros** componentes.

A **única** coisa que precisa ser **compartilhada** entre os **componentes** é o **protocolo** de **mensagens**.

Um **broker** se **divide** em:

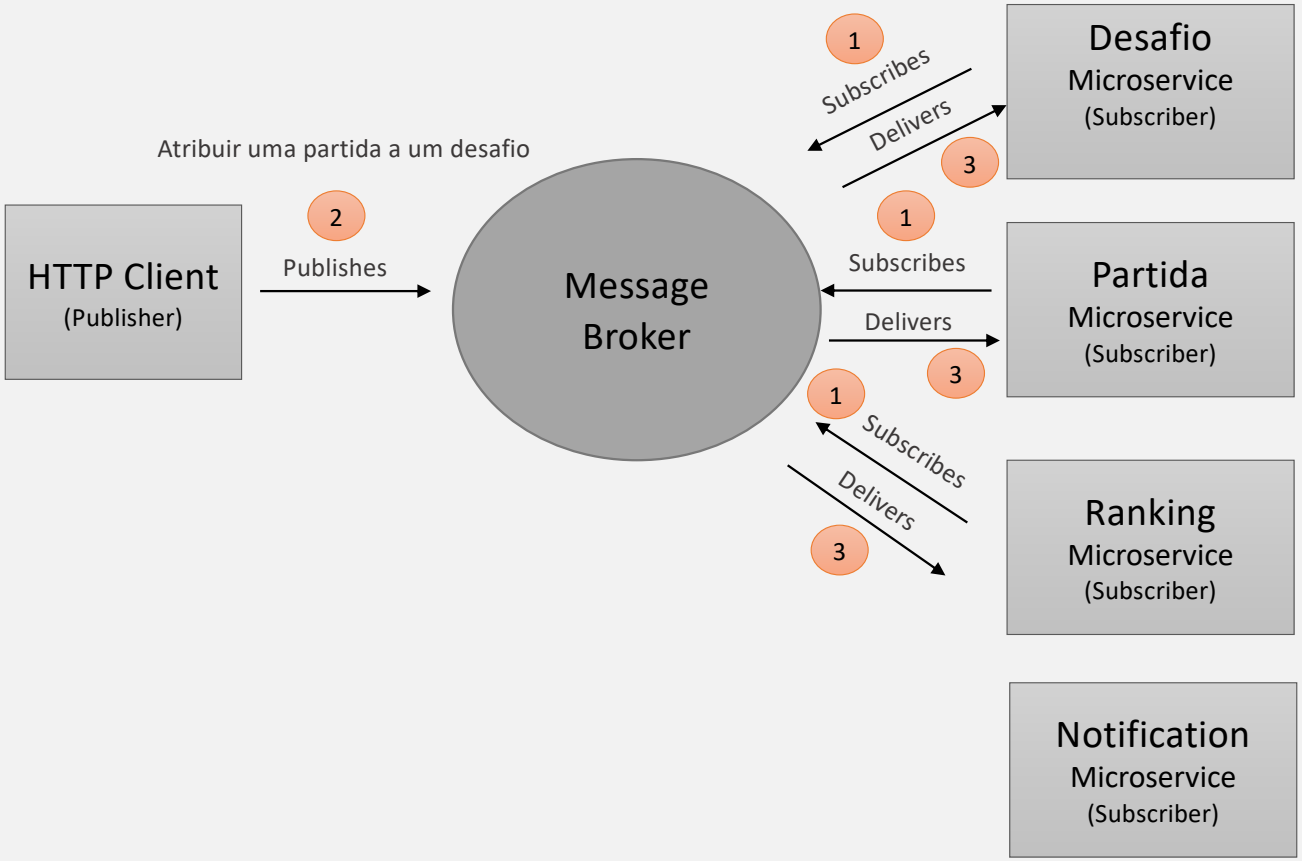
- **Broker Server:** Processo do lado do servidor, responsável por gerenciar a publicação, assinatura e entrega das mensagens aos clientes
- **Broker client API:** É disponibilizado em um package específico para cada linguagem (JavaScript, Java, Go, etc), fornecendo uma API para acessar o broker, a partir de aplicações clientes.

NestJS e o package microservices

Exemplo:

Como iremos atuar em relação ao processamento dos rankings?

Estilo do modelo de comunicação:
PUBLISH/SUBSCRIBE



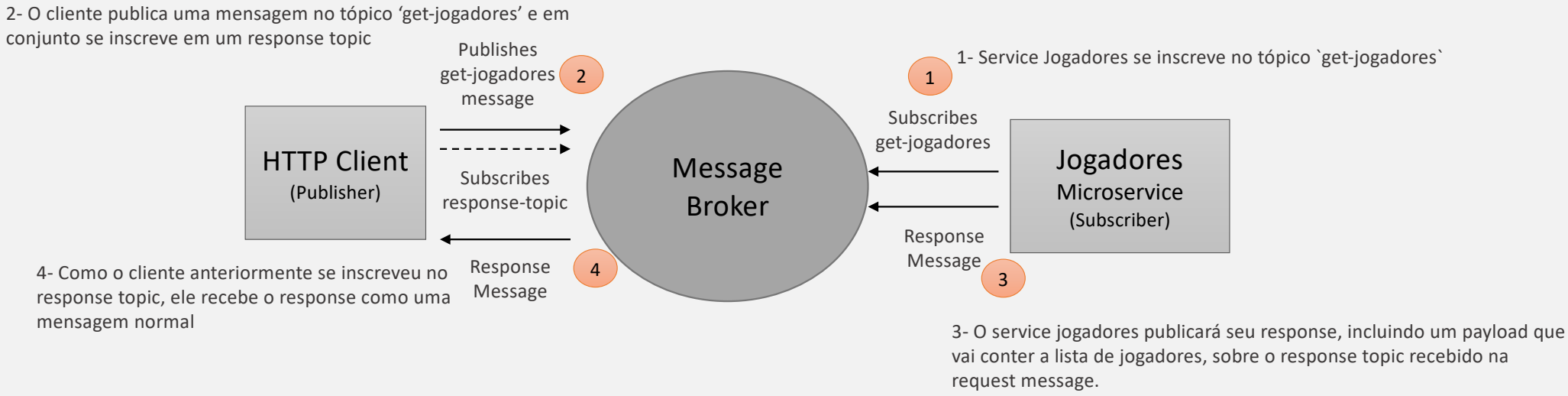
NestJS e o package microservices

Exemplo:

Como iremos atuar em relação ao resource ‘ get-jogadores ’ ?

Estilo do modelo de comunicação:
REQUEST/RESPONSE

Nosso broker resolve este problema de maneira prática, disponibilizando uma capacidade de request/response sobre o modelo de publish/subscribe



NestJS e o package microservices

Estilo do modelo de comunicação:
PUBLISH/SUBSCRIBE

Chamaremos de **event** os tipos de mensagens que podemos trocar com base nesse modelo

Event **emitter** – Componente que publica uma mensagem com um tópico (e um payload opcional). Trata-se de um message publisher

Event **subscriber** – Componente que registra o interesse em um tópico e recebe as mensagens (encaminhadas pelo broker) quando esta mensagem corresponde a um tópico publicado por um emitter.

Estilo do modelo de comunicação:
REQUEST/RESPONSE

Chamaremos de **request/response** os tipos de mensagens que podemos trocar com base nesse modelo

Requestor - Componente que publica uma mensagem que pretende ser tratada como uma request e também executa as etapas descritas anteriormente, ou seja, se inscreve em um response topic e inclui este response topic na mensagem publicada.

Responder - Componente que se inscreve em um topic, que pretende tratar como uma incoming request, produz um resultado e publica um response, incluindo o payload recuperado, para o response topic que recebeu na inbound request