

Arquitetura de Software

terça-feira, 25 de janeiro de 2022 09:41

Estilo Arquitetural

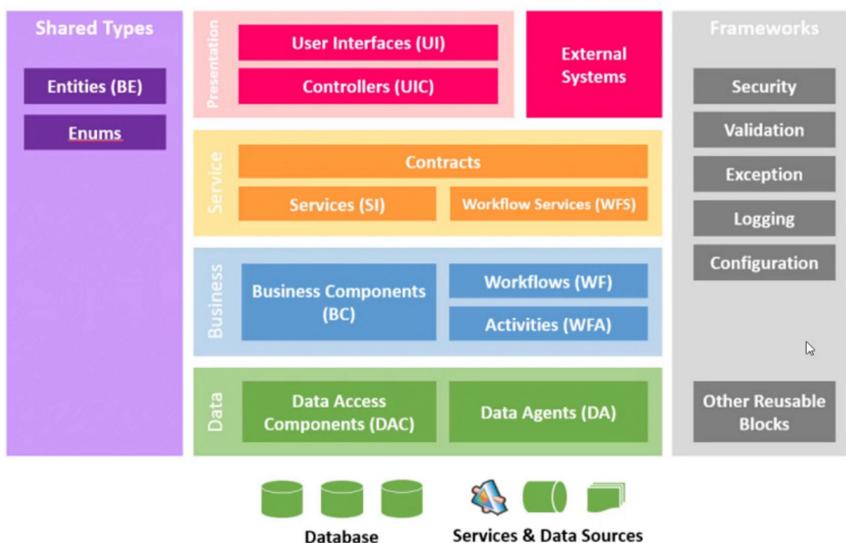
Um estilo arquitetural é uma abordagem de como projetar e entregar uma aplicação

Trata-se de como organizar os componentes responsáveis de um arquitetura, como eles irão interagir entre si e quais aspectos tecnológicos irão atender

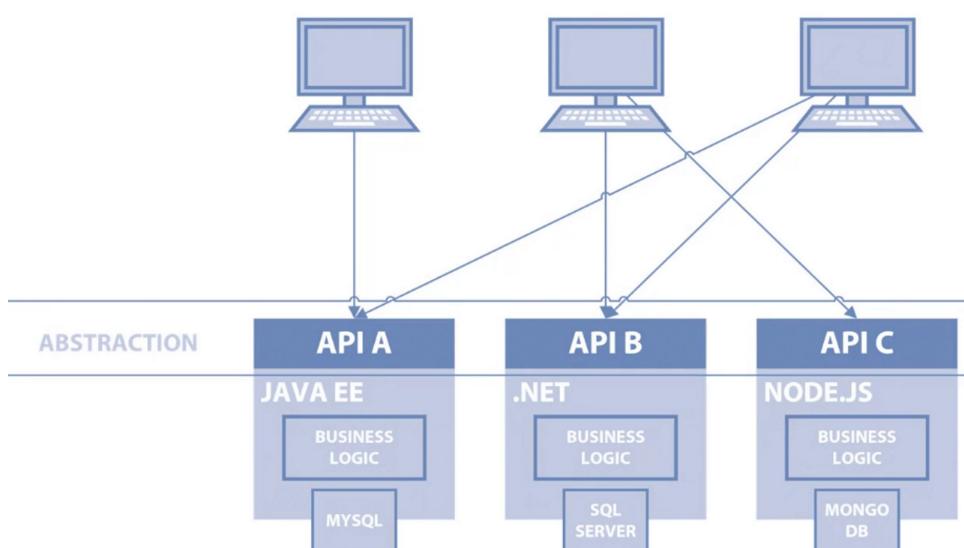
Arquitetura Monolítica



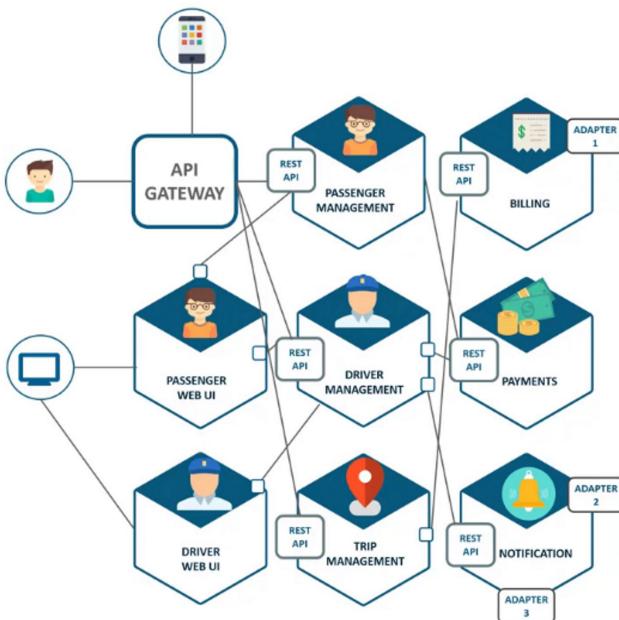
Arquitetura em camadas



Arquitetura REST



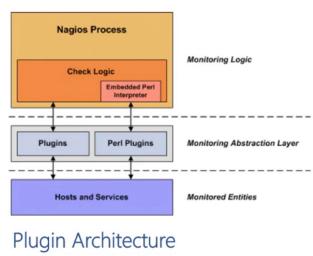
Arquitetura Microservices



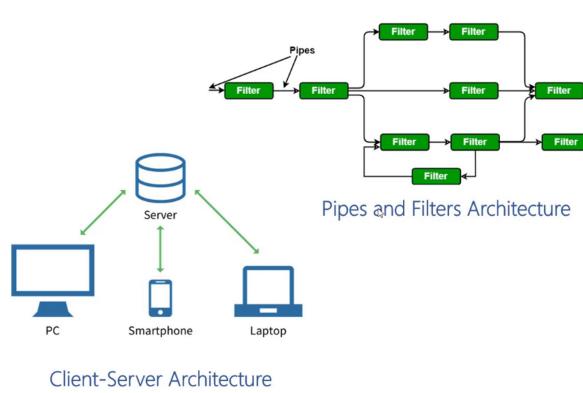
SOA (Service-Oriented Architecture) vs Microservices



Outras Arquiteturas:



Plugin Architecture



Pipes and Filters Architecture

Client-Server Architecture

Padrões arquiteturais:

Os padrões arquiteturais são semelhantes aos Design Patterns, mas possuem um escopo diferente.

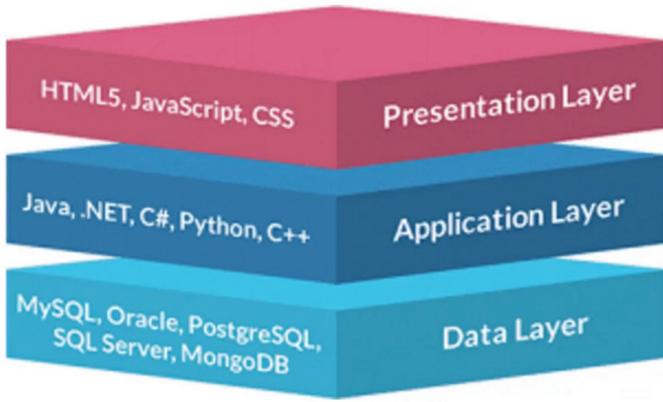
Padrões arquiteturais são estratégias de alto nível que dizem respeito a componentes de grande escala, as propriedades e mecanismos globais de um sistemas

Um projeto de arquitetura pode conter diversos estilos arquiteturais, e cada estilo arquitetural pode utilizar diversos padrões arquiteturais. Um padrão arquitetural pode ser um subconjunto de um estilo arquitetural visando um escopo específico.

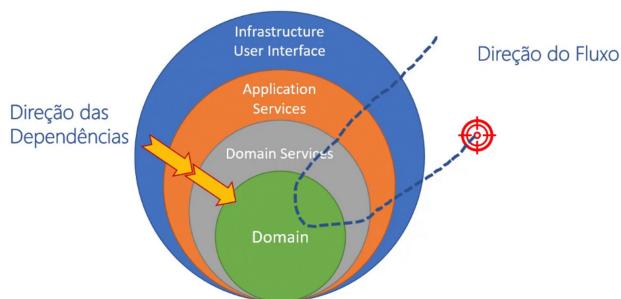
Um padrão arquitetural é uma solução geral e reutilizável para um problema em um contexto particular. É uma solução recorrente para um problema recorrente.

3-Tier Architecture

- Clássica maneira de distribuir responsabilidades
- Não deve ser menosprezada, pois nem sempre complexidade é a solução para um problema simples
- Pode ser aplicada em diversos cenários, porém geralmente é a mais encontrada em aplicações com foco comercial.

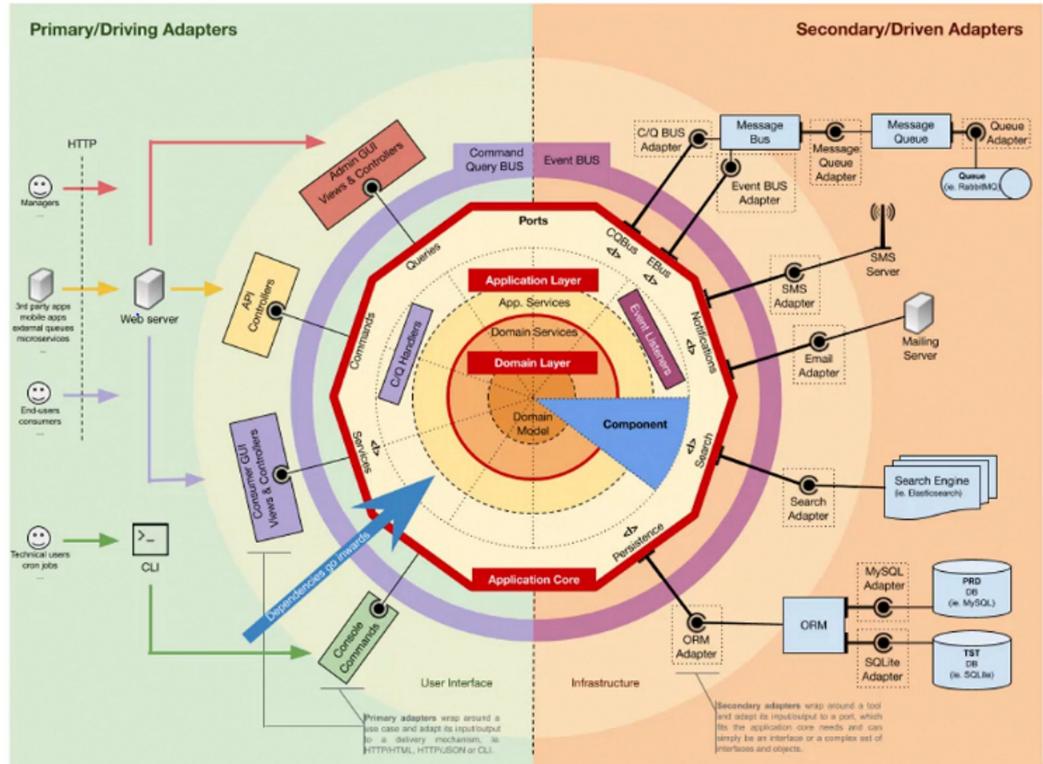


Onion Architecture - Clean Architecture



Hexagonal Architecture

Onion, DDD, Clean, CQRS tudo junto...



CQRS - Command Query Responsibility Segregation

- Um padrão arquitetural onde o foco principal é separar os meios de leitura e escrita dos dados. Alterações de dados são realizados via Commands e leitura de dados são realizados via Queries.
- O objetivo do CQRS é prover expressividade para aplicação, pois todos os Commands representam uma intenção e negócio.

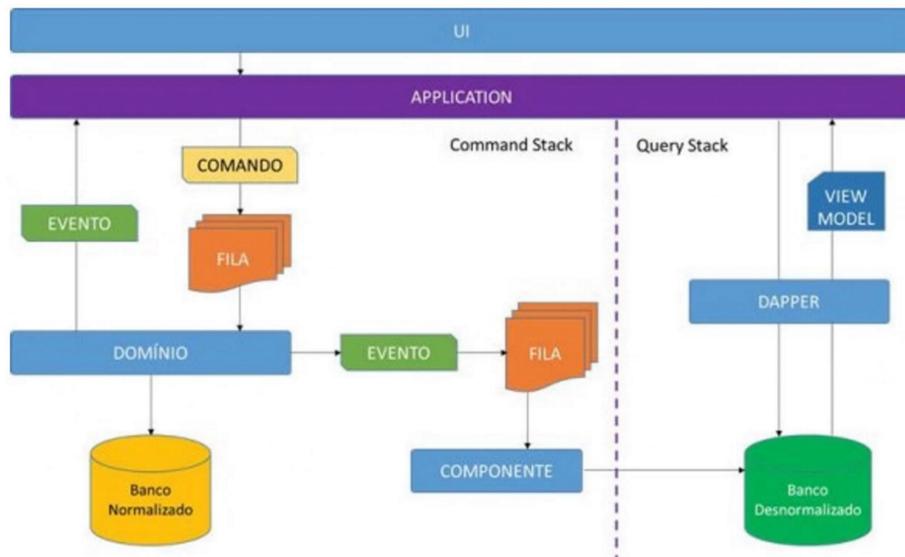
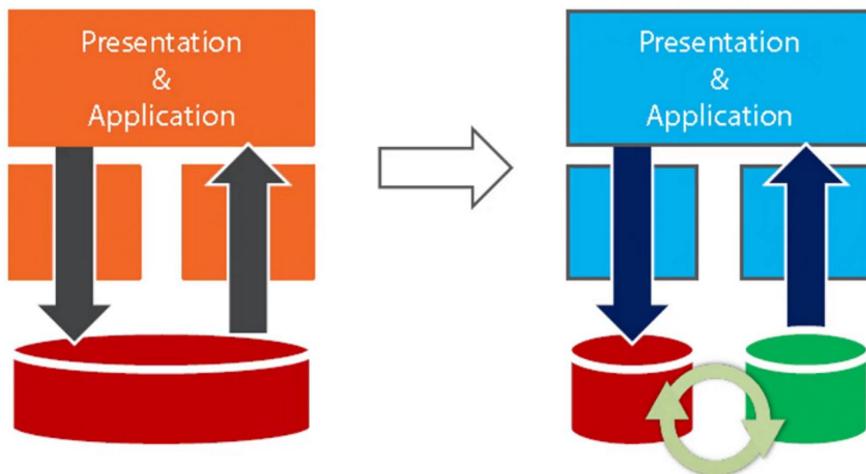
- CQRS promove a consistência eventual, que é quando possuímos um banco de leitura e outro de escrita com os mesmos dados, porém os dados não são consistidos exatamente o mesmo momento
- Muito aplicado em arquiteturas hexagonais, microservices ou em aplicações que possuem alta demanda de consumo de dados

Commands:

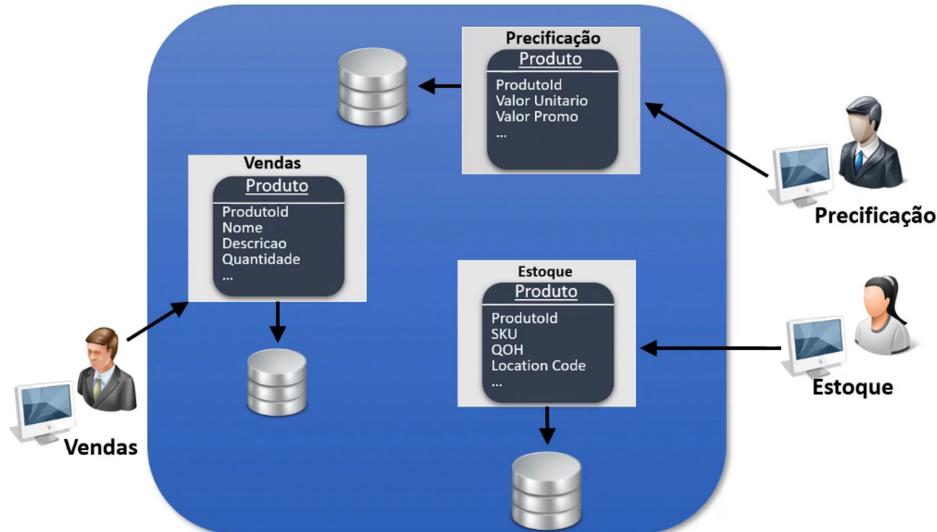
Representa uma intenção de mudança no estado de uma entidade. São expressivos e representam uma única intenção de negócios, ex:
AumentarSalarioFuncionarioCommand

Queries:

É a forma e obter dados de um banco origem de dados para atender as necessidades da aplicação



* Todo comando dispara um evento, seja de sucesso ou de falha



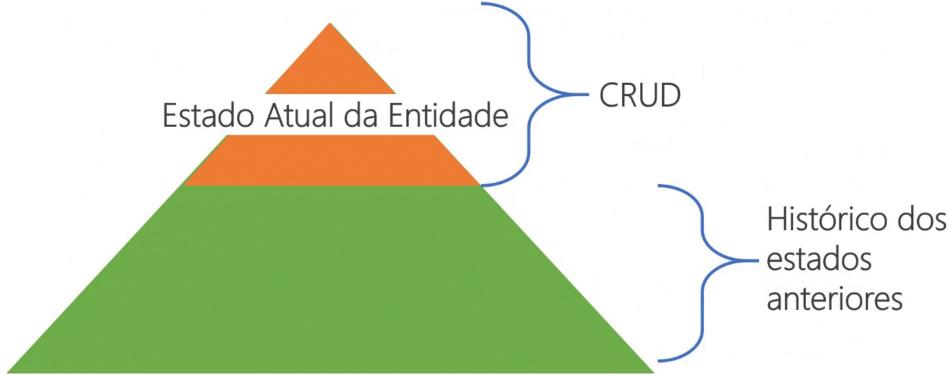
Event Sourcing

"Nós podemos buscar o estado de uma aplicação para encontrar o estado atual do mundo, e isso responde muitas perguntas. Entretanto há momentos que nós não só queremos ver onde nós estamos, mas também queremos saber como chegamos lá" - Martin Fowler

"Event sourcing assegura que todas as mudanças feitas no estado de uma aplicação são armazenadas como uma sequência de eventos.

"Não só podemos buscar esse eventos, mas também podemos usar este log de eventos para reconstruir estados passados e ajustar automaticamente o estado atual com mudanças retroativas" - Martin Fowler

A ideia central é persistir todos estados anteriores e uma entidade negócios desde o momento da sua criação. Com este dados em mãos é possível realizar o "replay" os fatos passados para entender o comportamento usuário, trabalhar BigData, Machine Learning, realizar testes integração cenários reais e simplesmente recrutar as entidades necessárias.



Domain-Driven Design
2003 - Eric Evans

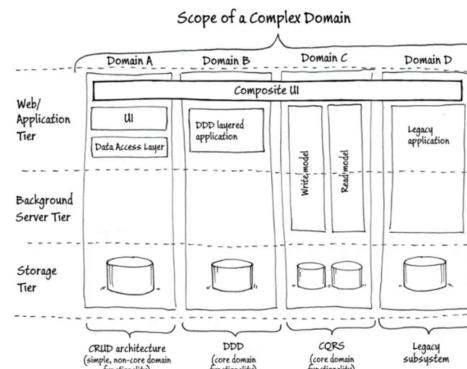
Indicados para aplicações complexas, com muitas entidades e regras de negócio

Razoavelmente fácil de entender, difícil de aplicar

Um guia para entender um negócio, organiza-lo em um conjunto de princípios, criar uma modelagem com base no negócio e implementar utilizando diversas boas práticas

Processo de "implementação" do DDD:

- ✓ Entender o Negócio
- ✓ Extrair a Linguagem Ubíqua
- ✓ Modelagem Estratégica
- ✓ Definir a Arquitetura
- ✓ Modelagem Tática



Linguagem Ubíqua:

- ✓ Vocabulário de todos os termos específicos do domínio
Nomes, verbos, adjetivos, jargões, apelidos, expressões idiomáticas e advérbios
- ✓ Compartilhado por todas as partes envolvidas no projeto
Primeiro passo para evitar desentendimentos
- ✓ Usadas em todas formas faladas e escritas de comunicação
A linguagem universal de um negócio é feita dentro da empresa



Modelagem Estratégica:

Extrair a Linguagem Ubíqua vai colaborar na visão e entendimento do negócio e como segregar seu domínio em partes menores e responsáveis.

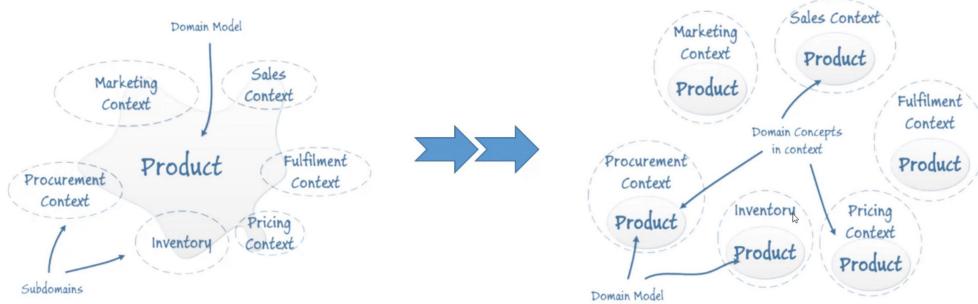
Para documentar estas segregações responsáveis utilizamos o Mapa de Contextos (Context Map) que pode ser representado através de imagens e uma simples documentação do tipo de relacionamento entre os contextos.

Cada contexto delimitado possui sua própria Linguagem Ubíqua, pois em contextos diferentes, os termos podem ter significados diferentes.

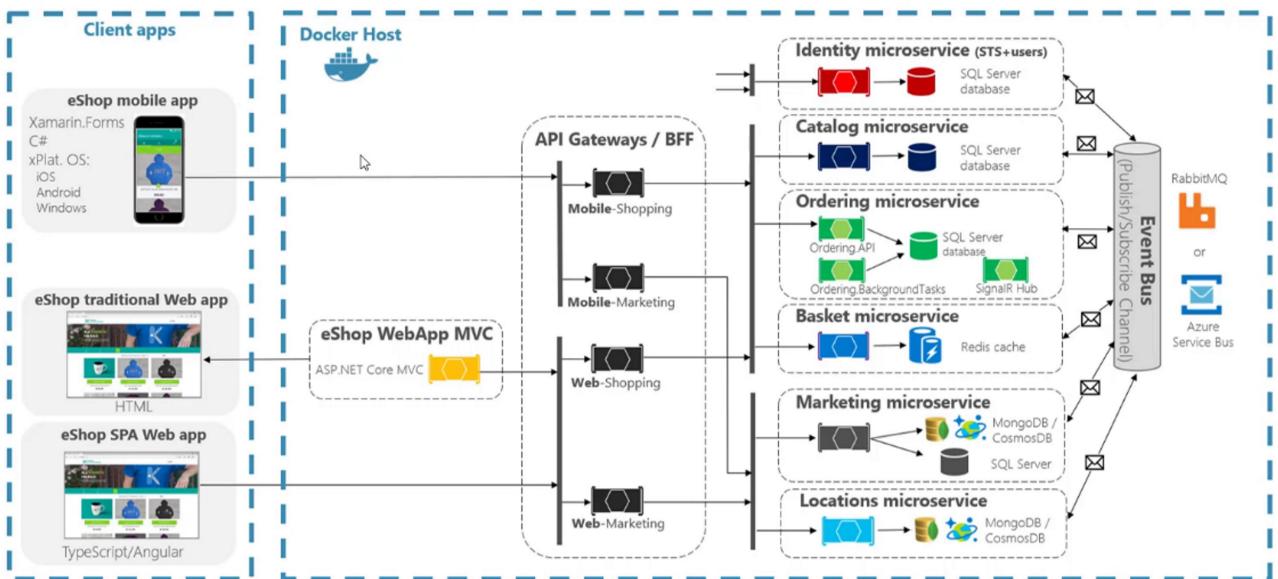


Modelagem Estratégica

Modelo de Negócio vs Modelo de Domínio:



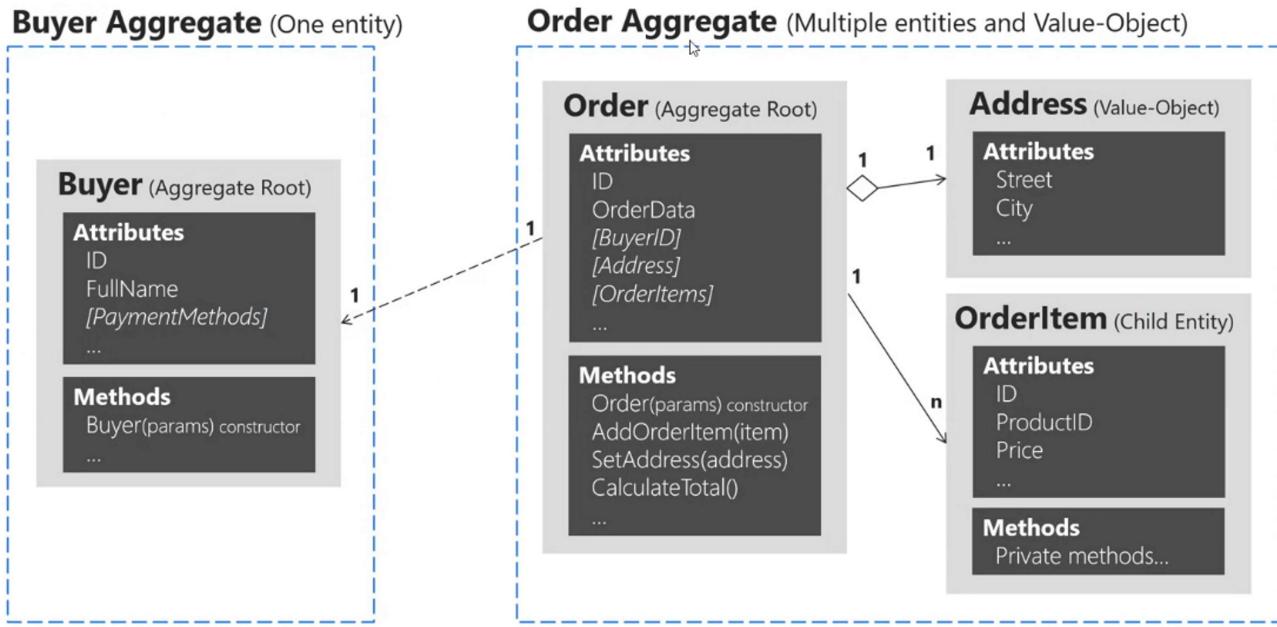
Definir a Arquitetura:



Modelagem Tática:

- ✓ **Aggregate Object**
Uma entidade que é a raiz agregadora de um processo do domínio que envolve mais de uma entidade.
- ✓ **Domain Model**
Uma entidade do domínio, possui estados e comportamentos, lógica de negócio, getters e setters AdHoc, etc.
- ✓ **Value Object**
Um objeto que agrupa valor às entidades, não possui identidade e é imutável.
- ✓ **Factory**
Classe responsável por construir adequadamente um objeto / entidade.
- ✓ **Domain Service**
Serviço do domínio que atende partes do negócio que não se encaixam em entidades específicas, trabalha com diversas entidades, realiza persistência através de repositórios e etc.
- ✓ **Application Service**
Serviço de aplicação que orquestra ações disparadas pela camada de apresentação e fornece DTOs para comunicação entre as demais camadas e para o consumo da camada de apresentação.
- ✓ **Repository**
Uma classe que realiza a persistência das entidades se comunicando diretamente com o meio de acesso aos dados, é utilizado apenas um repositório por agregação.
- ✓ **External Service**
Serviço externo que realiza a consulta/persistência de informações por meios diversos.

Modelagem Tática:



DDD é pensar no negocio do inicio ao fim.

DDD é um guia, cheio de boas práticas de como fazer você pensar no domínio e questionar como funciona, para que possamos de fato implementá-las de forma correta.