

- A classe PedidoPersistenceEntity abaixo faz parte da camada de persistência de uma aplicação e tem 2 atributos baseados em 2 classes @Embeddable, conforme demais classes que se seguem.
- A aplicação utiliza o conceito de modelo rico, no qual a classe Pedido é um aggregate root da camada de domínio que utiliza algumas classes object values, que se seguem.
- Na classe PedidoPersistenceEntity foram utilizados os atributos private FaturamentoEmbeddable faturamento e private EntregaEmbeddable entrega para fazer a ligação entre esta classe e suas classes object values InformacoesCobranca e InformacoesEntrega.
- Foram utilizadas as anotações @AttributeOverrides e @AttributeOverride para mapear os atributos e não dar erro de duplicidade.
- Seguindo o modelo iniciado no atributo faturamento, quais demais atributos deverão ser mapeados em faturamento e quais atributos deverão ser mapeados em entrega?
- Considere as demais classes demonstradas em seguida.

```
public class PedidoPersistenceEntity {

    @Id
    @EqualsAndHashCode.Include
    private Long id;
    private UUID clientId;
    private BigDecimal valorTotal;
    private Integer qtdeTotalItens;
    private String status;
    private String formaPagamento;
    private OffsetDateTime criadoEm;
    private OffsetDateTime pagoEm;
    private OffsetDateTime canceladoEm;
    private OffsetDateTime finalizadoEm;

    @CreatedBy
    private UUID criadoPorUsuarioID;

    @LastModifiedDate
    private OffsetDateTime ultimaModificacaoEm;

    @LastModifiedBy
    private UUID ultimaModificacaoPorUsuarioID;

    @Version
    private Long versaoPedido;

    @Embedded
    @AttributeOverrides(
    {
        @AttributeOverride(name = "nome", column = @Column(name = "faturamento_nome")),
        @AttributeOverride(name = "sobrenome", column = @Column(name = "faturamento_sobrenome"))
    }
    )
    private FaturamentoEmbeddable faturamento;

    @Embedded
    private EntregaEmbeddable entrega;
}

public class DestinatarioEmbeddable {
    private String nome;
    private String sobrenome;
    private String cpf;
```

```

    private String telefone;
}

public class EnderecoEmbeddable {
    private String logradouro;
    private String complemento;
    private String bairro;
    private String cidade;
    private String estado;
    private String cep;
}

public class EntregaEmbeddable {
    private BigDecimal valorEntrega;
    private LocalDate previsaoEntrega;

    @Embedded
    private DestinatarioEmbeddable cliente;

    @Embedded
    private EnderecoEmbeddable endereco;
}

public class FaturamentoEmbeddable {
    private String nome;
    private String sobrenome;
    private String cpf;
    private String email;
    private String telefone;

    @Embedded
    private EnderecoEmbeddable endereco;
}

public class Pedido implements AggregateRoot<Pedidold> {

    private Pedidold id;
    private Clienteld clienteld;
    private Moeda valorTotal;
    private Quantidade qtdeTotal;
    private OffsetDateTime feitoEm;
    private OffsetDateTime pagoEm;
    private OffsetDateTime canceladoEm;
    private OffsetDateTime finalizadoEm;
    private InformacoesEntrega entrega;
    private InformacoesCobranca faturamento;
    private PedidoStatus statusPedido;
    private FormasPagamento formasPagamento;
    private Set<PedidolItem> itens;
    private Long versaoPedido;
}

public interface AggregateRoot<ID> {
    ID getId();
}

public record ClienteEndereco(String logradouro, String complemento, String bairro,
    String cidade, String estado, ClienteCEP cep ) {

```

```

public ClienteEndereco {
    ValidacaoCampos.requerCampoPreenchido(logradouro);
    ValidacaoCampos.requerCampoPreenchido(bairro);
    ValidacaoCampos.requerCampoPreenchido(cidade);
    ValidacaoCampos.requerCampoPreenchido(estado);
    Objects.requireNonNull(cep);
}
}

public record InformacoesCobranca(ClienteNome nome, ClienteCPF cpf, ClienteEmail email,
    ClienteTelefone telefone, ClienteEndereco endereco) {

    public InformacoesCobranca {
        Objects.requireNonNull(nome);
        Objects.requireNonNull(cpf);
        Objects.requireNonNull(email);
        Objects.requireNonNull(telefone);
        Objects.requireNonNull(endereco);
    }
}

public record InformacoesEntrega(Moeda valorEntrega, LocalDate previsaoEntrega,
    Destinatario cliente, ClienteEndereco endereco) {

    public InformacoesEntrega {
        Objects.requireNonNull(valorEntrega);
        Objects.requireNonNull(previsaoEntrega);
        Objects.requireNonNull(cliente);
        Objects.requireNonNull(endereco);
    }
}

}

---

```

Boa pergunta 🙋.

A sua entidade de persistência `PedidoPersistenceEntity` está refletindo o **aggregate root** `Pedido`, que contém dois **value objects**:

- `InformacoesCobranca` → representado pelo `FaturamentoEmbeddable`
- `InformacoesEntrega` → representado pelo `EntregaEmbeddable`

Você já iniciou o mapeamento do faturamento com os atributos `nome` e `sobrenome`.

Agora, seguindo a mesma lógica, é preciso mapear **todos os atributos de cada** `@Embeddable`, tomando cuidado com os conflitos de nomes entre faturamento e entrega.

✅ 1. Para faturamento (`FaturamentoEmbeddable`)

A classe tem:

```

private String nome;
private String sobrenome;
private String cpf;
private String email;
private String telefone;

```

```
@Embedded
private EnderecoEmbeddable endereco;
```

Ou seja, além de nome e sobrenome que você já mapeou, faltam:

- cpf → faturamento_cpf
- email → faturamento_email
- telefone → faturamento_telefone

E para o **endereço** (EnderecoEmbeddable), também precisa mapear cada campo com prefixo faturamento_:

- logradouro → faturamento_logradouro
 - complemento → faturamento_complemento
 - bairro → faturamento_bairro
 - cidade → faturamento_cidade
 - estado → faturamento_estado
 - cep → faturamento_cep
-

2. Para entrega (EntregaEmbeddable)

A classe tem:

```
private BigDecimal valorEntrega;
private LocalDate previsaoEntrega;
```

```
@Embedded
private DestinatarioEmbeddable cliente;
```

```
@Embedded
private EnderecoEmbeddable endereco;
```

Aqui:

- valorEntrega → entrega_valor_entrega
- previsaoEntrega → entrega_previsao_entrega

Para o **destinatário** (DestinatarioEmbeddable), precisa mapear com prefixo entrega_cliente_:

- nome → entrega_cliente_nome
- sobrenome → entrega_cliente_sobrenome
- cpf → entrega_cliente_cpf
- telefone → entrega_cliente_telefone

Para o **endereço** (EnderecoEmbeddable), precisa mapear com prefixo entrega_:

- logradouro → entrega_logradouro
 - complemento → entrega_complemento
 - bairro → entrega_bairro
 - cidade → entrega_cidade
 - estado → entrega_estado
 - cep → entrega_cep
-

Resumindo

- **faturamento** precisa mapear: nome, sobrenome, cpf, email, telefone + todos os campos de endereço.
 - **entrega** precisa mapear: valorEntrega, previsaoEntrega, os campos de cliente (nome, sobrenome, cpf, telefone) e todos os campos de endereço.
-

Quer que eu monte o **código completo do** PedidoPersistenceEntity **com todas as** @AttributeOverrides já aplicadas, para você só copiar e colar no projeto?

Sim, monte o código completo da classe PedidoPersistenceEntity com todas as @AttributeOverrides já aplicadas.

Perfeito 🙌.

Seguindo a modelagem que você descreveu e evitando conflitos de nomes de colunas, aqui está a versão completa da classe PedidoPersistenceEntity com todas as @AttributeOverrides aplicadas:

```
import jakarta.persistence.*;
import lombok.EqualsAndHashCode;
import org.springframework.data.annotation.CreatedBy;
import org.springframework.data.annotation.LastModifiedBy;
import org.springframework.data.annotation.LastModifiedDate;
import java.math.BigDecimal;
import java.time.LocalDate;
import java.time.OffsetDateTime;
import java.util.UUID;
```

```
@Entity
public class PedidoPersistenceEntity {
```

```
    @Id
    @EqualsAndHashCode.Include
    private Long id;
```

```
    private UUID clientId;
    private BigDecimal valorTotal;
    private Integer qtdeTotalItens;
```

```

private String status;
private String formaPagamento;
private OffsetDateTime criadoEm;
private OffsetDateTime pagoEm;
private OffsetDateTime canceladoEm;
private OffsetDateTime finalizadoEm;

@CreatedBy
private UUID criadoPorUsuarioID;

@LastModifiedDate
private OffsetDateTime ultimaModificacaoEm;

@LastModifiedBy
private UUID ultimaModificacaoPorUsuarioID;

@Version
private Long versaoPedido;

// -----
// FATURAMENTO
// -----
@Embedded
@AttributeOverrides({
    @AttributeOverride(name = "nome", column = @Column(name = "faturamento_nome")),
    @AttributeOverride(name = "sobrenome", column = @Column(name = "faturamento_sobrenome")),
    @AttributeOverride(name = "cpf", column = @Column(name = "faturamento_cpf")),
    @AttributeOverride(name = "email", column = @Column(name = "faturamento_email")),
    @AttributeOverride(name = "telefone", column = @Column(name = "faturamento_telefone")),

    // Endereco do faturamento
    @AttributeOverride(name = "endereco.logradouro", column = @Column(name = "faturamento_logradouro")),
    @AttributeOverride(name = "endereco.complemento", column = @Column(name =
"faturamento_complemento")),
    @AttributeOverride(name = "endereco.bairro", column = @Column(name = "faturamento_bairro")),
    @AttributeOverride(name = "endereco.cidade", column = @Column(name = "faturamento_cidade")),
    @AttributeOverride(name = "endereco.estado", column = @Column(name = "faturamento_estado")),
    @AttributeOverride(name = "endereco.cep", column = @Column(name = "faturamento_cep"))
})
private FaturamentoEmbeddable faturamento;

// -----
// ENTREGA
// -----
@Embedded
@AttributeOverrides({
    @AttributeOverride(name = "valorEntrega", column = @Column(name = "entrega_valor_entrega")),
    @AttributeOverride(name = "previsaoEntrega", column = @Column(name = "entrega_previsao_entrega")),

    // Cliente (destinatário) da entrega
    @AttributeOverride(name = "cliente.nome", column = @Column(name = "entrega_cliente_nome")),
    @AttributeOverride(name = "cliente.sobrenome", column = @Column(name = "entrega_cliente_sobrenome")),
    @AttributeOverride(name = "cliente.cpf", column = @Column(name = "entrega_cliente_cpf")),
    @AttributeOverride(name = "cliente.telefone", column = @Column(name = "entrega_cliente_telefone")),

    // Endereco da entrega
    @AttributeOverride(name = "endereco.logradouro", column = @Column(name = "entrega_logradouro")),
    @AttributeOverride(name = "endereco.complemento", column = @Column(name = "entrega_complemento")),
    @AttributeOverride(name = "endereco.bairro", column = @Column(name = "entrega_bairro")),

```

```
@AttributeOverride(name = "endereco.cidade", column = @Column(name = "entrega_cidade")),
@AttributeOverride(name = "endereco.estado", column = @Column(name = "entrega_estado")),
@AttributeOverride(name = "endereco.cep", column = @Column(name = "entrega_cep"))
})
private EntregaEmbeddable entrega;
}
```

✓ Agora o PedidoPersistenceEntity cobre:

- **Faturamento** → dados pessoais (nome, sobrenome, cpf, email, telefone) + endereço completo.
- **Entrega** → informações de entrega (valorEntrega, previsaoEntrega) + dados do destinatário (nome, sobrenome, cpf, telefone) + endereço completo.

Quer que eu também monte a **DDL da tabela pedido em SQL** com todas essas colunas já mapeadas (para MySQL/PostgreSQL), para facilitar a conferência no banco?