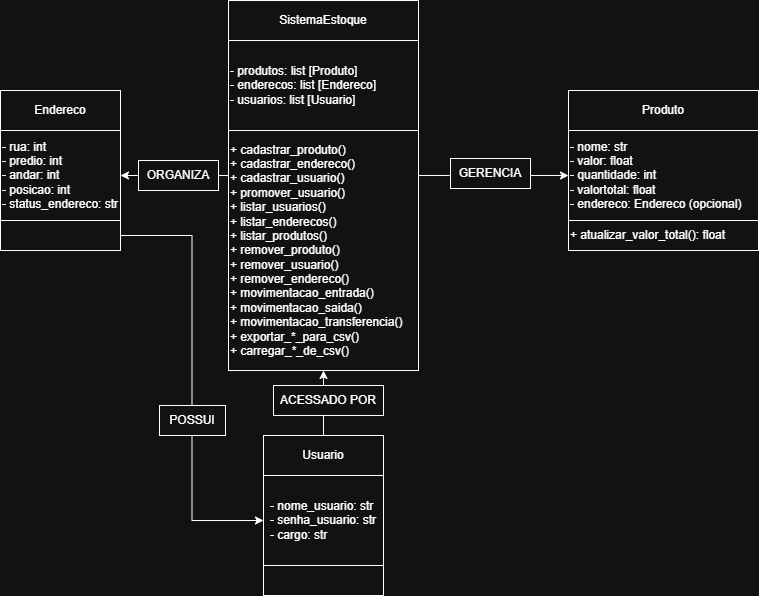
**Documentação Técnica**

**LOG WMS**

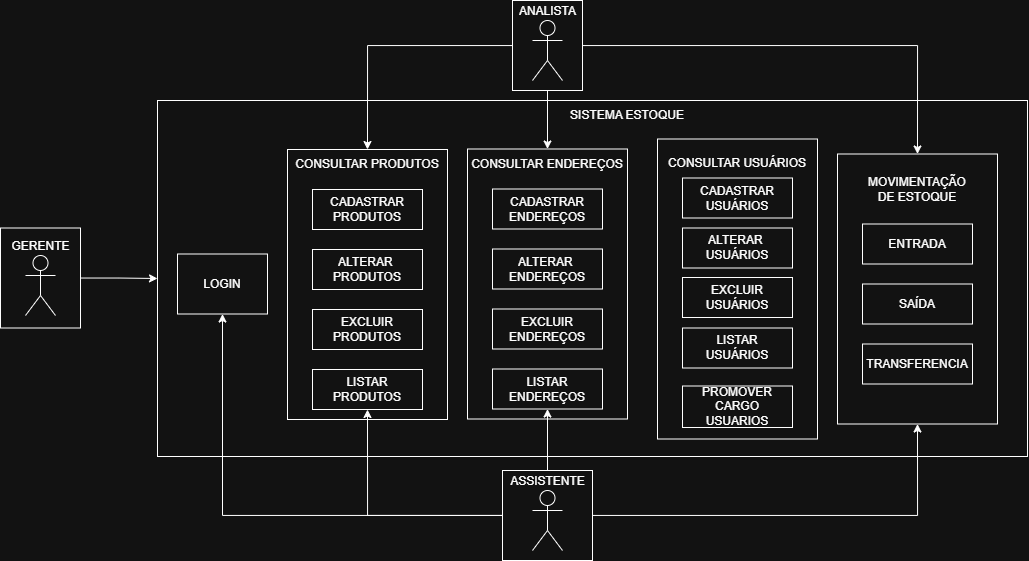
**Autor**: Fabrício Alves de Souza  
**Data**: 27/10/2025  
**Versão**: 1.0

## **1.DIAGRAMA DE CLASSES**



## 

## **CASOS DE USO**

****

### 2.1 Administrador

* **Gerenciar usuários**: criar, editar ou desativar contas.
* **Consultar relatórios**: produtos, estoque, transferências.
* **Configurar endereços e setores**: organizar o armazém.

### 2.2 Operador

* **Cadastrar produto**: adicionar novos produtos ao sistema.
* **Atualizar estoque**: registrar entrada, saída ou correção.
* **Transferir produto**: movimentar produtos entre endereços.
* **Consultar produto**: localizar produtos por nome, id ou endereço.

### 2.3 Sistema externo / Importador

* **Importar produtos**: carregar inventário via CSV.

## **3. ARQUITETURA DO PROJETO**

O projeto utiliza arquitetura em camadas:

* **Apresentação (UI)**: Interface Tkinter ou CLI para interação com usuário.
* **Serviços / Application Layer**: Regras de negócio e validação (ProdutoService, TransferenciaService, UsuarioService).
* **Domínio / Models**: Entidades principais (Produto, Usuario, Transferencia).
* **Persistência / Repository**: Isola detalhes de armazenamento em CSV ou JSON.

**Fluxo de transferência**: 1. Usuário solicita transferência. 2. TransferenciaService valida dados e permissões. 3. Atualiza quantidade no endereço de origem e destino. 4. Persiste alterações e registra log.

## **4. Bibliotecas Utilizadas**

| **Biblioteca** | **Versão** | **Função** |
| --- | --- | --- |
| Python | 3.11+ | Linguagem principal |
| tkinter | Nativa | Interface gráfica |
| csv / json / os | Nativa | Manipulação de arquivos |
| datetime | Nativa | Controle de datas |
| pandas (opcional) | 1.5+ | Relatórios avançados |
| pytest (opcional) | 7.0+ | Testes automatizados |

## **5. Estrutura de Pastas e Arquivos**

## **Commits Realizados**

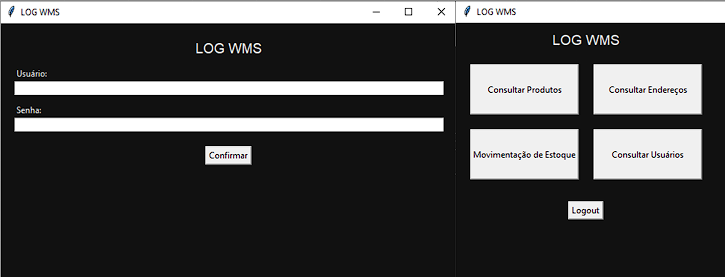
1. Colocar verificação na movimentação de entrada para checar se o endereço escolhido já possui o mesmo produto armazenado. (Antes, se fosse o mesmo produto, o sistema não permitia acrescentar.)
2. Criar método para promover e rebaixar usuários.
3. Adicionar verificação para impedir promoção/rebaixamento se o cargo selecionado for igual ao cargo atual.
4. Inserir verificação no método de remoção de endereço — caso o endereço possua saldo, o sistema bloqueia a exclusão.
5. Adicionar o atributo “tipo” à classe Produto para diferenciar categorias de produtos.

## **7. Instruções de Execução**

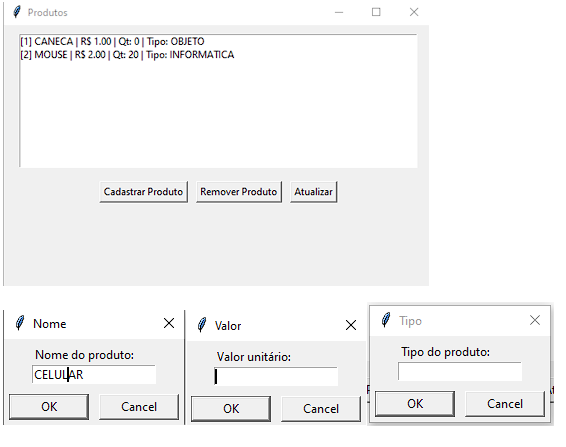
1. Baixe o projeto no GitHub.
2. Abra o terminal na pasta do projeto.
3. Execute o arquivo **log\_wms\_interface\_tkinter.py**
4. Caso seja a primeira execução, o sistema solicitará a criação de um **usuário administrador (gerente)**.
5. Faça login e acesse o menu de opções para interagir com o sistema.

## **Sistema em funcionamento**

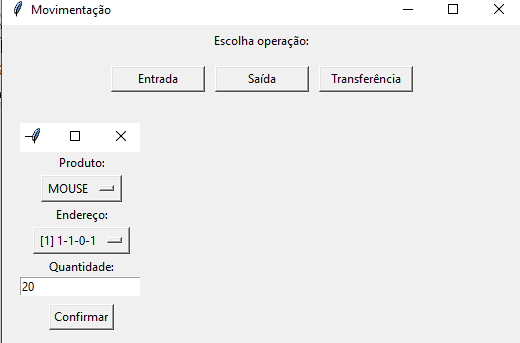
**Login e Menu**



**Consultar Produtos**



**Movimentação**



## **9. Referências**

* Documentação oficial do Python: https://docs.python.org/3/
* Conceitos de WMS (Warehouse Management System):  
  https://pt.wikipedia.org/wiki/Sistema\_de\_gerenciamento\_de\_armazenagem

**Fim da documentação técnica**