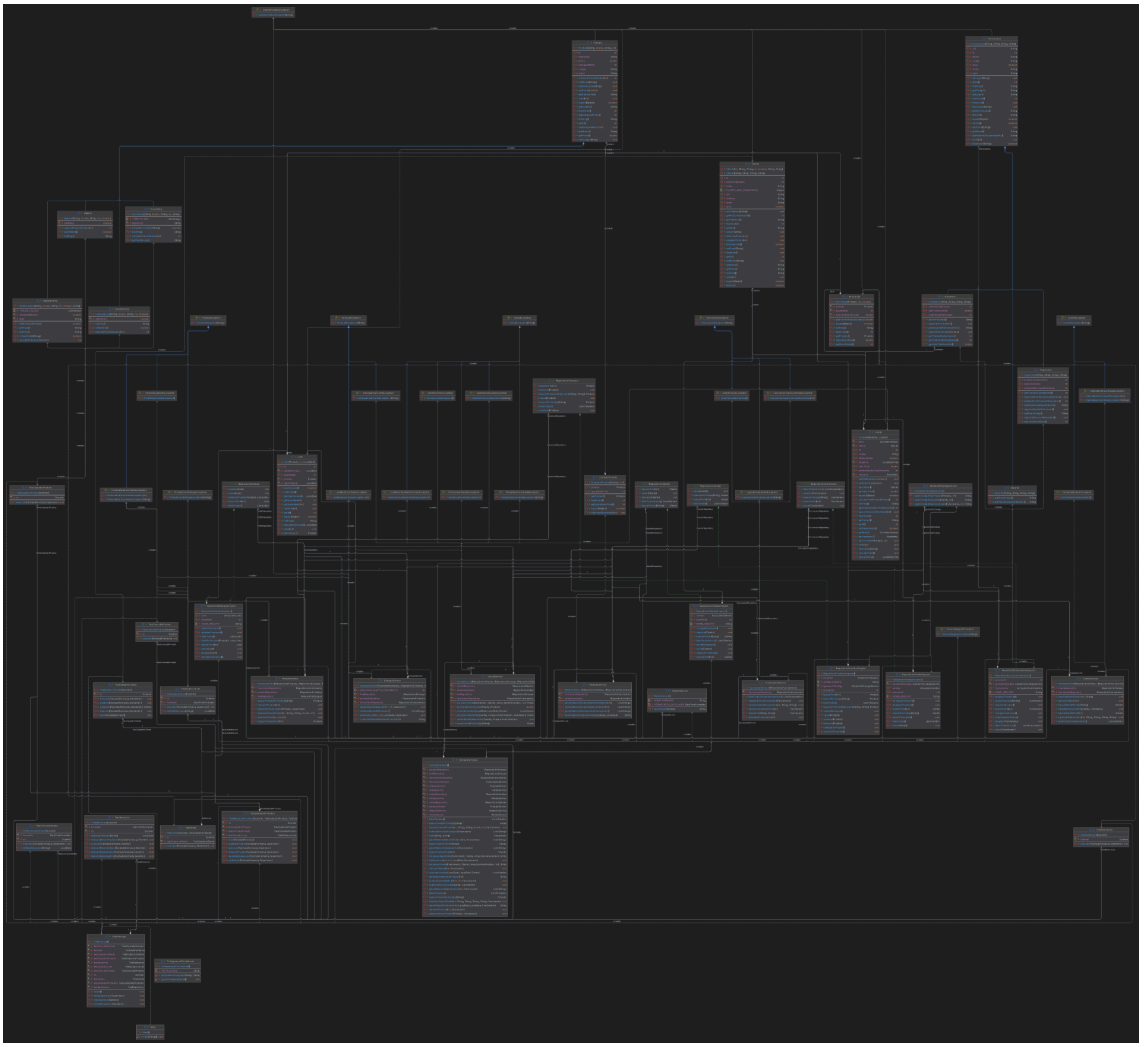
Universidade Federal do Agreste de Pernambuco
Curso: Bacharelado em Ciência da Computação.
Disciplina: Programação Orientada à Objetos.
Professora: Thaís Alves Burity Rocha.
Semestre: 2025.1.
Equipe: Guilherme Henrique Barbosa de Souza Lima,
João Vitor da Silva Moura,
Rafael Carvalho Rodrigues

3ª Entrega do Projeto

1. Diagrama de Classes



(Como o diagrama é muito grande optamos por exportar ele pra um site de imagens para download: https://imgur.com/a/cWshSPZ)

2. Arquitetura do sistema detalhada

Visão Geral das Camadas e da Fachada

O sistema é dividido em quatro camadas lógicas distintas, cada uma com um propósito bem definido. A comunicação entre a interface do usuário e a lógica de negócio é mediada por uma única classe, a FachadaFarmacia, que atua como um ponto de entrada simplificado e seguro para todas as operações do sistema.

1. **Camada de Apresentação** (UI - Pacote ui): Responsável por toda a interação com o usuário. Implementada como uma aplicação de terminal, é composta por uma classe TelaPrincipal que gerencia a navegação e delega as tarefas para "telas especializadas" (TelaVenda, TelaCadastroProduto, etc.). Esta camada não contém regras de negócio; sua única função é exibir dados e capturar a entrada do usuário, comunicando-se exclusivamente com a fachada.

2. **Camada de Fachada** (Fachada - Pacote fachada): A classe FachadaFarmacia é o único ponto de contato da UI com o restante do sistema. Ela realiza os serviços para atender às solicitações e implementa o controle de acesso, verificando o cargo do Funcionario logado antes de permitir a execução das operações.

3. **Camada de Negócio** (Pacote negocio): Esta camada contém a inteligência e as regras do sistema, e é subdividida em.

3.1. **Serviços** (negocio.servico): Classes como **VendaService**, **ProdutoService** e **RelatorioService** contêm a lógica de negócio. Por exemplo, o **VendaService** implementa o algoritmo de baixa de estoque FEFO (Primeiro que expira é o primeiro a sair).

3.2. **Entidades** (negocio.entidade): Representam os objetos do sistema, como **Produto**, **Cliente**, **Venda** e **Funcionario**. Elas encapsulam os dados e as regras de negócio.

3.3. **Exceções** (negocio.excecao): Uma hierarquia de exceções personalizadas (**ProdutoNaoEncontradoException**, **EstoqueInsuficienteException**, etc.) que modelam os erros de negócio.

4. **Camada de Dados** (Pacote dados): É a implementação concreta da persistência.

4.1. **Repositórios** (dados.repositorio):Interfaces como que definem os "contratos" de persistência de dados.

4.2. **Persistência** (dados.persistencia): Classes como **RepositorioProdutosArquivo** implementam as interfaces de repositório. Nesta implementação, elas gerenciam a lógica de salvar e carregar ArrayLists de objetos em arquivos .dat através da serialização Java.

5. **Análise das Classes Principais e Relacionamentos**

A classe abstrata Produto serve como base para tipos concretos como **Medicamento**, **Higiene** e **Cosmético**. Isso permite o uso de polimorfismo no método **calcularPontosGerados()**, onde a mesma chamada resulta em comportamentos diferentes dependendo do tipo real do produto.

Da mesma forma, **Funcionario** é a base para **Atendente**, **Supervisor** e **Gerente**, permitindo o tratamento polimórfico para o método **getResumoDeDesempenho()**.

O relacionamento central é o da **Venda**, que é composta por uma lista de objetos **ItemVenda**. A **Venda** também se associa a um **Cliente** e a um **Atendente**.

Por sua vez, cada **ItemVenda** se associa a um único **Produto**. Essa estrutura garante que cada linha de uma venda seja um registro histórico imutável, gravando inclusive o preço do produto no momento da transação.

O controle de estoque é modelado pela relação entre **Produto** (o item de catálogo) e **Lote** (à instância física com quantidade e validade). Um **Produto** pode ter múltiplos **Lotes** associados.

Ocultação de Informação (Encapsulamento): Conforme visível no diagrama, todos os atributos das classes de entidade são declarados como **private**. O acesso a eles é controlado por métodos públicos (**getters e setters**), onde validações são aplicadas. Um exemplo claro é na classe **Lote**, onde a quantidade só pode ser alterada pelos métodos darBaixa() ou setQuantidade(), que contêm regras para impedir um estoque negativo, garantindo assim a integridade do objeto.

3. Contribuições Individuais

Funcionalidade	**ID do Caso de Uso**	**Responsável**
Como Funcionário, eu quero **fazer login no sistema**, para acessar as funcionalidades do meu cargo.	UC21	Todos
Como Atendente, eu quero **processar uma venda**, para controlar o estoque e os pontos do cliente.	UC10	Guilherme
Como Atendente, eu quero **processar um reembolso**, para estornar o valor e retornar o item ao estoque.	UC11	Guilherme
Como Atendente, eu quero **calcular e atribuir pontos de fidelidade**, para incentivar futuras compras.	UC15	Guilherme
Como Atendente, eu quero **consultar informações de vendas passadas**, para resolver dúvidas ou gerenciar reembolsos.	UC12	Guilherme
Como Atendente, eu quero **adicionar novos clientes**, para que participem do programa de fidelidade.	UC06	Rafael
Como Atendente, eu quero **consultar as informações de um cliente**, para verificar seus dados e pontos.	UC07	Rafael
Como Atendente, eu quero **atualizar os dados de um cliente**, para manter o cadastro correto.	UC08	Rafael
Como Atendente, eu quero **remover o cadastro de um cliente**, para respeitar a privacidade do usuário.	UC09	Rafael
Como Atendente, eu quero **gerar um recibo da transação**, para fornecer o comprovante ao cliente.	UC13	Guilherme
Como Supervisor, eu quero	UC01	Todos

adicionar um novo produto ao catálogo, para que ele possa ser vendido.		
Como Atendente/Supervisor, eu quero **consultar as informações de um produto**, para verificar preço e disponibilidade.	UC02	Todos
Como Supervisor, eu quero **atualizar as informações de um produto**, para manter o catálogo correto.	UC03	Todos
Como Supervisor, eu quero **remover um produto do catálogo**, para corrigir irregularidades.	UC04	Todos
Como Supervisor, eu quero **consultar o estado geral do estoque**, para ter uma visão geral e planejar compras.	UC05	João Vitor
Como Supervisor, eu quero **ser notificado sobre vencimentos e estoque baixo**, para evitar perdas e planejar a reposição.	UC14	João Vitor
Como Supervisor, eu quero **adicionar um lote de um produto existente**, para dar entrada de mercadoria no estoque.	UC22	Todos
Como Supervisor, eu quero **ajustar a quantidade de um lote específico**, para corrigir o inventário.	UC23	João Vitor
Como Gerente, eu preciso **gerar um relatório** do faturamento do mês, para assim verificar o status do faturamento(na média,acima ou abaixo).	UC17	Rafael

Como Gerente, eu preciso **gerar um relatório** dos produtos com mais venda, para que eu possa otimizar as estratégias de compra de estoque.	UC18	João Vitor
Como Gerente, eu preciso **gerar um relatório** dos produtos menos vendidos, para assim tomar medidas para que esses produtos não expirem ou que não sejam comprados.	UC19	João Vitor
Como Gerente, eu preciso **gerar um relatório** mostrando o desempenho dos vendedores no mês, para assim poder verificar os funcionários mais produtivos e menos produtivos no setor de vendas.	UC20	Rafael