ROTEIRO - MIGRAR A ENTIDADE JOGADORES PARA O CONTEXTO BASEADO EM MICROSERVICES

Neste roteiro apresento as atividades necessárias para que possa migrar a entidade jogadores para o contexto baseado em microservices, onde são contempladas adequações tanto no API Gateway quanto no microservice admin-backend.
A intenção é que este material sirva como um guia, de modo que você possa conduzir essa atividade, antes de visualizar o vídeo em que apresento o resultado.

1- Começando pelo **API Gateway**
   a. Modularize sua aplicação
      i. [Refactoring] **Crie** o **módulo categorias** (Module e Controller). Migre a implementação da aula anterior para este módulo. Lembre-se de separar os Dtos.
      ii. **Crie** o **módulo jogadores** (Module e Controller). Neste momento crie apenas a estrutura padrão gerada pelo Nest
      iii. [Refactoring] Separe em um diretório comum os filters, interceptors e pipes
   b. [Refactoring] **Crie** um **provider** para **acomodar** o **ClientProxy**, isolando este componente em um módulo específico, de modo que você venha reaproveitar esta implementação nos módulos categorias e jogadores.
      i. Lembre-se de anotar sua classe com o decorator @Injectable()
      ii. Lembre-se de contar para seu novo módulo que sua classe é um provider
      iii. Lembre-se de exportar seu provider
      iv. Lembre-se de importar este módulo quando precisar utiliza-lo
   c. Agora realize o processo de **migração** da **entidade jogadores**:
      i. Realize a **migração** dos **Dtos**
         1. **atualizar-jogador.dto.ts**
         2. **criar-jogador.dto.ts**
            [Refactoring] Nosso modelo precisa de uma adequação. Precisamos exigir do cliente o ID da categoria na qual o jogador pertence.
      ii. Na **classe JogadoresController**, realize a **migração** dos **métodos**:
         1. Método **criarJogador**
            Encaminhe o jogador para o message broker, considerando o uso de um event emitter.
            [Desafio] Antes de encaminhar o jogador para o message broker, será necessário validar se a categoria informada pelo cliente realmente é válida. Caso não seja, será necessário lançar uma BadRequestException no API Gateway, retornando para o cliente a exceção.
         2. Método **consultarJogadores**
            Atenção ao uso do padrão requestor/responder
         3. Método **atualizarJogador**
            Encaminhe o Dto para o message broker, considerando o uso de um event emitter.
            [Desafio] Antes de encaminhar os dados de atualização do jogador para o message broker, será necessário validar se a categoria informada pelo cliente realmente é válida. Caso não seja, será necessário lançar uma BadRequestException no API Gateway, retornando para o cliente a exceção.
         4. Método **deletarJogador**

2- Agora vamos para nosso **microservice admin-backend**
   a. Modularize sua aplicação
      i. [Refactoring] **Crie** o **módulo categorias** (Controller, Module e Service). Migre a implementação da aula anterior para este módulo. Lembre-se de separar as interfaces.
      ii. Crie o módulo jogadores. Neste momento crie apenas a estrutura padrão gerada pelo Nest
   b. **Migrando** os métodos da **classe JogadoresService**
      i. Método **criarJogador**
      ii. Método **consultarTodosJogadores**
      iii. Método **consultarJogadorPeloId**
      iv. Método **atualizarJogador**
      v. Método **deletarJogador**

      Observação: Lembre-se de envolver seus métodos com o bloco try/catch, lançando uma RpcException, em caso de erro.
   c. **Migrando** a classe **JogadoresModule**

      Lembre-se de importar o MongooseModule, registrando o JogadorSchema
   d. Migrando a classe **JogadoresController**
      i. Método **criarJogador**

         Utilize um event subscriber para recuperar as mensagens que estão no RabbitMQ
      ii. Método **consultarJogadores**

         Utilize um responder para retornar um payload para o cliente
      iii. Método **atualizarJogador**

         Recupere o id do jogador e o Dto e realize a atualização do jogador
      iv. Método **deletarJogador**

      Observações:
      - Não se esqueça de envolver seus métodos com o bloco try/catch/finally
      - Não se esqueça de aplicar o acknowledge nas mensagens.