FATEC OSASCO

PROFESSOR PREFEITO HIRANT SANAZAR

ANDERSON MOREIRA BARBOSA

LEONARDO LEAL E SILVA

MARCOS PAULO FERREIRA DE CARVALHO

CURSO DSM – 3º SEMESTRE

TÉCNICAS DE PROGRAMAÇÃO II

PROF. VICKYBERT PESSOA FREIRE

OSASCO – SP

2024.1

CONTEXTUALIZAÇÃO

O sistema apresentado é um programa de gerenciamento de vendas de produtos escolares com conceito POO usando diagrama UML sem framework, desenvolvido em Técnicas de Programação I com linguagem PHP *(TPI=1 grupo de 6 | TPII=2 grupos de 3)*.

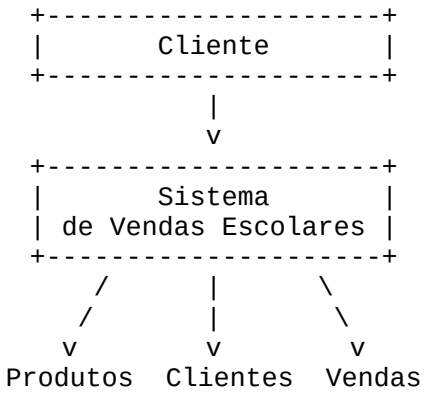
Este controle de vendas permite aos usuários registrar informações detalhadas sobre os clientes e produtos, realizar vendas associando itens aos clientes, calculando o total de vendas e gerando relatórios de vendas e clientes. Desta forma, o sistema resolve os seguintes problemas de forma organizada, automatizada e eficiente:

- ***Cadastro e Gerenciamento de Produtos –*** Permite aos usuários adicionarem novos produtos ao sistema, armazenar informações como descrição, estoque, preço e unidade de medida.
- ***Cadastro e Gerenciamento de Clientes –*** Permite o cadastro de clientes, armazenando informações pessoais e de contato.
- ***Registro de Vendas –*** Permite registrar vendas associadas a clientes cadastrados, adicionando produtos às vendas, aplicando descontos e calculando o valor total da venda.
- ***Listagem de Registros –*** Permite a listagem de todos os produtos, clientes e vendas cadastrados, facilitando a visualização e gerenciamento desses registros.
- ***Impressão de Pedidos –*** Permite a impressão de detalhes específicos de pedidos, incluindo informações do cliente e dos produtos comprados.

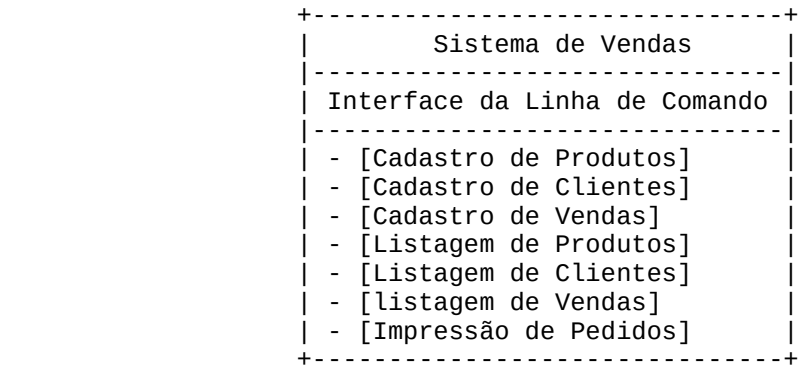
DIAGRAMAS

Estrutura do Sistema Atual

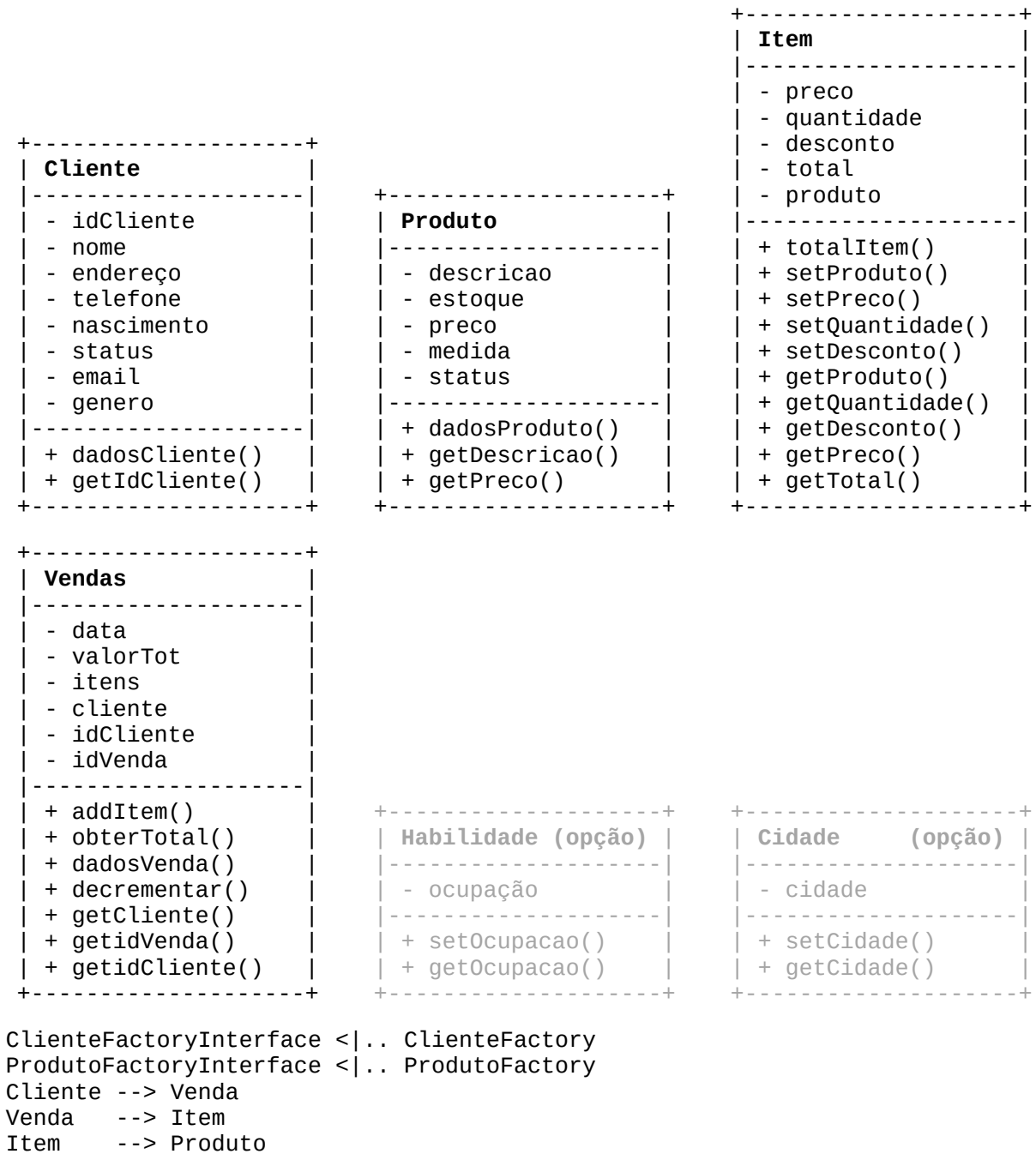
- Diagrama de Contexto (C4 Model - Nível 1)



- Diagrama de Containers (C4 Model - Nível 2)



- Diagrama de Classes do Sistema Atual



Padrões Patterns que serão empregados

- **Factory Pattern –** Facilita a criação de objetos Cliente e Produto, evitando a duplicação de código e promovendo a flexibilidade na criação de instâncias. É particularmente adequado porque separa a lógica de criação do resto do sistema, permitindo mudanças nas classes concretas sem afetar o código cliente.

- **Composite Pattern –** Permite tratar objetos individuais e composições de objetos de maneira uniforme. No contexto do sistema de vendas, isso é essencial para gerenciar uma venda composta de múltiplos itens. O uso do Composite torna a adição de novos tipos de produtos ou itens simples e escalável.

IMPLEMENTAÇÃO

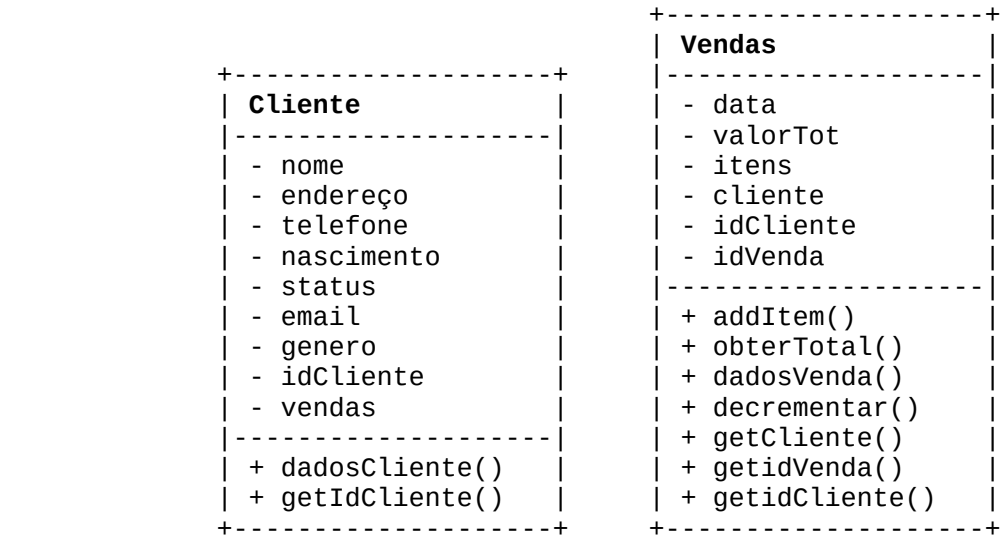
Refatoramos o script para implementar os padrões *FACTORY* e *COMPOSITE*, com separação de responsabilidades e melhorando a manutenção e extensão do código. Desta forma, implementamos ajustes como "*Item*" para representar um item individual de venda, enquanto "*CompositeItem*' para representar a coleção de itens. Isso permitirá que a classe "*Venda*" manipule tanto itens individuais quanto coleções de itens de maneira uniforme ao calcular o total da venda. Além disso, foram realizadas algumas correções e adaptações em todo o código para suportar as novas classes.

Descrição da funcionalidade

Além do aperfeiçoamento da parte de Cadastro de Cliente, permitindo o registro de novos clientes no sistema e na parte de Cadastro de Produtos, houve a readequação do Registro de Vendas, associando os produtos aos clientes visualizando os detalhes das vendas realizadas.

Diagramas de Classes Pré-Modificação

https://onlinegdb.com/3yRAMLo6G




```
Cliente --> Venda
Venda   --> Item
Item    --> Produto
```

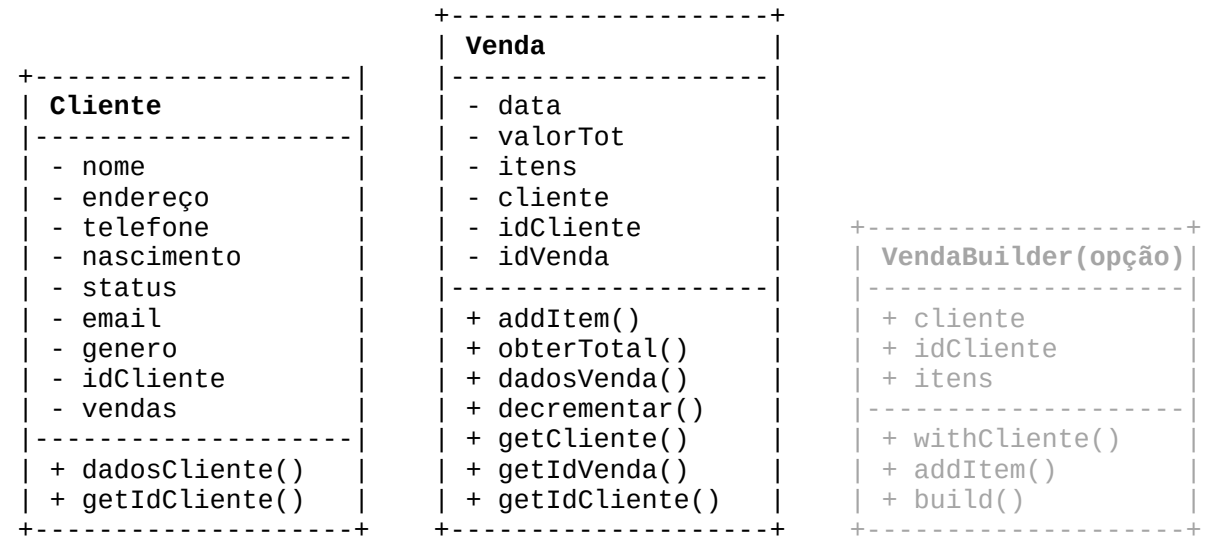
Descrição dos Patterns Aplicados

- **Factory Pattern**: O padrão FACTORY será utilizado para criar objetos ‘Cliente’ e ‘Produto’, através das classes ‘ClienteFactory’ e ‘ProdutoFactory’, proporcionando uma interface para a criação de objetos em uma superclasse, mas permitindo que subclasses alterem o tipo de objetos que serão criados.

- **Composite Pattern**: O padrão COMPOSITE será empregado exclusivamente na estrutura de ‘Venda’, onde ‘Venda’ é composta de múltiplos ‘Item’, e cada ‘Item’ está associado a um ‘Produto’.

Diagramas de Classes Pós-Modificação

https://onlinegdb.com/ItPPZJUqb




```
Cliente      --> Venda
Venda        --> Item
Item         --> Produto
VendaBuilder --> Venda
```

Trechos de Código Mais Relevantes

```
//INTERFACE

interface ClienteFactoryInterface {
    public function criarCliente($nome, $endereco, $telefone, $nascimento, $status, $email, $genero);
}


// CLASSES

class ClienteFactory implements ClienteFactoryInterface {
    public function criarCliente($nome, $endereco, $telefone, $nascimento, $status, $email, $genero) {
        return new Cliente($nome, $endereco, $telefone, $nascimento, $status, $email, $genero);
    }
}

class Venda {
    protected $data;
    protected $valorTot;
    protected $itens;
    protected $cliente;
    protected $idCliente;
    protected $idVenda;

    function __construct(Cliente $cliente, $idCliente){
        $this->idCliente = $idCliente;
        $this->data = date('Y-m-d H:i:s');
        $this->cliente = $cliente;

        self::$contador++;
        $this->idVenda = 'PED' . self::$contador;
        $this->itens = array();
    }
```

```
    public function addItem(Produto $produto, $quantidade, $desconto){
        $item = new Item();
        $item->setProduto($produto);
        $item->setQuantidade($quantidade);
        $item->setDesconto($desconto);
        $item->setPreco($produto->getPreco());
        $item->totalItem();
        $this->itens[] = $item;
    }
}

class VendaBuilder {
    private $cliente;
    private $idCliente;
    private $itens = [];

    public function withCliente(Cliente $cliente, $idCliente) {
        $this->cliente = $cliente;
        $this->idCliente = $idCliente;
        return $this;
    }

    public function addItem(Produto $produto, $quantidade, $desconto) {
        $item = new Item();
        $item->setProduto($produto);
        $item->setQuantidade($quantidade);
        $item->setDesconto($desconto);
        $item->setPreco($produto->getPreco());
        $item->totalItem();
        $this->itens[] = $item;
        return $this;
    }

    public function build() {
        $venda = new Venda($this->cliente, $this->idCliente);
        foreach ($this->itens as $item) {
            $venda->addItem($item->getProduto(), $item->getQuantidade(), $item->getDesconto());
        }
        return $venda;
    }
}
```

PROJETO ORIGINAL
https://onlinegdb.com/3yRAMLo6G

PROJETO COM IMPLEMENTAÇÃO
https://onlinegdb.com/ItPPZJUqb