



PÓS-GRADUAÇÃO MIT ENGENHARIA DE SOFTWARE COM .NET

Projeto de Bloco: Engenharia de Software e Modelagem

Integrantes:

ALUNO: ALEXANDER SILVA

ALUNO: MARLON BRAGA

ALUNO: PEDRO NOVAES




Projeto de Bloco: Engenharia de Software e Modelagem

MODELAGEM ESTRATÉGICA

Projeto de Bloco: Engenharia de Software e Modelagem


Aplicar os conceitos de DDD

Pontos Chave



Ubiquitous
Language

Uma linguagem onipresente, universal de um negócio dentro da empresa, compartilhada por todas as partes envolvidas no projeto, ou seja, termos específicos do domínio.



Bounded
Context

É uma parte da empresa ou do domínio do negócio que possui elementos ou conceitos com significados bem definidos, com linguagem própria, arquitetura e implementação específica.



Context
Map

É um mapa que representa todos os contextos mapeados através dos elementos da linguagem ubíqua, bem como seus subdomínios e relacionamentos.

Projeto de Bloco: Engenharia de Software e Modelagem

Aplicar os conceitos de DDD

Qual problema vai resolver;
O que vai implementar;
Para que serve;
Quem vai atender

Entender a fundo o negócio



Projeto de Bloco: Engenharia de Software e Modelagem

Aplicar os conceitos de DDD

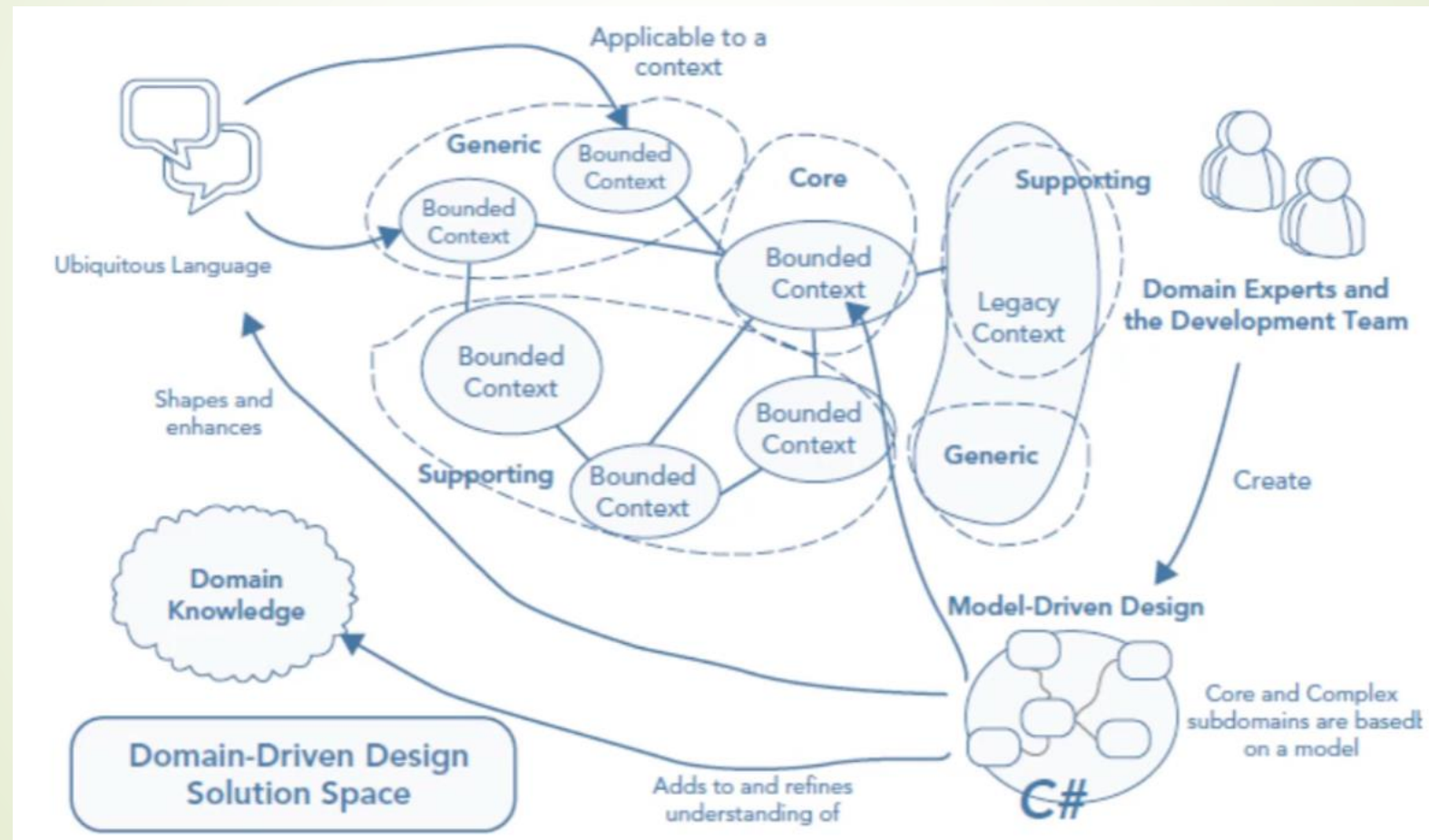
Atenção!!!
Deixar de extrair a
linguagem ubíqua é
um dos piores erros.

Primeiro passo extrair a linguagem ubíqua



Projeto de Bloco: Engenharia de Software e Modelagem

Aplicar os conceitos de DDD



Projeto de Bloco: Engenharia de Software e Modelagem

Aplicar os conceitos de DDD



- A loja virtual exibirá um catálogo de produtos de diversas categorias.
- Um cliente pode realizar um pedido contendo 1 ou N produtos.
- A loja realizará as vendas através de pagamento por cartão de crédito.
- O cliente irá realizar o seu cadastro para poder fazer pedidos.
- O cliente irá confirmar o pedido, endereço de entrega, escolher o tipo de frete e realizar o pagamento.
- Após o pagamento o pedido mudará de status conforme resposta da transação via cartão.
- Ocorrerá a emissão da nota fiscal logo após a confirmação de pagamento do pedido.

Projeto de Bloco: Engenharia de Software e Modelagem

Aplicar os conceitos de DDD



- A loja virtual exibirá um **catálogo** de **produtos** de diversas **categorias**.
- Um **cliente** pode realizar um **pedido** contendo 1 ou N produtos.
- A loja realizará as **vendas** através de **pagamento** por **cartão de crédito**.
- O cliente irá realizar o seu **cadastro** para poder fazer pedidos.
- O cliente irá confirmar o pedido, **endereço** de entrega, escolher o tipo de **frete** e realizar o pagamento.
- Após o pagamento o pedido mudará de **status** conforme resposta da **transação** via cartão.
- Ocorrerá a emissão da **nota fiscal** logo após a confirmação de pagamento do pedido.

Projeto de Bloco: Engenharia de Software e Modelagem

Aplicar os conceitos de DDD



- A loja virtual **exibirá um catálogo de produtos de diversas categorias.**
- Um cliente pode **realizar um pedido** contendo 1 ou N produtos.
- A loja **realizará as vendas** através de **pagamento por cartão de crédito.**
- O cliente irá **realizar o seu cadastro** para poder fazer pedidos.
- O cliente irá **confirmar o pedido**, endereço de entrega, **escolher o tipo de frete** e **realizar o pagamento.**
- Após o pagamento **o pedido mudará de status** conforme **resposta da transação via cartão.**
- Ocorrerá a **emissão da nota fiscal** logo após a **confirmação de pagamento** do pedido.

Projeto de Bloco: Engenharia de Software e Modelagem

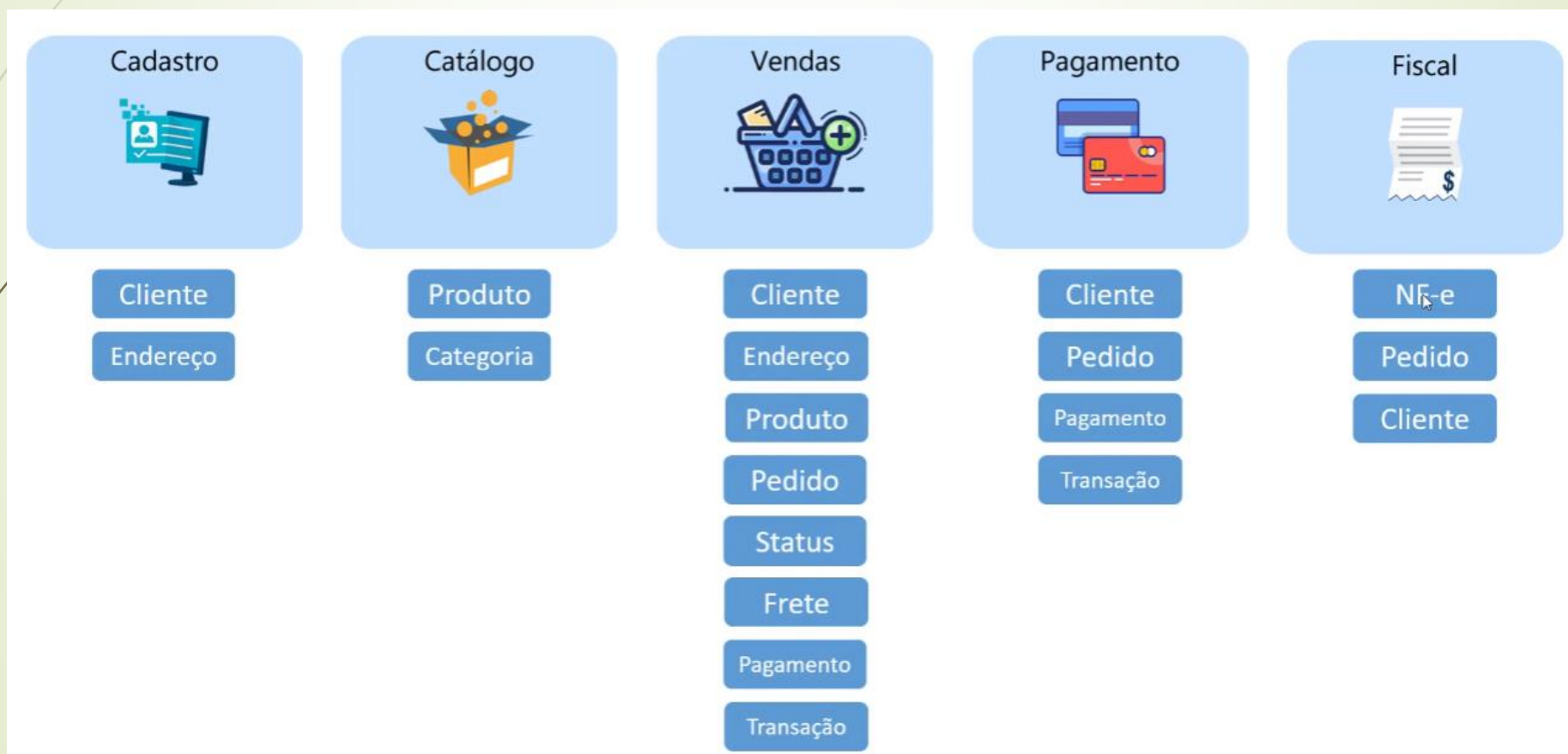
Aplicar os conceitos de DDD

Tipos de Domínios



Projeto de Bloco: Engenharia de Software e Modelagem

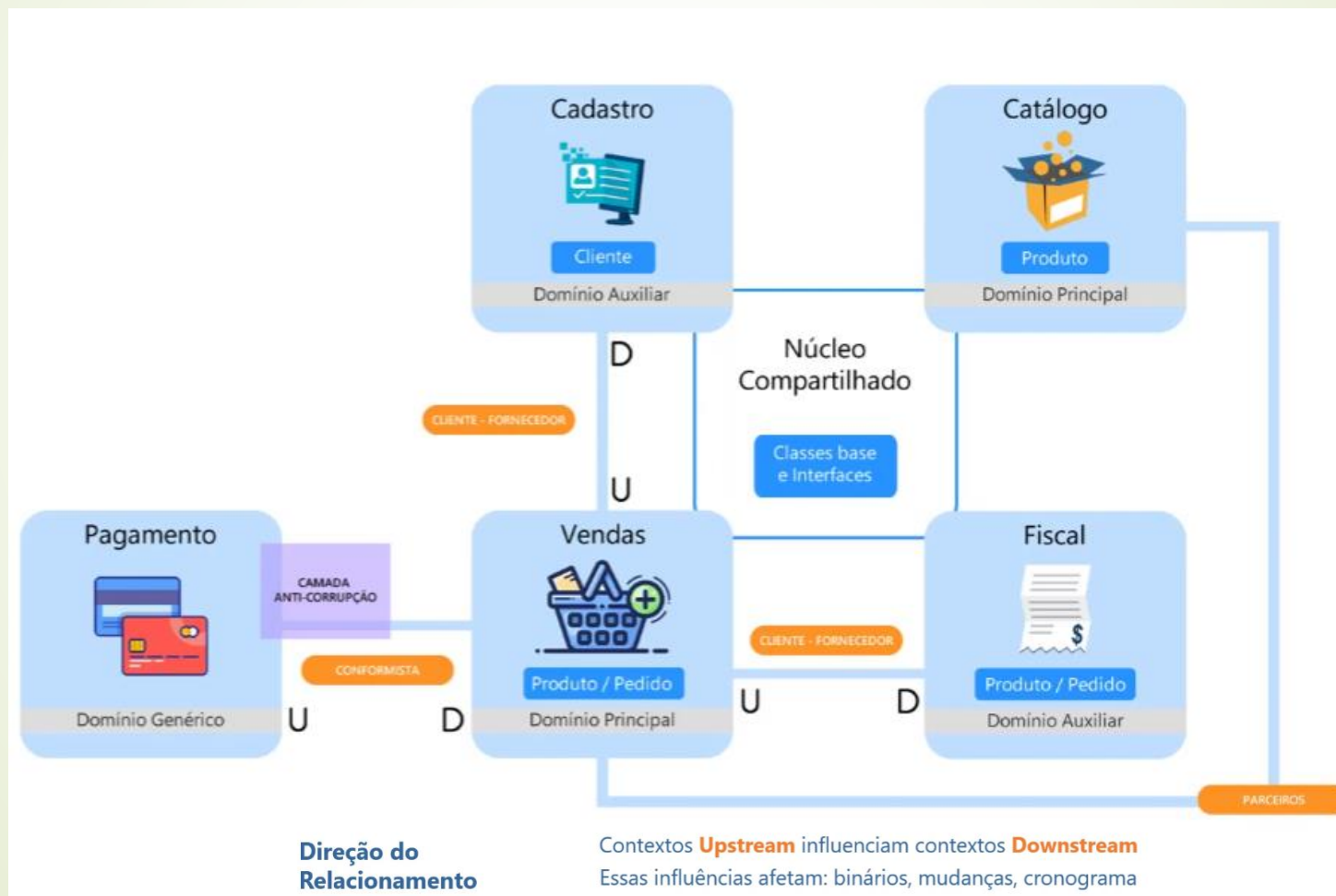
Aplicar os conceitos de DDD



Projeto de Bloco: Engenharia de Software e Modelagem

Aplicar os conceitos de DDD

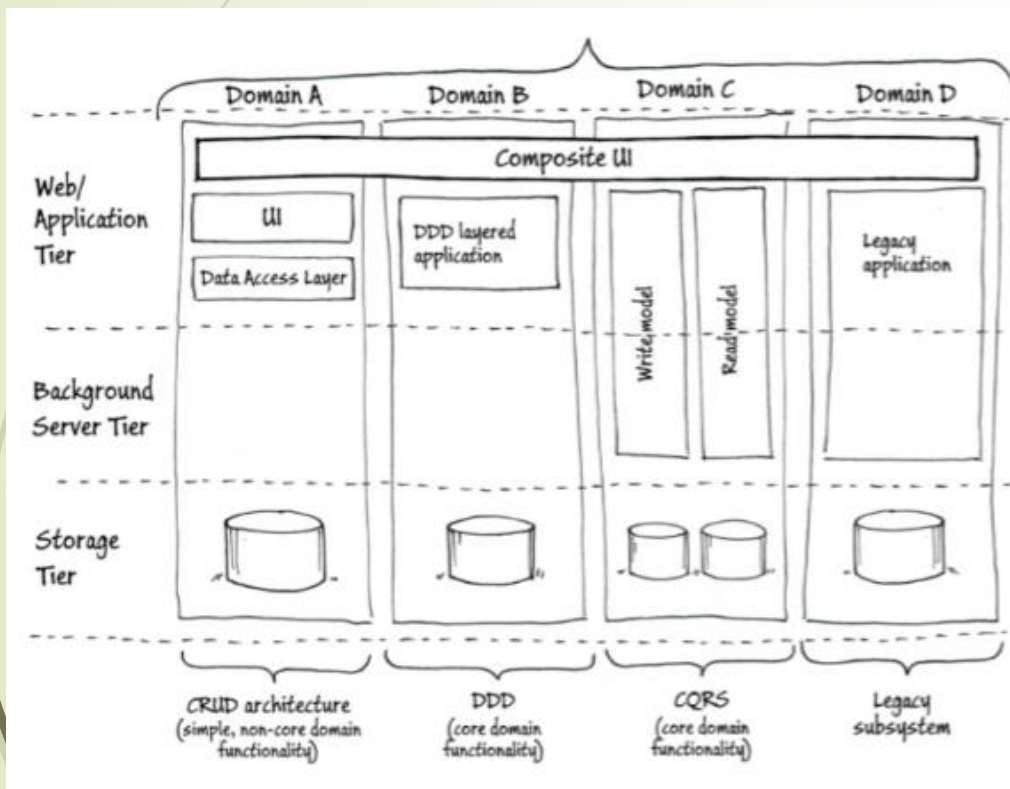
Mapa de Contextos



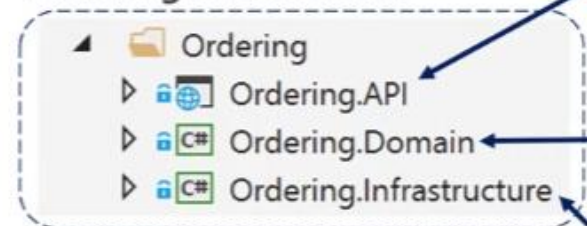
Projeto de Bloco: Engenharia de Software e Modelagem

Aplicar os conceitos de DDD

Definir arquitetura dos contextos



Ordering microservice



Application layer

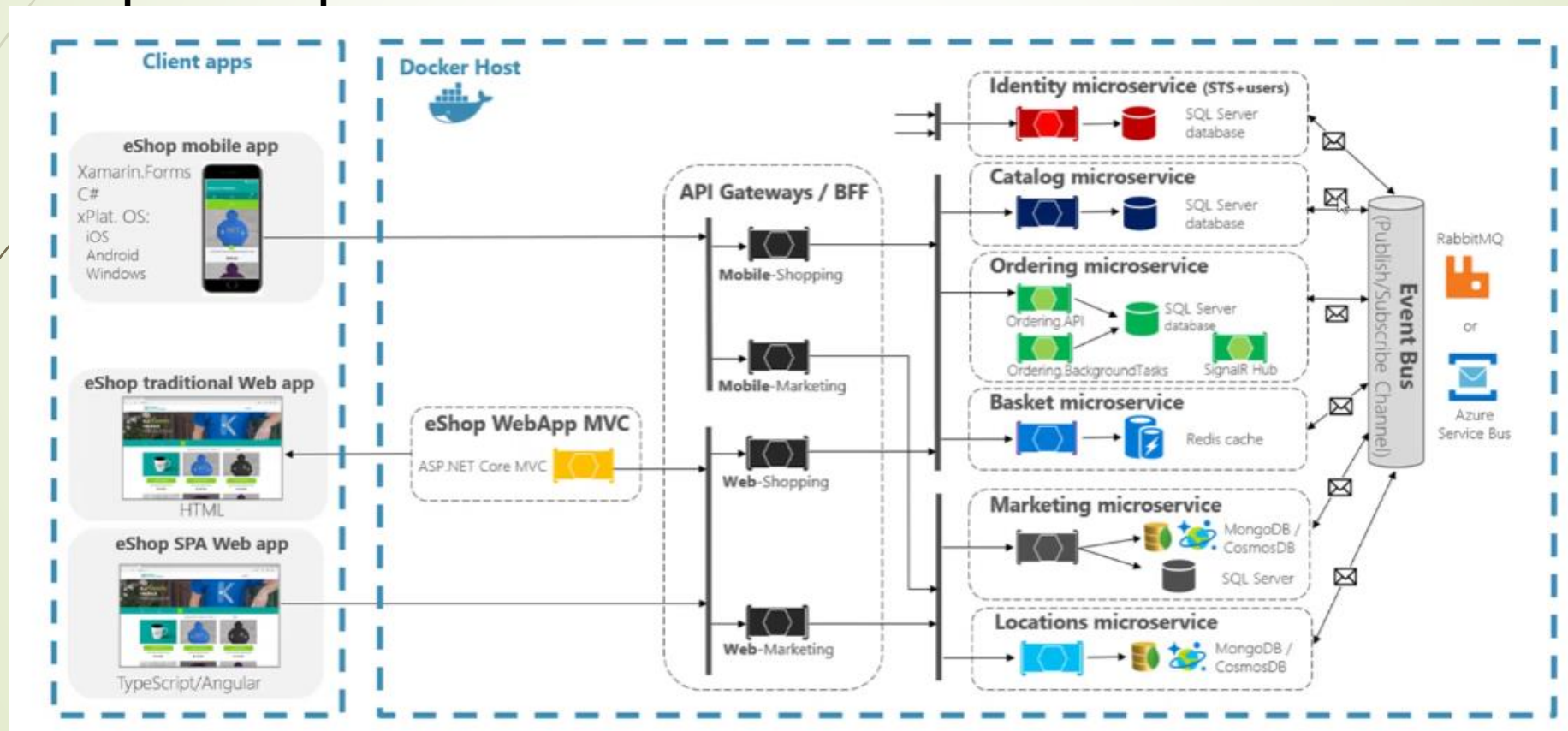
Domain model layer

Infrastructure layer

Projeto de Bloco: Engenharia de Software e Modelagem

Aplicar os conceitos de DDD

Exemplo de arquitetura dos contextos





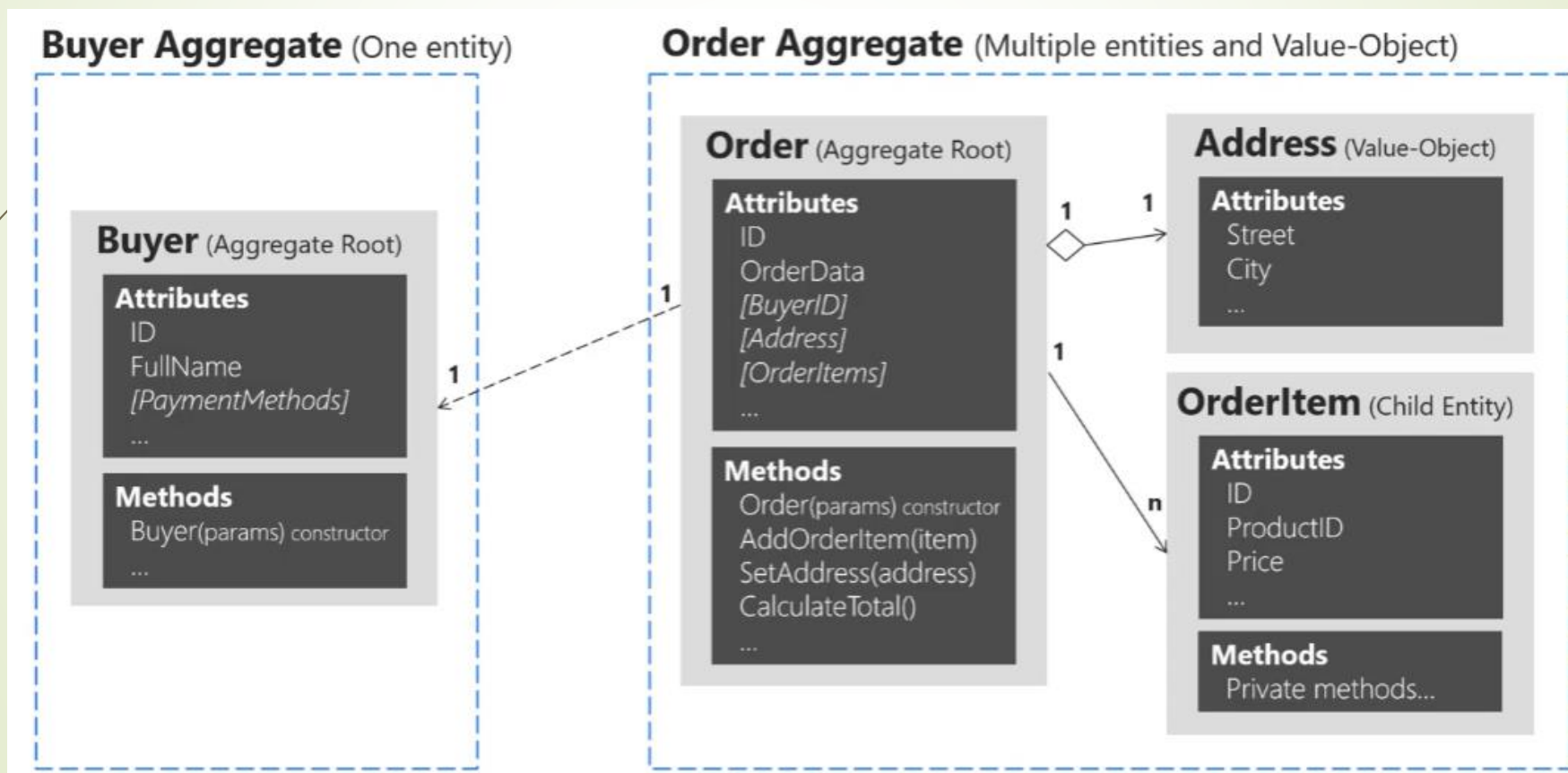
Projeto de Bloco: Engenharia de Software e Modelagem

MODELAGEM TÁTICA

Projeto de Bloco: Engenharia de Software e Modelagem

Aplicar os conceitos de DDD

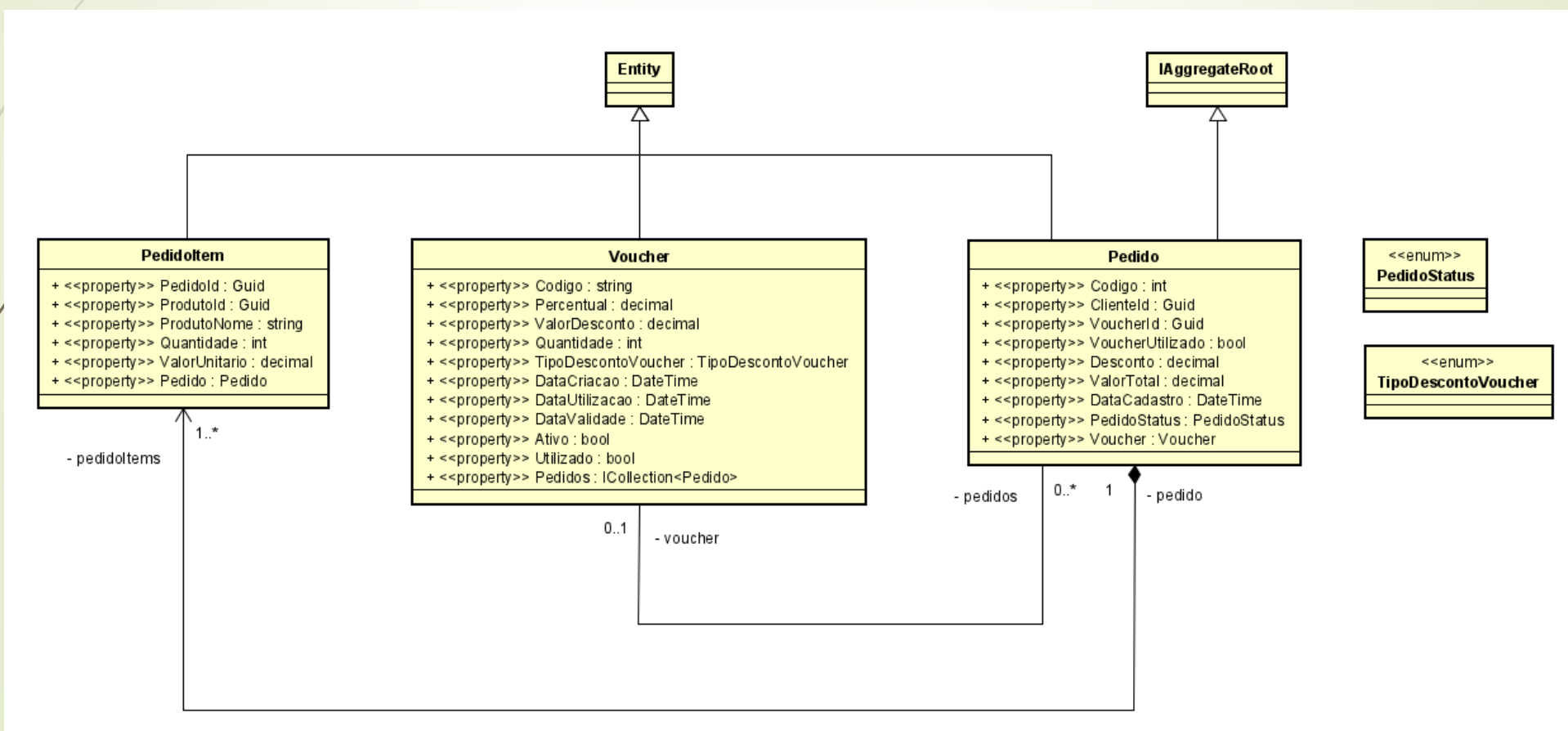
Agregação



Projeto de Bloco: Engenharia de Software e Modelagem

Aplicar os conceitos de DDD

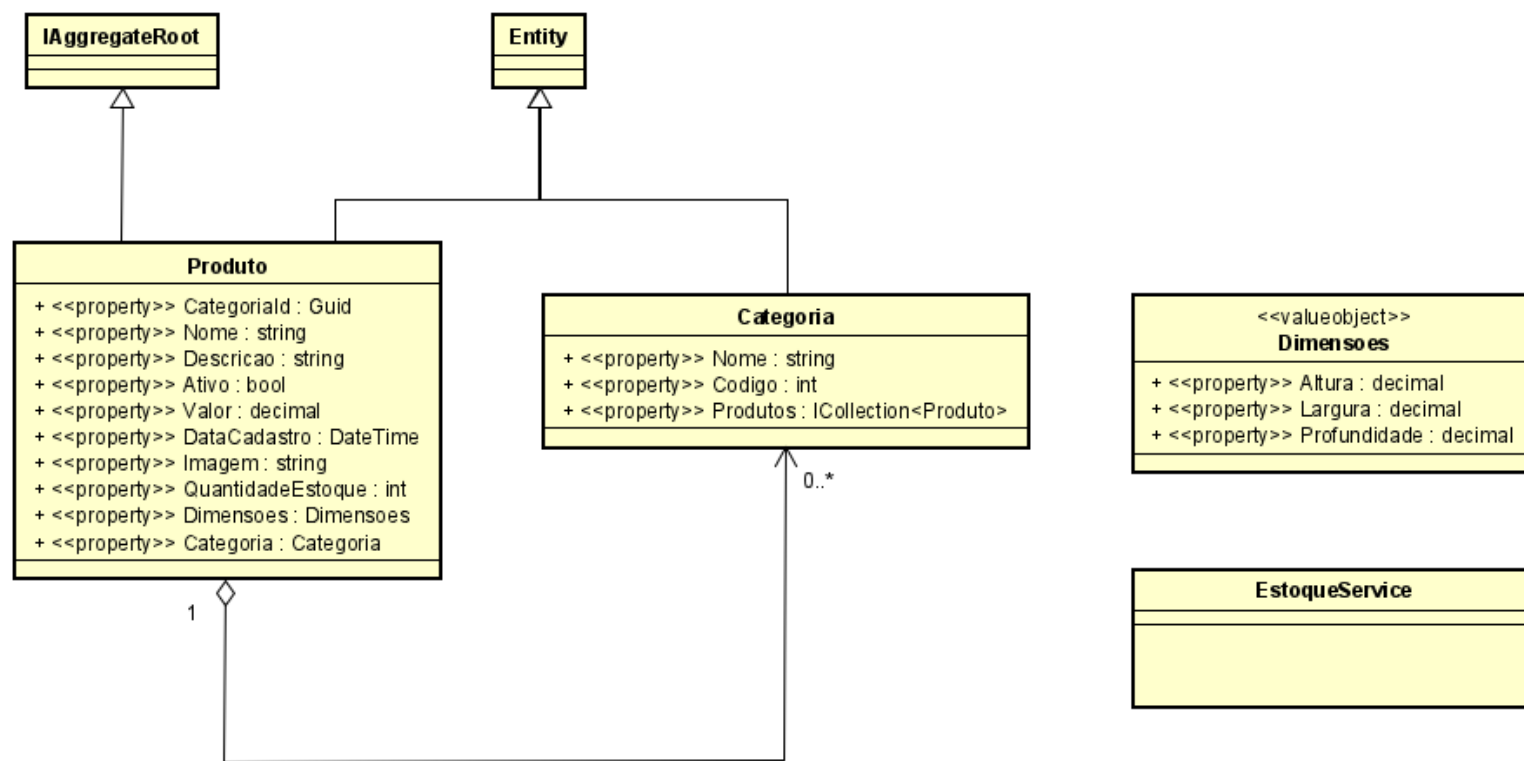
Modelo Conceitual – Contexto Vendas



Projeto de Bloco: Engenharia de Software e Modelagem

Aplicar os conceitos de DDD

Modelo Conceitual – Contexto Catálogo



Projeto de Bloco: Engenharia de Software e Modelagem

Aplicar os conceitos de DDD

Interface de marcação

```
1 namespace NerdStore.Core.DomainObjects
2 {
3     4 referências
4     public interface IAggregateRoot { }
```

```
4 namespace NerdStore.Core.Data
5 {
6     3 referências
7     public interface IRepository<T> : IDisposable where T : IAggregateRoot
8     {
9         18 referências
10        IUnitOfWork UnitOfWork { get; }
```

Raiz de agregação

```
7 namespace NerdStore.Vendas.Domain
8 {
9     21 referências
10    public class Pedido : Entity, IAggregateRoot
11    {
12        3 referências
13        public int Codigo { get; private set; }
14        8 referências
15        public Guid ClienteId { get; private set; }
16        3 referências
17        public Guid? VoucherId { get; private set; }
18        3 referências
19        public bool VoucherUtilizado { get; private set; }
20        4 referências
21        public decimal Desconto { get; private set; }
22        8 referências
23        public decimal ValorTotal { get; private set; }
24        1 referência
25        public DateTime DataCadastro { get; private set; }
26        8 referências
27        public PedidoStatus PedidoStatus { get; private set; }
28
29        private readonly List<PedidoItem> _pedidoItems;
30        10 referências
31        public IReadOnlyCollection<PedidoItem> PedidoItems => _pedidoItems;
32
33        // EF Rel.
34        9 referências
35        public Voucher Voucher { get; private set; }
```


Projeto de Bloco: Engenharia de Software e Modelagem

Aplicar os conceitos de DDD

Objeto de valor

```
1 using System;
2 using NerdStore.Core.DomainObjects;
3
4 namespace NerdStore.Catalogo.Domain
5 {
6     28 referências
7     public class Produto : Entity, IAggregateRoot
8     {
9         4 referências
10        public Guid CategoriaId { get; private set; }
11        4 referências
12        public string Nome { get; private set; }
13        4 referências
14        public string Descricao { get; private set; }
15        3 referências
16        public bool Ativo { get; private set; }
17        2 referências
18        public decimal Valor { get; private set; }
19        1 referência
20        public DateTime DataCadastro { get; private set; }
21        3 referências
22        public string Imagem { get; private set; }
23        5 referências
24        public int QuantidadeEstoque { get; private set; }
25        5 referências
26        public Dimensoes Dimensoes { get; private set; }
27        4 referências
28        public Categoria Categoria { get; private set; }
29    }
30 }
```

```
1 using NerdStore.Core.DomainObjects;
2
3 namespace NerdStore.Catalogo.Domain
4 {
5     10 referências
6     public class Dimensoes
7     {
8         4 referências
9         public decimal Altura { get; private set; }
10        4 referências
11        public decimal Largura { get; private set; }
12        4 referências
13        public decimal Profundidade { get; private set; }
14
15        7 referências
16        public Dimensoes(decimal altura, decimal largura, decimal profundidade)
17        {
18            Validacoes.ValidarSeMenorQue(valor: altura, minimo: 1, mensagem: "O campo Altura não pode ser menor ou igual a 0");
19            Validacoes.ValidarSeMenorQue(valor: largura, minimo: 1, mensagem: "O campo Largura não pode ser menor ou igual a 0");
20            Validacoes.ValidarSeMenorQue(valor: profundidade, minimo: 1, mensagem: "O campo Profundidade não pode ser menor ou igual a 0");
21
22            Altura = altura;
23            Largura = largura;
24            Profundidade = profundidade;
25        }
26
27        1 referência
28        public string DescricaoFormatada()
29        {
30            return $"LxAxP: {Largura} x {Altura} x {Profundidade}";
31        }
32
33        0 referências
34        public override string ToString()
35        {
36            return DescricaoFormatada();
37        }
38    }
39 }
```

Projeto de Bloco: Engenharia de Software e Modelagem

➤ Referências

MODELAGEM de Domínios Ricos. desenvolvedor.io/. Disponível em: <<https://desenvolvedor.io/curso/modelagem-de-dominios-ricos>>. Acesso em: 02 jun. 2022.

MODELAGEM de Software. lms.infnet.edu.br. Disponível em: <<https://lms.infnet.edu.br/moodle/course/view.php?id=5928>>. Acesso em: 10 fev. 2022.

ENGENHARIA de Software Aplicada. lms.infnet.edu.br. Disponível em: <<https://lms.infnet.edu.br/moodle/course/view.php?id=5927>>. Acesso em: 11 dez. 2022.

MODELAGEM de Dados UML. udemy.com. Disponível em: <<https://www.udemy.com/course/uml-diagrama-de-classes/learn/lecture/9881660?start=255#overview>>. Acesso em: 17 abr. 2022.