Gerenciamento de Estoque para Restaurante Universitário

Alunos: Ayran de Abreu Silva Duarte e Filipe Schack

1) Descrição do Problema

1.1 Escopo do Software

O software a ser desenvolvido visa aprimorar o gerenciamento de estoque de um restaurante universitário. O restaurante universitário é responsável por fornecer refeições diárias para estudantes, professores e funcionários. O escopo do software inclui:

- Registro e monitoramento de itens de estoque, como alimentos, bebidas e almoço.
- Rastreamento das entradas e saídas de estoque.
- Permitir visualizar o estoque, para futuras reposições.

1.2 Stakeholders

Os principais stakeholders envolvidos no projeto são:

- Gestores do Restaurante Universitário: Responsáveis pelo gerenciamento do restaurante.
- Equipe de Cozinha e Atendentes: Usuários cotidianos do software para atualizar o estoque e fazer pedidos.
- Fornecedores: Podem precisar interagir com o sistema para atualizar os preços e disponibilidade de produtos.
- Departamento de Compras: Responsável por gerar ordens de compra com base nas necessidades do estoque.

- Contabilidade da Universidade: Para fins de controle financeiro e prestação de contas.

2) Requisitos do Software

Os requisitos do software podem ser divididos em requisitos funcionais e não funcionais.

2.1 Requisitos Funcionais

- Registro de todos os itens de estoque, incluindo nome, descrição, quantidade, preço, funcionário responsável pela compra e fornecedor
- Rastreamento de entradas de estoque, com informações sobre fornecedor, data, quantidade e custo.
- Possibilidade de adicionar, remover e visualizar itens do estoque.
- Geração de informações de transação, com base nas necessidades de reabastecimento.
- Os funcionários devem ser divididos entre: comuns e financeiro.
- Apenas o funcionário “financeiro” pode visualizar as informações de transação.
- Apenas o funcionário “comum” pode visualizar, adicionar e remover produtos.
- O sistema deve possibilitar visualizar todas informações de transação feitas.
- As informações de transação devem conter: data, funcionário responsável, quantidade, custo e custo total (custo * quantidade).

2.2 Requisitos Não Funcionais

- O sistema deve ser de fácil utilização, com uma interface amigável para a equipe do restaurante.
- Deve ser seguro, protegendo informações sensíveis de estoque e transações.
- Deve ser escalável para lidar com o crescimento futuro do restaurante.

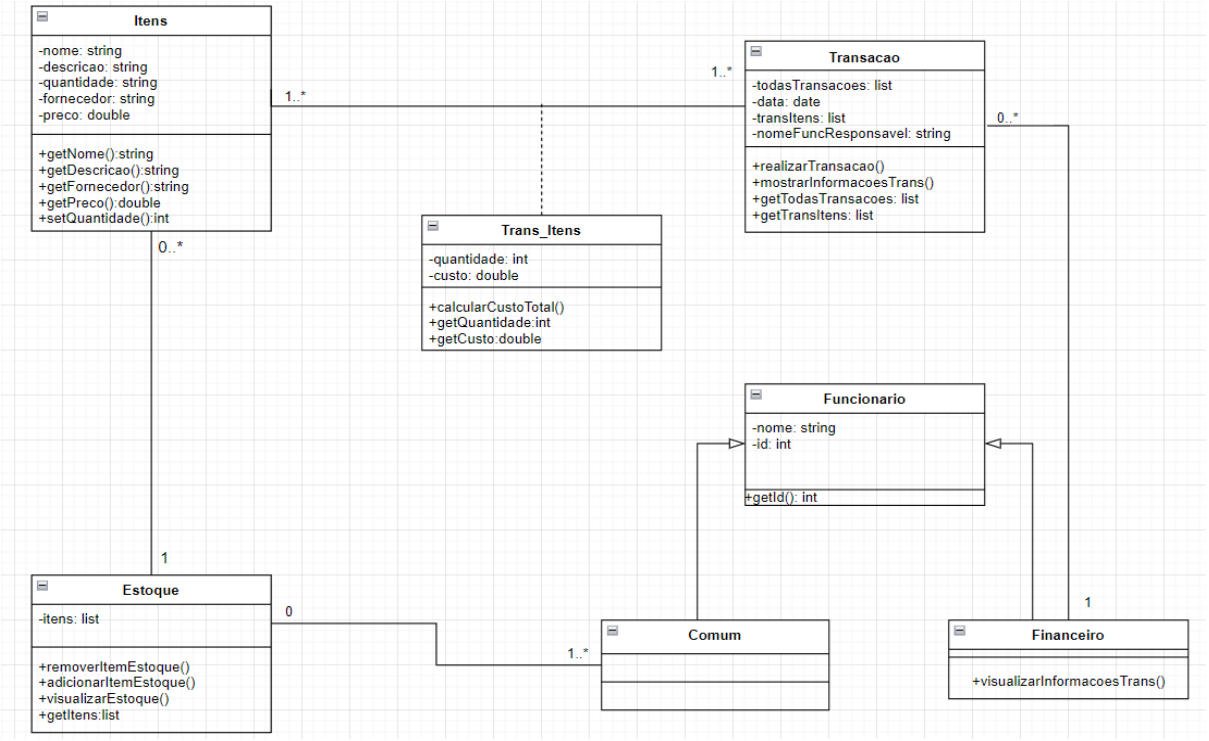
- Deve ser capaz de lidar com diferentes tipos de produtos.
- Deve ser eficiente em termos de desempenho para evitar atrasos na atualização de estoque.

3) Estimativa de duração do projeto completo (Cocomo)

- Está em um PDF separado, para melhor visualização.

4) Diagrama de Classes do Projeto UML

- Está em um PDF separado, para melhor visualização.



5) Testes Unitários

A seguir, explicações sobre cada teste feito no programa:

5.1 Classe EstoqueTest:

- **testAdicionarItemEstoque:** Teste feito com o intuito de verificar se o item foi adicionado corretamente, simulando uma possível entrada
- **testRemoverItemEstoque:** Teste feito com o intuito de verificar se o item foi removido corretamente, simulando uma possível entrada e após, a remoção do item adicionado

5.2 Classe FinanceiroTest:

- **testVisualizarInformacoesTrans:** Teste feito com o intuito de verificar se as saídas no console contém as informações esperadas sobre as informações de transação.

5.3 Classe ItemTest:

- **testItem:** Teste feito com o intuito de verificar se os métodos de acesso estão funcionando corretamente, simulando as entradas para criação de um item.

5.4 Classe TransItemTest:

- **testCalcularCustoTotal:** Teste feito com o intuito de verificar se o método “calcularCustoTotal” retorna o valor esperado.

5.4 TransacaoTest:

- **testMostrarInformacoesTrans:** Teste feito com o intuito de verificar se as informações de transações estam com as saídas corretas, adicionando um item fictício à transação e simulando a execução do método “mostrarInformacoesTrans”