Ministério da Educação
Universidade Federal do Cariri

UNIVERSIDADE FEDERAL DO CARIRI - UFCA
PRÓ-REITORIA DE GRADUAÇÃO - PROGRAD
CENTRO DE CIÊNCIA E TECNOLOGIA - CCT
CURSO DE BACHARELADO EM ENGENHARIA DE SOFTWARE

ESPECIFICAÇÃO PROJETO 1 - POO
TEMA 9 – SISTEMA DE LOJA VIRTUAL SIMPLIFICADA

1. Visão Geral

Desenvolver um sistema de **linha de comando (CLI)** ou **API mínima** (FastAPI/Flask, opcional) para uma **loja virtual**: **cadastro de produtos e clientes**, **carrinho**, **pedido**, **pagamento**, **cálculo de frete**, **emissão de nota/sumário de compra** e **relatórios de vendas**. A persistência deve ser simples (em **JSON** ou **SQLite**) e a modelagem orientada a objetos deve enfatizar **herança, encapsulamento, validações e composição**.

2. Requisitos Funcionais

1. **Produtos (CRUD)**
   - Campos: SKU (único), nome, categoria, preço unitário (>0), estoque (≥0), ativo/inativo.
   - Ajuste de estoque em entradas (compra/fornecedor) e baixas (pedido faturado).
2. **Clientes (CRUD)**
   - Campos: id, nome, email, CPF (ou doc.), endereço(s) com CEP/cidade/UF.
   - Impedir duplicidade de email/CPF.
3. **Carrinho**
   - Adicionar/remover/alterar quantidade de itens (produto + quantidade).
   - Cálculo de subtotal do carrinho.

- Validação: quantidade solicitada ≤ estoque disponível (no momento da criação do pedido).

4. **Cupons de desconto (opcional mas recomendado)**
   - Código, tipo (`VALOR` ou `PERCENTUAL`), valor/margem, **data de validade**, uso máximo e categorias elegíveis.
   - Aplicar apenas se regras forem atendidas.
5. **Pedido**
   - Criar pedido a partir do carrinho: cliente, itens (SKU, qtd, preço na data), **frete**, **desconto**, **total**.
   - Estados: `CRIADO`, `PAGO`, `ENVIADO`, `ENTREGUE`, `CANCELADO`.
   - Cancelar somente se `CRIADO` ou `PAGO` (com regras de estorno).
   - Gerar **resumo/nota** textual (itens, valores, endereço, formas de pagamento).
6. **Pagamento**
   - Registrar pagamento: data, forma (`PIX`, `CREDITO`, `DEBITO`, `BOLETO`), valor.
   - Validar: total pago ≥ total do pedido (após frete e descontos).
   - Marcar pedido como `PAGO` ao atingir o total devido.
7. **Frete**
   - Calcular frete por **faixa de CEP/UF** ou **tabela simples por cidade/UF** (parametrizado em `settings.json`).
   - Acrescentar prazo estimado (dias úteis).
8. **Expedição**
   - Gerar código de rastreio fictício ao mudar para `ENVIADO`.
   - Marcar como `ENTREGUE` com data de entrega.
9. **Relatórios (mín. 3)**
   - Faturamento por período.
   - Top N produtos mais vendidos e ticket médio.
   - Vendas por categoria/UF.
   - Pedidos por status (quantidade e percentual).
10. **Configurações** (`settings.json`)

- Tabela de fretes por UF/cidade/faixa de CEP, `top_n_produtos`, validade padrão de cupons, limites de parcelamento (se quiserem simular), política de cancelamento (janela em horas).

3. Requisitos Técnicos de POO (RT)

- **Modelagem e herança:**
  - `Produto`, `Cliente`, `Endereco`, `Carrinho`, `ItemCarrinho`, `Pedido`, `ItemPedido`, `Pagamento`, `Cupom`, `Frete`.
  - Subclasses opcionais de `Produto` (ex.: `ProdutoDigital` sem frete, `ProdutoFisico` com peso).
- **Encapsulamento e validações:**

  - `@property` para preço (>0), estoque (≥0), email válido, CPF válido (pode ser formato simples), quantidade (≥1).
  - Métodos que mantenham **consistência** entre carrinho/pedido/estoque.
- **Métodos especiais (≥ 4):**
  - `__str__/__repr__` em `Produto`/`Pedido`.
  - `__eq__` em `Produto` (por SKU) e `Cliente` (por CPF/email).
  - `__len__` em `Carrinho` (quantidade total de itens).
  - `__lt__` em `Produto` (ordenar por preço ou nome).
- **Cálculos (sem padrões):**
  - Funções para subtotal, desconto de cupom, frete e total final.
  - Função para **atualizar estoque** ao **faturar** (`PAGO`) e **repor** no cancelamento (quando aplicável).
- **Persistência:**
  - Módulo `dados.py` com funções simples para salvar/carregar **produtos, clientes, pedidos, cupons** em JSON **ou** uso direto de `sqlite3` (sem ORM).
  - Rotina de **seed** (produtos, clientes, tabela de frete, 1–2 cupons).
- **Testes (`pytest`):**
  - Adição/remoção/quantidade no carrinho; estoque insuficiente; cupom expirado ou inválido; cálculo de frete; pagamento parcial/total; transições de estado; cancelamento.
- **Interface:**
  - CLI com subcomandos, por exemplo:
    - `loja cadastrar-produto`, `loja cadastrar-cliente`,
    - `loja add-carrinho`, `loja fechar-pedido`,
    - `loja pagar`, `loja enviar`, `loja entregar`, `loja cancelar`
    - `loja relatorio faturamento`, `loja relatorio top`.
  - **Ou** API mínima com endpoints equivalentes.

4. Regras de negócio (essenciais)

- **Estoque:** não permitir criar pedido com quantidade > estoque disponível; baixa de estoque ao **confirmar pagamento**; **estorno de estoque** se pedido `CANCELADO` após pago.
- **Cupons:** aplicar apenas se válidos (data, uso máximo, categorias elegíveis); **não** permitir desconto que torne total < 0.
- **Frete:** obrigatoriamente calculado antes do pagamento; produtos digitais não somam frete.
- **Pagamentos e estados:**
  - `CRIADO → PAGO → ENVIADO → ENTREGUE`
  - `CRIADO → CANCELADO` ou `PAGO → CANCELADO` (com estorno de estoque).
- **Cancelamento:** só dentro da **janela configurada** ou se ainda não `ENVIADO`.
- **Endereço e CPF/email:** obrigatórios e válidos para fechar pedido.

- **Unicidade:** SKU único; email/CPF únicos.

5. Critérios de aceite

✅ **POO:** classes bem modeladas, encapsulamento com `@property`, **≥ 4 métodos especiais** aplicados.

✅ **Regras:** estoque, cupons, frete, pagamentos e estados corretos.

✅ **Persistência:** JSON/SQLite funcional (com seed).

✅ **Testes: ≥ 15** cobrindo casos felizes e de erro (estoque, cupom, frete, pagamento, cancelamento).

✅ **Relatórios: ≥ 3** implementados (faturamento, top N, ticket médio, por UF/categoria).

✅ **Documentação:** `README` com instruções, diagrama simples e decisões de implementação.

6. Cronograma

Semana 1 — 18/11/2025

Tema: Modelagem OO e definição do projeto
Entregas:

- UML textual (classes, atributos, métodos principais, relacionamentos).
- Arquivo `README.md` inicial com:
  - Descrição do projeto e objetivo.
  - Estrutura planejada de classes.
- Código inicial com classes vazias e docstrings de propósito.

Semana 2 — 25/11/2025

Tema: Implementação das classes base e encapsulamento
Entregas:

- `Produto`, `Cliente`, `Endereco`, `Carrinho`, `ItemCarrinho` com validações (`@property`).
- Métodos especiais principais (`__len__` em `Carrinho`, `__eq__`/`__repr__` onde couber).
- Testes básicos (`pytest`) para criação e manipulação de objetos.

Semana 3 — 02/12/2025

Tema: Herança, relacionamentos e persistência básica
Entregas:

- `Pedido`, `ItemPedido`, `Pagamento`, `Cupom`, `Frete`; fechar pedido a partir do carrinho.
- Persistência simples (JSON ou SQLite).
- Relatório inicial: **ocupação** por período.

Semana 4 — 09/12/2025

Tema: Regras de negócio e integração
Entregas:

- Estoque, cupons (validação completa), cálculo de frete, pagamento parcial/total, cancelamento com estorno.
- CLI ou API mínima funcional.
- Testes cobrindo fluxos principais e erros (estoque insuficiente, cupom expirado, endereço inválido).

Semana 5 — 16/12/2025 (Entrega final)

Tema: Padrões de projeto, refinamento e documentação final
Entregas:

- Relatórios consolidados: faturamento por período, top N, ticket médio, vendas por UF/categoria.
- `README` completo (execução, testes, exemplos) + diagrama final.
- Todos os testes passando (`pytest`).
- Repositório GitHub com tag `v1.0`.

7. Entrega

A entrega de cada etapa será feita via Classroom. Deve ser enviado o link da tag do github com a entrega até as 18:00h da data da entrega.

- Git: criando tags

8. Avaliação

A avaliação será feita seguindo os seguintes critérios: Roteiro de Avaliação