

**Pró-Reitoria Acadêmica
Curso de Engenharia de Software
Trabalho de Disciplina**

Sistema de Gestão de Estoque

**Autor(es):Vitor Hugo Sena dos Reis, Pedro Eduardo DA COSTA
NASCIMENTO,**

Professor: Jefferson Salomão Rodrigues

Sumário

Resumo.....	3
Palavras-chave.....	3
1. Introdução.....	3
2. Objetivo.....	3
3. Metodologia.....	3
4. Solução Desenvolvida.....	4
4.1 Descrição do Sistema.....	4
4.2 Funcionalidades Principais.....	4
5. Tecnologias Utilizadas.....	5
6. Modelagem e Banco de Dados.....	5
6.1 Estrutura das Tabelas.....	5
6.2 Mecanismos Implementados.....	5
6.3 Controle de Acesso.....	6
6.4 Entidades e Relacionamentos.....	6
7. Conclusão.....	6
Referências Bibliográficas.....	7

Resumo

Este projeto apresenta um sistema simples de gerenciamento de estoque, desenvolvido em **Java com Python com o uso do framework Flask** e integrado a um banco **relacional (MySQL)**. O sistema permite o cadastro e controle de produtos, atualização automática de estoque e registro de auditorias. A solução prioriza clareza, organização e boas práticas.

Palavras-chave

Gestão de estoque, Flask, MySQL, sistema web, auditoria, Engenharia de Software

1. Introdução

A gestão eficiente de estoque é um dos fatores críticos para o sucesso operacional de lojas de pequeno e médio porte, como as de materiais de construção. Muitas vezes, tais ambientes contam com equipes reduzidas e usuários com pouca familiaridade tecnológica, o que exige soluções simples, intuitivas e robustas. O sistema proposto busca simplificar esse processo com uma aplicação web intuitiva e uma base de dados estruturada, garantindo integridade das informações e facilidade de consulta.

2. Objetivo

- Desenvolver um sistema de gerenciamento de estoque para lojas ou galpões, com interface gráfica e funcionalidades completas de entrada, saída e consulta de produtos.
- Utilizar um banco de dados relacional para persistência estruturada e um banco NoSQL para casos específicos que exigem flexibilidade ou alta escalabilidade.
- Modelar o banco de dados relacional com diagrama entidade-relacionamento (DER), definindo entidades, atributos, relacionamentos, índices, views, triggers e funções/procedures conforme boas práticas.
- Implementar controle de acesso baseado em perfis de usuários (administrador, operador, etc), definindo regras claras de autorização.

- Justificar as escolhas tecnológicas adotadas (frontend, backend, SGBD relacional e NoSQL) em termos de adequação ao cenário do usuário final.

- Documentar e analisar o sistema desenvolvido, suas funcionalidades, arquitetura, tecnologias empregadas e desafios enfrentados.

3. Metodologia

1. Levantamento de requisitos — Uma análise de modelo de negócio visando estoques em geral.

2. Definição da arquitetura — Escolha das camadas (frontend, backend, persistência) e tecnologias baseadas no contexto escolar e de implementação.

3. Modelagem de dados — Criação do DER para o banco relacional, definição de entidades, atributos, relacionamentos.

4. Implementação — Desenvolvimento do frontend (interface), backend (API lógica de negócio) e integração com os bancos de dados (relacional e NoSQL).

5. Validação — Testes de funcionalidade, controle de acesso, fluxo típico de operação da loja.

6. Documentação — Elaboração deste relatório com descrições de sistema, justificativas tecnológicas, modelagem, controle de acesso, uso do banco NoSQL e conclusão.

4. Solução Desenvolvida

4.1 Descrição do Sistema

O sistema “Gerenciamento de Estoque” para loja de materiais de construção permite aos usuários gerenciar produtos, controlar entradas e saídas de estoque, visualizar relatórios simples e administrar usuários/perfis.

- Login de usuários com diferentes níveis de acesso.
- Cadastro, edição e exclusão de produtos.

- Registro de movimentações de entrada e saída de estoque.
- Atualização automática do saldo de produtos.

4.2 Funcionalidades Principais

- **Autenticação de usuários** e roteamento condicional conforme perfil (administrador, operador).
- **Cadastro de produtos** (nome, descrição, categoria, quantidade, preço de compra, preço de venda).
- **Registro de entrada de produtos** (data, quantidade, fornecedor) e saída de produtos (data, quantidade, motivo).
- **Consulta e listagem de estoque atual**, com filtros por categoria, faixa de quantidade.
- **Alerta** ou indicador para produtos com quantidade abaixo de determinado limiar.
- **Histórico de movimentações de estoque** (entadas/saídas) com data, usuário, quantidade.
- **Gerenciamento de usuários/perfis** e controle de acesso conforme nível de perfil.
- (Opcional) Relatório simples de vendas ou movimentações para análise rápida.
- **Relatórios** simplificados por meio de *views* SQL.

5. Tecnologias Utilizadas

Camada	Tecnologia	Justificativa
Frontend	React + CSS	Simples, responsivo

Backend	TypeScript Python.	+	Framework robusto e padronizado para aplicações web.
Banco Relacional	MySQL		Ideal para dados estruturados e relacionamentos.

6. Modelagem e Banco de Dados

6.1 Estrutura das Tabelas

Tabela: produtos

Descrição:

Armazena as informações de todos os produtos cadastrados no estoque. É a tabela-base para qualquer movimentação ou requisição.

Campos principais:

- **produto_id** – Identificador único do produto (chave primária).
- **nome** – Nome do produto. Possui restrição de unicidade para evitar duplicidade.
- **preco** – Valor unitário do produto.
- **estoque** – Quantidade atual disponível do produto.
- **categoria** – Classificação do produto (ex.: elétrico, hidráulico, pintura).

Função no sistema:

Servir como referência principal para entradas, saídas e requisições de produtos. Todas as movimentações dependem desta tabela.

Tabela: requisicoes_estoque

Descrição:

Armazena todas as solicitações feitas por funcionários para retirada de produtos do estoque.

Campos principais:

- **id_requisicao** – Identificador único da requisição (chave primária).
- **id_produto** – Referência ao produto solicitado (chave estrangeira para *produtos.product_id*).
- **quantidade_requisitada** – Quantidade solicitada pelo usuário.
- **data_requisicao** – Data e hora da solicitação, gerada automaticamente.
- **status** – Situação da requisição (ex.: "PENDENTE", "APROVADA", "NEGADA", "CONCLUIDA").

Função no sistema:

Controlar o fluxo de pedidos de retirada de produtos, permitindo acompanhar e gerenciar o status de cada solicitação.

Tabela: requisicoes_concluidas

Descrição:

Registra as requisições finalizadas, funcionando como um log histórico das movimentações realizadas no estoque.

Campos principais:

- **id_conclusao** – Identificador único da conclusão.
- **id_requisicao** – Referência para a requisição original (*requisicoes_estoque.id_requisicao*).
- **id_produto** – Produto movimentado (chave estrangeira para *produtos.product_id*).
- **quantidade_requisitada** – Quantidade retirada na conclusão.
- **data_conclusao** – Data e hora em que a requisição foi concluída.
- **concluida** – Campo booleano indicando se a requisição foi concluída (padrão: TRUE).

Função no sistema:

Registrar historicamente todas as requisições finalizadas, garantindo transparência, auditoria, rastreabilidade e controle das saídas do estoque.

Relacionamentos entre as Tabelas

- **produtos 1 —— N requisicoes_estoque**
- Um produto pode ser requisitado diversas vezes.
- Cada requisição está vinculada a apenas um **produto**.

- requisicoes_estoque 1 —— 1 requisicoes_concluidas

(não obrigatório, mas esperado quando concluída)

- Cada requisição pode ter **uma conclusão** associada.
- A conclusão registra o momento em que aquela requisição foi finalizada.
- produtos 1 —— N requisicoes_concluidas
- Um produto pode aparecer diversas vezes no histórico de conclusões.

6.2 Mecanismos Implementados

Justificativa para uso de índices, triggers, views, procedures/functions

- **Índices:** Criação de índices no campo ‘nome’ da entidade Produto para acelerar buscas e filtros; índice em Movimentação.dataMovimento para relatórios de movimentações numa faixa de datas.
- **Triggers:** Por exemplo, trigger após inserção em Movimentação-saída para reduzir automaticamente a quantidadeAtual de Produto e verificar limiar para emitir aviso.
- **Views:** Uma view “vw_EstoqueBaixo” que mostra todos os produtos com quantidadeAtual abaixo de limiarAviso — facilita o relatório para operador sem ter que escrever a junção manualmente.
- **Procedures/Functions:** Uma função “fn_CalculaQuantidadeAtual(productId)” que retorna a quantidade atual com base em soma de entradas e saídas; ou procedure “proc_RegistrarMovimentacao” que encapsula toda a lógica de entrada/saída, atualização de produto e auditoria.

6.3 Controle de Acesso

- **Operador:** pode consultar e registrar movimentações.
- **Gerente:** pode editar e inserir produtos.
- **Admin:** tem acesso completo ao sistema e banco.

Regras de acesso

- Somente administrador pode cadastrar usuários/perfis e alterar configurações do sistema.
- Operador pode apenas consultar relatórios, cadastrar entrada/saída e buscar produtos; não pode apagar produtos, categorias ou movimentações de outros usuários.
- Interface de login redireciona conforme perfil; sessões e tokens de autenticação implementados para manter o controle de acesso.
- Todas as ações críticas (como movimentações, deletar produto) são logadas com usuário, data/hora, tipo de ação para auditoria.

6.4 Entidades e Relacionamentos

- Cada **usuário** pertence a um **grupo de acesso**, que define suas permissões dentro do sistema.
- Os **produtos** são cadastrados com informações básicas (nome, categoria, preço, estoque atual).
- A tabela **movimentos** registra as **entradas** e **saídas** de produtos, vinculando cada operação ao usuário que a realizou.
- Essa estrutura garante integridade referencial e facilita a auditoria das ações.

7. Conclusão

O sistema cumpre seu objetivo de forma simples e eficaz: oferece controle de estoque confiável, com segurança de dados e histórico de operações. O uso conjunto de **flask** e **MySQL** resultou em uma solução moderna, modular e escalável.

Referências Bibliográficas

JOEL. **Triggers no SQL Server: teoria e prática aplicada em uma situação real.** DevMedia, 2013. Disponível em:

<https://www.devmedia.com.br/triggers-no-sql-server-teoria-e-pratica-aplicada-em-uma-situacao-real/28194>. Acesso em: 13 nov. 2025.

LUCSSLUCSSS. **Seu primeiro CRUD em Java com Spring.** Medium, 30 jan. 2024. Disponível em: <https://medium.com/@lucsslucsss/seu-primeiro-crud-em-java-com-spring-0c1dfd476a70>. Acesso em: 13 nov. 2025.

MICROSOFT. **CREATE TRIGGER (Transact-SQL).** Microsoft Learn, 2025. Disponível em: <https://learn.microsoft.com/pt-br/sql/t-sql/statements/create-trigger-transact-sql?view=sql-server-ver17>. Acesso em: 13 nov. 2025.

ORACLE. **MySQL 8.0 Reference Manual.** Redwood Shores: Oracle, 2025. Disponível em: <https://dev.mysql.com/doc/refman/8.0/en/>. Acesso em: 13 nov. 2025.