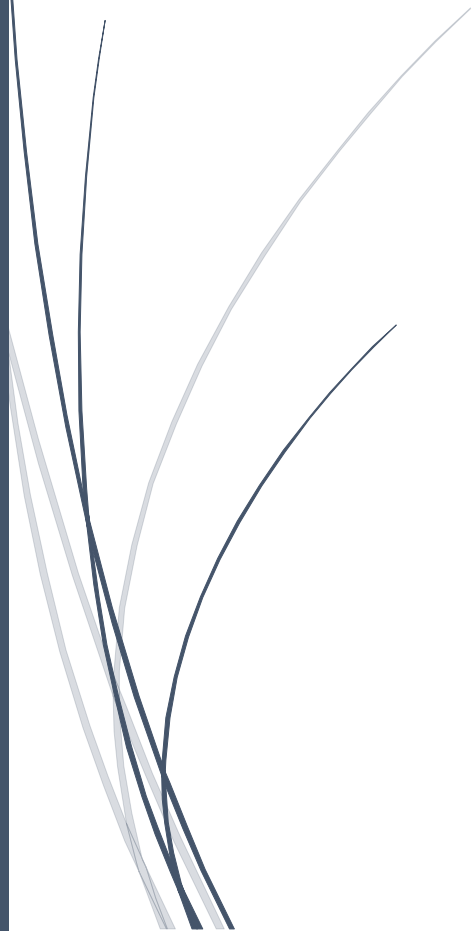
Engenharia de Software

11/10/2025

TP2

Domain-Driven Design (DDD) e Arquitetura de Softwares Escaláveis com Java

Professor(a): Armênio Torres Santiago Cardoso



Samuel Hermany
INSTITUTO INFNET

LINK GITHUB

https://github.com/faculdade-infnet/V-2-DDD_e_Arquitetura-de-Softwares-Escalaveis-com-Java/tree/main/TP2

1. EXPLIQUE DE FORMA SUCINTA QUAL A RAZÃO DE CRIAR AGREGADOS.

Para agrupar entidades e objetos de valor que precisam mudar juntos, garantindo a consistência das regras de negócio dentro de um limite transacional. O agregado possui uma entidade raiz que controla o acesso e mantém as invariantes do domínio, tornando o modelo mais coeso, consistente e fácil de manter.

2. O QUE SIGNIFICA "CONSISTÊNCIA TRANSACIONAL"?

Significa que todas as operações realizadas dentro de uma mesma transação devem ser concluídas com sucesso juntas ou todas devem ser desfeitas, garantindo que o sistema nunca fique em um estado inconsistente.

3. CITE AS 4 PROPRIEDADES CRUCIAIS QUE DEFINEM TRANSAÇÕES.

- **Atomicidade** garante que todas as operações dentro de uma transação sejam executadas por completo. Se uma falha ocorrer em qualquer etapa, todas as alterações são revertidas (rollback) , mantendo o sistema no estado anterior.
  Ou tudo acontece, ou nada acontece.

- **Consistência**, assegura que, após a transação, o banco de dados permaneça em um estado válido.
  Nenhuma regra de negócio ou integridade é violada após a transação.

- **Isolamento** mantém o efeito de uma transação invisível para outras pessoas até que ela seja confirmada, para evitar confusão.
  Evita leituras sujas e resultados incorretos durante execuções simultâneas.

- **Durabilidade** garante que as alterações de dados se tornem permanentes assim que a transação for confirmada.
  O que foi gravado com sucesso, permanece salvo.

4. O QUE SÃO "INVARIANTES DE NEGÓCIO"?

São regras de domínio ou condições que devem estar sempre verdadeiras dentro de um domínio, independentemente das operações realizadas. Exemplo:

- Em um pedido de compra, o valor total deve ser igual à soma dos itens.
- Uma conta bancária nunca pode ter saldo negativo, se a regra do domínio não permitir.

5. Porque um agregado só deve ter acesso a outro agregado pelo ID?

- **Isolamento**: cada agregado é uma unidade independente de consistência e transação. Se um agregado pudesse manipular diretamente outro, isso quebraria esse isolamento.
- **Baixo acoplamento**: acessar apenas o ID evita que um agregado dependa internamente da estrutura ou regras de outro, facilitando a manutenção e evolução do domínio.
- **Integridade transacional**: garante que cada transação afete apenas um agregado, evitando inconsistências em atualizações simultâneas.
- **Performance e escalabilidade**: acessar por ID evita carregamentos desnecessários de objetos grandes na memória, melhorando o desempenho.

6. Crie um trecho de código Java de uma entidade que representa um agregado e faça referência a outro agregado dentro do escopo do projeto Pet Friends.

Dentro da pasta do TP2 do github tem a pasta exercicio06_e_07 conforme link abaixo.

https://github.com/faculdade-infnet/V-2-DDD_e_Arquitetura-de-Softwares-Escalaveis-com-Java/tree/main/TP2/exercicio06_e_07

7. Elabore um trecho de código Java que mostre um método de negócio com a previsão de publicação de um evento de domínio dentro do escopo do projeto Pet Friends.

Dentro da pasta do TP2 do github tem a pasta exercicio06_e_07 conforme link abaixo.

https://github.com/faculdade-infnet/V-2-DDD_e_Arquitetura-de-Softwares-Escalaveis-com-Java/tree/main/TP2/exercicio06_e_07

8. O que é “evento de domínio”?

- São eventos significativos que ocorrem dentro do contexto do domínio e que podem ser capturados e processados pelo sistema.
- Em outras palavras, é uma mensagem que representa uma mudança de estado importante, geralmente disparada por um agregado após executar uma ação de negócio, tendo como principais características os itens abaixo:
    - Representa algo que já ocorreu *(ex.: PedidoCriado, PagamentoAprovado, EstoqueEsgotado)*
    - É imutável: Descreve um fato, não algo que ainda vai acontecer.
    - É usado para notificar outros componentes ou iniciar novos processos sem criar acoplamento direto.
    - Facilita a integração entre contextos delimitados (bounded contexts) e a comunicação assíncrona.

Exemplo prático:
Quando um pedido é confirmado, o agregado **“Pedido”** pode gerar um evento **“PedidoConfirmado”**, que será tratado por outro serviço responsável por enviar a nota fiscal ou atualizar o estoque.

9. Crie um trecho de código Java que mostre uma abstração de um objeto do tipo “evento de domínio”.

Dentro da pasta do TP2 do github tem a pasta exercicio09_e_10 conforme link abaixo.

https://github.com/faculdade-infnet/V-2-DDD_e_Arquitetura-de-Softwares-Escalaveis-com-Java/tree/main/TP2/exercicio09_e_10

10. Crie um trecho de código Java que mostre a implementação de um “evento de domínio” dentro do escopo do projeto Pet Friends.

Dentro da pasta do TP2 do github tem a pasta exercicio09_e_10 conforme link abaixo.

https://github.com/faculdade-infnet/V-2-DDD_e_Arquitetura-de-Softwares-Escalaveis-com-Java/tree/main/TP2/exercicio09_e_10

11. Qual é a diferença entre filas e tópicos e como estes elementos funcionam em conjunto?

A. Fila

- FIFO (First In, First Out), 1 remetente → 1 receptor.
- Ponto a ponto, alto acoplamento, apenas um consumidor recebe a mensagem.
- Cada mensagem é consumida por apenas um consumidor que deve-se associar a mensagem quem vai receber ela.
- **Exemplo**: uma tarefa enviada para processamento por um único serviço.

Ideal para balancear carga ou distribuir trabalho entre consumidores.

B. Tópico

- Similar a ondas de rádio: 1 remetente → vários receptores.
- Cada grupo de consumidores cria sua própria fila, permitindo múltiplas cópias da mesma mensagem.
- Cada mensagem é enviada a todos os assinantes interessados.
- **Exemplo**: um evento “PedidoConfirmado” notifica vários serviços — faturamento, estoque, entrega etc.

Ideal para disseminar eventos e manter sistemas sincronizados.

C. Funcionamento

- O tópico é o ponto central onde as mensagens são publicadas pelos produtores.
- Cada assinatura (subscription) criada para esse tópico atua como uma fila independente, garantindo que cada assinante receba sua cópia das mensagens.

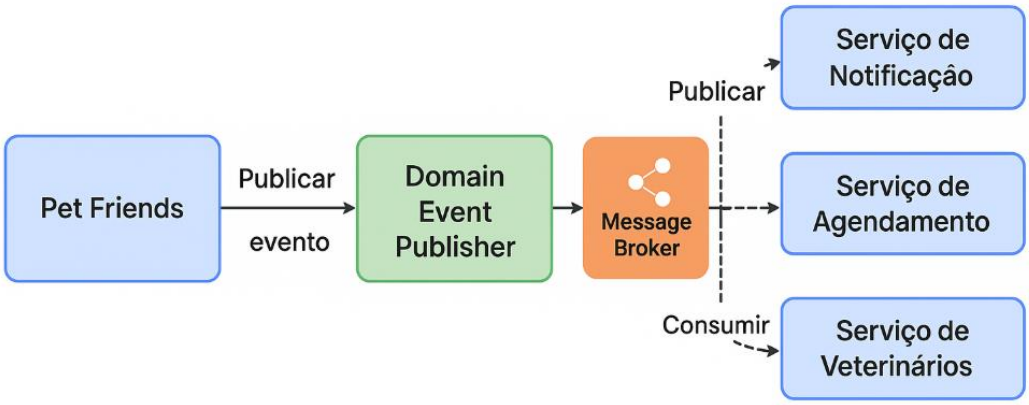
Em resumo:

- **O tópico distribui as mensagens.**
- **E cada fila (assinatura) entrega as mensagens a seus consumidores, de forma isolada e confiável.**

12. Dê um exemplo de solução de arquitetura para publicação de eventos de domínio dentro do escopo do projeto Pet Friends (ideal um desenho).

Dentro da pasta do projeto no github tem um arquivo **petfriends-event.puml**, que contém o **Desenho completo**.

- **Desenho simplificado**



- **Desenho Completo**