CENTRO UNIVERSITÁRIO DE CIÊNCIAS E TECNOLOGIA DO MARANHÃO-UNIFACEMA

CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS – 3º PERÍODO

AUGUSTO CÉSAR EVANGELISTA DOS SANTOS

CHRISTIAN SAMUEL SAMPAIO MARINHO

DEBORAH CRISTINA VIEIRA TRINDADE

GUILHERME PEREIRA DA SILVA

JULIANA LAYARA SANTOS

THUAN CAIQUE LIMA DE SOUSA

TRABALHO DISCENTE EFETIVO

RELATÓRIO

Trabalho Discente Efetivo (TDE) – Disciplina: Análise e Projeto Orientado a Objetos, Professor: Marcos Gomes da Silva Rocha , do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas - UNIFACEMA.

CAXIAS –MA

(2025)

LEVANTAMENTO DE REQUISITOS PARA CONSTRUÇÃO DE UM SISTEMA DE CONTROLE DE ESTOQUE

1. SUMÁRIO

LINK DO GITHUB: https://github.com/Sampaioooo/controle-estoque.git

2. INTRODUÇÃO SOBRE O LEVANTAMENTO DE REQUISITOS

O processo de levantamento de requisitos é uma das fases mais importantes do desenvolvimento de um software, pois define, com precisão, o que será construído. Através dessa etapa, é possível estabelecer claramente as expectativas do sistema, suas funcionalidades, comportamentos, restrições, e garantir que o produto final atenda aos objetivos propostos. No caso deste trabalho, que consiste no desenvolvimento de um **Sistema de Controle de Estoque**, essa fase assume papel central, pois o projeto será construído desde sua concepção até sua execução prática, com base na análise orientada a objetos.

A escolha do sistema de controle de estoque não foi aleatória. Esse tipo de solução é amplamente necessário em pequenos e médios comércios, que muitas vezes enfrentam dificuldades em manter o controle exato dos produtos armazenados. O sistema proposto será responsável por organizar e monitorar os produtos disponíveis em estoque, registrar entradas e saídas, atualizar quantidades automaticamente, emitir alertas de estoque baixo e armazenar o histórico de movimentações.

Todo o levantamento foi elaborado com o objetivo de criar um sistema funcional, priorizando a implementação da **camada de backend**, que é o foco principal da disciplina, sem deixar de lado uma interface simples, funcional e coerente. Abaixo estão descritos os **Requisitos Funcionais (RF)** e os **Requisitos Não Funcionais (RNF)**, com explicações completas.

3. REQUISITOS FUNCIONAIS

Os requisitos funcionais especificam **o que o sistema deve fazer**, ou seja, representam as funcionalidades que serão oferecidas ao usuário e que fazem parte do escopo funcional do software. A seguir, estão listados e descritos todos os requisitos funcionais identificados para o sistema de controle de estoque.

(RF01) Cadastro de Produtos
O sistema deve permitir o cadastro de novos produtos no estoque. Cada produto deve possuir as seguintes informações mínimas: nome, código único (gerado automaticamente ou informado), categoria, preço de venda e quantidade inicial. Essa funcionalidade é essencial para que os produtos possam ser gerenciados posteriormente, além de garantir consistência nos dados.

(RF02) Edição de Produtos
Deve ser possível editar informações de produtos já cadastrados. O usuário poderá alterar dados como nome, categoria, preço ou quantidade. Essa funcionalidade deve ser validada para evitar conflitos, como códigos duplicados ou valores inválidos.

(RF03) Exclusão de Produtos
O sistema deve possibilitar a exclusão de produtos do estoque. Essa operação deverá ser realizada com confirmação por parte do usuário, uma vez que pode comprometer o histórico de movimentações, que deve ser mantido mesmo após a remoção do item.

(RF04) Listagem de Produtos
O sistema deve apresentar uma listagem completa dos produtos em estoque, permitindo filtrar ou pesquisar por nome, código ou categoria. A listagem deve exibir os dados principais de cada produto, como quantidade atual, status (disponível, em falta, estoque baixo), e preço.

(RF05) Registro de Entrada de Produtos
O sistema deve permitir ao usuário registrar entradas de produtos, indicando a quantidade adicionada ao estoque. Esse registro atualizará automaticamente a quantidade do produto e deverá ser armazenado com data, hora e tipo de movimentação.

(RF06) Registro de Saída de Produtos
De forma análoga à entrada, o sistema deve registrar as saídas de produtos do estoque. O usuário informará o produto e a quantidade a ser retirada. O sistema validará se há quantidade suficiente e, em caso positivo, atualizará o estoque e registrará a movimentação.

(RF07) Controle de Quantidade Automático
Toda movimentação de entrada ou saída deve atualizar automaticamente a quantidade disponível de cada produto, garantindo que o sistema mantenha informações corretas em tempo real.

(RF08) Alerta de Estoque Baixo
Ao atingir um valor mínimo previamente definido para cada produto, o sistema deverá emitir um alerta visual indicando que o estoque está baixo. Essa funcionalidade é crucial para manter o abastecimento dos produtos e evitar rupturas no atendimento.

(RF09) Histórico de Movimentações
O sistema deve manter um histórico completo de todas as movimentações realizadas, contendo informações como: produto, tipo da movimentação (entrada ou saída), quantidade, data e hora. Esse histórico permitirá análises futuras e será parte da geração de relatórios.

(RF10) Geração de Relatórios
O sistema deve gerar relatórios básicos sobre o estado atual do estoque e sobre as movimentações realizadas em períodos específicos. Os relatórios devem conter informações claras, organizadas e exportáveis em formato visual (ex: HTML ou PDF futuramente).

(RF11) Categorização de Produtos
O sistema deve permitir a associação de produtos a categorias específicas, como "bebidas", "alimentos", "eletrônicos", etc. Essa categorização ajuda na organização, filtragem e geração de relatórios.

(RF12) Interface de Interação com o Usuário
Mesmo que o foco do trabalho seja o backend, o sistema deve disponibilizar uma interface simples para cadastro, listagem e movimentação dos produtos, possibilitando que qualquer usuário, mesmo sem conhecimentos técnicos, consiga operá-lo.

4. REQUISITOS NÃO FUNCIONAIS

Os requisitos não funcionais dizem respeito a **como o sistema deve funcionar**, abordando critérios técnicos, de desempenho, usabilidade e restrições de tecnologia. Abaixo estão descritos os principais RNFs identificados para o sistema:

(RNF01) Plataforma de Desenvolvimento
O sistema será desenvolvido utilizando a linguagem JavaScript no ambiente Node.js, com o framework Express.js. Essa stack foi escolhida por sua leveza, velocidade de desenvolvimento e ampla utilização no mercado.

(RNF02) Banco de Dados Relacional
Durante o desenvolvimento, será utilizado o banco de dados SQLite, por sua simplicidade e portabilidade. O sistema será estruturado de forma que a migração para um banco mais robusto como MySQL ou PostgreSQL possa ocorrer sem alterações significativas na lógica.

(RNF03) Arquitetura REST
A comunicação entre a interface do usuário e o backend será feita utilizando o padrão RESTful, com rotas organizadas por recursos (/produtos, /movimentacoes, etc.), facilitando o consumo por aplicações externas e testes com ferramentas como Postman.

(RNF04) Validação de Dados
Todos os dados enviados ao sistema deverão passar por processos de validação. O backend não aceitará valores inválidos ou campos obrigatórios ausentes, evitando que inconsistências sejam armazenadas no banco.

(RNF05) Interface Responsiva e Funcional
Ainda que simples, a interface será pensada para ser acessível em diferentes tamanhos de tela e navegadores, permitindo sua utilização em desktops, tablets ou celulares.

(RNF06) Tempo de Resposta
As operações realizadas no sistema deverão ser executadas em tempo inferior a 2 segundos, considerando o uso de conexões padrão e um volume médio de dados.

(RNF07) Modularização do Código
O projeto será desenvolvido seguindo o princípio da separação de responsabilidades, com módulos distintos para controle de rotas, lógica de negócios, manipulação de banco de dados e inicialização do servidor.

(RNF08) Documentação e Comentários
O código será entregue com documentação mínima, comentários explicativos e organização por pastas e arquivos, facilitando sua manutenção, leitura e avaliação por parte do professor.

5. MODELAGEM UML DO SISTEMA DE CONTROLE DE ESTOQUE

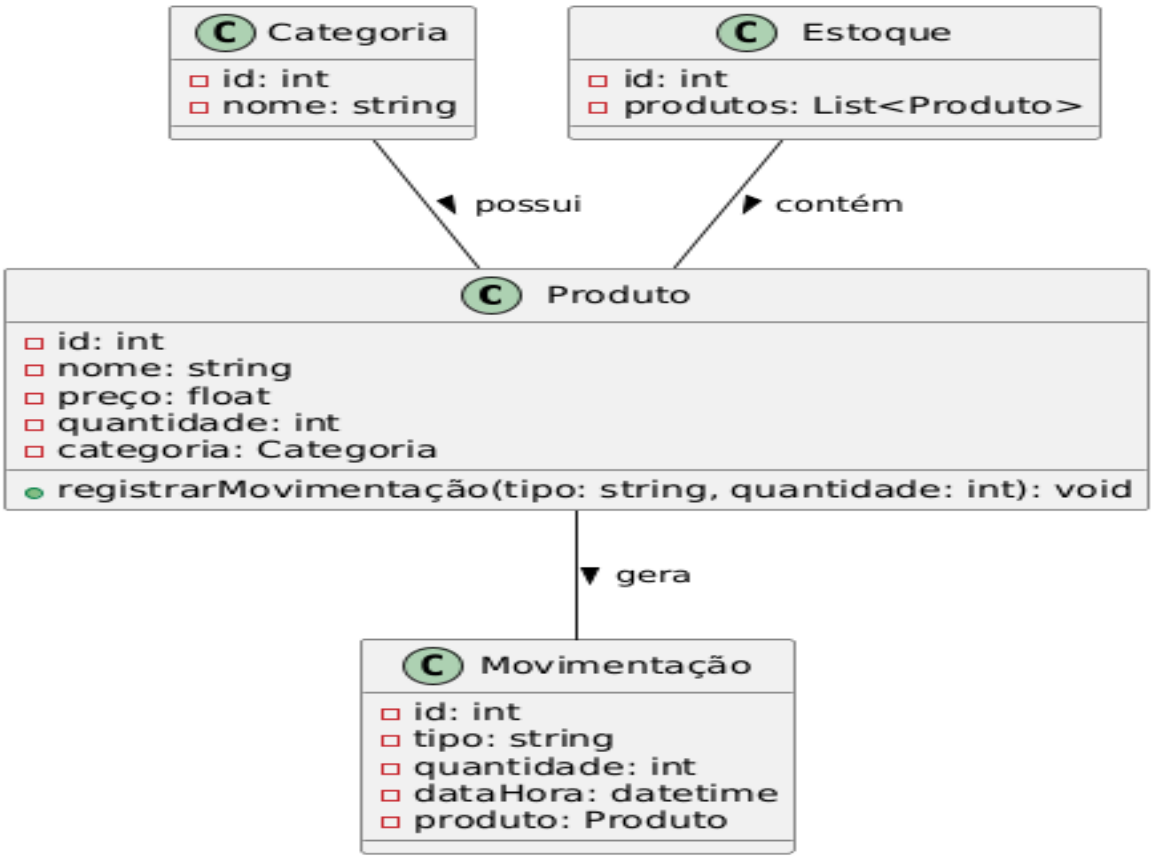
A modelagem orientada a objetos é uma prática essencial no processo de desenvolvimento de sistemas, pois permite compreender, representar e estruturar o comportamento e a organização dos elementos do software. nesta etapa, foram criados três diagramas UML fundamentais: o **Diagrama de Classe**, o **Diagrama de** Sequência e o **Diagrama de Estado**. cada um deles foi construído com base no levantamento de requisitos realizado anteriormente, com o objetivo de representar diferentes perspectivas do sistema de controle de estoque.

CLASSE	SEQUÊNCIA	ESTADO

class Produto {
- id: int
- nome: string
- preco: float
- quantidade: int
- categoria: Categoria
+ registrarMovimentacao(tipo: string, quantidade: int): void
}
 	
class Categoria {
- id: int
- nome: string
}
 	
class Movimentacao {
- id: int
- tipo: string
- quantidade: int
- dataHora: datetime
- produto: Produto
}
 	
class Estoque {
- id: int
- produtos: List<Produto>
}
	actor Usuario
	
	participant ProdutoController
	
	participant Produto
	
	participant Movimentacao
	
	Usuario -> ProdutoController : solicitarSaidaProduto(produtoId, quantidade)
	
	ProdutoController -> Produto : verificarEstoqueSuficiente(quantidade)
	
	ProdutoController -> Movimentacao : criarMovimentacao("Saída", quantidade)
	
	ProdutoController -> Produto : atualizarQuantidade(-quantidade)
	
	ProdutoController -> Usuario : confirmarOperação("Saída registrada")
		[*] --> Cadastrado
		
		Cadastrado --> Disponível : Entrada de estoque
		
		Disponível --> EstoqueBaixo : Quantidade < mínimo
		
		EstoqueBaixo --> Esgotado : Quantidade = 0
		
		EstoqueBaixo --> Disponível : Entrada de estoque
		
		Esgotado --> Disponível : Entrada de estoque
		
		Disponível --> Removido : Produto deletado
		
		EstoqueBaixo --> Removido : Produto deletado
		
		Esgotado --> Removido : Produto deletado

Categoria -- Produto : possui > Produto -- Movimentacao : gera > Estoque -- Produto : contém >		

6. DIAGRAMA DE CLASSE — ESTRUTURA ESTÁTICA DO SISTEMA



O diagrama de classe tem como principal finalidade representar a estrutura estática do sistema. Nele estão organizadas as principais **entidades (classes)**, seus **atributos**, **métodos** e os **relacionamentos** entre elas.

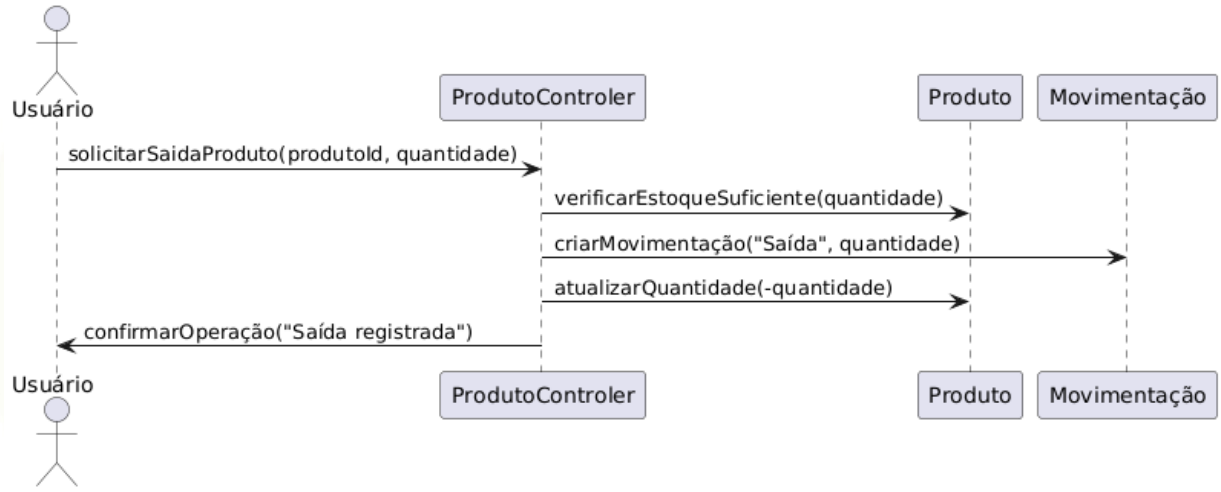
No sistema de controle de estoque, o núcleo está na classe **Produto**, que representa os itens armazenados. A classe contém atributos como id, nome, preço, quantidade e categoria. O método registrarMovimentacao() foi incluído para simbolizar a principal ação sobre o produto: a movimentação de entrada ou saída.

A classe **Categoria** representa a tipagem dos produtos, permitindo que cada produto pertença a uma determinada categoria (ex.: bebidas, limpeza, alimentos). O relacionamento entre Categoria e Produto é de **um para muitos**, pois uma categoria pode agrupar vários produtos.

A classe **Movimentacao** modela as alterações no estoque, seja por entrada ou saída. Cada movimentação registra o tipo, a quantidade, a data e hora da operação e o produto relacionado. O relacionamento entre Produto e Movimentacao é direto: um produto **gera** muitas movimentações.

Por fim, a classe **Estoque** aparece como uma estrutura de agregação lógica, agrupando os produtos do sistema. Ela pode futuramente ser usada como ponto de partida para relatórios ou estatísticas. O relacionamento com Produto é de **composição**, pois o estoque **contém** produtos, mas cada produto também é gerenciado de forma independente.

7. DIAGRAMA DE SEQUÊNCIA — INTERAÇÃO ENTRE OBJETOS



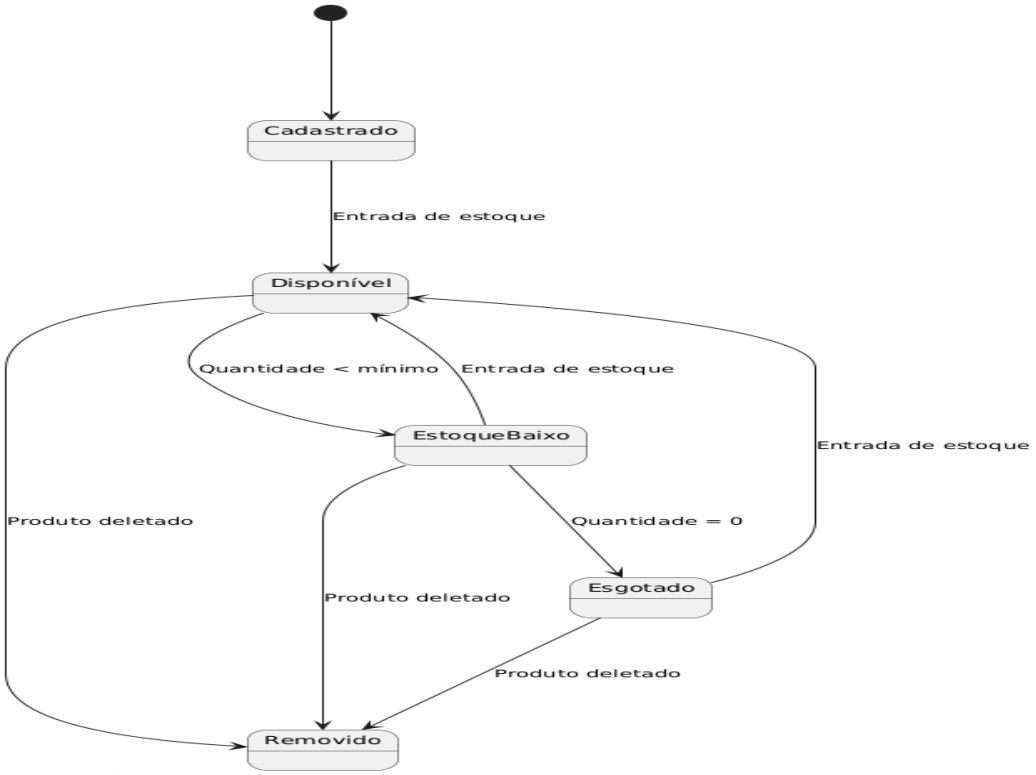
O diagrama de sequência descreve o fluxo de interação entre os objetos envolvidos em uma operação do sistema, focando na **ordem cronológica das mensagens trocadas**.

O caso de uso escolhido para representação foi o de **registro de saída de produto**, uma operação crítica em qualquer controle de estoque. Os objetos envolvidos são: **Usuário**, **ProdutoControler**, **Produto** e **Movimentação**.

A sequência inicia com o usuário solicitando uma saída, informando o **produtoId** e a quantidade. O **controler** verifica se há quantidade suficiente no estoque. Caso afirmativo, ele cria uma nova movimentação do tipo "Saída", atualiza a quantidade do produto e, por fim, retorna uma mensagem de confirmação para o usuário.

Este diagrama demonstra com clareza como os objetos do sistema se comunicam para executar uma operação, e também permite visualizar os métodos que precisam ser implementados no backend. Ele é útil tanto para desenvolvedores quanto para testadores que queiram compreender o comportamento do sistema em tempo de execução.

8. DIAGRAMA DE ESTADO — CICLO DE VIDA DO OBJETO PRODUTO



O diagrama de estado representa os **diferentes estados** que um objeto pode assumir ao longo do tempo e as **transições** entre esses estados, com base em eventos ou ações.

No contexto do sistema de controle de estoque, o objeto modelado foi o **Produto**. O ciclo de vida inicia-se no estado **Cadastrado**, e a primeira transição ocorre quando há **entrada de estoque**, levando o produto ao estado **Disponível**.

Caso a quantidade do produto caia abaixo de um valor mínimo definido, ele transita para o estado **Estoque Baixo**. Se a quantidade chegar a zero, o produto passa ao estado **Esgotado**. Em qualquer ponto, o produto pode ser excluído do sistema, assumindo o estado **Removido**.

O retorno ao estado **Disponível** ocorre sempre que há nova entrada de estoque. Essa lógica espelha um processo real de estoque físico e torna o sistema mais robusto ao antecipar e controlar mudanças de estado dos objetos.

Este diagrama ajuda a entender a dinâmica do sistema e foi utilizado como base para criar validações no backend que evitam inconsistências, como realizar saídas de produtos esgotados ou inativos.

??. REFERÊNCIAS