

Engenharia de Softwares Escaláveis

Design Patterns e Domain-Driven Design com Java

Agenda

Etapas 8: Projetando Aggregates.

- Modelagem de Agregados.



Modelagem de Agregados

Modelagem de Agregados

Existem algumas armadilhas esperando por você enquanto trabalha em seu modelo de domínio, implementando seus agregados.

Uma armadilha grande e desagradável é o **Modelo de Domínio Anêmico**.

É aqui que você está usando um modelo de domínio orientado a objetos e todos os seus agregados têm apenas acessadores públicos (getters e setters), mas nenhum comportamento do negócio real.

Isso tende a acontecer quando há um foco técnico em vez do foco no negócio durante a modelagem.

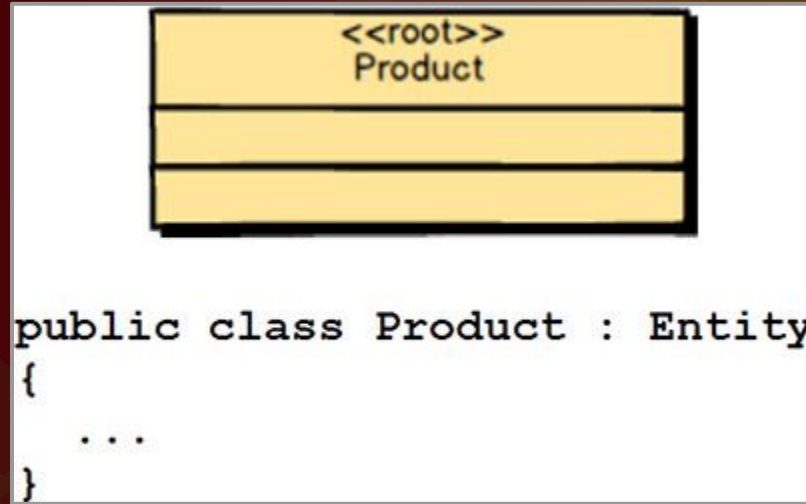
Modelagem de Agregados

Fique atento também ao vazamento de lógica de negócios nos serviços para além do seu modelo de domínio.

Pode acontecer sem ser detectado, assim como a anemia física.

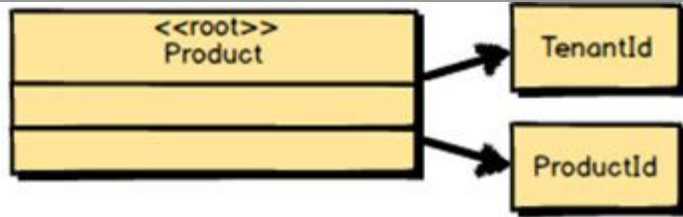
Delegar lógica de negócios de serviços para classes auxiliares/utilitárias também não vai funcionar bem.

Coloque sua lógica de negócios em seu modelo de domínio ou sofra bugs patrocinados por um **Modelo de Domínio Anêmico**.



A primeira coisa que você deve fazer é criar uma classe para sua entidade raiz agregada.

Aqui está uma representação UML (Unified Modeling Language) de Product.



```
public class Product : Entity
{
    private ProductId productId;
    private TenantId tenantId;
}
```

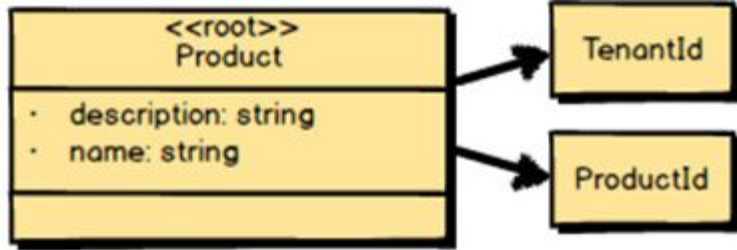
Cada entidade raiz agregada deve ter uma **identidade** globalmente exclusiva.

A classe **Product**, no contexto de gerenciamento ágil de projetos, tem duas formas de identidade globalmente exclusivas.

TenantId é escopo da entidade raiz dentro de uma determinada organização assinante.

A segunda identidade, que também é globalmente única, é o **ProductId**.

Esta segunda identidade de **Product** diferencia as instâncias de todas as outras dentro do mesmo locatário.

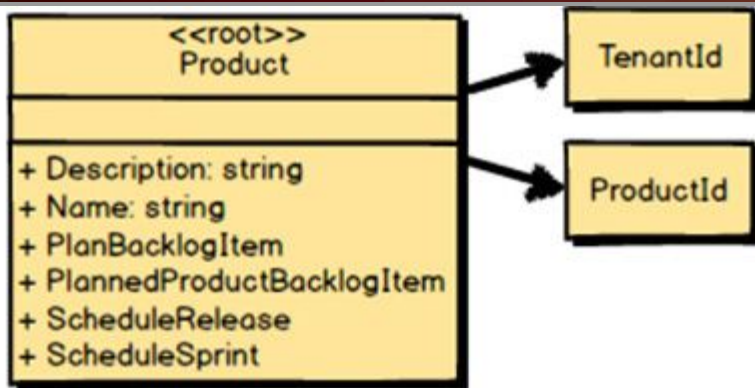


```
public class Product : Entity
{
    private string description;
    private string name;
    private ProductId productId;
    private TenantId tenantId;
}
```

Em seguida, você captura quaisquer atributos ou campos intrínsecos necessários para localizar o agregado.

No caso de **Product**, existem ambos **description** e **name**.

Os usuários podem pesquisar um ou ambos para encontrar cada um deles **Product**.



```
public class Product : Entity
{
    ...
    public void PlannedProductBacklogItem(...)
    {
        ...
    }
}
```

Finalmente, você adiciona qualquer comportamento complexo.

Aqui temos quatro novos métodos:

PlanBacklogItem()

PlannedProductBacklogItem()

ScheduleRelease()

ScheduleSprint()

Abstrações

Um modelo de software eficaz é sempre baseado em um conjunto de abstrações que abordam a maneira como o negócio faz as coisas.

Há, no entanto, a necessidade de escolher o nível de abstração apropriado para cada conceito que está sendo modelado.

Se você seguir a direção da sua **Linguagem Ubíqua**, geralmente criará as abstrações adequadas.

Por exemplo, no Contexto Ágil de Gerenciamento de Projetos estamos lidando com Scrum.

Faz sentido modelar os conceitos de **Product**, **BacklogItem**, **Release** e **Sprint** que estamos discutindo.

Mesmo assim, e se os desenvolvedores de software estivessem menos preocupados em modelar a **Linguagem Ubíqua** do Scrum e mais interessado em modelar uma solução para todos os conceitos atuais e futuros do Scrum?

Modelagem de Agregados

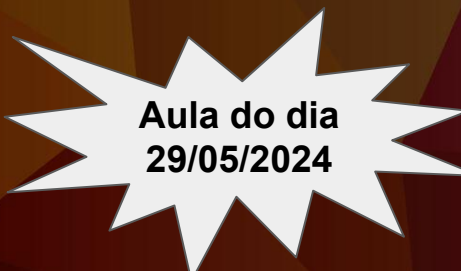
Modele a **Linguagem Ubíqua** explicitamente de acordo com o modelo mental dos **Especialistas do Domínio** que é refinado por sua equipe.

Ao **modelar o que a empresa precisa hoje**, você economizará uma quantidade considerável de tempo, orçamento, código e constrangimento.

Mais ainda, você prestará um ótimo serviço ao negócio ao modelar um **Contexto Limitado** preciso e útil que reflita um design eficaz.

Regras para Modelagem de Agregados

1. Proteja invariantes de negócios dentro dos limites agregados.
2. Projete pequenos agregados.
3. Faça referência a outros agregados apenas por identidade.
4. Atualize outros agregados usando consistência eventual.



**Aula do dia
29/05/2024**

Modelagem de Agregados

Concentre-se primeiro na segunda regra do design: “Projete pequenos **agregados**”.

Comece criando cada **agregado** com apenas uma **entidade**, que servirá como raiz do **agregado**.

Preencha cada uma das **entidades** com os campos/atributos/propriedades que você acredita estarem mais intimamente associados à única **entidade raiz**.

Uma grande dica aqui é definir cada campo/atributo/propriedade necessário para identificar e encontrar o **agregado**, bem como quaisquer campos/atributos/propriedades intrínsecos adicionais que são necessários para que o **agregado** seja construído e deixado em um estado inicial válido .

Modelagem de Agregados

Agora concentre-se na primeira regra do design: “Proteger invariantes de negócios dentro dos limites **agregados**”.

Você já afirmou na etapa anterior que, no mínimo, todos os campos/atributos intrínsecos devem estar atualizados quando o **agregado** de entidade único for persistido.

Mas agora você precisa examinar cada um dos seus **agregados**, um de cada vez. Ao fazer isso para o **Agregado A1**, pergunte aos **especialistas em domínio** se algum outro **agregado** que você definiu deve ser atualizado em reação às alterações feitas no **Agregado A1**.

Faça uma lista de cada um dos **agregados** e suas regras de consistência, que indicará os prazos para todas as atualizações baseadas em reações.

Em outras palavras, **Agregado A1** seria o cabeçalho de uma lista, e outros tipos de **agregados** seriam listados em **A1** se fossem atualizados em reação às atualizações de **A1**.

Modelagem de Agregados

Agora pergunte aos **especialistas do domínio** quanto tempo pode decorrer até que cada uma das atualizações baseadas em reações ocorra.

Isso levará a dois tipos de especificações: (a) imediatamente e (b) dentro de N segundos/minutos/horas/dias.

Uma forma possível de encontrar o limite de negócios correto é apresentar um prazo exagerado (como semanas ou meses) que é obviamente inaceitável.

Isso provavelmente fará com que os **especialistas em negócios** respondam em um prazo aceitável.

Modelagem de Agregados

Para cada um dos prazos imediatos, você deve considerar fortemente a composição dessas duas entidades dentro do mesmo limite agregado.

Isso significa, por exemplo, que o **Agregado A1** e o **Agregado A2** serão na verdade compostos em um novo **Agregado A**.

Agora os agregados **A1** e **A2** como foram definidos anteriormente não existirão mais. Existe apenas o **Agregado A**.

Modelagem de Agregados

Para cada um dos agregados reagentes que podem ser atualizados após um determinado tempo decorrido, você os atualizará usando a quarta regra do design: “Atualizar outros agregados usando consistência eventual”.

