

Documento de Envoltória - Sistema de Gerenciamento de Estoque

Identificação do Grupo

- Vitor Alves Titus Eskes / RM555137
- Nathan Craveiro Gonçalves Amin / RM555508
- Gabriel Matias Simões / RM556171
- Miguel Carmo / RM557622

Explicação das Hipóteses e Dados Considerados

Este documento detalha as hipóteses de funcionamento, a estrutura dos dados, a lógica de negócio e a interface de uso do sistema de gerenciamento de estoque desenvolvido em Python.

Hipóteses de Funcionamento do Sistema

O sistema de gerenciamento de estoque opera sob as seguintes hipóteses:

- **Estrutura do Estoque:** O estoque é organizado hierarquicamente por categorias. Cada categoria contém uma coleção de itens, e cada item possui atributos como quantidade atual, estoque mínimo e estoque ideal. A estrutura de dados principal é um dicionário aninhado, onde as chaves de nível superior são os nomes das categorias e os valores são outros dicionários contendo os itens.
- **Persistência de Dados:** O sistema assume que os dados do estoque precisam ser persistidos entre as execuções. Para isso, ele utiliza arquivos JSON (`estoque.json`) para salvar e carregar o estado do estoque. A persistência é manual, ou seja, o usuário deve explicitamente escolher a opção 'Salvar e Sair' no menu principal para que as alterações sejam gravadas. Caso o programa seja encerrado de forma inesperada, as últimas alterações não salvas serão perdidas.
- **Entrada de Dados:** Todas as interações com o usuário são realizadas via entrada de texto no terminal. O sistema espera que o usuário digite números inteiros para quantidades, mínimos e ideais, e strings para nomes de categorias e itens. Há tratamento básico de erros para entradas não numéricas, mas não há validação

extensiva para outros tipos de entrada (por exemplo, caracteres especiais em nomes).

- **Uso do Terminal:** O sistema é projetado para ser executado em um ambiente de terminal. Ele utiliza comandos do sistema operacional (`os.system('cls' if os.name == 'nt' else 'clear')`) para limpar a tela, proporcionando uma interface de usuário mais limpa e interativa. Isso implica que o sistema é dependente do ambiente de execução (Windows ou sistemas baseados em Unix/Linux).
- **Nomes Únicos:** Dentro de uma mesma categoria, os nomes dos itens devem ser únicos. Da mesma forma, os nomes das categorias devem ser únicos. O sistema realiza uma verificação básica para evitar a adição de itens ou categorias com nomes já existentes.
- **Movimentação de Estoque:** A movimentação de itens (entrada e saída) é baseada na quantidade. O sistema permite adicionar ou subtrair quantidades do estoque de um item existente. Não há controle de transações ou histórico de movimentações, apenas a atualização da quantidade atual.
- **Alertas de Estoque:** O sistema define dois níveis de alerta: 'FALTA' (quantidade abaixo do mínimo) e 'EXCESSO' (quantidade acima do ideal). Estes limites são definidos pelo usuário no momento da criação ou edição do item.

Dados Considerados

O sistema manipula dados de estoque que são estruturados da seguinte forma:

- **Estrutura Principal do Estoque:** O estoque é representado por um dicionário Python. As chaves deste dicionário são os nomes das **categorias** (strings, convertidas para `Title Case` na entrada do usuário para padronização). Os valores associados a cada categoria são, por sua vez, outros dicionários.
- **Itens:** Dentro de cada dicionário de categoria, as chaves são os **nomes dos itens** (strings, também convertidas para `Title Case`). Os valores associados a cada item são dicionários contendo os detalhes específicos do item.
- **Campos Esperados para Cada Item:** Cada item possui os seguintes campos:
 - `quantidade` : (Inteiro) Representa a quantidade atual do item em estoque. É o valor que é alterado durante as movimentações de entrada e saída.
 - `minimo` : (Inteiro) Define o limite inferior de estoque para o item. Se a `quantidade` cair abaixo deste valor, o item é considerado

em estado de `FALTA`. * `ideal` : (Inteiro) Define o limite superior de estoque para o item. Se a `quantidade` exceder este valor, o item é considerado em estado de `EXCESSO`.

- **Tipos Esperados e Validações:**
- **Nomes (Categorias e Itens):** Espera-se strings. O sistema converte a primeira letra de cada palavra para maiúscula (`.title()`) para padronização. Não há validação para caracteres especiais ou comprimento.
- **Quantidades, Mínimos e Ideais:** Espera-se números inteiros. O sistema utiliza `int(input())` para converter a entrada do usuário. Um bloco `try-except ValueError` é usado para capturar entradas não numéricas, informando ao usuário sobre a entrada inválida e abortando a operação. Não há validação para números negativos ou zero, o que significa que um usuário pode inserir uma quantidade, mínimo ou ideal negativo ou zero, o que pode levar a comportamentos inesperados.
- **Categorias e Itens Existentes:** Antes de adicionar uma nova categoria ou item, o sistema verifica se um nome idêntico já existe. Se sim, impede a criação e informa o usuário.
- **Remoção:** Ao remover itens ou categorias, o sistema solicita uma confirmação do usuário para evitar exclusões acidentais.

Como a Lógica de Negócio é Organizada

A lógica de negócio do sistema é organizada em módulos (arquivos Python) para promover a modularidade e a separação de responsabilidades. Abaixo, detalhamos a função de cada arquivo:

- **`main.py`:**
- **Função:** Ponto de entrada principal da aplicação. Sua única responsabilidade é iniciar o menu do sistema.
- **Detalhes:** Importa a função `menu` do módulo `menu.py` e a executa quando o script é chamado diretamente (`if __name__ == "__main__":`).
- **`menu.py`:**
- **Função:** Responsável por exibir o menu principal da aplicação, interagir com o usuário para receber a opção desejada e direcionar a execução para as funções apropriadas em outros módulos.
 - **Detalhes:**
 - Carrega o estoque ao iniciar (`carregar_estoque()`).

- Apresenta um loop contínuo que exibe as opções do menu.
- Limpa a tela do terminal a cada iteração (`os.system()`).
- Chama funções de `estoque.py` para gerenciar itens e categorias, e movimentar o estoque.
- Chama funções de `relatorios.py` para exibir itens em falta ou excesso.
- Chama `salvar_estoque()` antes de sair.

- **estoque.py :**

- **Função:** Contém as funções principais para manipulação do estoque, como adicionar/remover itens, consultar o estoque completo e gerenciar categorias.

- **Detalhes:**
- `gerenciar_item(estoque)` : Permite adicionar novos itens a uma categoria existente ou remover itens. Inclui validação básica para entradas numéricas e verifica a existência de itens.
- `consultar_estoque(estoque)` : Exibe todos os itens no estoque, organizados por categoria, mostrando a quantidade e o status (FALTA, EXCESSO, OK) de cada item.
- `movimentar_item(estoque, tipo)` : Gerencia a entrada (`tipo="entrada"`) e saída (`tipo="saida"`) de itens, atualizando a quantidade em estoque. Realiza validação de entrada numérica e verifica se o item existe.
- `gerenciar_categoria(estoque)` : Permite criar novas categorias ou remover categorias existentes. Solicita confirmação antes de remover uma categoria.
- Faz uso de funções auxiliares de `utils.py` como `verificar_status` e `escolher_categoria`.

- **utils.py :**

- **Função:** Contém funções utilitárias que são usadas por outros módulos para evitar duplicação de código e promover a reutilização.

- **Detalhes:**
- `verificar_status(quantidade, minimo, ideal)` : Retorna o status de um item (

● FALTA , ● EXCESSO ou ● OK) com base na quantidade atual, mínima e ideal. * `escolher_categoria(estoque)` : Exibe uma lista numerada das categorias disponíveis no estoque e permite ao usuário selecionar

uma categoria pelo número. Retorna o nome da categoria escolhida ou `None` se a escolha for inválida ou não houver categorias.







- **relatorios.py :**
- **Função:** Responsável por gerar relatórios específicos sobre o estado do estoque.
 - **Detalhes:**
 - `itens_em_alerta(estoque, tipo="falta")` : Exibe uma lista de itens que estão em estado de `FALTA` (abaixo do mínimo) ou `EXCESSO` (acima do ideal), dependendo do parâmetro `tipo`. Utiliza a função `verificar_status` de `utils.py`.
- **persistencia.py :**
- **Função:** Lida com a leitura e escrita dos dados do estoque em um arquivo, garantindo que o estado do estoque seja salvo e carregado.
 - **Detalhes:**
 - `carregar_estoque()` : Tenta carregar o estoque de um arquivo JSON (`estoque.json`). Se o arquivo não existir ou houver um erro na leitura, retorna um dicionário vazio, inicializando o estoque.
 - `salvar_estoque(estoque)` : Salva o estado atual do dicionário `estoque` em um arquivo JSON (`estoque.json`).

Essa organização modular facilita a manutenção, o teste e a expansão do sistema, pois cada módulo tem uma responsabilidade bem definida.

Interface de Uso Esperada

O sistema de gerenciamento de estoque é projetado para ser utilizado através de uma interface de linha de comando (CLI) no terminal. A interação com o usuário segue um padrão de menu-driven, onde o usuário escolhe opções numeradas para realizar as ações desejadas.

- **Entrada do Usuário via Terminal:** Todas as entradas são feitas digitando texto no terminal e pressionando `Enter`. Isso inclui a seleção de opções de menu, nomes de categorias e itens, e quantidades.
- **Opções de Menu:** O menu principal é exibido de forma clara, com cada opção numerada e uma breve descrição da funcionalidade. Sub-menus são apresentados para ações que requerem escolhas adicionais (ex: gerenciar itens, gerenciar categorias).

- **Mensagens e Feedback:** O sistema fornece feedback constante ao usuário através de mensagens no terminal:
- **Confirmação de Sucesso:** Mensagens como "✅ Item 'X' adicionado..." ou "✅ Categoria 'Y' criada." indicam que uma operação foi concluída com êxito.
- **Alertas e Erros:** Mensagens como "❌ Opção inválida!", "❌ Item já existe..." ou "❌ Entrada inválida." informam o usuário sobre problemas ou entradas incorretas.
- **Status do Estoque:** Ao consultar o estoque ou verificar itens em alerta, o sistema exibe o status ( FALTA ,  EXCESSO ,  OK) de cada item.
- **Instruções:** Mensagens como "Pressione ENTER para voltar ao menu..." guiam o usuário sobre como prosseguir.
- **Limpeza de Tela:** A tela do terminal é limpa antes de exibir o menu principal ou após certas operações, garantindo que a interface permaneça organizada e fácil de ler.
- **Interatividade:** Embora seja uma interface de texto, o uso de emojis (como  ,  , ) e formatação básica (linhas de separação) tenta tornar a experiência um pouco mais amigável e visualmente distinta.

Em resumo, a interface é simples e direta, focada na funcionalidade e na clareza das informações apresentadas ao usuário via terminal.