UNIVERSIDADE DE BRASÍLIA

FACULDADE DE ECONOMIA, ADMINISTRAÇÃO, CONTABILIDADE E GESTÃO DE POLÍTICAS PÚBLICAS - FACE

DEPARTAMENTO DE ADMINISTRAÇÃO

JULIANO POPIOLEK SASSE

Um sistema de gestão de estoque com inteligência artificial

BRASÍLIA - DF

2025

JULIANO POPIOLEK SASSE

Um sistema de gestão de estoque com inteligência artificial

Monografia apresentada ao Departamento de Administração como requisito parcial à obtenção do título de Bacharel em Administração.

Professor Orientador: Professor Dr., Victor Rafael Rezende Celestino.

BRASÍLIA – DF

2025

JULIANO POPIOLEK SASSE

Um sistema de gestão de estoque com inteligência artificial

A Comissão Examinadora, abaixo identificada, aprova o Trabalho de Conclusão do Curso de Administração da Universidade de Brasília do (a) aluno (a)

Juliano Popiolek Sasse

Professor Dr. Victor Rafael Rezende Celestino
Professor-Orientador

Profa. Dr.ª Silvia Araújo dos Reis,
Professor-Examinador

Prof. Dr. João Gabriel de Moraes Souza,
Professor-Examinador

Brasília, 07 de julho de 2025.

RESUMO

A gestão eficiente de estoques é um dos principais desafios para pequenas e médias empresas, dado que grande parte do capital imobilizado está concentrado nesse setor e decisões inadequadas podem comprometer toda a operação. Este trabalho apresenta o desenvolvimento de um sistema de gestão de estoque baseado em inteligência artificial, com arquitetura modular, voltado para automatizar o registro, análise e otimização dos estoques de forma acessível. O sistema utiliza o Google Sheets como base de dados central, integrando-se via API Python para garantir atualização em tempo real e colaboração. As rotinas analíticas combinam a previsão de demanda com Prophet, algoritmos de otimização com Gurobi, geração de dashboards interativos e relatórios automatizados. Para potencializar diagnósticos e recomendações, o sistema incorpora OCR para extração de dados de imagens de relatórios, além de fluxos inteligentes por meio do *framework LangChain*. O resultado é uma solução flexível, escalável e de baixo custo, capaz de entregar *insights* acionáveis para a tomada de decisão, democratizando o acesso à inteligência artificial aplicada à gestão de estoques.

Palavras-chave: Gestão de Estoque; Inteligência Artificial; Otimização; Gurobi; Previsão de Demanda; OCR; LangChain; Automação; Pequenas Empresas**.**

ABSTRACT

Efficient inventory management is a key challenge for small and medium-sized enterprises, as much of their fixed capital is allocated to stock and poor decisions can jeopardize operations. This study presents the development of an artificial intelligence-based inventory management system with a modular architecture, designed to automate registration, analysis, and inventory optimization in an accessible way. The system uses Google Sheets as its central database, integrating via Python API for real-time updates and collaboration. Analytical routines combine demand forecasting with Prophet, optimization algorithms with Gurobi, interactive dashboards, and automated reports. To enhance diagnostics and recommendations, the system incorporates OCR for extracting data from report images and intelligent flows through the LangChain framework. The result is a flexible, scalable, and low-cost solution capable of delivering actionable insights for decision-making, democratizing access to artificial intelligence in inventory management.

Keywords: *Inventory Management; Artificial Intelligence; Optimization; Gurobi; Demand Forecasting; OCR; LangChain; Automation; SMEs..*

LISTA DE TABELAS

Tabela 1 - comparativa modelo determinístico e estocástico 17

Tabela 2 - Tipos de Artefatos 30

Tabela 3 - Exemplo Banco de Dados 37

Tabela 4 - Saída estoque 41

Tabela 5 - Saldo estoque 42

Tabela 6 - Resumo rápido do saldo por produto 43

Tabela 7 - Itens com estoque abaixo de 20 unidades 59

LISTA IMAGENS

Imagem 1 - Dashboard Interativo - Estoque Filtrado 45

Imagem 2 - Tabela de Saldo Atual 45

Imagem 3 - Estoque completo 46

Imagem 4 - Estoque Consolidado Atual 46

Imagem 5 - resultado teste 1 51

Imagem 6 - Resultado teste 2 52

Imagem 7 - resultado teste 3 53

Imagem 8 - resultado teste 5, saldo atual 56

Imagem 9 - resultado teste 5, movimentação diária 57

Imagem 10 - resultado teste 5, movimentação último semestre 57

Imagem 11 - resultado teste 6 58

LISTA DE ABREVIATURAS E SIGLAS

DS - *Design Science*

DSR – *Design Science Research*

EOQ - *Economic Order Quantity (*Quantidade económica do pedido*)*

ERP - *Enterprise Resource Planning*

GANs - Redes Generativas Adversariais

IA - Inteligência Artificial

IoT - *Internet of Things*

JIT - *Just In Time*

LLM - *Large Language Model* (Modelo de linguagem grande porte)

ML - *Machine Learning*

OCR - *Optical Character Recognition*

PMEs - Pequenas e médias empresas

VAE - *Variational Autoencoders*

SUMÁRIO

1 INTRODUÇÃO..... 11

1.1 Contextualização..... 11

1.2 Formulação Do Problema..... 12

1.3 Objetivo Geral..... 12

1.4 Objetivos Específicos..... 12

1.5 Justificativa..... 13

2 REFERENCIAL TEÓRICO..... 15

2.1 Modelos De Gestão De Estoque..... 15

2.1.1 Definições de gestão de estoque..... 15

2.1.2 Modelo determinístico e estocástico..... 16

2.1.3 Gurobi - Otimização de estoque..... 16

2.2 Inteligência Artificial..... 20

2.2.1 Modelos de Inteligência Artificial..... 20

2.2.2 IA Generativa..... 23

2.2.3 OCR..... 24

2.2.4 LangChain..... 24

2.3 Ia e Estoque..... 25

2.4 Open source e democratização do acesso tecnológico..... 26

3 METODOLOGIA..... 28

3.1 Tipologia e descrição geral dos métodos de pesquisa..... 28

3.2 Caracterização e descrição dos instrumentos de pesquisa..... 29

3.3 Caracterização da organização pesquisada..... 32

3.4 Procedimentos de coleta de dados..... 32

3.5 Análise de dados..... 32

4 RESULTADO - VISUALIZADOR ESTOQUE..... 34

4.1 Caminho inicial e primeiros testes..... 34

4.2 Banco de dados.....36
4.3 Bloco 1: instalações, imports e configurações iniciais.....37
4.4 Bloco 2: integração com google sheets.....39
4.5 Bloco 3: dashboards, visualizações e exportação..... 42
4.6 Bloco 4: backend, automação de análises e endpoints web.....46
4.7 Bloco 5: ocr, ia para análise automática dos dashboards e publicação de apis com ngrok.....48
4.8 Bloco 6: testes..... 50
4.9 Respostas llm a respeito do estoque.....60
4.10 funcionalidades entregues:..... 61
5 RESULTADO - OTIMIZAÇÃO ESTOQUE.....63
5.1 - Conexão Google Drive e leitura do arquivo..... 63
5.2 - Pré-processamento de dado..... 64
5.3 - Salva tabela de demanda no drive.....64
5.4 - Instalação e licença Gurobi.....65
5.5 - Preparação dos dados.....65
5.6 - Resolução do modelo com variável Y binária e quantidade Q.....66
5.7 - Solução e relatório resumido..... 66
5.8 - Funcionalidades entregues.....67
6 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS..... 69
6.1 Objetivos atendidos da pesquisa.....69
6.2 Limitações da pesquisa..... 70
6.3 Sugestões de trabalhos futuros.....71
REFERÊNCIAS.....73
APÊNDICES.....78
Apêndice 1 - Bloco 1: instalações, imports e configurações iniciais..... 78
Apêndice 2 - Bloco 2: funções de integração com o google sheets..... 81
Apêndice 3 - Bloco 3: funções dashboard, filtros interativos, gráficos e funções auxiliares..... 83
Apêndice 4 - Bloco 4: backend flask com dashboard e ia/ocr.....95

Apêndice 5 - Bloco 5: ocr+ia para análise automática dos dashboards e publicação do link ngrok para testes e subir apis.....101

Apêndice 6 - Bloco 6: testes mostrando a funcionalidade do código.....104

Apêndice 7 - Conexão Google Drive e leitura do arquivo..... 107

Apêndice 8 - Pré-processamento de dado..... 108

Apêndice 9 - Salva tabela de demanda no drive.....109

Apêndice 10 - Instalação e licença Gurobi.....110

Apêndice 11 - Preparação dos dados.....111

Apêndice 12 - Resolução do modelo com variável Y binária e quantidade Q.....113

Apêndice 13 - Solução e relatório resumido.....115

1 INTRODUÇÃO

1.1 Contextualização

A gestão de estoques é objeto de estudo há muito tempo. Sua primeira formulação matemática conhecida, a fórmula do lote econômico, foi desenvolvida por F. W. Harris em 1915 (*apud* Santoro; Miguel, 2007). Durante as revoluções industriais, e especialmente nas duas grandes guerras mundiais, esses estudos ganharam impulso. O motivo foi a intensificação da produção em massa e a necessidade de distribuição eficiente. Nesse contexto, surgiram métodos para aprimorar a lucratividade e o controle dos inventários, com o objetivo de atender à crescente demanda global (Ballou, 2015).

No século XX, os avanços na administração e tecnologia transformaram a gestão de estoques em uma área de estudo formal. Marcados por modelos como o JIT (*just in time*), (Ohno, T. 1988), desenvolvido no Japão pela Toyota, revolucionaram o setor ao enfatizar a redução de estoques por meio de uma reposição eficiente e sincronizada com a demanda. Com o advento dos estudos, modelos computacionais começaram a aparecer para auxiliar em tais processos. Em sua evolução, deram origem à Inteligência Artificial (IA), levando a gestão de estoques para uma próxima fase, marcada por sua automação e eficiência.

A gestão do estoque desempenha um papel fundamental dentro das organizações, uma vez que influencia diretamente a eficiência e a lucratividade, um estoque gerido de forma errônea pode ocasionar um alto custo de armazenagem ou, até mesmo, a perda de clientes e de vendas devido à falta do material. Empresas de pequeno e médio porte, no geral, utilizam-se de planilhas e métodos tradicionais para realizar tal análise, (Bowersox; Closs; Cooper, 2014), contudo esses métodos corriqueiramente falham, usualmente devido a erro humano.

Nesse ínterim, as IAs vieram para auxiliar em sua gestão, tanto para agilizar processos quanto para a sua manutenção. Podemos utilizar ferramentas, como a IA generativa, para analisar grandes consumos de dados históricos e analisar padrões de consumos sazonais (Russell; Norvig,2021). Com base nessa análise, é possível direcionar os esforços de forma mais estratégica, determinando quais clientes ou mercados demandam mais atenção. Consequentemente, a IA não apenas viabiliza a redução dos custos de estocagem, mas também serve como um poderoso suporte para as equipes de compras e comercial.

O desenvolvimento de uma IA para gestão de estoque, como proposto neste trabalho, irá beneficiar pequenas e médias empresas que buscam soluções simples, possibilitando o aumento de sua competitividade no mercado.

1.2 Formulação Do Problema

A administração dos estoques constitui um elemento central para a manutenção da eficiência operacional e da competitividade empresarial, sendo determinante em empresas cuja atividade depende diretamente da disponibilidade de produtos (Ballou, 2015). Em pequenas e médias empresas (PMEs), são utilizados, em grande parte, controles manuais e ferramentas simplificadas, especialmente pelo elevado custo de sistemas robustos e falta de processos no planejamento do estoque (Bowersox; Closs; Cooper, 2014). Tais condições frequentemente ocasionam desequilíbrios, seja pelo acúmulo de estoques, seja pela falta de produtos, o que repercute negativamente sobre a lucratividade e o nível de serviço ao cliente (Chopra; Meindl, 2011).

O mercado revela-se mais dinâmico, marcado por mudanças bruscas, rápidas e pela sazonalidade (Mittal, 2024). Embora grandes corporações contem com sistemas informatizados potentes, observa-se que, entre as PMEs, persistem barreiras à adoção de tecnologias avançadas, decorrentes, além das restrições orçamentárias, da limitada capacidade técnica dos colaboradores e gestores (Bezerra de Menezes, 2007).

Assim, surge a questão central deste trabalho: como desenvolver um sistema com inteligência artificial voltado para estoques que permita prever a demanda e otimizar o controle do inventário, de forma eficiente e acessível? Sua resposta implica não somente utilizar as tecnologias disponíveis, mas, além disso, adotar soluções práticas e uma interface de fácil usabilidade, possibilitando a sua adoção.

1.3 Objetivo Geral

O objetivo deste trabalho é desenvolver um sistema voltado para gestão de estoque com inteligência artificial generativa, sendo este capaz de prever a demanda de produtos com base nos dados históricos, possibilitar a análise humana dos itens com maior e menor saída, gerar relatórios e *dashboards* a respeito do estoque e, por fim, criar *insights* por meio da IA.

1.4 Objetivos Específicos

Pretende-se alcançar os seguintes objetivos:

- Coletar dados do histórico de ao menos quatro meses da empresa amostral;
- Modelar o diagrama de fluxo do usuário na solução;
- Projetar e implementar um sistema *web* com ferramenta *No Code*, integrando IA generativa;
- Testar e avaliar a plataforma.

1.5 Justificativa

A gestão de estoque representa um pilar estratégico que pode definir a rentabilidade de uma empresa. A eficiência nessa área é vital, uma vez que um estoque bem gerenciado minimiza o capital imobilizado e mitiga os riscos financeiros associados à perda pelo excesso ou pela falta de determinados produtos. Muitas organizações de médio e pequeno porte possuem dificuldades nessa gestão, porque se utilizam de métodos manuais suscetíveis a erros de análise humana e problemas de abastecimento.

O advento da era da informação, caracterizado pela ampla disponibilidade de dados, posiciona a Inteligência Artificial como uma ferramenta poderosa para solucionar esse desafio. Análise de padrões de consumo e automação de processos podem ajudar a otimizar os estoques e melhorar a eficiência operacional. Contudo, essa tecnologia ainda possui um elevado custo, sendo acessível apenas para grandes empresas, o que gera uma lacuna que precisa ser preenchida (Chopra; Meindl, 2011).

Este trabalho busca uma solução de fácil acesso para pequenas e médias empresas, com o desenvolvimento de um sistema que mescla funcionalidades essenciais de gestão de inventário com recursos de Inteligência Artificial. Para garantir a viabilidade e facilitar a adoção do sistema, o projeto prioriza o uso de tecnologias gratuitas.

No âmbito social, a adoção de um sistema para gestão de estoque de baixo custo em PMEs pode contribuir para a sustentabilidade e para o sucesso do negócio, questões fundamentais para geração de emprego e renda dentro de uma sociedade. O fortalecimento da gestão dentro dessas empresas auxilia em seu crescimento sustentável, impactando a comunidade empresarial e seus colaboradores de forma positiva.

Dentro da perspectiva tecnológica, o avanço da Inteligência Artificial amplia a inovação em diversos setores, dentre os quais encontram-se os estoques, trazendo um grande poder de análise de dados, de identificação de padrões de consumo e de auxílio nas decisões de seus gestores. Apesar disso, a aplicação prática dessa tecnologia ainda é restrita, principalmente em razão de seu elevado custo de implementação, sendo mais comum em

grandes corporações com maior capacidade de investimento (Chopra; Meindl, 2011). Dessa forma, surge uma lacuna tecnológica: as PMEs que, em função das limitações de recursos e conhecimento técnico, permanecem à margem dessa revolução digital.

Outrossim, no quesito acadêmico, a pesquisa e o desenvolvimento de ferramentas acessíveis que aliem fundamentos clássicos da administração de estoques a recursos avançados de inteligência artificial têm potencial para enriquecer o debate e ampliar o campo de aplicação do conhecimento científico na área de gestão. O desenvolvimento de soluções que equilibrem robustez técnica, custo reduzido e facilidade de uso atende a uma demanda real do mercado e aproxima a produção acadêmica da realidade empresarial das PMEs.

Nesse sentido, este projeto busca democratizar o acesso de boas ferramentas para a gestão de estoque, oferecendo às empresas menores a possibilidade de conseguir um estoque organizado e eficiente.

2 REFERENCIAL TEÓRICO

2.1 Modelos De Gestão De Estoque

2.1.1 Definições de gestão de estoque

A gestão de estoques é, basicamente, o conjunto de práticas, processos e estratégias empregadas no planejamento, organização e controle dos produtos de uma empresa, ou seja, os insumos e produtos finais dela.

Segundo Ballou (2015), a administração de estoques deve maximizar o fluxo de materiais na cadeia de suprimentos, o que viabiliza alta disponibilidade de produtos ao menor custo possível. Tal atividade trata de tomadas de decisões sobre a quantidade ideal a ser mantida em estoques, periodicidades de reposição e sobre como organizar fisicamente o produto. Essa estratégia deve equilibrar dois objetivos frequentemente conflitantes: garantir estoques suficientes para evitar faltas e, ao mesmo tempo, minimizar o capital investido e os custos de operação.

Há também modelos matemáticos como o EOQ (*Economic Order Quantity*), desenvolvido por Harris (1915, *apud* Erlenkotter, 2003), que servem para determinar a melhor quantidade de pedidos para estoque, levando em consideração os custos de aquisição, de manutenção dos estoques e de faltas de produtos. Nesse sentido, hoje, temos ferramentas, como Gurobi, que servem para solucionar problemas matemáticos, facilitando a resolução lógica de modelos como o EOQ, tal ferramenta será explicada mais adiante.

Entretanto, a gestão de estoques não se limita apenas ao controle de itens, uma vez que envolve, também, aspectos estratégicos da organização, à medida que impacta diretamente áreas como produção, vendas e financeiro. Uma gestão inadequada pode resultar em perdas financeiras significativas ou no acúmulo de estoques obsoletos. Chopra e Meindl (2011) chamam a atenção para a relevante importância da gestão de estoques para a otimização da cadeia de suprimentos. Uma gestão eficiente pode resultar em reduções de custos, melhorias do nível de serviços e aumento da competitividade da empresa. Uma gestão de estoques efetiva é aquela que resulta da integração de várias áreas da empresa, tais como: compras, produção, logística, comercial e financeira.

2.1.2 Modelo determinístico e estocástico

Existem diversos modelos de estoques, não obstante, grande parte da escolha desses modelos parte da divisão de duas frentes de roteirização, a modelagem determinística ou estocástica.

No modelo determinístico (Cauchick, 2018), a demanda por um produto e tempo de entrega de reposição são consideradas conhecidas e constantes ao longo do tempo, desse modo, a empresa sabe quanto de um determinado produto será vendido durante seu tempo, assim, se a organização possui um vasto conhecimento do mercado, fica simples determinar como será o seu abastecimento. Tal sistema é adequado para situações em que há fornecimento constante de matérias-primas, como ocorre no caso do petróleo ou de autopeças.

Entretanto, o modelo estocástico segue uma lógica oposta. Segundo Higle (2005), sua demanda e tempo de entrega de reposição são incertos e envolve inúmeras variáveis, como a sazonalidade dos pedidos, a constante alteração de preços, a falta de estoque nos fornecedores e os prazos de entrega, que podem variar continuamente em função da demanda. Esse sistema é complexo de prever, visto que varia, constantemente, com a demanda, pondo em pauta, até mesmo, o risco o país. Ele é utilizado em produtos que possuem uma certa sazonalidade e variabilidade em sua necessidade. A título de exemplificação, tem-se uma empresa de brindes personalizados, que trabalha mediante à inovação e à demanda de eventos do mercado, sendo uma possibilidade o uso da fórmula de pedido único.

Tabela 1 - Tabela comparativa: modelo determinístico e estocástico

Característica	**Modelo Determinístico**	**Modelo Estocástico**
Demanda	Conhecida e constante	Incerta e variável
Complexidade	Baixa	Alta
Cálculos	Simples	Mais complexos, envolvendo probabilidade e estatística
Aplicabilidade	Produtos com demanda estável	Produtos com demanda variável
Nível de serviço	Geralmente mais baixo	Permite definir um nível de serviço desejado

2.1.3 Gurobi - Otimização de estoque

Em operações de estoque, lidar com múltiplos produtos, demandas que mudam ao longo do tempo e limitações de espaço ou orçamento é sempre um desafio. Muitas vezes, métodos tradicionais não conseguem capturar toda a complexidade do ambiente real,

principalmente quando é preciso decidir não somente quanto pedir, mas quando e quais produtos priorizar em cada reposição. Foi nesse cenário que optou-se por utilizar o Gurobi[1], que mostrou-se excelente para otimização matemática.

O Gurobi é amplamente citado na literatura por sua eficiência na resolução de problemas de programação linear e inteira, que são, justamente, os tipos de modelos mais aplicados em logística e gestão de estoques. Ele permite estruturar cenários reais, levando em conta diversas restrições — seja o espaço limitado no estoque, um orçamento apertado para compras do mês ou mesmo preferências da empresa por determinados itens. Diferente de modelos fechados e com fórmulas fixas, o Gurobi permite adaptar as regras do jogo conforme as necessidades de cada negócio (Gurobi Optimization, LLC, 2023).

Dentro de um sistema de gestão de estoque, o problema de otimização pode ser usado para:

- Definir automaticamente a quantidade ideal de cada item a ser reposto, equilibrando custos de armazenagem, custos de pedido e o risco de falta de produtos;
- Priorizar compras, caso haja limitação de recursos ou espaço, ajudando o gestor a focar nos itens mais críticos naquele momento;
- Avaliar diferentes cenários de demanda: com o auxílio de previsões feitas por modelos como o Prophet, o Gurobi ajusta as decisões a cada nova realidade de consumo, simulando desde meses de alta até períodos mais estáveis;
- Minimizar o custo total do estoque, levando em conta descontos por volume, custos logísticos e penalidades por ruptura.

Essas aplicações não se restringem ao plano teórico: estudos acadêmicos evidenciam a utilização do Gurobi em contextos práticos de gestão de estoques e cadeias de suprimentos. Li e Jiao (2022) desenvolveram um modelo de programação estocástica em duas fases para o problema de roteamento e reposição de estoques com demandas incertas no setor de combustíveis, utilizando o Gurobi como *solver* principal. Os resultados indicaram que a solução baseada no Gurobi, além de eficiente, permitiu avaliar o desempenho de algoritmos customizados para cenários complexos e realistas de inventário. Isso demonstra sua relevância para a tomada de decisão em ambientes logísticos. Aftermarket Analytics (2021) firmou parceria com a Gurobi para desenvolver uma plataforma de otimização de inventário,

[1] https://www.gurobi.com/

destacando o *solver* como elemento central para operacionalização de estratégias logísticas complexas

Seu diferencial está, precisamente, em sua flexibilidade e velocidade, permitindo criar modelos personalizados e resolver problemas em poucos segundos, algo essencial para quem trabalha com estoques dinâmicos e não pode esperar por soluções demoradas.

Por fim, vale destacar que, na prática deste trabalho, o *solver* mostrou-se especialmente útil para sugerir políticas de reposição que consideram o saldo atual do estoque, além de restrições reais do dia a dia da empresa. Sua aplicação permitiu auxiliar a definição de variáveis de decisão, definição de restrições, construção da função objeto, leitura dos resultados e recomendações.

Nesse sentido foram utilizadas as seguintes fórmulas matemáticas para cálculo de otimização dentro do estoque, tais fórmulas foram criadas com o auxílio do Gurobot[2], uma IA treinada para resolver problemas de otimização com o Gurobi:

Conjuntos e Índices:

- **P**: Conjunto de produtos (tipos de camisetas)
- **T**: Conjunto de períodos de tempo (meses)
- **K = P × T**: Conjunto de todas as combinações produto-período

Parâmetros

- C_p: Custo de pedido por unidade para o produto p∈Pp \in Pp∈P
- H_p: Custo de manutenção de estoque por unidade para o produto p∈P
- $D_{p,t}$: Demanda do produto p no período t
- L_{min}: Lote mínimo (quantidade mínima por pedido)
- C: Limite de capacidade de armazenamento
- B_t: Orçamento disponível no período ttt

Variáveis de Decisão

- $Y_{pt} \in \{0, 1\}$: Variável binária que indica se um pedido deve ser realizado para o produto p no período t
- $Q_{p,t} \geq 0$: Quantidade inteira do produto ppp a ser pedida no período ttt

[2] https://portal.gurobi.com/iam/chat/

- $I_{p,t} \geq 0$: Nível de estoque do produto p ao final do período t

Função objetivo:

$$min \sum_{p \in P} \sum_{t \in T} (C_p \cdot Q_{p,t} + H_p \cdot I_{p,t})$$

Restrições Indicadoras (Ordem Lógica):

$$Y_{p,t} = 0 \Rightarrow Q_{p,t} = 0 \; \forall p \in P, t \in T$$

Lote mínimo:

$$Q_{p,t} \geq L_{min} \cdot Y_{p,t} \; \forall p \in P, t \in T$$

Balanço de estoque

- Para o primeiro período (t = 1):

  $$I_{p,1} = Q_{p,1} - D_{p,1} \; \forall p \in P$$

- Para os períodos subsequentes (t > 1):

  $$I_{p,t} = I_{p,t-1} + Q_{p,t} - D_{p,t} \; \forall p \in P, t \in T, t > 1$$

Capacidade de estoque:

$$\sum_{p \in P} I_{p,t} \leq C \; \forall t \in T$$

Orçamento mensal:

$$\sum_{p \in P} C_p \cdot Q_{p,t} \leq B_t \; \forall t \in T$$

Satisfação de demanda:

$$Q_{p,t} \geq D_{p,t} \; \forall p \in P, t \in T$$

Domínios das Variáveis:

- $Y_{p,t} \in \{0, 1\} \; \forall p \in P, t \in T$
- $Q_{p,t} \in Z_+ \; \forall p \in P, t \in T$
- $I_{p,t} \in R_+ \; \forall p \in P, t \in T$

2.2 Inteligência Artificial

2.2.1 Modelos de Inteligência Artificial

No ramo da Inteligência Artificial temos diversas formas de realizar seu desenvolvimento, aqui, abordaremos algumas delas e explicaremos suas vantagens e desvantagens, ressaltando essa importância dentro da proposta do trabalho na área de gestão de estoques.

Machine Learning (ML):

Alpaydin(2020), afirma que o Machine Learning é uma subárea da inteligência artificial que permite os sistemas computacionais aprenderem a partir de uma grande gama de dados, sem, obrigatoriamente,a necessidade de serem programados para tal atividade específica. O aprendizado é baseado em dados e algoritmos que permitem que a máquina faça a análise de padrões, tomando sua decisão dentro dessa base de dados.

Podemos dividir a ML em três principais categorias (Murphy, 2012):

- Aprendizado supervisionado: ele é treinado com dados tanto de entrada quanto de saída conhecidos. Pode ser utilizado para previsão de demanda de produtos e classificação de itens;
- Aprendizado não supervisionado: não existem dados definidos de entrada e de saídas, este modelo busca aprender ou identificar padrões ocultos. Pode ser utilizado na segregação de clientes ou agrupamento de produtos;
- Aprendizado por reforço: já este modelo aprende por tentativa e erro, sendo recompensado ou punido pelas tentativas. Podemos utilizá-lo no controle de inventário.

Os modelos mais utilizados do ML são:

- Regressão linear: modelo básico, utilizado para a previsão de valores contínuos;
- Árvore de decisões: baseado em regras para a tomada de decisões, dividindo os dados em segmentos menores buscando facilitar a tomada de decisão por meio desses “galhos”;

- Rede neurais artificiais: utilizado para identificar padrões complexos em dados não lineares;
- *Support vector machine*: classificação de regressões, utilizado em problemas de alta dimensionalidade.

Vantagens do ML:

- Grande capacidade de processamento de dados;
- Identificação de padrões;
- Automatização com base em dados históricos.

Desvantagens do ML:

- Necessita de uma grande quantidade de dados com qualidade;
- Os dados podem ser enviesados;
- Necessidade de um *hardware* para processar os algoritmos.

Modelos Preditivo (ML)

Brownlee (2017), demonstra que os modelos preditivos são algoritmos utilizados para prever eventos futuros com dados passados. Eles são utilizados, em grande parte, na previsibilidade de demanda e análise comportamental. Dentro do contexto de estoque. são usados para prever demandas, sazonalidades e otimizar o reabastecimento. Sua base é o *Machine learning*, dessa forma, seus principais algoritmos são os mesmos da regressão linear, da árvore de decisões, das redes neurais artificiais e do *support vector machine*.

Modelo de Classificação (ML)

Levando em conta o contexto de ML, o modelo de algoritmos de classificação é feito para atribuir categorias a dados com base em padrões identificados a partir de exemplos anteriores. Essa abordagem é amplamente empregada em aplicações como reconhecimento de imagens, detecção de fraudes e triagem médica (Zhang *et al.*, 2020; Mahmood *et al.*, 2022). No contexto da área de gestão de estoques, a classificação pode ser utilizada para segmentar produtos de acordo com seu giro ou valor, favorecendo decisões assertivas quanto ao armazenamento e à reposição dos itens.

***Deep Learning*:**

De acordo com Goodfellow (2016), *Deep Learning* é uma subárea do *Machine Learning* que é focado no uso de redes neurais com diversas camadas (disso vem o termo "profundo") para processar grandes volumes de dados, principalmente não estruturados, como imagens, áudios e textos. Tal modelo vem se destacando pela complexa capacidade de aprendizagem, algo que se assemelha ao cérebro humano. Vale ressaltar que existem diversas classificações possíveis para as arquiteturas de *Deep Learning*, dependendo do critério adotado, sendo a seguir apresentada uma das abordagens reconhecidas na literatura.

Separado em quatro principais arquiteturas:

- Rede neurais convolucionais: utilizada em processamento de dados especiais, como imagens;
- Redes neurais recorrentes: voltada para lidar com dados sequenciais, como séries temporais;
- Redes generativas: composta por duas redes que convergem entre si, uma geradora e outra discriminadora. Uma cria e a outra autentica os dados. Como, por exemplo, a criação de imagens ou dados.
- *Transformers*: essa foi uma arquitetura revolucionária, visto que lida com uma grande quantidade de texto, como o *ChatGPT* da *Open AI*.

Essa classificação, baseada em Goodfellow, Bengio e Courville (2016), representa apenas uma possível estrutura de categorização, sendo que a área continua em rápida evolução com o surgimento de novas arquiteturas e aplicações.

Vantagens do *Deep Learning*:

- O modelo é capaz de extrair, automaticamente, as características mais relevantes dos dados, dispensando a necessidade de intervenção manual na definição desses atributos;
- Alta precisão e capacidade de avaliação de dados;
- Versatilidade, visto que podemos aplicar em textos, imagens e áudios.

Desvantagens do *Deep Learning*:

- Necessidade de um grande volume de dados com qualidade;
- Elevado custo computacional;
- Dificuldade na interpretação de alguns resultados (como foi tomada tal decisão?).

Sistemas de recomendação

Sistemas de recomendação dentro do *Deep Learning*, segundo Aggarwal (2016)**,** são modelos que sugerem itens ou ações relevantes com dados históricos passados das movimentações realizadas dentro da organização.

Seus principais algoritmos são:

- Matriz de fatoração: decompõe as interações usuário-produto em fatores latentes;
- Modelos baseados em rede neural: utilizam do *Deep Learning*.
- Filtragem colaborativa: sugere itens com base no comportamento de outros usuários;
- Modelos híbridos: combinam filtragem colaborativa e baseada em conteúdo para maior precisão.

2.2.2 IA Generativa

A IA Generativa é uma abordagem que busca criar conteúdo ou dados a partir de um grande volume de informações obtidas anteriormente. Esta se distingue do modelo preditivo, visto que ele apenas estima os resultados com base na entrada dos dados e, por sua vez, a IA Generativa "cria" novas informações, como texto, vídeo, áudio a partir de algum *prompt* (comando dados à IA).

Segundo Brown (2020), a Inteligência Artificial Generativa utiliza modelos baseados em Redes Neurais Artificiais complexas, que aprendem representações dos dados e são capazes de gerar novas informações com características similares.

Exemplo de modelagens:

- GANs (Redes Generativas Adversariais): Criadas por Ian Goodfellow em 2014, produzem dados com base em redes neurais que competem entre si, dessa forma, os dados produzidos são baseados nos conjuntos originais;
- VAE (*Variational Autoencoders*): utilizam uma estrutura de codificação e decodificação para aprender a distribuição dos dados e, com isso, podem gerar dados semelhantes.
- Modelos Autoregressivos e *Transformers*: o surgimento dos modelos autoregressivos baseados em *Transformers*, como proposto por Vaswani *et al*. (2017), representou um avanço marcante na IA generativa. Esses modelos utilizam mecanismos de atenção para gerar sequências de texto coerentes e contextuais a partir de comandos (Radford *et al.*, 2019; Devlin *et al.*, 2019). Atualmente, os *Transformers* viabilizam sistemas capazes de produzir textos longos, traduzir idiomas e sintetizar código, ampliando o

escopo e a sofisticação da IA generativa, como por exemplo o GPT (Lin; Zhou; Wang, 2023; Liu *et al.*, 2023).

2.2.3 OCR

O Reconhecimento Óptico de Caracteres (OCR — *Optical Character Recognition*) é uma tecnologia consolidada que transforma imagens contendo texto em dados digitais editáveis, sendo amplamente utilizada para agilizar o processamento automatizado de informações. Em sistemas de gestão de estoque, o OCR permite que *dashboards*, relatórios ou etiquetas — muitas vezes salvos como imagens — sejam convertidos em texto, facilitando o uso pelas camadas de inteligência artificial e reduzindo o consumo de *tokens* ao evitar o envio de grandes volumes de dados para análise. Em artigos publicados no âmbito da ICRA 2021, como *"OCR-based Inventory Management Algorithms Robust to Damaged Images"*, demonstra-se a eficácia de algoritmos OCR em cenários industriais, inclusive quando os rótulos estão danificados, ressaltando sua relevância prática e robustez operacional (Choi *et al*., 2021)

2.2.4 LangChain

O LangChain[3] é um *framework* emergente que facilita a criação de aplicações baseadas em modelos de linguagem, permitindo conexões com diferentes fontes de dados, tratamento de contextos extensos e definição de fluxos inteligentes de consulta e geração textual. Sobhani e Singh (2023) exemplificam seu uso na automação de atendimento ao cliente, na qual a ferramenta foi utilizada para construir um *chatbot* corporativo, aproveitando o GPT e bases de conhecimento específicas da organização. Já Li *et al.* (2023) discutem a aplicação de modelos de linguagem na otimização da cadeia de suprimentos, destacando que *frameworks* como o LangChain podem tornar mais intuitiva a interpretação de resultados complexos de sistemas de planejamento, por meio de interações em linguagem natural. Essas experiências demonstram que ele não só simplifica a integração de grandes modelos de linguagem com dados heterogêneos, como também melhora, significativamente, a acessibilidade e dinamicidade das análises automatizadas — o que foi observado na implementação do sistema de estoque, neste o LangChain permitiu a geração de diagnósticos e relatórios inteligentes a partir de dados consolidados.

[3] https://www.langchain.com/

2.3 Ia e Estoque

A gestão de estoque tradicional, frequentemente baseada em processos manuais ou dependentes da análise humana, pode apresentar desafios, como o excesso ou a falta de produtos. A integração entre a IA e o estoque vem revolucionado a maneira como as organizações otimizam os seus processos estratégicos de compra/produção e logístico.

Principais funcionalidades que podemos incorporar:

- Previsão de demanda: Chopra e Meindl (2016) demonstram que podemos utilizar modelos preditivos baseados em *Machine Learning* para analisar a sazonalidade e previsão das vendas e tal previsão pode auxiliar na manutenção do estoque;
- Automatização no reabastecimento: com uma integração no ERP, a IA pode sugerir ou automatizar os pedidos de reposição com base nos padrões dos dados fornecidos a ela;
- Cadastro de produto simplificado: utilização de *prompts* simples.

Silver, Pyke, Thomas (2017) detalham estratégias de gestão de estoque com o apoio de tecnologias modernas, assim, destacam alguns desafios e benefícios de se trabalhar com a IA.

Benefícios:

- Redução de custos: reduz custos operacionais;
- Tomada de decisão embasada em dados: *insights* gerados pela IA possibilitam a tomada de decisão baseadas em dados concisos;
- Eficiência operacional: a automação de processos reduz o tempo gasto em processos repetitivos;
- Satisfação do cliente: aumenta a possibilidade da disponibilidade do produto;
- Previsibilidade: a capacidade de antecipar a demanda torna a operação mais previsível e saudável;
- Financeiro: com a previsibilidade de demanda conseguimos ter uma previsão do capital de giro necessário para manter o estoque girando com saúde.

Desafios:

- Qualidade dos dados: para conseguir atingir todos esses objetivos precisamos ter uma qualidade e integridade dos dados, caso contrário, o sistema não funcionará corretamente;
- Capacitação técnica e custo inicial: a capacitação técnica para gestão desse sistema e um elevado custo podem inviabilizar tal aplicação, contudo, este trabalho busca contornar esse desafio;
- Adaptação contínua às mudanças do cenário: o mercado é dinâmico e pode ser impactado por fatos imprevisíveis;
- Dependência da tecnologia: o uso excessivo pode criar uma dependência da IA, afetando a capacidade crítica e analítica dos gestores, além disso, a falha ou travamento do sistema pode gerar a paralisação da operação ou interferir nas tomadas de decisões.

2.4 *Open source* e democratização do acesso tecnológico

O movimento *open source* tem se consolidado como um vetor essencial na democratização de tecnologias avançadas, especialmente para pequenas e médias empresas (PMEs). Ao eliminar custos de licenciamento e reduzir a dependência de fornecedores, *softwares* de código aberto oferecem a essas empresas condições similares aos grandes sistemas para inovar em áreas como gestão de estoques e inteligência artificial.

Fagboola *et al.* (2022) apresenta um estudo de caso em que uma plataforma *web,* baseada em código aberto, foi implementada para automatizar o gerenciamento de estoques em uma PME, resultando em maior precisão nos registros de entradas e saídas e redução significativa de erros operacionais. Essa experiência evidencia a capacidade das soluções *open source* de substituir métodos manuais, trazendo ganhos relevantes em eficiência com baixo investimento inicial.

Ademais, um estudo de Salah *et al.* (2023) documenta a adoção de sistemas *open source* combinados com IoT para monitoramento em tempo real de produtos em estoque. A solução permitiu à empresa detectar faltas e excessos de forma dinâmica, reduzindo rupturas e otimizando a reposição.

A análise de migração de PMEs para tecnologias *open source*, realizada por Helo e Hao (2021), mostra que a adoção de *frameworks* abertos de IA fortalece a resiliência das cadeias de suprimentos. Os autores identificam melhorias tanto na visibilidade quanto na adaptação operativa, reforçando o valor competitivo das PMEs.

Um estudo de Albayrak Ünal, Erkayman e Usanmaz (2023) confirma o crescente interesse por inteligência artificial em gestão de estoques, apontando que muitos projetos utilizam ferramentas *open source* em estágios iniciais de adoção tecnológica. Os autores observam que, embora ainda exista uma lacuna entre pesquisa e prática, estratégias de implementação *open source* já apresentam resultados promissores.

Por fim, o relatório da Linux Foundation (2025) evidencia que cerca de 94 % das organizações que adotam IA, incluindo PMEs, integram componentes *open source* em suas soluções. O estudo realça ganhos de produtividade, inovação e acessibilidade que visam nivelar o campo de atuação entre pequenas empresas e grandes corporações.Em suma, a literatura respalda a relevância e a eficácia do *open source* como estratégia de democratização tecnológica, ao oferecer às PMEs meios acessíveis e flexíveis para modernizar sua gestão de estoque e implementar inteligência artificial sem grandes custos e dependência de ferramentas proprietárias.

3 METODOLOGIA

3.1 Tipologia e descrição geral dos métodos de pesquisa

O método utilizado para o desenvolvimento desta pesquisa foi a *Design Science Research* (DSR), derivado do paradigma da *Design Science* (DS).

O DSR busca uma abordagem de resolução prática, orientada a problemas existentes buscando uma melhoria na solução ou artefatos que auxiliem na eficiência na sociedade ou em organizações. (Van Aken, 2005; Winter; Aier, 2016, *apud* Cauchick, 2019).

Logo, este trabalho tem como objetivo desenvolver uma solução para a gestão de estoques com o auxílio de uma IA generativa, capaz de contribuir para a previsibilidade da demanda e fornecer insights sobre reposições, de forma acessível para pequenas e médias empresas. Ao utilizar o DSR para resolver problemas complexos, é necessário criar e desenvolver artefatos, que são objetos destinados à produção de conhecimento científico e cuja constituição se fundamenta em métodos científicos (Hevner *et al.*, 2004).De acordo com Cauchick (2019), o pesquisador que utiliza a DS como abordagem deve estruturar sua investigação com foco na solução do problema identificado no contexto analisado. Para isso, é necessário identificar, projetar, avaliar e/ou desenvolver um ou mais artefatos que contribuam para resolver o problema ou promover melhorias significativas, tanto no âmbito organizacional quanto na sociedade.

Outrossim, compreende-se que a funcionalidade dos artefatos reside na sua capacidade de mediar a interação entre um ambiente interno e um ambiente externo, tendo, como finalidade atender a um objetivo em específico (Simon, 1996, *apud* Cauchick, 2019). O ambiente interno corresponde à maneira como o artefato deve operar para atender aos requisitos necessários à resolução de um problema. Por sua vez, o ambiente externo refere-se ao contexto que proporciona as condições para o funcionamento desse artefato. Assim, mudanças no ambiente externo podem alterar as condições de operação do artefato, exigindo ajustes em sua configuração para que continue adequado à solução do problema proposto, o que implica, consequentemente, alterações no ambiente interno (Van Aken *et al*., 2012, *apud* Cauchick, 2019).

A Tabela 2, evidencia os tipos de artefatos desenvolvidos sob o paradigma da DS, bem como as suas respectivas definições.

Tabela 2 - Tipos de Artefatos

Tipo de artefato	Definição
Constructo	Conceitos utilizados para descrever problemas ou especificar as respectivas soluções.
Modelo	Conjunto de proposições que expressam as relações entre constructos e representam situações problema e solução.
Método	Conjunto organizado de passos para realizar uma tarefa, baseado em constructos e modelos.
Instanciação	Execução do(s) artefato (s) em seu ambiente real, evidenciando a viabilidade e eficácia dos artefatos.
Design Proposition	Regras tecnológicas ou regras de projeto, consideradas contribuições teóricas da *Design Science*.

Fonte: Metodologia Científica para Engenharia (Cauchick, Paulo, 2019)

3.2 Caracterização e descrição dos instrumentos de pesquisa

De acordo com Van Aken (*apud* Cauchick, 2019), a *DSR* é uma metodologia que pode ser empregada para guiar a realização de investigações científicas em várias áreas do conhecimento. É recomendado quando se tem, como objetivo da pesquisa, desenvolver a resolução de problemas com criação de artefatos ou com a elaboração de novos conhecimentos, sejam eles científicos ou técnicos (Pimentel, 2020).

Com a aplicação do método DSR na presente pesquisa, identificam-se cinco etapas fundamentais para aprofundar o entendimento sobre o artefato a ser desenvolvido (Lacerda *et al.*, 2013). Essas etapas são: conscientização, sugestão, desenvolvimento, avaliação e, por fim, conclusão.

Conscientização

A etapa de conscientização do DSR fundamenta-se na compreensão da questão a ser abordada (Lacerda, 2013, *apud* Romme, 2003). Nessa fase, o problema deve ser analisado em profundidade, buscando identificar suas principais causas e a relação com o contexto interno e externo. Além disso, espera-se que o artefato seja delineado, explicando como poderá

contribuir para a solução do problema. Nesta pesquisa, o artefato será o de desenvolver um sistema *Web* com uma Inteligência Artificial para o Controle de Estoque, sendo um Sistema de Gestão com Previsão Automatizada de Reposição. Desse modo, o objetivo é utilizar a IA generativa em um sistema para a gestão de estoque, de modo que esse sistema seja capaz, baseado em dados passados, de auxiliar na previsibilidade de demanda, dando possibilidades de reposições inteligentes. Possibilitando a empresa a utilizar métodos robustos e "espertos" para auxiliar em seu estoque.

Sugestão

A etapa de *Sugestão* no *Design Science Research* (DSR) é o momento em que o pesquisador propõe soluções iniciais para o problema identificado na fase de relevância. Essas soluções, embasadas em teorias e conhecimentos prévios, servem como base para o desenvolvimento do artefato que será avaliado posteriormente (Hevner *et al.,* 2004). O objetivo é criar um protótipo conceitual que conecte o problema identificado ao artefato, garantindo que ele atenda às necessidades do contexto estudado.

Peffers *et al.* (2007) destacam que essa etapa requer fundamentação teórica sólida para orientar a formulação das soluções. Hevner *et al.* (2004) enfatizam que o processo exige criatividade e rigor, pois as propostas devem ser viáveis e inovadoras, respeitando as restrições e especificidades do ambiente de aplicação. Além disso, nessa fase, é importante definir critérios para avaliar a eficácia do artefato, assegurando que ele resolva o problema de forma eficiente.

Neste trabalho, um dos artefatos utilizados será o modelo de IA Generativa, responsável por receber os dados de estoques e gerar sugestões com base nessas informações, além de possibilitar o cadastro completo por meio da plataforma. Ferramentas como o ChatGPT participarão desse processo, tanto no desenvolvimento quanto na geração de *insights* a partir dos dados de estoque, garantindo uma IA treinada para oferecer essas recomendações. Outro artefato será composto por ferramentas *no-code*, empregadas no desenvolvimento dos processos do cliente dentro da plataforma.

Desenvolvimento

Nesse ponto, inicia-se o desenvolvimento do artefato. Podem ser utilizadas diferentes técnicas, como algoritmos computacionais, representações gráficas, protótipos, maquetes, entre outras. Nessa fase, o pesquisador se dedica à criação do ambiente interno do artefato, conforme proposto por Simon (1996) (*apud* Dresch *et al.,* 2015).

Segundo Peffers *et al.* (2007), o desenvolvimento deve seguir um planejamento bem estruturado, integrando conhecimentos teóricos e práticos para garantir a eficácia do artefato. Hevner *et al.* (2004) reforçam que o rigor científico deve estar presente em cada passo, garantindo que as decisões de design estejam alinhadas com os objetivos do estudo e os critérios definidos previamente.

Nesse sentido, essa etapa terá, como foco, a criação de um sistema inteligente para gestão de estoques. O desenvolvimento envolve transformar os requisitos funcionais e as propostas conceituais em um sistema *web* plenamente operacional. O objetivo é construir um artefato capaz de cadastrar produtos, registrar movimentações, categorizar itens, sugerir reposições e gerar relatórios fundamentados em análises preditivas, coma finalidade de aproveitar o potencial da IA Generativa para executar essas tarefas. A documentação detalhada do processo também será essencial, registrando as decisões técnicas e os resultados obtidos. Assim, o desenvolvimento do sistema permitirá validar as soluções propostas e consolidar o conhecimento adquirido ao longo do projeto. Nessa etapa serão realizados diversos testes, tanto de coerência quanto da qualidade dos dados obtidos para verificar a confiabilidade da IA.

Avaliação

Nesse momento, será conduzida a análise dos resultados relacionados ao desempenho do artefato para solucionar o problema proposto. Para isso, serão estabelecidas métricas específicas que permitirão avaliar sua eficácia e compreender tanto os aspectos bem-sucedidos quanto as falhas observadas. Com base nesses resultados, serão realizadas as adaptações necessárias para aprimorar o artefato (Lacerda, 2013).Nesse processo, será realizada uma análise para verificar se o objetivo geral do trabalho foi efetivamente alcançado e se o sistema apresenta confiabilidade em seus relatórios, além de ser de fácil utilização.

Conclusão

Como orientado por Dresch *et al.* (2015), as conclusões da pesquisa devem explicitar os desafios e aprendizados encontrados ao longo da elaboração do trabalho. Nesse ponto, serão consolidados os resultados alcançados e dadas orientações para pesquisas futuras dentro do contexto trabalhado, renovando o ciclo proposto pela *Design Science Research*.

3.3 Caracterização da organização pesquisada

A amostra deste trabalho foi composta pela empresa *Sasse Gifts*, fundada em 1993 em Brasília, e especializada em brindes personalizados e confecção têxtil, como camisetas, bonés, sacolas e uniformes. A empresa possui escritório e fábrica em Brasília, uma unidade fabril em Santa Catarina e um polo estratégico em São Paulo. Seu foco está no atendimento a grandes eventos, como *Na Praia, Lollapalooza e Mega da Virada*, além de prestar serviços para empresas de grande porte que realizam ações de marketing e endomarketing, dentre as quais estão Banco do Brasil, Caixa Econômica Federal, Sabin e Brasal.

A fábrica em Brasília mantém um amplo estoque de materiais prontos, aguardando apenas o processo de personalização, e dispõe de um extenso maquinário de impressão, tanto para produtos têxteis quanto para a linha de brindes. Essa estrutura é voltada especialmente para atender pedidos urgentes e que exigem atenção diferenciada.

Já a fábrica em Florianópolis foi estrategicamente instalada em Santa Catarina, um importante polo têxtil do país, e tem como foco abastecer a produção de Brasília, além de realizar trabalhos de costura especializada, como a confecção de uniformes profissionais.

Por sua vez, o polo em São Paulo é direcionado à linha de brindes, mantendo contato direto com importadores e parceiros, o que facilita a comercialização dos produtos. Conta, ainda, com uma equipe comercial robusta, voltada para explorar as oportunidades do vasto mercado paulista.

3.4 Procedimentos de coleta de dados

Os dados coletados correspondem aos registros de pedidos de compra e venda de produtos que transitam pela fábrica da Sasse, em Brasília, envolvendo necessariamente a entrada e a saída de itens no estoque. Serão considerados dados de, no mínimo, quatro meses, para possibilitar uma análise preliminar no sistema, com o auxílio de uma IA generativa.

A coleta será realizada diretamente no banco de dados referente às produções efetuadas no período definido para o estudo.

3.5 Análise de dados

Neste trabalho, a análise de dados se conduziu de maneira estruturada, com o objetivo de validar a proposta de um sistema de gestão de estoque. O procedimento de análise foi

dividido em etapas sequenciais, acompanhando a lógica dos blocos de desenvolvimento do *notebook*, detalhados, posteriormente, nos resultados.

A etapa inicial consistiu na instalação e configuração do ambiente computacional, priorizando a utilização do *Google Colab*[4], de modo a assegurar tanto a acessibilidade ao sistema quanto a replicabilidade dos experimentos realizados. Em seguida, serão realizadas as integrações necessárias entre o sistema e o *Google Sheets*[5], que atuará como banco de dados para o registro das movimentações de estoque.

Com a infraestrutura estabelecida, foi possível importar e organizar os dados históricos, cobrindo, ao menos, quatro meses de operações da empresa amostral. A manipulação e análise desses dados foi realizada, principalmente, com o suporte da biblioteca pandas, permitindo estruturação, filtragem e consolidação das informações para o uso analítico.

Na etapa seguinte, o foco está na criação de *dashboards* interativos e relatórios visuais, utilizando bibliotecas como *Matplotlib*[6] e *Seaborn*[7] para gerar visualizações que auxiliem no monitoramento do saldo de estoque e na identificação de produtos críticos. A exportação desses *dashboards* em formatos como PNG e Excel proporciona flexibilidade para análises posteriores. A análise preditiva, elemento central deste trabalho, será operacionalizada por meio do Prophet[8], voltado à previsão de demanda a partir das séries temporais de vendas. Paralelamente, a biblioteca gurobipy será empregada nos algoritmos de otimização, tencionando sugerir políticas de reposição baseadas em múltiplos cenários e restrições do negócio.

A integração da automação inteligente é implementada por meio da construção de um *backend* utilizando Flask[9], o que permitirá a criação de endpoints web para consultas automatizadas de análises e relatórios. Paralelamente, será incorporada uma camada de OCR, empregando o *pytesseract*[10], para possibilitar a extração de texto a partir de imagens de *dashboards*. Essa extração viabiliza a análise automatizada por modelos de linguagem, como o GPT, facilitando a geração de recomendações personalizadas.

Assim, todos os componentes foram submetidos a testes, assegurando a funcionalidade das ferramentas propostas. Essa abordagem metodológica permite que cada etapa do

[4] https://colab.google/
[5] https://docs.google.com/spreadsheets/u/0/
[6] https://matplotlib.org/
[7] https://seaborn.pydata.org/
[8] https://facebook.github.io/prophet/
[9] https://flask.palletsprojects.com/en/stable/
[10] https://pypi.org/project/pytesseract/

desenvolvimento, da coleta de dados brutos até à sua entrega, seja acompanhada, avaliada e aprimorada, resultando em um sistema transparente, além de, por ser de código aberto, assegurar constante aprimoramento e adaptação.

4 RESULTADO - VISUALIZADOR ESTOQUE

Esse capítulo aborda, de forma abrangente, o processo de construção do código do sistema, contemplando desde a ideação do projeto e as diferentes tentativas, até a implementação da solução final. Destarte, são demonstradas as ferramentas selecionadas e os caminhos percorridos ao longo do desenvolvimento, bem como as adaptações necessárias diante dos desafios e restrições de tempo impostos pelo cronograma do projeto. O objetivo é apresentar, de maneira transparente, o percurso adotado.

Além da etapa inicial, o capítulo será estruturado em blocos, seguindo a lógica do código apresentado no apêndice, com explicações detalhadas de cada etapa até a realização dos testes finais, sendo este capítulo focando para o código do visualizador do estoque.

4.1 Caminho inicial e primeiros testes

A trajetória de desenvolvimento do sistema de gestão de estoque com inteligência artificial foi marcada por avanços incrementais, decisões técnicas estratégicas e sucessivos desafios práticos. Desde o início, buscou-se priorizar a construção de uma solução acessível, flexível e realista, capaz de se adaptar às demandas cotidianas de pequenas e médias empresas.

O primeiro passo foi a escolha do *Google Sheets* como banco de dados principal. Essa ferramenta mostrou-se vantajosa por sua facilidade de acesso, integração nativa com *Python* (via API *gspread*[11]), compartilhamento em nuvem e atualização em tempo real. Essa opção permitiu centralizar e organizar todas as informações do estoque, incluindo entradas, saídas, atributos dos produtos e demais registros necessários.

A partir dessa base, tornou-se evidente a necessidade de automatizar a tomada de decisão na reposição de itens. Para isso, foi feito outro código, abordado no capítulo 5, com o *solver* chamado Gurobi, um otimizador matemático amplamente reconhecido em aplicações logísticas.

[11] https://docs.gspread.org/en/latest/

Durante o processo, buscou-se ampliar o grau de automação do sistema por meio de integrações com plataformas externas, como o *Make*[12], para orquestrar rotinas automatizadas entre a base de dados, a camada de inteligência e eventuais frentes de visualização. Além disso, foi avaliada a utilização do *Dify*[13] como interface de IA conversacional, corporificando a intenção de fornecer respostas automáticas e facilitar a interação com o usuário. No entanto, ambos os recursos apresentaram obstáculos significativos, como erros de integração, limitações no fluxo de dados e instabilidade no funcionamento, em especial quando o volume de dados aumentava. Diante disso, foi necessário reavaliar as escolhas e decidir por manter o desenvolvimento concentrado no ambiente *Google Colab*, utilizando bibliotecas *Python*, a fim de garantir, assim, maior controle, estabilidade e adaptabilidade do sistema.

Com a maturação do projeto, surgiu um novo desafio relacionado ao custo operacional da análise de dados via inteligência artificial. Consultar toda a base de dados diretamente com modelos de linguagem (GPT[14]/OpenAI) resultava em consumo elevado de *tokens* e, consequentemente, aumento nos custos e lentidão do processo. A solução adotada foi modificar o fluxo: em vez de enviar grandes volumes de dados à IA, passou-se a consolidar os *dashboards* acadêmicos e relatórios de estoque em imagens (formato PNG), utilizando-os como fonte de análise. Nesse ponto, entrou em cena o OCR (*Optical Character Recognition*), tecnologia implementada para extrair o texto dessas imagens e convertê-las em dados estruturados para análise da IA. O OCR foi decisivo para reduzir o custo e o tempo das análises, porque permitiu que a IA realizasse diagnósticos e sugerisse recomendações a partir de relatórios resumidos, o que manteve a precisão e a utilidade dos *insights*.

Em paralelo, para potencializar ainda mais a geração de diagnósticos automáticos, foi integrado ao sistema o LangChain. Esse *framework* permite conectar modelos de linguagem avançados a bancos de dados textuais e documentos, facilitando consultas inteligentes, respostas contextualizadas e elaboração automatizada de relatórios e *insights*, tudo de maneira natural e eficiente. O LangChain foi utilizado, principalmente, no bloco de análise de *dashboards* e geração de relatórios automáticos, agregando valor ao sistema ao permitir uma interação dinâmica com os dados consolidados.

Durante toda essa trajetória, o projeto evoluiu por meio de tentativa, erro e aprimoramento contínuo, sempre pautado em critérios de eficiência operacional, acessibilidade, custo-benefício e viabilidade técnica. O resultado desse processo foi um

[12] https://make.com/
[13] https://cloud.dify.ai/signin
[14] https://chatgpt.com/

sistema de fácil manutenção, praticamente pronto para integração com futuras interfaces, e que utiliza das ferramentas modernas de inteligência artificial aplicadas à gestão de estoque.

Em síntese, o Google Sheets foi utilizado como uma base de dados acessível, colaborativa e de fácil integração. No que lhe concerne, o Gurobi auxiliou o processo para a tomada de decisão para as reposições. O OCR viabilizou a análise ágil e econômica de relatórios consolidados, convertendo imagens em texto para processamento pela IA. Além disso, o LangChain potencializou o uso da inteligência artificial, permitindo análises avançadas e a geração automática de relatórios a partir dos dados consolidados.

A construção desse sistema evidencia como a integração criteriosa de diferentes tecnologias, aliada à adaptação frente a obstáculos práticos, pode resultar em uma solução relevante tanto para o contexto empresarial quanto para a pesquisa acadêmica.

4.2 Banco de dados

Como mencionado, o banco de dados selecionado para este trabalho foi o Google Sheets, principalmente em detrimento de sua alta familiaridade com o público em geral, curta curva de aprendizagem e baixo custo.

Tabela 3 - Exemplo Banco de Dados

Produto	Código	Cor	Tecido	Tamanho	Data	Entrada	Saída	Vendedor	Cliente	Pedido	Período	Chave
Camiseta Tradicional	000.012	Rosa	Clássico	G	13/11/2024		4	Giovanni			2024-11	G_Camiseta Tradicional_Rosa_Clássico+000.012_2024-11
Camiseta Tradicional	000.012	Rosa	Clássico	GG	13/11/2024		2	Giovanni			2024-11	GG_Camiseta Tradicional_Rosa_Clássico+000.012_2024-11
Camiseta Tradicional	000.012	Branco	Clássico	PP	21/11/2024		40	Giovanni			2024-11	PP_Camiseta Tradicional_Branco _Clássico+000.012_2024-11
Camiseta Tradicional	000.012	Branco	Clássico	P	21/11/2024		460	Giovanni			2024-11	P_Camiseta Tradicional_Branco _Clássico+000.012_2024-11
Camiseta Tradicional	000.012	Branco	Clássico	M	21/11/2024		500	Giovanni			2024-11	M_Camiseta Tradicional_Branco _Clássico+000.012_2024-11
Camiseta Tradicional	000.012	Branco	Clássico	G	21/11/2024		500	Giovanni			2024-11	G_Camiseta Tradicional_Branco _Clássico+000.012_2024-11
Camiseta Tradicional	000.012	Branco	Clássico	GG	21/11/2024		500	Giovanni			2024-11	GG_Camiseta Tradicional_Branco _Clássico+000.012_2024-11
Camiseta Tradicional	000.012	Branco	Clássico	P	22/11/2024		1.000	Juliano			2024-11	P_Camiseta Tradicional_Branco _Clássico+000.012_2024-11
Camiseta Tradicional	000.012	Branco	Clássico	M	22/11/2024		2.000	Juliano			2024-11	M_Camiseta Tradicional_Branco _Clássico+000.012_2024-11

Fonte: Elaboração própria

Na tabela 3 vemos um exemplo de como esse banco de dados deve ser construído, nela, há, aproximadamente, 960 linhas — com os futuros registros, esse número tende a aumentar — e 13 colunas.

Descrição das variáveis (colunas):

- Produto: nome do item;
- Código: referência alfanumérico que remete ao nome do produto;
- Cor: cor do item;
- Tecido: somente em itens da linha têxtil;
- Tamanho: grade de tamanho das camisetas, usado somente para as camisetas;
- Data: data do registro;

- Entrada: entrada de mercadoria no estoque;
- Saída: saída de mercadoria do estoque, geralmente uma venda;
- Vendedor: vendedor responsável;
- Cliente;
- Pedido: Número do pedido dentro do sistema da empresa;
- Período: mês e ano do registro da linha, para eventuais filtros, fórmula dentro do Sheets - =TEXTO(F2;"yyyy-mm")
- Chave: junção das principais colunas para eventuais filtros, fórmula dentro do Sheets - =CONCATENAR(E2;"_";A2;"_";C2;"_";D2;"+";B2;"_";L2)

4.3 Bloco 1: instalações, *imports* e configurações iniciais

Antes de qualquer avanço prático no desenvolvimento do código, foi indispensável dedicar tempo à preparação do ambiente de trabalho. Essa etapa inicial pode parecer simples, mas é fundamental para que todas as demais fases do projeto aconteçam de maneira fluida e sem imprevistos. Todo o projeto foi estruturado no Google Colab, precisamente por ser um ambiente acessível, prático e difundido entre estudantes e profissionais, além de permitir o uso de recursos em nuvem e o fácil compartilhamento do código.

Foi nesse contexto que, no primeiro bloco do código, foram concentrados esforços na instalação das bibliotecas e configurações das integrações que dariam sustentação ao sistema. Algumas dessas bibliotecas merecem destaque pelo papel que desempenham no funcionamento do projeto:

- gspread e oauth2client[15]: fundamentais para a integração do sistema com o Google Sheets, viabilizando a autenticação e o acesso seguro às planilhas de dados. Essas ferramentas tornaram possível automatizar todo o processo de leitura e atualização das informações de estoque, eliminando a necessidade de inserção manual e minimizando riscos de erros.

- pandas e openpyxl[16]: essenciais para manipulação, análise e exportação dos dados em formato tabular. O *pandas* tornou-se o "centro nervoso" do processamento de dados, permitindo organizar, filtrar e consolidar grandes volumes de informações de forma

[15] Disponível em: https://pypi.org/project/gspread/ e https://pypi.org/project/oauth2client/
[16] Disponível em: https://pypi.org/project/pandas/ e https://pypi.org/project/openpyxl/

prática e eficiente, enquanto o *openpyxl* viabilizou a exportação das análises em formato Excel, facilitando o compartilhamento dos resultados.

- matplotlib e seaborn[17]: utilizadas para a construção dos *dashboards* e visualizações interativas que permitem monitorar o estoque em tempo real. A clareza desses gráficos não só facilita a tomada de decisão como também agrega valor à apresentação dos resultados no âmbito acadêmico e empresarial.

- Prophet[18]: ferramenta para previsão de demanda baseada em séries temporais, fundamental para o módulo de previsão inteligente do sistema. Ao integrar o Prophet, o sistema ganhou capacidade de antecipar cenários de consumo e sugerir ações proativas para evitar rupturas ou excessos de estoque.

- flask e flask-ngrok[19]: empregadas para criar e expor *endpoints web* do sistema, facilitando a integração futura com interfaces *front-end* e automações externas. Com essas bibliotecas, foi possível transformar funções do sistema em serviços acessíveis via API, abrindo espaço para futuras expansões e integrações.

- pytesseract e Pillow[20]: necessárias para o OCR, possibilitando a extração de texto a partir de imagens de *dashboards* e relatórios consolidados. Esse recurso inovador permitiu analisar relatórios gerados em formato visual (como PNG), convertendo-os em texto para posterior processamento automatizado pela inteligência artificial.

- LangChain[21] e suas extensões: ampliaram as capacidades analíticas do sistema, permitindo a utilização de modelos de linguagem avançados para análise dos dados. A integração dessa tecnologia foi fundamental para que o sistema pudesse gerar diagnósticos automáticos, responder perguntas e criar relatórios inteligentes de maneira rápida e contextualizada.

Além de instalar e configurar essas bibliotecas, também foi preciso definir variáveis de ambiente, caminhos para credenciais e parâmetros de acesso às APIs utilizadas, assegurando

[17] Disponível em: https://pypi.org/project/matplotlib/ e https://pypi.org/project/seaborn/
[18] Disponível em: https://pypi.org/project/prophet/
[19] Disponível em: https://pypi.org/project/Flask/ e https://pypi.org/project/flask-ngrok/
[20] Disponível em: https://pypi.org/project/pytesseract/ e https://pypi.org/project/Pillow/
[21] Disponível em: https://python.langchain.com/

que todo o ecossistema do projeto estivesse conectado e operante. Essa preparação cuidadosa foi decisiva tanto para prevenir conflitos de versões e falhas inesperadas quanto para assegurar que o sistema mantenha sua escalabilidade e seja passível de manutenção ao longo do tempo.

Assim, a etapa inicial do projeto foi mais do que um simples procedimento técnico: ela se revelou um verdadeiro alicerce, sobre o qual todo o sistema de gestão de estoque foi construído. Essa preocupação com a base do ambiente contribuiu diretamente para o sucesso das etapas seguintes, refletindo o cuidado e o rigor essenciais para a condução de um trabalho acadêmico e profissional consistente.

Resultado do bloco:

Deverá aparecer a instalação de todas as dependências e autorizações das autenticações, finalizado da seguinte forma:

✅ Todas as dependências instaladas e chaves configuradas!

4.4 Bloco 2: integração com google sheets

Uma vez com o ambiente pronto, a prioridade foi garantir que o sistema pudesse se comunicar, de maneira eficiente, com o Google Sheets. O Sheets tornou-se o banco de dados, já que é nele que tudo acontece: entradas, saídas, registros de produtos e seus detalhes. Tal ferramenta foi selecionada, sobretudo, pela facilidade de acesso, pelo baixo custo e, claro, pela fácil usabilidade.

Uma integração bem executada é fundamental para o sucesso do projeto. Foi necessário dedicar tempo de qualidade aos ajustes e processos de autenticação, com o objetivo de automatizar a coleta de informações e importar os registros diretamente para o sistema, eliminando a necessidade de digitação manual ou troca de e-mails. A utilização do pandas nessa etapa trouxe grande flexibilidade, permitindo que os dados, já estruturados em um *DataFrame*, fossem facilmente filtrados, agrupados e transformados de acordo com as necessidades específicas de cada análise. Ocorreram alguns desafios ao longo desse processo. Padronizar campos como nomes de produtos, tamanhos e cores exigiu atenção, pois até mesmo pequenas variações poderiam atrapalhar os filtros e as análises futuras. O cuidado com o banco de dados logo no início poupou muito trabalho nas fases seguintes, principalmente quando chegou o momento de gerar relatórios automáticos ou gráficos para a tomada de decisão.

Deixar o sistema integrado ao Google Sheets trouxe, além de praticidade, um ganho grande de transparência. As informações ficam acessíveis e sempre atualizadas para qualquer usuário autorizado, o que elimina a confusão de versões e permite que a equipe trabalhe olhando para o mesmo dado, em tempo real.

O segundo bloco do projeto foi decisivo para estruturar uma base sólida de dados, capaz de suportar todas as funcionalidades de análise, previsão e otimização que viriam a seguir. Essa integração com o Google Sheets representou um avanço significativo, não apenas em termos de automação, mas principalmente em acessibilidade e segurança. Isso mostrou, na prática, como tecnologias simples podem transformar a gestão de estoques no contexto empresarial.

Resultado do bloco:

Após a leitura e organização dos dados pelo sistema, é possível realizar, por exemplo, consultas automatizadas, identificar rapidamente produtos com saldo crítico, rastrear a origem de eventuais problemas de abastecimento e gerar relatórios customizados para diferentes áreas da empresa. Esses *outputs* permitem a tomada de decisões mais ágil, fundamentada em informações precisas, cumprindo a proposta de elevar o nível de inteligência e automação da gestão de estoques.

Tabela 4 - Saída estoque

Produto	Código	Cor	Tecido	Tamanho	Data Entrada
Camiseta Tradicional	0.012	Branco	Clássico	PP	20-08-01
Camiseta Tradicional	0.012	Branco	Clássico	P	20-08-01

Camiseta Tradicional	0.012	Branco	Clássico	M	20-08-01
Camiseta Tradicional	0.012	Branco	Clássico	G	20-08-01
Camiseta Tradicional	0.012	Branco	Clássico	GG	20-08-01

Fonte: Elaboração própria

Tabela 5 - Saldo estoque

PP_Camiseta Tradicional_Branco _Clássico+000.0..	-100
P_Camiseta Tradicional_Branco _Clássico+000.0..	-100
M_Camiseta Tradicional_Branco _Clássico+000.0..	-400
G_Camiseta Tradicional_Branco _Clássico+000.0..	-200
GG_Camiseta Tradicional_Branco _Clássico+000.0..	-50

Fonte: Elaboração própria

Tabela 6 - Resumo rápido do saldo por produto:

Boné	-1607
Camiseta Polo	-1357
Camiseta Tradicional	-15974
Caneca Porcelana	-578

Copo Stanley	-1070
Saco Mochila	-4638
Sacola TNT	-6803
Sacola Tecido	-6655
Viseira	-1230

Fonte: Elaboração própria

Além disso, será baixado tanto o estoque consolidado em formato de tabela quanto em *dashboard* para futuramente ser transformado em OCR e ser analisado por uma IA.

4.5 Bloco 3: *dashboards*, visualizações e exportação

Essa etapa do projeto foi dedicada ao desenvolvimento das funcionalidades responsáveis pela visualização dos dados, geração de *dashboards* (em formato acadêmico), criação de relatórios e exportação dos resultados para diferentes formatos. Tais ferramentas são essenciais para transformar os dados brutos do estoque em informações acessíveis e úteis para a tomada de decisão.

Uma das principais funcionalidades desse bloco é o *dashboard* acadêmico, que possibilita ao usuário filtrar os dados de estoque por critérios como produto, cor, tamanho, tecido e período. Com esses filtros, é possível analisar recortes específicos do estoque e compreender rapidamente a situação de diferentes grupos de itens, ajustando os parâmetros conforme as demandas da gestão.

Além da filtragem, o sistema apresenta gráficos e tabelas consolidadas que fornecem uma visão clara do saldo atual dos produtos, movimentações de entrada e saída, itens com estoque crítico e outros indicadores relevantes. Os principais gráficos incluem:

- Saldo atual por produto;
- Itens em situação crítica de estoque (abaixo de determinado limiar);
- Movimentação mensal.

Essas visualizações são fundamentais para que gestores possam identificar, rapidamente, gargalos, excessos, oportunidades de reposição e tendências de consumo, tudo de forma visual e intuitiva.

Outro destaque do bloco é a possibilidade de exportação dos resultados. O sistema permite salvar dashboards como imagens no formato PNG e tabelas como arquivos Excel (.xlsx). Essa funcionalidade é relevante tanto para o compartilhamento dos dados com outros setores quanto para fins de documentação, auditoria ou apresentação de resultados. O processo de exportação é realizado de maneira automatizada, garantindo que a informação visualizada seja exatamente a mesma que é exportada, sem necessidade de retrabalho manual.

O bloco também incorpora recursos de OCR, possibilitando extrair texto de imagens de *dashboards* ou tabelas consolidadas. Essa tecnologia foi integrada para facilitar a análise automatizada dos dados por modelos de inteligência artificial, reduzindo custos computacionais ao trabalhar com resumos visuais em vez de planilhas completas.

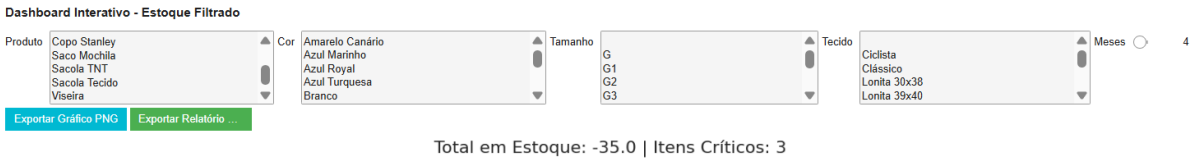
Por fim, todas essas funcionalidades estão integradas de modo a proporcionar um ambiente de gestão visual, no qual o usuário tem autonomia para analisar, exportar e compartilhar informações do estoque em tempo real, de maneira simples e eficiente. O Bloco 3, portanto, cumpre um papel estratégico no sistema, atuando como ponte entre os dados operacionais e a tomada de decisão baseada em informações visuais e precisas.

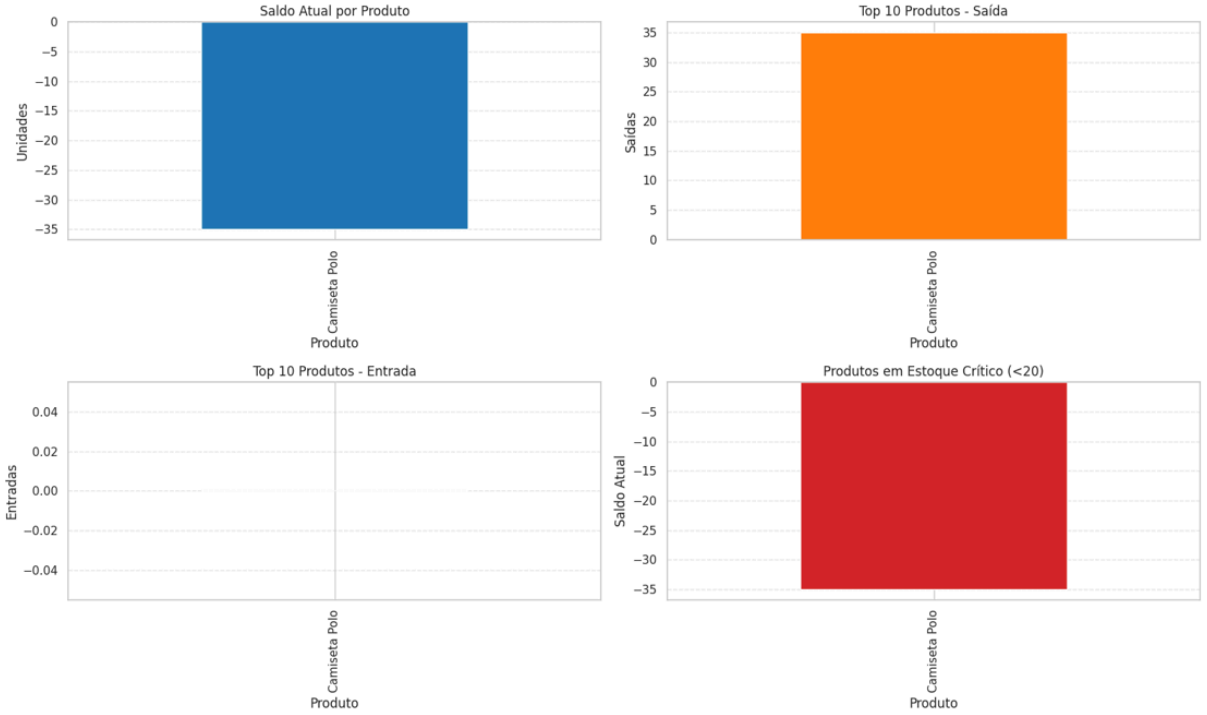
Resultado do bloco:

Nesse bloco, o autor montou um *dashboard* acadêmico com filtros, estoque atual, consolidado e os itens em estado crítico, todos com a possibilidade para serem baixados tanto em excel quanto em PNG, visando compor um relatório.

Ao consultar o *dashboard*, o gestor pode, celeremente, identificar, por exemplo, que determinado produto apresenta saldo abaixo do ideal e agir de forma preventiva. Da mesma forma, a visualização de movimentações por período permite analisar tendências de consumo e ajustar estratégias de reposição. As funcionalidades de exportação garantem que a informação analisada possa ser facilmente compartilhada e documentada.

Imagem 1 - Dashboard Interativo - Estoque Filtrado





Fonte: Elaboração própria

Imagem 2 - Tabela de Saldo Atual

	Código	Produto	Cor	Tamanho	Tecido	Saldo Atual	Status
0	0.014	Camiseta Polo	Preto	G	Polo Clássica	-15.0	Crítico
1	0.014	Camiseta Polo	Preto	M	Polo Clássica	-10.0	Crítico
2	0.014	Camiseta Polo	Preto	P	Polo Clássica	-10.0	Crítico

Imagem da tabela consolidada salva como estoque_consolidado.png (258 linhas)

Fonte: Elaboração própria

Imagem 3 - Estoque completo

	Produto	Código	Cor	Tecido	Tamanho	Data	Entrada	Saída	Vendedor	Cliente	Pedido	Periodo	Chave	Saldo Atual	Sku
0	Camiseta Tradicional	0.012	Branco	Clássico	PP	2024-08-01	0.0	100.0	Giovanni	Banco do Brasil		2024-08	PP_Camiseta Tradicional_Branco _Clássico+000.0...	-100.0	Camiseta Tradicional | Clássico | Branco | PP
1	Camiseta Tradicional	0.012	Branco	Clássico	P	2024-08-01	0.0	100.0	Giovanni	Banco do Brasil		2024-08	P_Camiseta Tradicional_Branco _Clássico+000.01...	-100.0	Camiseta Tradicional | Clássico | Branco | P
2	Camiseta Tradicional	0.012	Branco	Clássico	M	2024-08-01	0.0	400.0	Giovanni	Banco do Brasil		2024-08	M_Camiseta Tradicional_Branco _Clássico+000.01...	-400.0	Camiseta Tradicional | Clássico | Branco | M
3	Camiseta Tradicional	0.012	Branco	Clássico	G	2024-08-01	0.0	200.0	Giovanni	Banco do Brasil		2024-08	G_Camiseta Tradicional_Branco _Clássico+000.01...	-200.0	Camiseta Tradicional | Clássico | Branco | G
4	Camiseta Tradicional	0.012	Branco	Clássico	GG	2024-08-01	0.0	50.0	Giovanni	Banco do Brasil		2024-08	GG_Camiseta Tradicional_Branco _Clássico+000.0...	-50.0	Camiseta Tradicional | Clássico | Branco | GG
5	Camiseta Tradicional	0.012	Branco	Clássico	G1	2024-08-01	0.0	50.0	Giovanni	Banco do Brasil		2024-08	G1_Camiseta Tradicional_Branco _Clássico+000.0...	-50.0	Camiseta Tradicional | Clássico | Branco | G1
6	Camiseta Tradicional	0.012	Azul Turquesa	Clássico	M	2024-08-06	0.0	2.0	Giovanni			2024-08	M_Camiseta Tradicional_Azul Turquesa_Clássico+...	-2.0	Camiseta Tradicional | Clássico | Azul Turques...
7	Camiseta Tradicional	0.012	Azul Turquesa	Clássico	GG	2024-08-06	0.0	4.0	Giovanni			2024-08	GG_Camiseta Tradicional_Azul Turquesa_Clássico...	-4.0	Camiseta Tradicional | Clássico | Azul Turques...
8	Camiseta Tradicional	0.012	Azul Turquesa	Clássico	G1	2024-08-06	0.0	1.0	Giovanni			2024-08	G1_Camiseta Tradicional_Azul Turquesa_Clássico...	-1.0	Camiseta Tradicional | Clássico | Azul Turques...
9	Camiseta Tradicional	0.012	Amarelo Canário	Clássico	M	2024-08-06	0.0	50.0	Juliano			2024-08	M_Camiseta Tradicional_Amarelo Canário_Clássic...	-50.0	Camiseta Tradicional | Clássico | Amarelo Caná...

Mostrando as 10 primeiras linhas de um total de 961.

--- OCR Estoque consolidado (tabela): ---

Fonte: Elaboração própria

Imagem 4 - Estoque Consolidado Atual

	Código	Produto	Cor	Tamanho	Tecido	Saldo Atual	Status
253	505.001e	Boné	Azul Marinho		Ciclista	-700.0	Crítico
254	505.001e	Boné	Branco		Ciclista	-857.0	Crítico
255	505.001e	Boné	Preto		Ciclista	-50.0	Crítico
69	0.014	Camiseta Polo	Azul Marinho	G	Polo Clássica	-147.0	Crítico
70	0.014	Camiseta Polo	Azul Marinho	G1	Polo Clássica	-23.0	Crítico
...	...	...	...	...	...	...	...
240	503.135x	Sacola Tecido	Natural		Lonita 30x38	-2465.4	Crítico
229	503.001e	Sacola Tecido	Natural		Lonita 39x40	-600.0	Crítico
239	503.134x	Sacola Tecido	Natural		Lonita 42x41x10	-1660.0	Crítico
256	Ainda não Definido	Viseira	Amarelo Canário		Viseira Tactel	-500.0	Crítico
257	Ainda não Definido	Viseira	Branco		Viseira Tactel	-730.0	Crítico

258 rows × 7 columns

Total de linhas: 258

PNG salvo automaticamente como estoque_consolidado.png (258 linhas)

Exportar Excel | Exportar PNG (de n...

Fonte: Elaboração própria

4.6 Bloco 4: *backend*, automação de análises e *endpoints web*

O Bloco 4 do projeto tem como foco principal a automação de análises inteligentes e a disponibilização das principais funcionalidades do sistema por meio de um *backend* acessível via API. Esse bloco possibilita a flexibilidade e o alcance do sistema, permitindo que suas ferramentas sejam utilizadas por aplicações externas, como *front-ends* personalizados ou plataformas de visualização.

Entre as principais funcionalidades desenvolvidas nesse bloco, destacam-se:

Servidor *Flask* para APIs

O núcleo do bloco é a implementação de um servidor *web* baseado em *Flask*. Esse servidor oferece múltiplos *endpoints*, cada um responsável por uma operação específica do sistema. Dentre os principais serviços disponibilizados via API, destacam-se:

- *Dashboards* dinâmicos: geração automática de gráficos (saldo atual, movimentação diária, visão geral do estoque) de acordo com os parâmetros recebidos na requisição;

- Alertas automáticos: *endpoint* dedicado à consulta rápida de itens com estoque crítico, facilitando a gestão proativa do estoque;

- Previsão de demanda: serviço que utiliza os dados históricos para projetar a necessidade futura de cada Stock Keeping Unit (SKU), Unidade de Manutenção de Estoque em português;

- Relatórios consolidados: geração de relatórios textuais e estruturados, reunindo informações como itens críticos, tendências e previsões;

- Consulta de estoque consolidado: retorno do estoque atual agrupado por código e atributos, pronto para análise ou exportação.

Integração e Automação de Fluxos:

Por meio desses *endpoints*, o sistema passa a ser acessível não apenas pelo ambiente Colab, mas também por outros *softwares*, ferramentas de automação (como *Make* ou *Zapier*), e futuras interfaces *front-end*. Isso significa que as análises, relatórios e recomendações

podem ser consumidos automaticamente por qualquer aplicação capaz de fazer requisições HTTP, ampliando o potencial de automação e integração do sistema com outros processos da empresa.

Automação de Análises Inteligentes:

Além das consultas tradicionais, o bloco também implementa mecanismos de automação para análise de *dashboards* e relatórios gerados pelo próprio sistema. Utilizando o OCR para extrair texto de imagens de *dashboards*, o conteúdo visual é enviado para análise automática por modelos de linguagem avançados, como GPT, que retornam insights, alertas e recomendações diretamente para o usuário ou aplicação consumidora.

Prontidão para Escalabilidade e Customização:

O *backend* foi planejado para ser escalável e facilmente adaptável. Novos *endpoints* podem ser adicionados conforme surgirem novas necessidades, mantendo o sistema pronto para evoluir junto ao negócio ou à pesquisa. A abordagem modular do código facilita tanto a manutenção quanto a expansão das funcionalidades.

O Bloco 4 cumpre o papel de transformar as análises e operações internas do sistema em serviços acessíveis e reaproveitáveis, tornando possível a automação de processos, a integração com outros sistemas e a utilização dos recursos de inteligência artificial de forma transparente e eficiente. Dessa maneira, a solução deixa de ser apenas uma ferramenta interna de análise e passa a atuar como uma plataforma inteligente, apta a ser utilizada em diferentes cenários e aplicações dentro do contexto empresarial.

Resultado do bloco

Esse bloco não possui um resultado visível em si, ele é utilizado para montar automações de processos, integração via API e configurações da integração com a IA. Assim, é utilizado para configurações internas das APIs.

Na prática, a abertura de *endpoints* para consumo externo representa um diferencial competitivo, possibilitando a integração contínua com novos fluxos de trabalho, aplicações e automações, de acordo com as demandas da empresa.

4.7 Bloco 5: ocr, ia para análise automática dos *dashboards* e publicação de apis com *ngrok*

O Bloco 5 do sistema tem como foco a automação avançada das análises, aliando recursos de OCR (Optical Character Recognition), inteligência artificial e publicação de APIs para permitir diagnósticos rápidos, inteligentes e acessíveis dos dados de estoque.

Principais funcionalidades do Bloco 5:

Extração de dados via OCR

A primeira etapa deste bloco consiste em aplicar o OCR, por meio da biblioteca *pytesseract*, sobre imagens de *dashboards* e relatórios consolidados do estoque. Isso permite converter o conteúdo visual, como tabelas e gráficos salvos em PNG, em texto editável. Com essa abordagem, o sistema passa a analisar informações resumidas dos *dashboards*, otimizando recursos e acelerando o diagnóstico, já que processar imagens consolidadas demanda menos uso de API e recursos computacionais.

Análise automática por Inteligência Artificial

Após a extração do texto, essas informações são encaminhadas para modelos de linguagem como o GPT-3.5-turbo[22]. A IA interpreta o conteúdo extraído, identifica padrões, reconhece situações críticas de estoque e elabora recomendações ou alertas automáticos. O processo ocorre de maneira automatizada, proporcionando diagnósticos claros e objetivos, mesmo para usuários sem domínio técnico em análise de dados.

Disponibilização de *endpoints* para análise automática

O sistema implementa *endpoints* **web** (através do *Flask*) capazes de receber imagens de *dashboards* via *upload*. Assim que um arquivo é enviado, o *backend* executa o OCR, processa os dados por meio da IA e retorna os *insights* diretamente ao usuário. Essa solução torna a análise visual.

Publicação do *backend* via *Ngrok*

Para garantir que o sistema possa ser testado e validado remotamente, o bloco utiliza o *Ngrok* para criar um link público temporário para os *endpoints* da API. Isso facilita não só os

[22] Disponível em: https://platform.openai.com/docs/models/gpt-3-5

testes e demonstrações do sistema, mas também sua integração com plataformas externas ou fluxos de automação, sem depender de infraestrutura própria de servidores.

Ao automatizar a análise dos dashboards, o Bloco 5 garante que o gestor tenha acesso a diagnósticos e recomendações baseados em dados, sem a necessidade de consultas manuais ou processamento local. Além disso, a publicação das APIs por *Ngrok* proporciona flexibilidade e praticidade tanto para experimentação quanto para integração do sistema em diferentes contextos de uso.

Resultado do bloco:

Neste *output*, o sistema utiliza o OCR (*Optical Character Recognition*) para extrair automaticamente os dados de *dashboards* salvos como imagens (em formato PNG) e, em seguida, submete essas informações a modelos de linguagem natural (LLMs), integrando inteligência artificial para análise e geração de *insights* sobre o estoque. Além disso, por meio do *Flask* e do *Ngrok*, as rotinas analíticas podem ser acessadas remotamente via APIs, facilitando a integração com outras plataformas ou automações externas.

Requirement already satisfied: pytesseract in /usr/local/lib/python3.11/dist-packages (0.3.13)

Requirement already satisfied: pillow in /usr/local/lib/python3.11/dist-packages (11.2.1)

Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.11/dist-packages (from pytesseract) (24.2)

⏳ Aguardando Flask subir…

* Serving Flask app '__main__'

* Debug mode: off

INFO:werkzeug:**WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.**

* Running on all addresses (0.0.0.0)

* Running on http://127.0.0.1:5000

* Running on http://172.28.0.12:5000

INFO:werkzeug:Press CTRL+C to quit

Authtoken saved to configuration file: /root/.config/ngrok/ngrok.yml

🔗 API disponível em: NgrokTunnel: "https://4ded-34-125-80-14.ngrok-free.app" -> "http://localhost:5000"

4.8 Bloco 6: testes

Este bloco foi desenvolvido para validar, testar e demonstrar todas as funcionalidades implementadas nos blocos anteriores, assegurando que o sistema de gestão de estoque opere conforme o esperado. Trata-se de uma etapa fundamental para garantir robustez, confiabilidade e alinhamento com os objetivos do projeto.

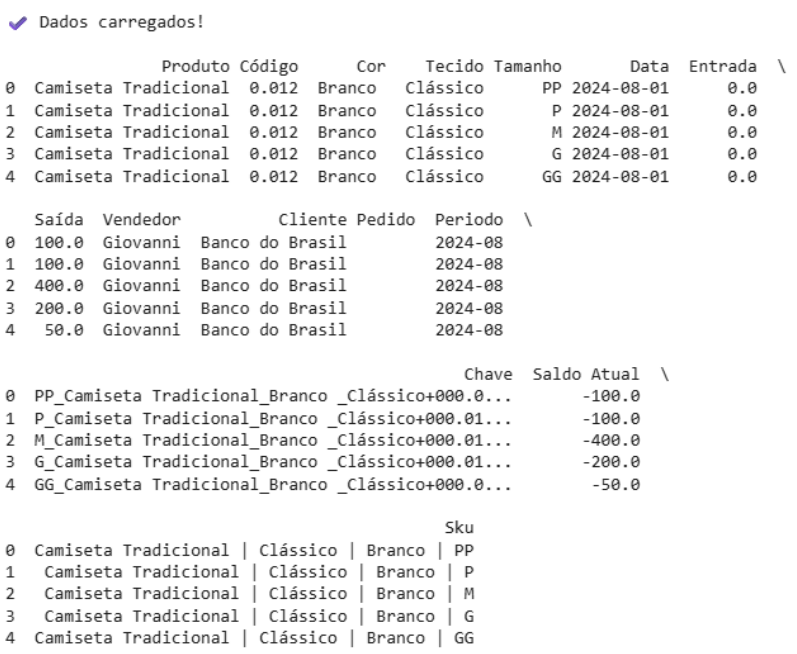
Teste 1. Carregamento de Dados:

Nesta etapa, o sistema executa rotinas que comprovam a correta integração com o Google Sheets ou arquivos CSV, carregando os dados de estoque e verificando se todas as informações essenciais estão disponíveis e estruturadas de acordo com o planejado. Esse teste é importante para identificar eventuais falhas de conexão, problemas de autenticação ou inconsistências nos dados originais.

Resultado teste 1:

Após a execução do bloco de carregamento, o sistema retorna como *output* uma amostra inicial do DataFrame, exibindo, geralmente, as primeiras linhas da matriz de dados. Este extrato inclui variáveis presentes no banco de dados.

Imagem 5 - Resultado Teste 1



✔ Dados carregados!

	Produto	Código	Cor	Tecido	Tamanho	Data	Entrada	Saída	Vendedor	Cliente	Pedido	Periodo	Chave	Saldo Atual	Sku
0	Camiseta Tradicional	0.012	Branco	Clássico	PP	2024-08-01	0.0	100.0	Giovanni	Banco do Brasil		2024-08	PP_Camiseta Tradicional_Branco _Clássico+000.0...	-100.0	Camiseta Tradicional | Clássico | Branco | PP
1	Camiseta Tradicional	0.012	Branco	Clássico	P	2024-08-01	0.0	100.0	Giovanni	Banco do Brasil		2024-08	P_Camiseta Tradicional_Branco _Clássico+000.01...	-100.0	Camiseta Tradicional | Clássico | Branco | P
2	Camiseta Tradicional	0.012	Branco	Clássico	M	2024-08-01	0.0	400.0	Giovanni	Banco do Brasil		2024-08	M_Camiseta Tradicional_Branco _Clássico+000.01...	-400.0	Camiseta Tradicional | Clássico | Branco | M
3	Camiseta Tradicional	0.012	Branco	Clássico	G	2024-08-01	0.0	200.0	Giovanni	Banco do Brasil		2024-08	G_Camiseta Tradicional_Branco _Clássico+000.01...	-200.0	Camiseta Tradicional | Clássico | Branco | G
4	Camiseta Tradicional	0.012	Branco	Clássico	GG	2024-08-01	0.0	50.0	Giovanni	Banco do Brasil		2024-08	GG_Camiseta Tradicional_Branco _Clássico+000.0...	-50.0	Camiseta Tradicional | Clássico | Branco | GG

Fonte: Elaboração própria

Teste 2. Demonstração do *Dashboard* Interativo:

O presente bloco realiza a execução do *dashboard* interativo, permitindo ao usuário experimentar os filtros, visualizar os gráficos e analisar os dados consolidados em tempo real. A demonstração reforça a usabilidade e a praticidade do sistema, confirmando que a experiência proposta nos *dashboards* corresponde à realidade do usuário.

Resultado teste 2:

A correta apresentação dos *dashboards* confirma a eficácia dos processos de integração e manipulação dos dados, além de evidenciar o valor do sistema para a gestão do estoque. A possibilidade de aplicar filtros e visualizar informações consolidadas facilita a identificação de gargalos, tendências de consumo e oportunidades de reposição, alinhando-se diretamente ao propósito do trabalho de tornar a gestão de estoque mais visual e analítica.

Imagem 6 - Resultado Teste 2

Fonte: Elaboração própria

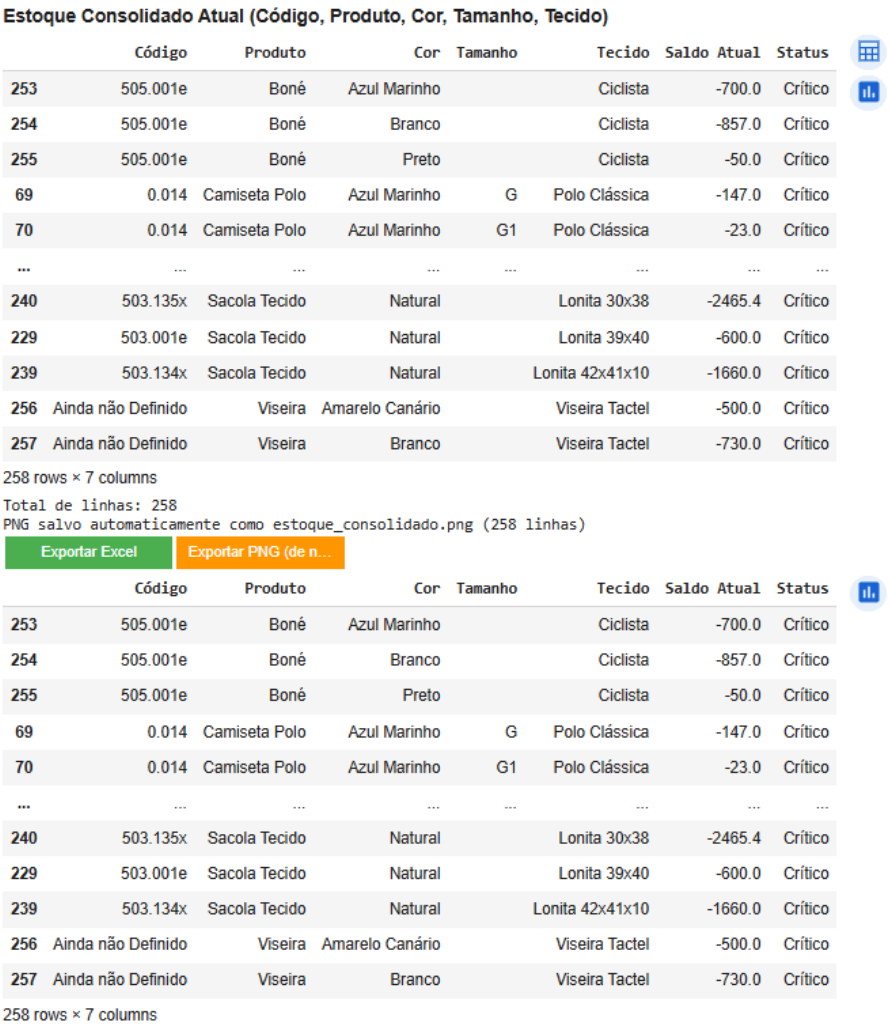
Teste 3. Visualização consolidada do estoque:

Dentre os testes realizados, este busca trazer, de forma visual, o estoque atual, demonstrando os itens em estado crítico e sua quantidade atual, adicionalmente, traz a possibilidade de exportar esse relatório tanto em Excel quanto em PNG.

Resultado teste 3:

A validação dessa funcionalidade é fundamental para assegurar que o sistema seja capaz de entregar informações específicas conforme a necessidade do usuário. Tornando o acompanhamento do estoque mais preciso e aderente à realidade.

Imagem 7 - resultado teste 3



Estoque Consolidado Atual (Código, Produto, Cor, Tamanho, Tecido)

	Código	Produto	Cor	Tamanho	Tecido	Saldo Atual	Status
253	505.001e	Boné	Azul Marinho		Ciclista	-700.0	Crítico
254	505.001e	Boné	Branco		Ciclista	-857.0	Crítico
255	505.001e	Boné	Preto		Ciclista	-50.0	Crítico
69	0.014	Camiseta Polo	Azul Marinho	G	Polo Clássica	-147.0	Crítico
70	0.014	Camiseta Polo	Azul Marinho	G1	Polo Clássica	-23.0	Crítico
...	...	...	...	...	...	...	...
240	503.135x	Sacola Tecido	Natural		Lonita 30x38	-2465.4	Crítico
229	503.001e	Sacola Tecido	Natural		Lonita 39x40	-600.0	Crítico
239	503.134x	Sacola Tecido	Natural		Lonita 42x41x10	-1660.0	Crítico
256	Ainda não Definido	Viseira	Amarelo Canário		Viseira Tactel	-500.0	Crítico
257	Ainda não Definido	Viseira	Branco		Viseira Tactel	-730.0	Crítico

258 rows × 7 columns

Total de linhas: 258
PNG salvo automaticamente como estoque_consolidado.png (258 linhas)

Exportar Excel | Exportar PNG (de n...

	Código	Produto	Cor	Tamanho	Tecido	Saldo Atual	Status
253	505.001e	Boné	Azul Marinho		Ciclista	-700.0	Crítico
254	505.001e	Boné	Branco		Ciclista	-857.0	Crítico
255	505.001e	Boné	Preto		Ciclista	-50.0	Crítico
69	0.014	Camiseta Polo	Azul Marinho	G	Polo Clássica	-147.0	Crítico
70	0.014	Camiseta Polo	Azul Marinho	G1	Polo Clássica	-23.0	Crítico
...	...	...	...	...	...	...	...
240	503.135x	Sacola Tecido	Natural		Lonita 30x38	-2465.4	Crítico
229	503.001e	Sacola Tecido	Natural		Lonita 39x40	-600.0	Crítico
239	503.134x	Sacola Tecido	Natural		Lonita 42x41x10	-1660.0	Crítico
256	Ainda não Definido	Viseira	Amarelo Canário		Viseira Tactel	-500.0	Crítico
257	Ainda não Definido	Viseira	Branco		Viseira Tactel	-730.0	Crítico

258 rows × 7 columns

Fonte: Elaboração própria

Teste 4. Funções de filtro, alerta, e consolidado:

Neste teste, são executadas rotinas que produzem visualizações e documentos a partir dos dados filtrados ou consolidados, como: saldo por produto, filtros e estoque crítico. O *output,* apresentado na Tabela 5, demonstra que o sistema permite ao usuário realizar filtros dinâmicos por diferentes variáveis.

Resultado teste 4:

O resultado desse teste evidencia que o sistema é capaz de entregar informações valiosas para a tomada de decisão. Assim, a possibilidade de filtrar e visualizar recortes específicos do estoque facilita o monitoramento de produtos com giro elevado, identifica rapidamente itens com estoque crítico e permite um acompanhamento detalhado das movimentações.

Resultado Teste 4

Filtro por produto Camiseta:

	Produto	Código	Cor	Tecido	Tamanho	Data Entrada	\
0	Camiseta Tradicional	0.012	Branco	Clássico	PP	2024-08-01	0.0
1	Camiseta Tradicional	0.012	Branco	Clássico	P	2024-08-01	0.0
2	Camiseta Tradicional	0.012	Branco	Clássico	M	2024-08-01	0.0

	Saída	Vendedor	Cliente	Pedido	Periodo	\
0	100.0	Giovanni	Banco do Brasil		2024-08	
1	100.0	Giovanni	Banco do Brasil		2024-08	
2	400.0	Giovanni	Banco do Brasil		2024-08	

	Chave	Saldo Atual	\
0	PP_Camiseta Tradicional_Branco _Clássico+000.0...	-100.0	
1	P_Camiseta Tradicional_Branco _Clássico+000.01...	-100.0	
2	M_Camiseta Tradicional_Branco _Clássico+000.01...	-400.0	

Sku

0 Camiseta Tradicional | Clássico | Branco | PP

1 Camiseta Tradicional | Clássico | Branco | P

2 Camiseta Tradicional | Clássico | Branco | M

Alerta de baixo estoque:

	Código	Produto
0	0.012	Camiseta Tradicional
1	0.012	Camiseta Tradicional
2	0.012	Camiseta Tradicional

Sku Saldo Atual

0	Camiseta Tradicional | Clássico | Branco | PP	-100.0
1	Camiseta Tradicional | Clássico | Branco | P	-100.0
2	Camiseta Tradicional | Clássico | Branco | M	-400.0

[960 rows x 4 columns]

Estoque consolidado (API):

	Código	Produto	Cor	Tamanho	Tecido
253	505.001e	Boné	Azul Marinho		Ciclista
254	505.001e	Boné	Branco		Ciclista
255	505.001e	Boné	Preto		Ciclista
69	0.014	Camiseta Polo	Azul Marinho	G	Polo Clássica
70	0.014	Camiseta Polo	Azul Marinho	G1	Polo Clássica

	Saldo Atual	Status

253 -700.0 Crítico

254 -857.0 Crítico

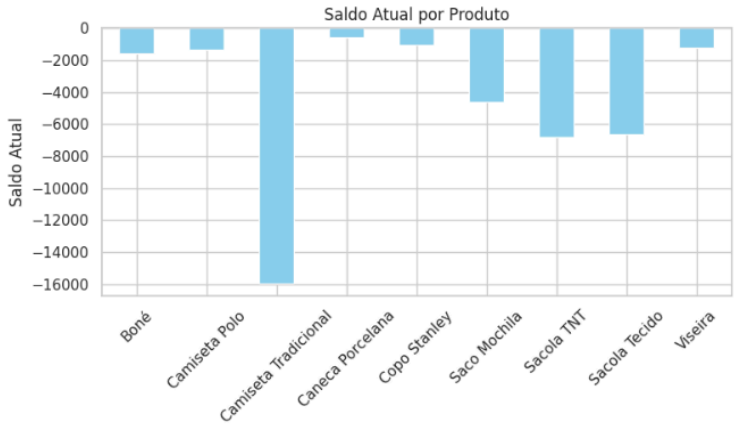
Teste 5. Geração de gráficos individuais:

O bloco atual avalia a geração de gráficos individuais, como saldo por produto, movimentação diária e visão geral do estoque, além de checar a exportação correta de relatórios. Isso certifica que todos os *outputs* visuais e documentais sejam produzidos fielmente ao que foi visualizado no sistema.

Resultado teste 5:

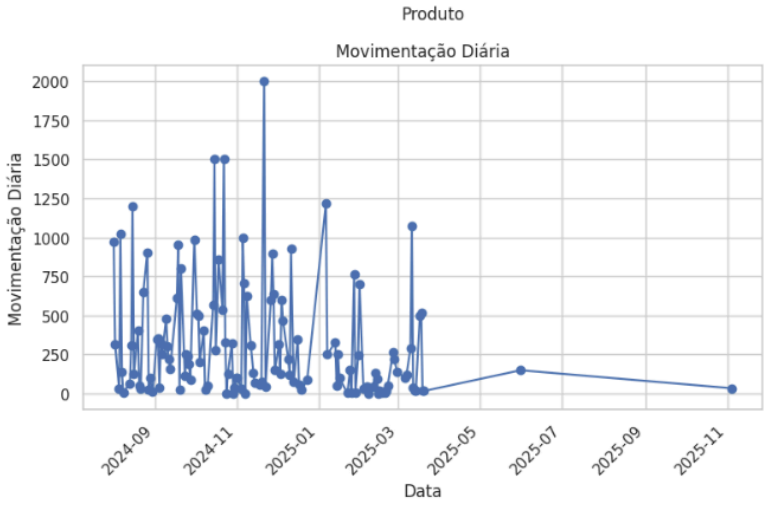
A exportação automatizada desses materiais representa um avanço significativo em termos de eficiência e padronização dos processos de gestão de estoque. O sistema garante que as informações visualizadas na plataforma sejam fielmente replicadas nos arquivos exportados, eliminando retrabalho manual e o risco de divergências de dados.

Imagem 8 - Resultado Teste 5, Saldo Atual



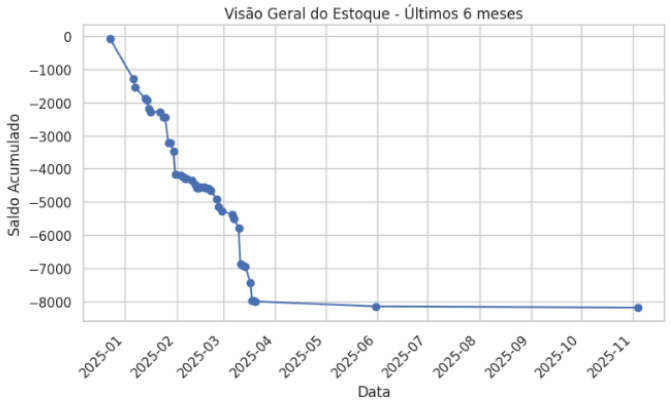
Fonte: Elaboração própria

Imagem 9 - Resultado Teste 5, Movimentação Diária



Fonte: Elaboração própria

Imagem 10 - Resultado Teste 5, Movimentação Último Semestre



Fonte: Elaboração própria

Teste 6. Previsão *Prophet*:

Com o propósito de verificar se o *prophet* está funcionando corretamente para realizar as previsões, é necessário dispor de, no mínimo, dois meses de histórico de movimentação do produto. Caso não haja dados suficientes, a ferramenta não conseguirá gerar as projeções . Quando os dados mínimos requeridos são obtidos, o sistema recebe o histórico de movimentação de um produto específico (por exemplo, “Camiseta Tradicional | Clássico | Branco | PP”) e utiliza o *Prophet* para projetar a quantidade esperada de saída para os

próximos meses. O *output* é apresentado em forma de tabela, com as datas previstas (“ds”) e os valores de demanda estimada (“yhat”).

Resultado teste 6:

Imagem 12 - Resultado Teste 6

Previsão para SKU: Camiseta Tradicional | Clássico | Branco | PP

ds	yhat
2024-08-01	44,39
2024-09-01	58,89
2024-10-01	72,92
2024-11-01	87,42
2024-11-30	100,98
2024-12-31	115,48
2025-01-31	129,98

Fonte: Elaboração própria

Interpretação:

- Para cada data futura (no início/fim de cada mês), o Prophet está prevendo a quantidade de demanda esperada (yhat) para o SKU *Camiseta Tradicional* | *Clássico* | *Branco* | *PP*;
- Por exemplo, para 01/08/2024, a previsão de demanda é de 44 unidades (aproximadamente). Para 31/01/2025, a previsão é de cerca de 130 unidades;
- Tendência: o modelo identificou uma tendência de crescimento na demanda desse SKU ao longo dos próximos meses.

Teste 7. Relatório de estoque/alerta:

Avalia a funcionalidade do sistema na geração de relatórios gerenciais que combinam, em um único *output*, alertas automáticos de produtos em situação crítica e saldo do estoque.

Resultado teste 7:

A validação deste teste demonstra que o sistema é capaz de entregar, de forma consolidada, informações essenciais a respeito do estoque. Ao unir alertas preventivos e uma visão geral em um único relatório, o sistema facilita a rotina do gestor.

Tabela 7 - Itens com estoque abaixo de 20 unidades

⚠️ Itens com estoque abaixo de 20 unidades:

Código	Produto	SKU	Saldo Atual
0.012	Camiseta Tradicional	Camiseta Tradicional | Clássico | Branco | PP	-100
0.012	Camiseta Tradicional	Camiseta Tradicional | Clássico | Branco | P	-100
0.012	Camiseta Tradicional	Camiseta Tradicional | Clássico | Branco | M	-100
0.012	Camiseta Tradicional	Camiseta Tradicional | Clássico | Branco | G	-100

Fonte: Elaboração própria

Teste 8. Validação do OCR, IA e dos *Endpoints Web*:

Este teste verifica a funcionalidade de envio de imagens de *dashboards* para análise automática por OCR e inteligência artificial, bem como a resposta dos *endpoints web* disponibilizados para integração com plataformas externas. Essas validações são fundamentais para garantir que a camada de automação e inteligência opere de forma confiável em cenários práticos.

Resultado teste 8:

Insight IA – Alerta: há vários produtos com estoque crítico, apresentando níveis muito baixos ou negativos. Essa condição representa risco de ruptura, podendo afetar as vendas e prejudicar a experiência do cliente. Recomenda-se ação imediata para normalizar os estoques e evitar impactos no negócio.

Tendência: alguns produtos estão com saldos positivos, indicando uma boa saída e aceitação no mercado. É importante monitorar esses produtos para garantir que continuem sendo bem recebidos pelos clientes.

Ações sugeridas:

- Realizar um levantamento detalhado dos produtos com saldos críticos e fazer pedidos de reposição imediatamente;
- Analisar a demanda dos produtos com saldos positivos e considerar aumentar o estoque desses itens para atender à procura;
- Implementar um sistema de controle de estoque mais eficiente para evitar futuras situações de saldo crítico;
- Avaliar a possibilidade de promoções ou estratégias de marketing para impulsionar a venda dos produtos com saldos críticos;
- Manter uma comunicação constante com os fornecedores para garantir a disponibilidade dos produtos mais procurados.

Teste 9. *Endpoints flask* (requisições HTTP)

Avaliou-se a integração do sistema com a tecnologia *Flask* para a disponibilização de *endpoints web*, por meio dos quais é possível realizar requisições HTTP que acessam diretamente as funcionalidades implementadas para o controle de estoque. O status “200” significa que determinada requisição está pronta para operar.

Resultado teste 9:

O teste evidencia que o sistema alcançou um grau de flexibilidade, uma vez que as funcionalidades não estão restritas ao ambiente local do *notebook*, podendo ser acessadas e integradas a outras plataformas, sistemas externos ou aplicações *front-end*.

```
INFO:werkzeug:127.0.0.1 - - [19/Jun/2025 18:35:15] "GET /dashboard/saldo HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [19/Jun/2025 18:35:15] "GET /alertas HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [19/Jun/2025 18:35:15] "GET /estoque_consolidado HTTP/1.1" 200
```

Teste 10. *Endponit* de OCR+IA

Este teste segue a mesma lógica do Teste 8. No entanto, em vez de converter uma tabela em OCR para análise pela IA, será utilizado um *dashboard*, que também será transformado em OCR antes de ser processado pela inteligência artificial. Dessa forma, o executor poderá comparar os resultados e decidir se prefere realizar a análise a partir de tabelas ou de *dashboards*, conforme a base avaliativa gerada.

Resultado teste 10:

Resposta da IA:

"Com base nos dados fornecidos, é possível identificar que o saldo atual de estoque está em 16.000 unidades. Observando o saldo atual por produto, é importante notar se há algum produto com um estoque muito baixo ou muito alto em comparação com os demais.\n\nUm alerta que pode ser identificado é se algum produto estiver com um estoque muito baixo, o que pode indicar a necessidade de reabastecimento ou ajuste na previsão de demanda. Por outro lado, um estoque muito alto pode indicar um excesso de produtos que podem ficar obsoletos ou ocupar espaço desnecessário. Uma tendência que pode ser observada é se o saldo atual está aumentando ou diminuindo ao longo do tempo. Isso pode indicar se as estratégias de gestão de estoque estão sendo eficazes ou se é necessário fazer ajustes. Com base nessas informações, possíveis ações para o gestor incluem revisar as previsões de demanda, otimizar os níveis de estoque de cada produto, identificar oportunidades de redução de custos e melhorar a eficiência na gestão de estoques. É importante manter um equilíbrio saudável entre a oferta e a demanda para garantir a eficiência operacional da empresa."

O bloco de testes funciona como um ambiente controlado para garantir que todas as funcionalidades desenvolvidas operem corretamente, assegurando segurança ao usuário final e possibilitando ajustes finos antes da implantação em ambiente real. Esse processo reforça o compromisso do projeto com qualidade, transparência e eficiência, servindo tanto para auditorias quanto para demonstrar, na prática, o potencial do sistema.

4.9 Respostas llm a respeito do estoque

A aplicação da inteligência artificial generativa, integrada ao sistema de gestão de estoque via análise automatizada de dados extraídos por OCR, mostrou-se eficiente ao apontar gargalos operacionais, tendências e possíveis ações para a gestão do estoque. A comparação entre os dois métodos de entrada (tabela consolidada e *dashboard* acadêmico) evidencia diferenças importantes quanto à precisão, detalhamento e aplicabilidade prática das análises.

No teste 8, a análise foi realizada a partir de uma tabela estoque, convertida para texto por meio de OCR. A resposta da IA destacou, de forma objetiva, a existência de produtos em situação crítica, identificando riscos de ruptura que impactam diretamente as vendas e a satisfação do cliente. O modelo também reconheceu, para além disso, tendências nos itens com saldos maiores ou menores, sugerindo o monitoramento constante desses produtos. As recomendações da IA foram coerentes, englobando ações de reposição imediata e ajustes de estoque. Tais recomendações foram feitas a partir do GPT 3.5-turbo e com um *prompt* genérico englobando todo o estoque.

Em contrapartida, no teste 10, a entrada utilizada foi um *dashboard* gráfico do estoque, igualmente submetido ao OCR antes da análise pela IA. Embora a resposta tenha captado a tendência geral do saldo de estoque e levantado alertas quanto a níveis críticos ou excessivos. A perda de precisão se deve à menor densidade de informação textual no *dashboard* quando comparado à tabela, além de possíveis erros de leitura na conversão gráfica para texto, próprios das limitações técnicas do OCR nesse tipo de visualização. Assim, evidencia-se que a utilização do OCR em *dashboards* pode comprometer a riqueza analítica e a acurácia dos diagnósticos, sobretudo quando detalhes quantitativos por SKU são essenciais.

Outro fator relevante para a qualidade das análises geradas é a formulação do *prompt* enviado à IA. *Prompts* mais específicos, com perguntas bem direcionadas e dados contextualizados, tendem a resultar em respostas precisas e relevantes para a gestão. Ressalta-se ainda que os testes foram conduzidos com o modelo GPT-3.5-turbo, versão anterior dos modelos de linguagem disponíveis. Versões recentes, como o GPT-4 e suas derivações, apresentam melhorias significativas na compreensão contextual, síntese e detalhamento das respostas, sendo preferíveis para diagnósticos críticos em ambientes empresariais.

Portanto, a experiência prática demonstra que a IA aplicada à análise de estoque, principalmente quando fundamentada em dados consolidados e *prompts* bem elaborados, pode proporcionar diagnósticos automáticos confiáveis, contribuindo para decisões ágeis e alinhadas às demandas do negócio. No entanto, a adoção de visualizações (*dashboards*) como fonte primária para o OCR deve ser repensada e realizada com cautela, reconhecendo suas limitações para uso analítico.

4.10 funcionalidades entregues:

O quarto capítulo se dedicou à apresentação detalhada do processo de desenvolvimento e validação do sistema de visualização de estoques, estruturado a partir da integração de tecnologias acessíveis e inteligência artificial. Ao longo deste capítulo, foram descritos, de modo metódico e fundamentado, os caminhos trilhados desde a ideação do projeto, os desafios superados, as soluções implementadas e os testes realizados para garantir a transparência.

O percurso inicial do desenvolvimento foi marcado pela busca de ferramentas que proporcionam flexibilidade e acessibilidade. O Google Sheets foi selecionado como banco de dados central, não apenas por sua familiaridade, mas também por sua integração nativa com Python, via API gspread, possibilitando o armazenamento em nuvem e compartilhamento colaborativo dos registros de estoque. A escolha dessa plataforma conferiu ao sistema uma

base sólida e adaptável, fundamental para sustentar as etapas posteriores da visualização do estoque.

O capítulo demonstrou que o sucesso do sistema dependeu, em grande medida, de uma criteriosa preparação do ambiente computacional. A estruturação do projeto no Google Colab viabilizou o uso de recursos em nuvem, democratizando o acesso e simplificando o compartilhamento do código-fonte e das análises.

O tratamento dos dados incluiu desde a padronização de campos até a consolidação dos registros em matrizes e dashboards acadêmicos, apoiando a elaboração de relatórios customizados e a identificação rápida de situações críticas, como saldos negativos. Destaca-se que a exportação dos dashboards e tabelas, em formatos Excel e PNG, além de atender às necessidades de documentação, tornou as informações facilmente compartilháveis entre setores.

As funcionalidades de visualização, desenvolvidas no Bloco 3, mostraram-se essenciais para traduzir dados brutos em conhecimento gerencial acionável. A integração de recursos de OCR e inteligência artificial elevou o grau no auxílio da análise, tornando possível a extração de informações para análise automatizada por modelos de linguagem natural, como o GPT.

O código em flaks, possibilitou o alcance e a escalabilidade do projeto, permitindo que suas principais funcionalidades fossem consumidas via API posteriormente. O que abre caminhos para uma futura elaboração de um front-end.

O bloco de testes desempenhou papel fundamental na validação de todas as funcionalidades desenvolvidas. A realização de experimentos controlados, desde o carregamento dos dados até a geração e exportação dos relatórios, assegurou o alinhamento entre os objetivos propostos e os resultados efetivamente entregues. Testes específicos atestaram o desempenho dos módulos de OCR e análise por IA.

5 RESULTADO - OTIMIZAÇÃO ESTOQUE

A busca pela eficiência na gestão de estoques é um desafio recorrente em empresas. No contexto deste trabalho, o desenvolvimento do sistema de gestão de estoque com inteligência artificial foi orientado desde o início para não apenas registrar e analisar dados, mas, sobretudo, apoiar a tomada de decisão quanto à reposição.

Conforme descrito no item 4.1, o projeto avançou de forma incremental, priorizando soluções práticas, de baixo custo e de fácil integração. Após a escolha do Google Sheets como banco de dados central e a superação dos desafios iniciais de automação e visualização dos dados, tornou-se evidente a necessidade de incorporar uma ferramenta capaz de sugerir políticas de reposição de maneira automatizada e alinhada com restrições reais do negócio.

Com isso, a utilização do Gurobi como solver matemático para otimização de estoque foi fundamental para a construção de um sistema inteligente e operado por formulações matemáticas. O Gurobi possibilitou implementar um modelo capaz de determinar a quantidade ideal a ser pedida de cada item, em cada período, levando em conta fatores como custo de pedido, custo de estoque, demanda prevista, lotes mínimos, limites de capacidade e orçamento disponível. Essa abordagem possibilitou simular cenários diversos e oferecer recomendações embasadas em modelos matemáticos.

5.1 - Conexão Google Drive e leitura do arquivo

Diferentemente do capítulo 4, em que fizemos a integração do Google Sheets via API, nesta etapa adotou-se o Google Drive[23] como ambiente de armazenamento central dos arquivos. Inicialmente, foi realizada a conexão do ambiente Google Colab ao Google Drive, permitindo o acesso direto aos arquivos de planilhas necessários para as análises subsequentes. Esse procedimento garante que o fluxo de trabalho permaneça acessível e sincronizado em nuvem, favorecendo a replicabilidade dos experimentos e a colaboração.

Após a configuração do ambiente, a etapa seguinte consistiu na leitura da base de dados de estoque, armazenada em formato Excel. Para isso, utilizou-se a biblioteca pandas. O arquivo foi carregado a partir da pasta definida no Google Drive, permitindo que o DataFrame resultante servisse como insumo principal para as análises, previsões e processos de otimização abordados nos tópicos posteriores.

Adicionalmente, visando a padronização e a integridade das informações, foi implementada uma rotina para remoção de acentos e espaços em nomes de produtos e demais campos textuais relevantes. Esta normalização dos dados, por meio de função própria baseada nos módulos “unicodedata” e “re” da linguagem Python, busca minimizar inconsistências que poderiam prejudicar agrupamentos, filtros ou análises. Dessa forma, estabeleceu-se uma base de dados limpa, consistente e pronta para servir aos modelos de previsão de demanda e otimização matemática.

[23] Dísponivel em: https://drive.google.com/

5.2 - Pré-processamento de dado

No intuito de garantir a robustez e a relevância das análises, foi conduzido um pré-processamento dos dados de estoque. Inicialmente, os registros foram filtrados de modo a considerar exclusivamente produtos classificados como “Camiseta Tradicional” ou “Camiseta Polo”, por se tratarem das principais linhas de interesse para o estudo. Essa escolha assegura que os resultados obtidos estejam alinhados à dinâmica real do negócio e à demanda mais representativa.

Para simplificar o tratamento das variáveis e facilitar as etapas subsequentes de agrupamento, criou-se a coluna “CorSimplificada”, categorizando as camisetas como “Branco” ou “Colorido”, conforme o valor informado originalmente. Da mesma forma, os tamanhos dos produtos foram segmentados entre “tamanho normal” (para as opções PP, P, M, G e GG) e “tamanho especial” (para demais graduações), permitindo análises diferenciadas para categorias de comportamento logístico e custo distintos.

A fim de viabilizar agrupamentos consistentes e eliminar possíveis ruídos causados por variações na grafia dos campos, foi construído um identificador único para cada produto. O “ProdutoID” resulta da concatenação dos principais atributos, produto, cor simplificada, tecido e tamanho simplificado, todos convertidos para minúsculas e desprovidos de acentuação. Essa padronização garante maior precisão nos cruzamentos e comparações ao longo da análise.

Adicionalmente, padronizou-se a informação de data para o formato “AnoMês” (YYYYMM), facilitando a organização dos dados em séries temporais e a posterior geração de previsões de demanda mensais.

Com isso, os dados foram agregados e reorganizados em uma estrutura matricial (*pivot table*), onde cada linha representa um produto específico e cada coluna corresponde a um determinado mês. Os valores contidos refletem o total de saídas do produto em cada período. Essa estrutura é fundamental para a aplicação dos algoritmos de previsão e otimização desenvolvidos nas etapas seguintes do trabalho, permitindo análises mais precisas e segmentadas por produto e por tempo.

5.3 - Salva tabela de demanda no drive

Com o objetivo de assegurar a persistência e a acessibilidade das informações processadas, procedeu-se à exportação da tabela agregada de demanda mensal por produto em dois formatos amplamente utilizados: CSV e Excel. Para tanto, foram definidos diretórios específicos no Google Drive para o armazenamento dos arquivos gerados, favorecendo a organização dos resultados e a fácil localização dos dados em etapas posteriores do projeto.

A matriz resultante do pré-processamento, que consolida o histórico de saídas mensais por produto, foi salva em ambos os formatos por meio das funções “to_csv” e “to_excel” da biblioteca pandas. Essa estratégia visa atender diferentes necessidades de manipulação e

compartilhamento, uma vez que o formato CSV é leve e facilmente integrável a outros sistemas, enquanto o Excel é mais adequado para análises visuais e ajustes manuais por parte do usuário.

A confirmação dos caminhos de salvamento é apresentada ao pesquisador diretamente no ambiente do notebook, conferindo transparência ao processo e garantindo que as bases de dados estejam devidamente armazenadas para futuras consultas, validações ou integrações com os módulos de previsão e otimização desenvolvidos ao longo do trabalho.

5.4 - Instalação e licença Gurobi

Com o intuito de otimizar, por meio de modelo matemático, o sistema de gestão de estoque, foi necessária a instalação e configuração do solver Gurobi no ambiente Google Colab. Inicialmente, procedeu-se à instalação da biblioteca gurobipy[24], a qual fornece as interfaces necessárias para definição, manipulação e resolução dos modelos de programação utilizados neste trabalho.

Na sequência, realizou-se a configuração da licença acadêmica fornecida pela Gurobi, por meio do Web License Service (WLS). Esse procedimento envolveu a criação de um ambiente personalizado para o solver, no qual foram inseridos os parâmetros de autenticação exigidos. Essa etapa é fundamental para garantir a conformidade com as políticas de uso da ferramenta e possibilitar a plena execução dos algoritmos de otimização.

5.5 - Preparação dos dados

Com o objetivo de alimentar o modelo de otimização com informações adequadas, procedeu-se à preparação e organização dos dados de entrada. Inicialmente, a matriz de demanda mensal por produto, previamente construída e salva em formato CSV, foi carregada e estruturada em listas e dicionários que facilitam o processamento pelo *solver* Gurobi. Foram extraídos os nomes dos produtos e dos períodos de análise, convertendo-os para formatos mais apropriados, a fim de garantir a compatibilidade com as restrições de nomenclatura do ambiente de otimização.

Considerando que o Gurobi impõe limites no comprimento dos nomes das variáveis, foi realizada a conversão dos identificadores de produtos e meses para versões abreviadas, acompanhadas por mapas de referência cruzada que asseguram a rastreabilidade dos resultados após a execução do modelo. Em seguida, as demandas mensais de cada produto foram consolidadas em um dicionário, otimizando o acesso durante a formulação das restrições e da função objetivo do modelo matemático.

Foram definidos também os parâmetros de custo associados a cada produto, tanto para os pedidos de reposição quanto para a manutenção em estoque, respeitando as especificidades de cada item analisado. Adicionalmente, foram estabelecidas as capacidades máximas de armazenagem — calculadas como um acréscimo percentual sobre o maior volume mensal

[24] https://pypi.org/project/gurobipy/

observado, bem como os orçamentos disponíveis para cada período, garantindo que as soluções propostas respeitem os limites operacionais da empresa.

Por fim, parâmetros auxiliares, como o lote mínimo de pedido, o tempo máximo de execução e o gap de otimização, foram configurados para assegurar o desempenho e a viabilidade computacional do modelo. Todo esse preparo possibilita que o algoritmo de otimização opere sobre uma base de dados confiável e bem estruturada, maximizando a utilidade prática das recomendações geradas.

5.6 - Resolução do modelo com variável Y binária e quantidade Q

A formulação do modelo de otimização de estoque foi realizada com base nos princípios de programação matemática, utilizando Gurobi para a resolução eficiente do problema. Inicialmente, foram definidas as variáveis de decisão: variáveis binárias para indicar a realização de pedidos em cada produto e período, variáveis inteiras para determinar a quantidade de itens a ser solicitada e variáveis contínuas para o controle dos níveis de estoque ao final de cada mês.

A função objetivo do modelo busca minimizar o custo total do sistema do estoque, abrangendo tanto os custos associados aos pedidos de reposição quanto os custos de manutenção dos estoques. Para garantir a aderência às restrições operacionais da empresa, foram inseridas condições que garantem que pedidos só sejam realizados com um lote mínimo, além de impor que a quantidade solicitada seja nula caso não haja pedido autorizado no período correspondente.

O balanço de estoque foi modelado de forma a refletir, com precisão, a dinâmica de entradas e saídas mensais de cada produto, permitindo acompanhar a evolução do estoque em toda a série histórica analisada. Restrições adicionais foram implementadas para limitar a capacidade máxima de armazenamento, bem como para garantir que o custo dos pedidos realizados em cada mês permaneça dentro do orçamento disponível para o período.

Por fim, parâmetros de otimização como o tempo máximo de execução e o gap de solução foram configurados, assegurando que o modelo retorne soluções de alta qualidade em prazos razoáveis, compatíveis com a necessidade prática da gestão empresarial. Tal abordagem proporciona à organização um mecanismo robusto e automatizado de tomada de decisão, alinhando a gestão de estoques às melhores práticas recomendadas pela literatura de pesquisa operacional. Fórmula matemática descrita de forma clara dentro do capítulo 2.1.3.

5.7 - Solução e relatório resumido

Após a definição e parametrização do modelo matemático, a resolução do problema de otimização foi conduzida por meio do comando de execução do Gurobi, responsável por encontrar a melhor solução possível diante das restrições e objetivos estabelecidos. O procedimento de otimização processa todas as combinações de produtos e períodos,

determinando os momentos ideais para a realização de pedidos, bem como as quantidades a serem solicitadas e os estoques finais a serem mantidos em cada ciclo.

Com o objetivo de conferir transparência e facilitar a interpretação dos resultados, foi implementado um relatório resumido que apresenta o custo ótimo total encontrado pelo modelo, além de detalhar, para cada produto, os períodos em que foram emitidos pedidos de reposição. Para esses casos, são destacados o valor da variável binária indicativa do pedido, a quantidade efetivamente solicitada e o nível de estoque projetado para o final do respectivo mês.

Essa abordagem permite ao gestor não apenas visualizar a economia proporcionada pelo modelo de otimização, mas também compreender, de forma granular, o comportamento ideal de reposição para cada item do portfólio ao longo do horizonte analisado. Os resultados obtidos, portanto, servem como base para a tomada de decisão, promovendo uma gestão de estoques mais eficiente, embasada em dados e alinhada às melhores práticas de pesquisa operacional.

Resultado Solução e relatório resumido:
Sendo

- Y: quantidade do lote a ser comprado;
- q: Quantidade unitária da compra;
- I: Sobra mensal do estoque.

```
Custo ótimo: 397,313.88
Produto: camisetapolo_branco_poloclassica_tamanhonormal
  2024-10: y = 1, q = 110, I = 100
Produto: camisetapolo_colorido_poloclassica_tamanhoespecial
  2024-08: y = 1, q = 100, I = 85
Produto: camisetapolo_colorido_poloclassica_tamanhonormal
  2024-08: y = 1, q = 131, I = 0
  2024-09: y = 1, q = 100, I = 0
  2024-10: y = 1, q = 100, I = 40
  2024-11: y = 1, q = 100, I = 15
  2024-12: y = 1, q = 100, I = 0
  2025-01: y = 1, q = 246, I = 0
  2025-02: y = 1, q = 135, I = 61
Produto: camisetapolo_colorido_polopremium_tamanhoespecial
  2024-09: y = 1, q = 100, I = 98
```

5.8 - Funcionalidades entregues

O capítulo 5 consolidou a aplicação prática do modelo de otimização de estoque, evidenciando como a integração entre técnicas de ciência de dados e métodos matemáticos avançados pode transformar a gestão operacional de empresas. Inicialmente, optou-se pelo uso do Google Drive como ambiente de armazenamento centralizado. Por meio do Google

Colab, foi realizada a leitura eficiente da base de dados, com posterior normalização textual para evitar inconsistências durante as análises.

O pré-processamento dos dados foi fundamental para garantir a qualidade e relevância das informações utilizadas na modelagem. Esse processo envolveu a filtragem dos produtos mais representativos (“Camiseta Tradicional” e “Camiseta Polo”), a criação de variáveis simplificadas para cor e tamanho, e o desenvolvimento de identificadores padronizados para os itens. Com a agregação mensal das saídas de estoque, os dados foram organizados em uma matriz (*pivot table*), proporcionando uma visão temporal detalhada por produto, estrutura essencial para alimentar os algoritmos de previsão e otimização.

Buscando flexibilidade e segurança nos registros, a tabela de demanda mensal foi salva em formatos CSV e Excel, facilitando o acesso, à conferência e o reaproveitamento dos dados em diferentes etapas do projeto.

Na preparação dos dados para otimização, foi elaborado índices reduzidos, dicionários de custos e parâmetros operacionais (como orçamento mensal e lote mínimo de compra), assegurando que todas as informações estivessem compatíveis com os requisitos técnicos do Gurobi e refletissem as restrições reais do negócio.

A formulação do modelo de otimização contemplou variáveis binárias para a decisão de pedidos, variáveis inteiras para a quantidade a ser solicitada e variáveis contínuas para o controle dos estoques finais. A função objetivo foi estruturada para minimizar o custo global do sistema, somando custos de pedidos e de manutenção em estoque, respeitando restrições como lote mínimo, balanço de estoque, capacidade máxima do armazém e limites orçamentários mensais.

A resolução do modelo ocorreu de forma eficiente e transparente, culminando na geração de um relatório detalhado que apresenta o custo ótimo obtido, os meses em que foram realizados pedidos, as quantidades solicitadas e o saldo de estoque mensal para cada produto analisado. O relatório evidencia, de maneira clara e acessível, as principais recomendações geradas pelo modelo, possibilitando ao gestor visualizar oportunidades de economia, evitar excessos e otimizar a tomada de decisão de reposição.

Em síntese, o capítulo 5 demonstrou, por meio de uma sequência lógica de preparação, modelagem e análise de resultados, como otimização matemática pode ser aplicada de forma integrada à gestão de estoques, elevando o patamar de suporte à decisão dentro do ambiente empresarial.

6 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

6.1 Objetivos atendidos da pesquisa

Este trabalho teve, como objetivo geral, desenvolver um sistema de gestão de estoque com inteligência artificial generativa, visando não só auxiliar o tomador de decisão do estoque, mas também inovar na automação de diagnósticos. De modo geral, o sistema cumpriu seu propósito central, reunindo automação, análise inteligente de dados e integração com bases acessíveis e populares, como o Google Sheets.

Em relação aos objetivos específicos:

- **Coleta de dados históricos:**

  Foi possível consolidar uma base real de dados de movimentações de estoque, contemplando entradas, saídas e atributos relevantes dos itens, o que permitiu alimentar e testar os módulos analíticos do sistema.

- **Modelagem do fluxo do usuário:**

  O fluxo de interação entre usuário e sistema foi mapeado e executado, partindo do registro dos itens até a geração de *dashboards acadêmicos*, relatórios e *insights*, garantindo coerência lógica e operacional à solução.

- **Projeto de uma interface *web* intuitiva e integração via *no-code*:**

  Este objetivo foi atingido apenas de forma parcial. A proposta inicial contemplava o desenvolvimento de uma interface *web* própria, capaz de abstrair a complexidade do *backend* e proporcionar uma experiência mais intuitiva ao usuário. Contudo, a versão final do sistema foi totalmente implementada no ambiente Google Colab, exigindo a execução de comandos por meio de células de código interativo. A ausência de um *front-end* dedicado impossibilitou a criação de uma interface gráfica amigável, o que representa uma limitação significativa para usuários sem familiaridade com programação ou com ambientes de *notebooks* interativos. Essa restrição pode impactar a escalabilidade e a adoção do sistema em contextos mais amplos, uma vez que reduz sua acessibilidade para públicos não técnicos.

- **Testes práticos e avaliação da plataforma:**

Os testes dos módulos implementados foram realizados com sucesso, validando funcionalidades como a coleta e análise de dados, dashboards, OCR e automação de *insights* por IA. No entanto, como a plataforma não foi concluída com uma interface gráfica ou *front-end*, os testes foram restritos ao ambiente do Colab, sem uma experiência de uso final comparável a sistemas comerciais.

6.2 Limitações da pesquisa

Realizar este trabalho foi, sem dúvida, uma experiência transformadora em termos de aprendizado prático e maturidade acadêmica. Ao longo do desenvolvimento, este autor enfrentou questões que vão além do código-fonte, passando pelas decisões técnicas, pelas limitações do contexto e, principalmente, pelas consequências dessas escolhas na experiência do usuário. Dessa forma, demonstra-se como a teoria e prática podem andar em vias separadas, de início, a ideação parecia simples, no decorrer do projeto, porém, as limitações foram surgindo.

A principal restrição do sistema é, sem dúvida, a ausência de uma interface dedicada. Por mais que o Google Colab atenda às necessidades de prototipação, a solução acaba exigindo conhecimentos técnicos que, no âmbito da gestão, em muitas empresas, não fazem parte do cotidiano. Isso restringe, consideravelmente, o alcance do sistema e, de certa forma, distancia o projeto de um cenário de adoção real em empresas fora do ambiente acadêmico ou tecnológico.

Outro aspecto que merece destaque diz respeito à infraestrutura escolhida. O uso de ferramentas gratuitas, como Google Sheets e Colab, em conjunto com Gurobi como ferramenta paga, foi decisivo para a viabilidade inicial do projeto, porém traz, consigo, delimitações que ficaram mais claras conforme os módulos foram implementados. A manipulação de grandes volumes de dados, por exemplo, não é tão fluida quanto seria em soluções empresariais robustas e, alguns processos, principalmente os que envolvem integrações com APIs, OCR e IA generativa, acabam sujeitos a instabilidades ou à necessidade de ajustes frequentes. Essas dificuldades não invalidam o sistema, mas deixam evidente que, para escalar ou integrar com outras plataformas, seriam necessários investimentos adicionais e maior planejamento estrutural.

Do ponto de vista da usabilidade, fica evidente que o projeto cumpriu seu papel enquanto “prova de conceito” e, acima de tudo, como plataforma para testes e experimentação. Ainda assim, para transformar essa base em um produto voltado ao mercado,

seria fundamental investir em automação completa das etapas, simplificação das configurações e uma experiência de usuário mais fluida e acessível.

Vale observar também que, apesar dos resultados obtidos com dados reais, a solução ainda não foi validada em larga escala, nem testada com usuários finais de diferentes perfis. Essa etapa, que demanda mais tempo e recursos, é essencial para identificar pontos de melhoria que não aparecem em ambiente controlado ou simulado.

Em resumo, este autor reconhece que o sistema atinge os objetivos principais do trabalho e apresenta inovações relevantes, mormente, ao demonstrar como recursos acessíveis de inteligência artificial podem ser aplicados na gestão de estoques. Contudo, as limitações técnicas e de usabilidade apontam para a necessidade de evoluir o projeto, seja na direção de uma interface amigável, seja na busca por maior robustez e integração com sistemas já adotados no mercado. Mais do que uma limitação, essa percepção representa, igualmente, um convite à continuidade do desenvolvimento e à colaboração interdisciplinar.

6.3 Sugestões de trabalhos futuros

Considerando as etapas abordadas no trabalho e as continências, existem diversos caminhos para o desenvolvimento futuro do sistema. Um dos principais pontos diz respeito à criação de uma interface gráfica própria, *web* e/ou *mobile*, que permita, a qualquer usuário, mesmo com um conhecimento básico em sistemas, operar a ferramenta de maneira intuitiva. Tal avanço não apenas aumentaria a acessibilidade, como facilitaria a manutenção do estoque por meio dos colaboradores.

No contexto logístico, há oportunidades claras para integrar o acompanhamento de transportes, fretes e recebimentos de mercadorias. Funcionalidades como rastreamento em tempo real de pedidos, monitoramento do *status* de entregas, cálculo de custos logísticos e geração automática de alertas para atrasos podem transformar o sistema em uma plataforma completa de gestão, não restrita ao estoque físico, mas conectada a toda a cadeia de suprimentos.

É interessante o desenvolvimento de integrações nativas ou via APIs com ERPs e plataformas de *marketplace*, facilitando a importação automática de notas fiscais, a conciliação de recebimentos e o cruzamento de dados entre vendas, estoque e logística. Além disso, um ponto que pode ser explorado é o algoritmo, aprimorando o *Machine Learning* visando deixar as previsões mais precisas.

Por fim, seria importante submeter o sistema a testes com usuários de diferentes perfis e setores, coletando *feedbacks* para ajustes de usabilidade, performance e segurança. Estudos de caso em empresas de portes variados, inclusive em ambientes de alta complexidade logística, contribuíram para validar o modelo e apontar melhorias práticas.

Em resumo, há um vasto campo para evolução do sistema, tanto no aperfeiçoamento das funcionalidades já existentes quanto na ampliação para áreas logísticas e possíveis integrações. Esses avanços aumentariam o valor prático da solução e, para além, seu potencial de impacto em diferentes segmentos do mercado.

REFERÊNCIAS

AGGARWAL, C. C. **Recommender Systems: The Textbook.** Springer, 2016. Disponível em: https://link.springer.com/book/10.1007/978-3-319-29659-3 . Acesso em: 16 jun. 2025.

ALBAYRAK ÜNAL, Özge; ERKAYMAN, Burak; USANMAZ, Bilal. **Applications of artificial intelligence in inventory management: a systematic review of the literature.** Archives of Computational Methods in Engineering, 2023. Disponível em: https://link.springer.com/article/10.1007/s11831-022-09879-5. Acesso em: 16 jun. 2025.

ALPAYDIN, E. **Introduction to Machine Learning**. Cambridge: MIT Press, 2020. Disponível em: https://books.google.com.br/books?hl=pt-BR&lr=&id=uZnSDwAAQBAJ&oi=fnd&pg=PR7&dq=ALPAYDIN,+E.+Introduction+to+Machine+Learning.+Cambridge:+MIT+Press,+2020.&ots=xOvRsuMovX&sig=IlG1Tk-_cggDLSx5dPKjpDewypY#v=onepage&q=ALPAYDIN%2C%20E.%20Introduction%20to%20Machine%20Learning.%20Cambridge%3A%20MIT%20Press%2C%202020.&f=false . Acesso em: 16 jun. 2025.

BALLOU, Ronald H**. Logística empresarial: transportes, administração de materiais e distribuição física**. 1. ed. rev. e atual. São Paulo: Atlas, 2015. Disponível em: https://repositorio.usp.br/item/003226134 . Acesso em: 16 jun. 2025.

BELFIORE, Patrícia Prado; COSTA, Oswaldo Luiz do Valle; FÁVERO, Luiz Paulo Lopes. **Um modelo de estoque e roteirização com demanda determinística e estocástica.** Gestão da Produção Operações e Sistemas, v. 4, n. 2, p. 67-83, 2009. Disponível em: https://revista.feb.unesp.br/gepros/article/view/173/121. Acesso em: 18 dez. 2024.

BEZERRA DE MENEZES, Adriana Mendes Pereira. **Uso da tecnologia da informação nas micro e pequenas empresas do setor de confecção do estado do Rio de Janeiro**. 2007. 150 f. Dissertação (Mestrado em Engenharia de Sistemas e Computação) – Coordenação dos Programas de Pós-Graduação de Engenharia, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2007. Disponível em: https://cos.ufrj.br/uploadfile/publicacao/1955.pdf. Acesso em: 16 jun. 2025.

BOWERSOX, D. J.; CLOSS, D. J.; COOPER, M. B. **Gestão logística de cadeias de suprimentos**. São Paulo: Bookman Editora, 2014.

BROWN, T. B.; et al. **Language Models are Few-Shot Learners.** Advances in Neural Information Processing Systems (NeurIPS), 2020. Disponível em: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf . Acesso em: 16 jun. 2025.

BROWNLEE, J**. Introduction to time series forecasting with python: how to prepare data and develop models to predict the future**. Machine Learning Mastery,, 2017.Disponível em: https://books.google.com.br/books?hl=pt-BR&lr=&id=-AiqDwAAQBAJ&oi=fnd&pg=PP1&dq=BROWNLEE,+J.+Introduction+to+Time+Series+Forecasting+with+Python.+Machine+Learning+Mastery,+2017.&ots=Xgwku1YxHq&sig=hpEXfXX5uxhFF2Xka2E5rt4aOu8#v=on

epage&q=BROWNLEE%2C%20J.%20Introduction%20to%20Time%20Series%20Forecastin g%20with%20Python.%20Machine%20Learning%20Mastery%2C%202017.&f=false .. Acesso em: 16 jun. 2025.

CAUCHICK, P. **Metodologia Científica para Engenharia**. 1ª edição. Rio de Janeiro: Elsevier Editora Ltda, 2019. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788595150805/. Acesso em: 17 dez. 2024.

CAUCHICK, Paulo. **Metodologia de Pesquisa em Engenharia de Produção e Gestão de Operações.** 3ª edição. Rio de Janeiro: Elsevier Editora Ltda, 2018. E-book. ISBN 978-85-352-9135-3. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788595150805/. Acesso em: 18 dez. 2024.

CHOPRA, Sunil; MEINDL, Peter. **Gestão da cadeia de suprimentos: estratégia, planejamento e operações**. 4. ed. São Paulo: Pearson Prentice Hall, 2011.

DEVLIN, Jacob et al. **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.** In: Proceedings of NAACL-HLT 2019, p. 4171–4186, 2019. Disponível em: https://aclanthology.org/N19-1423.pdf. Acesso em: 16 jun. 2025.

DRESCH, Aline; LACERDA, Daniel P.; JÚNIOR, José A. V. A. **Design Science Research: método de pesquisa para avanço da ciência e tecnologia**. Grupo A, 2015. E-book. ISBN 9788582605530. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788582605530/. Acesso em: 17 dez. 2024.

ERLENKOTTER, D. Ford W. **Harris's economic order quantity**. Operations Management, v. 35, n. 7, p. 9,2003. Disponível em: https://books.google.com.br/books?hl=pt-BR&lr=&id=cGzxgjWUkaYC&oi=fnd&pg=PA9&dq=ERLENKOTTER,+D.+An+early+classic+misplaced:+Ford+W.+Harris%27s+economic+order+quantity+model+of+1915.+Operations+Research&ots=1ZjCb8UkBG&sig=FR9_rW8EFHHFhwTMcKYN6LdSCag#v=onepage&q=ERLENKOTTER%2C%20D.%20An%20early%20classic%20misplaced%3A%20Ford%20W.%20Harris's%20economic%20order%20quantity%20model%20of%201915.%20Operations%20Research&f=false . Acesso em: 17 jun. 2025.

FAGBOOLA, F. et al. **Development of a web‑based platform for automating an inventory management of a small and medium enterprise**. Journal of Logistics, Informatics and Service Science, v. 11, n. 1, p. 13-26, 2022. Disponível em: https://www.researchgate.net/publication/368258907_DEVELOPMENT_OF_A_WEB-BASED_PLATFORM_FOR_AUTOMATING_AN_INVENTORY_MANAGEMENT_OF_A_SMALL_AND_MEDIUM_ENTERPRISE. Acesso em: 16 jun. 2025.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, **A. Deep Learning**. Cambridge: MIT Press, 2016.

GUROBI OPTIMIZATION, LLC. **Aftermarket Analytics selects Gurobi as exclusive partner to revolutionize supply chain optimization application development**. Beaverton (OR) & Pueblo (CO), 28 set. 2021. Disponível em: https://www.gurobi.com/news/aftermarket-analytics-selects-gurobi-as-exclusive-partner-to-revolutionize-supply-chain-optimization-application-development/. Acesso em: 16 jun. 2025.

Gurobi Optimization, LLC. **Gurobi Optimizer Reference Manual**, Version 10.0. 2023. Disponível em: https://www.gurobi.com/documentation/10.0/refman/index.html. Acesso em: 11 jun. 2025.

HELO, Petri; HAO, Yukun. **Artificial intelligence in operations management and supply chain management: an exploratory case study**. Production Planning & Control, v. 33, n. 2-3, p. 193-206, 2021. Disponível em: https://www.tandfonline.com/doi/full/10.1080/09537287.2021.1882690. Acesso em: 16 jun. 2025.

HEVNER, A. et al. **Design Science in Information Systems Research**. Management Information Systems Quarterly, v. 28, n. 1, p. 75-106, 2004. Disponível em: https://www.researchgate.net/publication/201168946_Design_Science_in_Information_Systems_Research/. Acesso em: 17 dez. 2024.

HIGLE, J. L. **Stochastic Programming: Optimization When Uncertainty Matters**. Tutorials in Operations Research INFORMS - New Orleans 2005, 2005. Disponível em: https://pubsonline.informs.org/doi/abs/10.1287/educ.1053.0016 . Acesso em: 17 jun. 2025.

HILLIER, Frederick S.; LIEBERMAN, Gerald J. **Introdução à Pesquisa Operacional. Capítulo 18: Teoria dos Estoques.** Porto Alegre: AMGH, 2013. Disponível em: https://integrada.minhabiblioteca.com.br/reader/books/9788580551198/pageid/822. Acesso em: 18 dez. 2024.

LACERDA, Daniel Pacheco et al. **Design Science Research: método de pesquisa para a engenharia de produção.** Gestão & Produção, v. 20, p. 741-761, 2013. Disponível em: https://www.scielo.br/j/gp/a/3CZmL4JJxLmxCv6b3pnQ8pq/?lang=pt . Acesso em: 17 jun. 2025.

LI, B.; MELLOU, K.; ZHANG, B.; PATHURI, J.; MENACHE, I. **Large Language Models for Supply Chain Optimization**. arXiv Preprint, arXiv:2307.03875, 2023. Disponível em: https://arxiv.org/abs/2307.03875. Acesso em: 13 jun. 2025.

LI, Zhenping; JIAO, Pengbo. **Two‐stage stochastic programming for the inventory routing problem with stochastic demands in fuel delivery.** International Journal of Industrial Engineering Computations, v. 13, n. 4, p. 507–522, 2022. DOI: 10.5267/j.ijiec.2022.7.004. Disponível em: https://pdfs.semanticscholar.org/ca5c/3269f49818add492dcc698ea06fb80d96f69.pdf. Acesso em: 17 jun. 2025.

LIN, Kevin Z.; ZHOU, Zihang; WANG, Chen. **Generative Pre-trained Transformers: A Survey**. IEEE Transactions on Neural Networks and Learning Systems, 2023. DOI: 10.1109/TNNLS.2023.3275684. Disponível em: https://ieeexplore.ieee.org/document/10134146. Acesso em: 16 jun. 2025.

LINUX FOUNDATION. **The Economic and Workforce Impacts of Open Source AI**. San Francisco: Linux Foundation, 2025. Disponível em: https://www.linuxfoundation.org/hubfs/LF%20Research/lfr_market_impact_052025a.pdf. Acesso em: 16 jun. 2025.

LIU, Pengfei et al. **Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing**. ACM Computing Surveys, v. 55, n. 9, p. 1-35, 2023. Disponível em: https://dl.acm.org/doi/full/10.1145/3560815. Acesso em: 16 jun. 2025.

MITTAL, Utkarsh. **Managing uncertainty in fashion supply chains: an AI-based analysis of demand variability and forecast precision**. International Research Journal of Engineering and Technology (IRJET), v. 11, n. 1, p. 481-488, jan. 2024. Disponível em: https://www.irjet.net/archives/V11/i1/IRJET-V11I177.pdf. Acesso em: 16 jun. 2025.

MURPHY, K. P. **Machine Learning: A Probabilistic Perspective.** Cambridge: MIT Press, 2012. Díposnivel em: https://books.google.com.br/books?hl=pt-BR&lr=&id=RC43AgAAQBAJ&oi=fnd&pg=PR7&dq=MURPHY,+K.+P.+Machine+Learning:+A+Probabilistic+Perspective.+Cambridge:+MIT+Press,+2012.&ots=unkubAPv3d&sig=y6GAc5_XHZksRfP7JI454Z4Bv9g#v=onepage&q&f=false . Acesso em: 16 jun. 2025.

OHNO, Taiichi. **Toyota Production System: Beyond Large-Scale Production**. Productivity Press, 1988.

PEFFERS, K.; et al. **A Design Science Research Methodology for Information Systems Research**. Journal of Management Information Systems, v. 24, n. 3, p. 45-77, 2007.Disponível em: https://www.tandfonline.com/doi/epdf/10.2753/MIS0742-1222240302?needAccess=true . Acesso em: 17 jun. 2025.

PEREIRA, L. C. C. **Modelo de otimização estocástica para o controle de reposição de estoque em sistema de duas camadas sob incerteza.** 2017. 88 f. Dissertação (Mestrado em Engenharia Industrial) — Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2017. Disponível em: https://www.dbd.puc-rio.br/pergamum/tesesabertas/1312439_2017_completo.pdf. Acesso em: 18 dez. 2024.

PIMENTEL, Mariano; FILIPPO, Denise; SANTOS, Thiago Marcondes dos. **Design Science Research: pesquisa científica atrelada ao design de artefatos.** Revista Brasileira de Informática na Educação, v. 3, n. 1, p. 37-61, 2020. DOI: https://doi.org/10.34627/vol3iss1pp37-61. Acesso em: 17 jun. 2025.

RADFORD, Alec et al. **Language Models are Unsupervised Multitask Learners**. OpenAI Technical Report, 2019. Disponível em: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf. Acesso em: 16 jun. 2025.

SALAH, Ahmed; HAMEED, Mohanad; KAREEM, Haider; ABD, Ali. **Implementing an automated inventory management system for small and medium-sized enterprises using open source and IoT**. Iraqi Journal for Computer Science and Mathematics, v. 4, n. 1, p. 37-50, 2023. Disponível em: https://ijcsm.researchcommons.org/cgi/viewcontent.cgi?article=1085&context=ijcsm. Acesso em: 16 jun. 2025.

SANTORO, Miguel Cezar; FREIRE, Gilberto. **Análise comparativa entre modelos de estoque.** EPUSP 2007. Disponível em: https://doi.org/10.1590/S0103-65132008000100007. Acesso em: 17 jun. 2024.

SILVER, E. A.; PYKE, D. F.; THOMAS, D. J. **Inventory and Production Management in Supply Chains**. CRC Press, 2017.

SOBHANI, S.; SINGH, N. **Automating Customer Service using LangChain: Building custom open-source GPT Chatbot for organizations**. arXiv Preprint, arXiv:2310.05421, 2023. Disponível em: https://arxiv.org/abs/2310.05421. Acesso em: 13 jun. 2025.

VASWANI, Ashish et al. **Attention is all you need. Advances in Neural Information** Processing Systems, v. 30, p. 5998-6008, 2017. Disponível em: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html. Acesso em: 16 jun. 2025.

VILARDO, Noelle Borges. **Modelagem e simulação de um sistema de estoques visando ao planejamento econômico**. 2005. Trabalho de Conclusão de Curso (Graduação em Engenharia de Produção) — Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2016. Disponível em: https://pantheon.ufrj.br/bitstream/11422/18197/1/monopoli10016573.pdf. Acesso em: 18 dez. 2024.

ZHANG, Chiyuan; BENGIO, Samy; HARDT, Moritz; RECHT, Benjamin; VALIANT, Oriol. **Understanding deep learning (still) requires rethinking generalization**. Communications of the ACM, v. 64, n. 3, p. 107-115, 2021. Disponível em: https://dl.acm.org/doi/10.1145/3446776. Acesso em: 16 jun. 2025.

APÊNDICES

Apêndice 1 - Bloco 1: instalações, imports e configurações iniciais

Instalação de todas as bibliotecas necessárias, reúne todas as importações e as constantes de configuração (caminhos, nomes de planilha, webhook, parâmetros de RAG e modelo). Dentro do Google Colab é necessário fazer a instalação toda vez que for rodar o código

```
# ——————————————————————————————————————
# Instale (ou reinstale) todas as bibliotecas necessárias
# ——————————————————————————————————————

!pip install flask flask-ngrok gspread oauth2client pandas openpyxl prophet kaleido
gurobipy matplotlib
!pip install -U flask gspread oauth2client pandas prophet matplotlib \
        chromadb tiktoken pyngrok
!pip install --upgrade langchain openai edgedb flask-ngrok PyPDF2
!pip install pyngrok
!pip install --upgrade openai==1.68.2
!pip install -U langchain-community
!pip install -U langchain-chroma
!pip install -U langchain-openai
!pip install -U langchain
!pip install pytesseract
!apt-get update
!apt-get install -y tesseract-ocr
!apt-get install -y tesseract-ocr-por  # Para suporte ao português
!pip install dataframe-image


# ——————————————————————————————————————————————————————
# Imports Gerais
# ——————————————————————————————————————————————————————

import io
import time
import logging
import os
```

```
        import pandas as pd
        import matplotlib.pyplot as plt
        from threading import Thread
        from flask import Flask, request, jsonify, send_file
        import requests
        import gspread
        from oauth2client.service_account import ServiceAccountCredentials
        import pytesseract
        from PIL import Image

        # Prophet para previsão de demanda
        from prophet import Prophet

        # Gurobi para otimização
        import gurobipy as gp

        # LangChain e OpenAI para IA e análises de dashboard
        from langchain.text_splitter import CharacterTextSplitter
        from langchain.vectorstores import Chroma
        from langchain.embeddings.openai import OpenAIEmbeddings
        from langchain.schema import Document
        from langchain.chat_models import ChatOpenAI
        from langchain.chains import RetrievalQA
        from langchain_community.vectorstores import Chroma

        # NGROK para expor o Flask
        from pyngrok import ngrok

        # Subprocesso para garantir compatibilidade openai (evita problemas de cache do
Colab)
        import importlib, sys, subprocess
        subprocess.check_call([sys.executable, "-m", "pip", "install", "--upgrade",
"openai==1.68.2"])
        importlib.invalidate_caches()
        openai = importlib.import_module("openai")
        from openai import RateLimitError, OpenAIError
        print("✅ Versão openai:", openai.__version__)
```

```
# ——————————————————————————————————————
# CONFIGURAÇÕES DE CHAVES, PATHS E VARIÁVEIS GLOBAIS
# ——————————————————————————————————————

# 🔵 NGROK: chave (copie a sua do painel do ngrok)
!ngrok config add-authtoken "SEU TOKE AQUI"

# 🟣 GUROBI: credenciais para ambiente WLS (preencha com as suas)
env = gp.Env(empty=True)
env.setParam('SUA CREDENCIAL AQUI')
env.setParam('SUA CREDENCIAL AQUI')
env.setParam('SUA CREDENCIAL AQUI', N° USER)
env.start()

# Google Sheets: path do JSON, nome da planilha e da aba
CAMINHO_JSON_SHEETS = "SEU CAMINHO JSON"
NOME_PLANILHA       = "NOME PLANILHA"
NOME_ABA            = "NOME ABA"

# RAG (vector store local para análise IA)
CHROMA_DIR          = "chroma_db"
RAG_K               = 4     # quantos docs relevantes buscar
CHUNK_SIZE          = 500   # tamanho do chunk
CHUNK_OVERLAP       = 50    # overlap dos chunks

# LLM OpenAI
LLM_MODEL           = "gpt-3.5-turbo"  # usando GPT-3.5-turbo para economizar
tokens
# OpenAI API KEY
os.environ["OPENAI_API_KEY"] = "SUA API KEY"

print("✅ Todas as dependências instaladas e chaves configuradas!")
```

Apêndice 2 - Bloco 2: funções de integração com o google sheets

Este bloco contém carregar estoque, ele vai carregar e organizar o estoque para futuras planilhas o Print, é só para validar se deu certo, se quiser pode tirar

```
# ———————————— CONFIGURAÇÕES GERAIS ————————————
    CAMINHO_JSON_SHEETS = "CAMINHO PLANILHA"   # Troque se o caminho for diferente
    NOME_PLANILHA = "NOME PLANILHA"
    NOME_ABA = "NOME DA ABA"

# ——————————— FUNÇÃO PARA CARREGAR O ESTOQUE DO SHEETS ———

    def carregar_estoque():
        """
        Carrega os dados de estoque da aba do Google Sheets em um DataFrame,
        normaliza colunas e gera colunas úteis (Saldo Atual, SKU, etc).
        """
        # 1. Autenticação e abertura da planilha
        scope = ["https://spreadsheets.google.com/feeds", "https://www.googleapis.com/auth/drive"]
        creds = ServiceAccountCredentials.from_json_keyfile_name(CAMINHO_JSON_SHEETS, scope)
        client = gspread.authorize(creds)
        sheet = client.open(NOME_PLANILHA).worksheet(NOME_ABA)
        data = sheet.get_all_records()
        df = pd.DataFrame(data)

        # 2. Normalização das colunas
        df.columns = df.columns.str.strip().str.title().str.replace(' +', ' ', regex=True)
        df["Entrada"] = pd.to_numeric(df.get("Entrada", 0), errors="coerce").fillna(0)
        df["Saída"] = pd.to_numeric(df.get("Saída", 0), errors="coerce").fillna(0)
        df["Saldo Atual"] = df["Entrada"] - df["Saída"]

        # 3. Conversão de data (se existir)
        if "Data" in df.columns:
            df["Data"] = pd.to_datetime(df["Data"], dayfirst=True, errors="coerce")
```

```
    # 4. Gera campo SKU único
    df["Sku"] = (
        df["Produto"].str.strip() + " | "
        + df["Tecido"].str.strip() + " | "
        + df["Cor"].str.strip() + " | "
        + df["Tamanho"].fillna("").astype(str)
    )

    return df

# ———————————— CARREGAR DADOS E TESTAR ————————————

df_estoque = carregar_estoque()
print(df_estoque.head())

# Exemplo: Saldo por produto
print("\nResumo rápido do saldo por produto:")
print(df_estoque.groupby("Produto")["Saldo Atual"].sum())
```

Apêndice 3 - Bloco 3: funções dashboard, filtros interativos, gráficos e funções auxiliares

Funções de gráficos: Plots para saldo, movimentação, filtros avançados.

Dashboard Interativo: O usuário pode filtrar produto, cor, tamanho, tecido, tudo via dropdown.

Salvamento de gráficos: Qualquer gráfico pode ser salvo para análise posterior ou para enviar à IA.

OCR pronto: Basta passar o caminho do PNG do gráfico e o texto é extraído para análise automatizada.

```
# ———————————— IMPORTS ————————————
import matplotlib.pyplot as plt
import seaborn as sns
from IPython.display import display, clear_output, HTML
import ipywidgets as widgets
import pandas as pd
import pytesseract
from PIL import Image

sns.set_theme(style="whitegrid")

# ———————————— FUNÇÃO CONSOLIDAR ESTOQUE ————

def estoque_consolidado(df):
    """
    Agrupa e soma o saldo disponível para cada SKU (incluindo Código).
    """
    df_group = (
        df.groupby(['Código', 'Produto', 'Cor', 'Tamanho', 'Tecido'], dropna=False)['Saldo Atual']
```

```
            .sum()
            .reset_index()
            .sort_values(['Produto', 'Cor', 'Tamanho', 'Tecido'])
        )
        df_group['Status'] = df_group['Saldo Atual'].apply(lambda x: 'Crítico' if x < 20 else 'OK')
        return df_group

# ———————— DASHBOARD INTERATIVO + SALDO ATUAL ————

    def dashboard_interativo(estoque_df):
        """
        Exibe dashboard interativo com filtros, gráficos e tabela de saldo atual.
        Permite exportar gráfico (PNG) e relatório (Excel) filtrados.
        """
        df = estoque_df.copy()
        produtos = sorted(df['Produto'].dropna().unique())
        cores = sorted(df['Cor'].dropna().unique())
        tamanhos = sorted(df['Tamanho'].dropna().unique())
        tecidos = sorted(df['Tecido'].dropna().unique())

        w_produto = widgets.SelectMultiple(options=produtos, description='Produto', layout=widgets.Layout(width='45%'), style={'description_width': 'initial'})
        w_cor = widgets.SelectMultiple(options=cores, description='Cor', layout=widgets.Layout(width='45%'), style={'description_width': 'initial'})
        w_tamanho = widgets.SelectMultiple(options=tamanhos, description='Tamanho', layout=widgets.Layout(width='45%'), style={'description_width': 'initial'})
        w_tecido = widgets.SelectMultiple(options=tecidos, description='Tecido', layout=widgets.Layout(width='45%'), style={'description_width': 'initial'})
```

```
        w_meses = widgets.IntSlider(value=4, min=1, max=24, step=1,
description='Meses', style={'description_width': 'initial'})


        botao_exportar_grafico = widgets.Button(description="Exportar Gráfico PNG",
button_style="info")
        botao_exportar_relatorio = widgets.Button(description="Exportar Relatório Excel",
button_style="success")
        output = widgets.Output()


        def filtrar_df():
            fdf = df.copy()
            if w_produto.value:
                fdf = fdf[fdf['Produto'].isin(w_produto.value)]
            if w_cor.value:
                fdf = fdf[fdf['Cor'].isin(w_cor.value)]
            if w_tamanho.value:
                fdf = fdf[fdf['Tamanho'].isin(w_tamanho.value)]
            if w_tecido.value:
                fdf = fdf[fdf['Tecido'].isin(w_tecido.value)]
            if w_meses.value:
                fdf['Data'] = pd.to_datetime(fdf['Data'])
                min_date = fdf['Data'].max() - pd.DateOffset(months=w_meses.value)
                fdf = fdf[fdf['Data'] >= min_date]
            return fdf


        def plotar_dashboard(fdf):
            saldo_atual = fdf.groupby(['Código', 'Produto', 'Cor', 'Tamanho', 'Tecido'],
dropna=False)['Saldo Atual'].sum().reset_index()
```

```
            saldo_atual['Status'] = saldo_atual['Saldo Atual'].apply(lambda x: 'Crítico' if x < 20 else 'OK')
            total_estoque = saldo_atual['Saldo Atual'].sum()
            produtos_criticos = saldo_atual[saldo_atual['Status'] == 'Crítico']

            top_saida = fdf.groupby('Produto')['Saída'].sum().sort_values(ascending=False)
            top_entrada = fdf.groupby('Produto')['Entrada'].sum().sort_values(ascending=False)

            fig, axs = plt.subplots(2, 2, figsize=(16, 10))
            saldo_produto = saldo_atual.groupby('Produto')['Saldo Atual'].sum().sort_values(ascending=False)
            saldo_produto.plot(kind='bar', color='tab:blue', ax=axs[0,0])
            axs[0,0].set_title('Saldo Atual por Produto')
            axs[0,0].set_ylabel('Unidades')
            axs[0,0].grid(axis='y', linestyle='--', alpha=0.5)

            top_saida.head(10).plot(kind='bar', color='tab:orange', ax=axs[0,1])
            axs[0,1].set_title('Top 10 Produtos - Saída')
            axs[0,1].set_ylabel('Saídas')
            axs[0,1].grid(axis='y', linestyle='--', alpha=0.5)

            top_entrada.head(10).plot(kind='bar', color='tab:green', ax=axs[1,0])
            axs[1,0].set_title('Top 10 Produtos - Entrada')
            axs[1,0].set_ylabel('Entradas')
            axs[1,0].grid(axis='y', linestyle='--', alpha=0.5)

            produtos_criticos_prod = produtos_criticos.groupby('Produto')['Saldo Atual'].sum().sort_values()
```

```
            produtos_criticos_prod.plot(kind='bar', color='tab:red', ax=axs[1,1])
            axs[1,1].set_title('Produtos em Estoque Crítico (<20)')
            axs[1,1].set_ylabel('Saldo Atual')
            axs[1,1].grid(axis='y', linestyle='--', alpha=0.5)

            plt.suptitle(f'Total em Estoque: {total_estoque} | Itens Críticos:
{produtos_criticos.shape[0]}', fontsize=16, y=1.02)
            plt.tight_layout()
            plt.show()
            return fig, saldo_atual

        def ao_mudar_filtros(change=None):
            with output:
                clear_output(wait=True)
                fdf = filtrar_df()
                fig, saldo_atual = plotar_dashboard(fdf)
                display(widgets.HTML("<b>Tabela de Saldo Atual (Filtrada):</b>"))
                display(saldo_atual)

        def ao_exportar_grafico(b):
            fdf = filtrar_df()
            fig, saldo_atual = plotar_dashboard(fdf)
            nome_arquivo = "dashboard_filtrado.png"
            fig.savefig(nome_arquivo, format='png')
            print(f"Gráfico salvo como {nome_arquivo}")

        def ao_exportar_relatorio(b):
            fdf = filtrar_df()
```

```
            saldo_atual = fdf.groupby(['Código', 'Produto', 'Cor', 'Tamanho', 'Tecido'],
dropna=False)['Saldo Atual'].sum().reset_index()
            saldo_atual['Status'] = saldo_atual['Saldo Atual'].apply(lambda x: 'Crítico' if x <
20 else 'OK')
            top_saida =
fdf.groupby('Produto')['Saída'].sum().sort_values(ascending=False).reset_index().rename(colu
mns={'Saída': 'Total Saídas'})
            top_entrada =
fdf.groupby('Produto')['Entrada'].sum().sort_values(ascending=False).reset_index().rename(co
lumns={'Entrada': 'Total Entradas'})
            resumo_categoria = fdf.groupby('Tecido')['Saldo
Atual'].sum().reset_index().rename(columns={'Saldo Atual': 'Total em Estoque'})
            resumo_geral = pd.DataFrame({
                'Total de Itens': [saldo_atual['Produto'].nunique()],
                'Total em Estoque': [saldo_atual['Saldo Atual'].sum()],
                'Produtos Críticos': [saldo_atual[saldo_atual['Status'] == 'Crítico'].shape[0]]
            })

            nome_arquivo = "relatorio_filtrado.xlsx"
            with pd.ExcelWriter(nome_arquivo) as writer:
                saldo_atual.to_excel(writer, sheet_name='Estoque Atual', index=False)
                top_saida.to_excel(writer, sheet_name='Top Saída', index=False)
                top_entrada.to_excel(writer, sheet_name='Top Entrada', index=False)
                resumo_categoria.to_excel(writer, sheet_name='Resumo por Tecido',
index=False)
                resumo_geral.to_excel(writer, sheet_name='Resumo Geral', index=False)
            print(f"Relatório salvo como {nome_arquivo}")

        w_produto.observe(ao_mudar_filtros, names='value')
        w_cor.observe(ao_mudar_filtros, names='value')
```

```
        w_tamanho.observe(ao_mudar_filtros, names='value')
        w_tecido.observe(ao_mudar_filtros, names='value')
        w_meses.observe(ao_mudar_filtros, names='value')
        botao_exportar_grafico.on_click(ao_exportar_grafico)
        botao_exportar_relatorio.on_click(ao_exportar_relatorio)

        painel_filtros = widgets.HBox([w_produto, w_cor, w_tamanho, w_tecido,
w_meses])
        painel_botoes = widgets.HBox([botao_exportar_grafico, botao_exportar_relatorio])

        display(widgets.HTML("<h3>Dashboard Interativo - Estoque Filtrado</h3>"))
        display(widgets.VBox([painel_filtros, painel_botoes, output]))
        ao_mudar_filtros()


    # ———————— TRANSFORMAR DADOS DASHBORD EM PNG ————————

    def gerar_dashboard_filtrado_png(df, nome_arquivo="dashboard_filtrado.png"):
        """
        Gera um gráfico simples de saldo atual por produto, salva como PNG.
        """
        saldo_atual = df.groupby(['Produto'])['Saldo Atual'].sum().reset_index()
        fig, ax = plt.subplots(figsize=(12, 6))
        ax.bar(saldo_atual['Produto'], saldo_atual['Saldo Atual'])
        plt.title("Saldo Atual por Produto")
        plt.ylabel("Saldo Atual")
        plt.xticks(rotation=45, ha='right')
        plt.tight_layout()
```

```
        plt.savefig(nome_arquivo)
        plt.close(fig)
        print(f"Dashboard filtrado salvo como {nome_arquivo}")

    # Execute para gerar o PNG:
    gerar_dashboard_filtrado_png(df_estoque)

    # ———— EXPORTAR PLANILHA CONSOLIDADA COMO PNG ————

    def exportar_consolidado_png_tabela(df, nome_arquivo="estoque_consolidado.png"):
        """
        Exporta toda a tabela consolidada como PNG (ideal para IA que lê tabela).
        """
        tabela = estoque_consolidado(df)
        n_rows = tabela.shape[0]
        n_cols = tabela.shape[1]
        fig_height = min(1.2 + 0.30 * n_rows, 120)
        fig, ax = plt.subplots(figsize=(min(20, n_cols*2.5), fig_height))
        ax.axis('off')
        tbl = ax.table(cellText=tabela.values, colLabels=tabela.columns, loc='center',
cellLoc='center')
        tbl.auto_set_font_size(False)
        tbl.set_fontsize(10)
        tbl.scale(1, 1.3)
        plt.tight_layout()
        plt.savefig(nome_arquivo, bbox_inches='tight')
        plt.close(fig)
        print(f"Imagem da tabela consolidada salva como {nome_arquivo} ({n_rows}
linhas)")
```

```
        return nome_arquivo


    # ———————— VISUALIZADOR ESTOQUE CONSOLIDADO ————————
    def visualizar_estoque_consolidado(df):
        """
        Exibe a tabela de estoque consolidado no notebook,
        exporta automaticamente o PNG, e fornece botões para exportar Excel/PNG.
        Retorna o DataFrame para integração front-end ou API.
        """
        from IPython.display import display, HTML
        import ipywidgets as widgets
        import matplotlib.pyplot as plt

        tabela = estoque_consolidado(df)
        display(HTML("<h3>Estoque Consolidado Atual (Código, Produto, Cor, Tamanho, Tecido)</h3>"))
        display(tabela)
        print(f"Total de linhas: {len(tabela)}")

        # Exportação automática de PNG ao exibir
        n_rows = tabela.shape[0]
        n_cols = tabela.shape[1]
        fig_height = min(1.2 + 0.30 * n_rows, 120)
        fig, ax = plt.subplots(figsize=(min(20, n_cols*2.5), fig_height))
        ax.axis('off')
        tbl = ax.table(cellText=tabela.values, colLabels=tabela.columns, loc='center', cellLoc='center')
        tbl.auto_set_font_size(False)
        tbl.set_fontsize(10)
```

```
            tbl.scale(1, 1.3)
            plt.tight_layout()
            nome_arquivo = "estoque_consolidado.png"
            plt.savefig(nome_arquivo, bbox_inches='tight')
            plt.close(fig)
            print(f"PNG salvo automaticamente como {nome_arquivo} ({n_rows} linhas)")

            # Botão para exportar Excel
            def exportar_excel(b):
                nome_excel = "estoque_consolidado.xlsx"
                tabela.to_excel(nome_excel, index=False)
                print(f"Arquivo salvo como {nome_excel}")

            # Botão para exportar PNG manualmente de novo, se quiser
            def exportar_png(b):
                fig, ax = plt.subplots(figsize=(min(20, n_cols*2.5), fig_height))
                ax.axis('off')
                tbl = ax.table(cellText=tabela.values, colLabels=tabela.columns, loc='center',
cellLoc='center')
                tbl.auto_set_font_size(False)
                tbl.set_fontsize(10)
                tbl.scale(1, 1.3)
                plt.tight_layout()
                plt.savefig(nome_arquivo, bbox_inches='tight')
                plt.close(fig)
                print(f"Imagem salva como {nome_arquivo} (manual)")

            botao_excel = widgets.Button(description="Exportar Excel",
button_style="success")
```

```
        botao_png = widgets.Button(description="Exportar PNG (de novo)",
button_style="warning")
        botao_excel.on_click(exportar_excel)
        botao_png.on_click(exportar_png)
        display(widgets.HBox([botao_excel, botao_png]))

        # Retorna o DataFrame para integração via front-end/API
        return tabela

    # ———————— VISUALIZADOR RÁPIDO DE DATAFRAME ————————

    def visualizar_tabela(df, titulo=None, linhas=10):
        """
        Exibe um DataFrame de forma amigável no Colab, com título opcional e número de
linhas customizável.
        """
        if titulo:
            display(HTML(f"<h3>{titulo}</h3>"))
        display(df.head(linhas))
        print(f"Mostrando as {min(linhas, len(df))} primeiras linhas de um total de
{len(df)}.")
    # ———————————————— OCR PARA ANÁLISE POR IA ————————
    def extrair_texto_ocr(path_img):
        """
        Usa OCR para extrair texto de dashboards/tabelas salvos em PNG/JPG.
        Retorna string com o texto extraído.
        """
        img = Image.open(path_img)
        texto = pytesseract.image_to_string(img, lang="por")
```

```
        return texto
    #landingAI
    # ———————————————— COMO USAR ————————————————
    # 1. Carregue o DataFrame do estoque (exemplo)
    df_estoque = carregar_estoque()  # use seu método real

    # 2. Dashboard interativo (para uso humano)
    dashboard_interativo(df_estoque)

    # 3. Exportar planilha consolidada como PNG (para IA via OCR)
    exportar_consolidado_png_tabela(df_estoque,
nome_arquivo="estoque_consolidado.png")

    # 4. Visualização rápida do DataFrame
    visualizar_tabela(df_estoque, titulo="Estoque Completo", linhas=10)

    # 5. OCR da tabela consolidada para análise IA
    texto_tab = extrair_texto_ocr("estoque_consolidado.png")
    print("\n--- OCR Estoque consolidado (tabela): ---\n", texto_tab[:1500])

    # 6 Estoque consolidado
    tabela_consol = visualizar_estoque_consolidado(df_estoque)
```

Apêndice 4 - Bloco 4: backend flask com dashboard e ia/ocr

O backend Flask serve dashboards como PNG e aceita uploads de gráficos para análise automática via IA/OCR.

O endpoint /analisar-dashboard permite que uma IA analise qualquer gráfico enviado, retornando texto e um insight automático.

Tudo pronto para ser chamado externamente, inclusive por outras ferramentas (n8n, Make, Zapier, etc).

```
# ———————————————————— BLOCO 4 ————————————————————
# Dashboards interativos, filtros, alertas automáticos, previsão Prophet, endpoints Flask
# ----------------------------------------------------------------

import io
import pandas as pd
import matplotlib.pyplot as plt
from flask import Flask, request, send_file, jsonify

# ========= 1. Funções auxiliares de filtros e alertas ==========

def filtrar_estoque(df, produto=None, cor=None, tamanho=None, tecido=None):
    df_filtrado = df.copy()
    if produto:
        df_filtrado = df_filtrado[df_filtrado['Produto'].str.contains(produto, case=False, na=False)]
    if cor:
        df_filtrado = df_filtrado[df_filtrado['Cor'].str.contains(cor, case=False, na=False)]
    if tamanho:
        df_filtrado = df_filtrado[df_filtrado['Tamanho'].astype(str).str.contains(str(tamanho), case=False, na=False)]
    if tecido:
        df_filtrado = df_filtrado[df_filtrado['Tecido'].str.contains(tecido, case=False, na=False)]
    return df_filtrado
```

```
        def alertas_baixo_estoque(df, limiar=20):
            alertas = df[df['Saldo Atual'] < limiar]
            # Inclui o código no alerta
            return alertas[['Código', 'Produto', 'Sku', 'Saldo Atual']] if not alertas.empty else
None

        # ========= 2. Dashboards simples com filtros ==========

        def plot_saldo_atual_por_produto(df, produto=None, cor=None, tamanho=None,
tecido=None):
            df_f = filtrar_estoque(df, produto, cor, tamanho, tecido)
            saldo = df_f.groupby('Produto')['Saldo Atual'].sum()
            fig, ax = plt.subplots(figsize=(8,5))
            saldo.plot(kind='bar', ax=ax, color='skyblue')
            ax.set_ylabel('Saldo Atual')
            ax.set_title('Saldo Atual por Produto')
            plt.xticks(rotation=45)
            plt.tight_layout()
            return fig

        def plot_movimentacao_diaria(df, produto=None):
            df_f = filtrar_estoque(df, produto)
            if 'Data' not in df_f.columns: return None
            df_f = df_f.copy()
            df_f['Movimentacao'] = df_f['Entrada'] + df_f['Saída']
            diario = df_f.groupby('Data')['Movimentacao'].sum()
            fig, ax = plt.subplots(figsize=(8,5))
            diario.plot(ax=ax, marker='o')
            ax.set_ylabel('Movimentação Diária')
            ax.set_title('Movimentação Diária')
            plt.xticks(rotation=45)
            plt.tight_layout()
            return fig

        def plot_visao_geral_estoque(df, meses=6, produto=None):
            df_f = filtrar_estoque(df, produto)
```

```
        if 'Data' not in df_f.columns: return None
        df_f = df_f.copy()
        df_f['Data'] = pd.to_datetime(df_f['Data'], errors='coerce')
        if meses:
            data_min = pd.Timestamp.today() - pd.DateOffset(months=meses)
            df_f = df_f[df_f['Data'] >= data_min]
        saldo_diario = df_f.groupby('Data')['Saldo Atual'].sum().cumsum()
        fig, ax = plt.subplots(figsize=(8,5))
        saldo_diario.plot(ax=ax, marker='o')
        ax.set_ylabel('Saldo Acumulado')
        ax.set_title(f'Visão Geral do Estoque - Últimos {meses} meses')
        plt.xticks(rotation=45)
        plt.tight_layout()
        return fig

    # ========= 3. Previsão de demanda Prophet ==========
    from prophet import Prophet

    def prever_demanda(df, sku, meses=3):
        df_sku = df[df['Sku'] == sku]
        if df_sku.empty or 'Data' not in df_sku.columns:
            return None
        df_mensal = df_sku.groupby(pd.Grouper(key='Data',
freq='MS'))['Saída'].sum().reset_index()
        if len(df_mensal) < 2:
            return None
        df_mensal = df_mensal.rename(columns={'Data':'ds', 'Saída':'y'})
        modelo = Prophet()
        modelo.fit(df_mensal)
        futuro = modelo.make_future_dataframe(periods=meses, freq='M')
        previsao = modelo.predict(futuro)
        return previsao[['ds', 'yhat']]

    # ========= 4. Relatório de alertas + previsão de demanda ==========
    def gerar_relatorio_estoque(df, sku=None):
        alertas = alertas_baixo_estoque(df)
        relatorio = ""
```

```
        if alertas is not None:
            relatorio += "⚠️ Itens com estoque abaixo de 20 unidades:\n"
            relatorio += alertas.to_string(index=False)
        else:
            relatorio += "✅ Nenhum item com estoque crítico.\n"
        if sku:
            previsao = prever_demanda(df, sku)
            if previsao is not None:
                relatorio += f"\n\n🔮 Previsão de demanda para SKU {sku}:\n"
                relatorio += previsao.tail(3).to_string(index=False)
            else:
                relatorio += f"\n\n⚠️ Não há dados suficientes para previsão de {sku}."
        return relatorio

    # ========= 5. Estoque Consolidado (com Código) ==========
    def estoque_consolidado(df):
        df_group = (
            df.groupby(['Código', 'Produto', 'Cor', 'Tamanho', 'Tecido'], dropna=False)['Saldo Atual']
            .sum()
            .reset_index()
            .sort_values(['Produto', 'Cor', 'Tamanho', 'Tecido'])
        )
        df_group['Status'] = df_group['Saldo Atual'].apply(lambda x: 'Crítico' if x < 20 else 'OK')
        return df_group

    # ==== 6. Endpoints Flask para dashboards, alertas, filtros, previsão, consolidados ======
    app = Flask(__name__)
    @app.route('/dashboard/saldo')
    def api_dash_saldo():
        produto = request.args.get('produto')
        cor = request.args.get('cor')
        tamanho = request.args.get('tamanho')
        tecido = request.args.get('tecido')
        fig = plot_saldo_atual_por_produto(df_estoque, produto, cor, tamanho, tecido)
```

```
    buf = io.BytesIO()
    fig.savefig(buf, format='png')
    buf.seek(0)
    return send_file(buf, mimetype='image/png')

@app.route('/dashboard/movimentacao')
def api_dash_mov():
    produto = request.args.get('produto')
    fig = plot_movimentacao_diaria(df_estoque, produto)
    buf = io.BytesIO()
    fig.savefig(buf, format='png')
    buf.seek(0)
    return send_file(buf, mimetype='image/png')

@app.route('/dashboard/visao_geral')
def api_dash_visao():
    produto = request.args.get('produto')
    meses = int(request.args.get('meses', 6))
    fig = plot_visao_geral_estoque(df_estoque, meses, produto)
    buf = io.BytesIO()
    fig.savefig(buf, format='png')
    buf.seek(0)
    return send_file(buf, mimetype='image/png')

@app.route('/alertas')
def api_alertas():
    alertas = alertas_baixo_estoque(df_estoque)
    if alertas is None:
        return jsonify({'status': 'ok', 'mensagem': 'Nenhum item crítico!'})
    else:
        return alertas.to_json(orient='records')

@app.route('/previsao')
def api_previsao():
    sku = request.args.get('sku')
    meses = int(request.args.get('meses', 3))
    previsao = prever_demanda(df_estoque, sku, meses)
```

```
    if previsao is None:
        return jsonify({"erro": "Não há dados suficientes para previsão."})
    return previsao.to_json(orient='records')

@app.route('/relatorio')
def api_relatorio():
    sku = request.args.get('sku')
    relatorio = gerar_relatorio_estoque(df_estoque, sku)
    return jsonify({'relatorio': relatorio})

@app.route('/estoque_consolidado')
def api_estoque_consolidado():
    """Retorna o estoque consolidado (com código) em JSON."""
    df_cons = estoque_consolidado(df_estoque)
    return df_cons.to_json(orient='records')
# ———————————— Pronto para subir Flask+Ngrok depois! ————————————
```

Apêndice 5 - Bloco 5: ocr+ia para análise automática dos dashboards e publicação do link ngrok para testes e subir apis

```
# ———————————————— BLOCO 5 ————————————————
# OCR (pytesseract) + Análise IA (OpenAI GPT) dos dashboards gerados
# ----------------------------------------------------------------
# Instale as dependências extras se ainda não fez (pode rodar novamente sem problemas)

!pip install pytesseract pillow
import pytesseract
from PIL import Image

def ocr_from_image(image_bytes):
    """
    Recebe bytes de uma imagem, faz OCR e retorna o texto extraído."""
    image = Image.open(io.BytesIO(image_bytes))
    texto = pytesseract.image_to_string(image, lang="por")  # Use 'por' para português
    return texto.strip()

def ia_analise_dashboard(texto_dashboard):
    """
    Usa o modelo GPT para analisar o texto extraído do dashboard
    e retorna um insight automatizado.
    """


    prompt = f"""
    Você é um assistente de gestão de estoques. Analise os dados abaixo e forneça insights práticos, identificando alertas, tendências e possíveis ações para o gestor:

    {texto_dashboard}
    Responda de forma clara, objetiva e em português.
    """
    import openai
    response = openai.chat.completions.create(
```

```
            model="gpt-3.5-turbo",  # Troque se desejar outro modelo
            messages=[{"role": "user", "content": prompt}],
            temperature=0.3,
            max_tokens=512,
        )
        return response.choices[0].message.content.strip()

    # ———— Endpoints Flask para análise automática de dashboard ————
    @app.route('/analise_dashboard', methods=['POST'])
    def api_analise_dashboard():
        """
        Recebe uma imagem de dashboard (PNG, JPG), faz OCR e pede insight à IA.
        Use: POST com campo 'file' (form-data).
        """
        if 'file' not in request.files:
            return jsonify({"erro": "Nenhuma imagem enviada!"}), 400
        file = request.files['file']
        img_bytes = file.read()
        texto_dashboard = ocr_from_image(img_bytes)
        if not texto_dashboard:
            return jsonify({"erro": "Não foi possível extrair dados da imagem!"}), 400
        insight = ia_analise_dashboard(texto_dashboard)
        return jsonify({
            "texto_extraido": texto_dashboard,
            "insight_ia": insight
        })

# ========== THREAD FLASK COM NGROK PARA USO NO COLAB ======
    def run_app_ngrok():
        app.run(host='0.0.0.0', port=5000, debug=False, use_reloader=False)
    from threading import Thread
    from pyngrok import ngrok
    import time
    # Inicie Flask numa thread
    thread = Thread(target=run_app_ngrok, daemon=True)
    thread.start()
    # Aguarde o Flask subir
```

```
print("⏳ Aguardando Flask subir…")
time.sleep(5)
# Abra túnel Ngrok para a porta 5000
!ngrok config add-authtoken "CHAVE NGROK"
public_url = ngrok.connect(5000, bind_tls=True)
print("🔗 API disponível em:", public_url)
```

Apêndice 6 - Bloco 6: testes mostrando a funcionalidade do código

Teste 1: Carregamento do estoque (Sheets)

```
df_estoque = carregar_estoque()
print("✔️ Dados carregados!\n")
print(df_estoque.head())
```

Teste 2: Dashboard Interativo (visual/manual)

```
# Apenas execute para abrir o painel
dashboard_interativo(df_estoque)
```

Teste 3: Visão consolidada do estoque

```
visualizar_estoque_consolidado(df_estoque)
```

Teste 4: Funções de filtro, alerta e consolidado (bloco 4)

```
# Teste do filtro
df_filtrado = filtrar_estoque(df_estoque, produto="Camiseta")
print("Filtro por produto Camiseta:\n", df_filtrado.head())

# Teste do alerta de baixo estoque
print("\nAlerta de baixo estoque:")
print(alertas_baixo_estoque(df_estoque))

# Teste do estoque consolidado (função API)
print("\nEstoque consolidado (API):")
print(estoque_consolidado(df_estoque).head())
```

Teste 5: Geração de gráficos individuais

```
# Saldo atual por produto (mostra gráfico)
plot_saldo_atual_por_produto(df_estoque)
plt.show()

# Movimentação diária (se houver coluna Data)
plot_movimentacao_diaria(df_estoque)
plt.show()

# Visão geral estoque (acumulado)
plot_visao_geral_estoque(df_estoque, meses=6)
plt.show()
```

Teste 6: Previsão Prophet

```
# Pegue um SKU válido (exemplo: primeira linha do df)
sku_exemplo = df_estoque['Sku'].iloc[0]
prev = prever_demanda(df_estoque, sku_exemplo, meses=3)
print(f"Previsão para SKU: {sku_exemplo}")
print(prev if prev is not None else "Sem dados suficientes para previsão")
```

Teste 7: Relatório de estoque/alerta + previsão

```
# Gera relatório para o SKU de exemplo
print(gerar_relatorio_estoque(df_estoque, sku_exemplo))
```

Teste 8: OCR e análise de IA

```
# Certifique-se de já ter rodado a exportação do consolidado:
# exportar_consolidado_png_tudo(df_estoque)

with open("estoque_consolidado.png", "rb") as f:
    img_bytes = f.read()
texto = ocr_from_image(img_bytes)
print("Texto extraído da imagem:", texto)

# 2. Mande o texto extraído para a IA
insight = ia_analise_dashboard(texto)
print("Insight IA:", insight)
```

Teste 9: Endpoints Flask (Requisições HTTP)

```
import requests

# Substitua pela URL do seu ngrok/localhost:
url_base = "http://localhost:5000"  # ou ngrok

# Dashboard saldo atual (imagem)
r = requests.get(f"{url_base}/dashboard/saldo")
assert r.status_code == 200
print("✔️ Endpoint dashboard/saldo OK (imagem PNG)")

# Alertas JSON
```

```
r = requests.get(f"{url_base}/alertas")
print("Alertas:", r.json())

# Estoque consolidado JSON
r = requests.get(f"{url_base}/estoque_consolidado")
print("Estoque consolidado:", r.json()[:3])  # Mostra só os 3 primeiros

# Previsão
sku = df_estoque['Sku'].iloc[0]
r = requests.get(f"{url_base}/previsao", params={"sku": sku})
print("Previsão:", r.json())

# Relatório de texto
r = requests.get(f"{url_base}/relatorio", params={"sku": sku})
print("Relatório:", r.json()['relatorio'])
```

Teste 10: Endpoints de OCR+IA

```
# Com um arquivo de dashboard (exemplo PNG)
files = {'file': open('dashboard_filtrado.png', 'rb')}
r = requests.post(f"{url_base}/analise_dashboard", files=files)
print(r.json())
```

Apêndice 7 - Conexão Google Drive e leitura do arquivo

```
      # Conexão do Google Drive
      try:
          from google.colab import drive
          drive.mount('/content/drive')
      except ModuleNotFoundError:
          pass

      %cd /content/drive/MyDrive/Colab Notebooks/
      # Leitura do arquivo com os dados de saída (demanda) do Excel

      import os, pandas as pd, unicodedata, re

      def remove_acentos(txt: str) -> pd.Series:
          """Remove diacríticos de séries pandas ou strings."""
          nfkd = unicodedata.normalize("NFKD", str(txt))
          return re.sub(r'\s+', '', ''.join(c for c in nfkd
                                            if unicodedata.category(c)
!= 'Mn'))

      excel_path = '/content/drive/MyDrive/Colab
Notebooks/Modelo_Estoque_Gurubi.xlsx'
      df = pd.read_excel(excel_path, sheet_name='Estoque')
```

Apêndice 8 - Pré-processamento de dado

```
      # Pré-processamento dos dados

      # Filtra Camiseta Tradicional ou Polo
      mask =
df['Produto'].str.contains(r'Camiseta\s+Tradicional|Camiseta\s+Polo',
                                          case=False, na=False)
      df = df[mask].copy()

      # Cor simplificada
      df['CorSimplificada'] = df['Cor'].apply(
          lambda x: 'Branco' if str(x).strip().lower() == 'branco' else
'Colorido'
      )

      # Tamanho simplificado
      normais = {'PP', 'P', 'M', 'G', 'GG'}
      df['TamanhoSimplificado'] = df['Tamanho'].apply(
          lambda x: 'tamanho normal'
          if str(x).strip().upper() in normais else 'tamanho especial'
      )

      # Identificador sem acentos
      df['ProdutoID'] = (
          df['Produto'].apply(remove_acentos).str.lower() + '_' +
          df['CorSimplificada'].str.lower() + '_' +
          df['Tecido'].apply(remove_acentos).str.lower() + '_' +
          df['TamanhoSimplificado'].str.replace(' ', '').str.lower()
      )

      # Data → YYYYMM
      df['AnoMes'] = pd.to_datetime(df['Data']).dt.strftime('%Y%m')

      # Agrega e pivoteia
      pivot = (
          df.groupby(['ProdutoID', 'AnoMes'])['Saída']
            .sum()
            .reset_index()
            .pivot(index='ProdutoID', columns='AnoMes', values='Saída')
            .fillna(0)
            .astype(int)
            .sort_index()
      )
```

Apêndice 9 - Salva tabela de demanda no drive

```
        # # Salva tabela de demanda no Drive
        output_csv  = '/content/drive/MyDrive/Colab
Notebooks/demanda_mensal_produto.csv'
        output_xlsx = '/content/drive/MyDrive/Colab
Notebooks/demanda_mensal_produto.xlsx'

        pivot.to_csv(output_csv, index=True)       # CSV
        pivot.to_excel(output_xlsx, index=True)    # Excel

        print(f"Tabela salva em:\n  • {output_csv}\n  • {output_xlsx}")
```

Apêndice 10 - Instalação e licença Gurobi

```
# Instalar Gurobi no Google Colab (execute esta célula primeiro)
!pip install -U gurobipy
 Configuração da licença WLS para Gurobi
import gurobipy as gp
from gurobipy import Model, GRB, quicksum

env = gp.Env(empty=True)
env.setParam('WLSACCESSID', 'Sua licença aqui')
env.setParam('WLSSECRET', 'Sua licença aqui')
env.setParam('LICENSEID', Sua licença aqui)
env.start()
```

Apêndice 11 - Preparação dos dados

```
      # Preparação de Dados

      # 1. Dados de entrada
      pivot = pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/demanda_mensal_produto.csv',
                          index_col=0)

      produtos = pivot.index.tolist()
      meses_raw = pivot.columns.astype(str).tolist()              #
['202408', ...]
      meses = [f"{s[:4]}-{s[4:]}" for s in meses_raw]             #
['2024-08', ...]

      # Limitação de caracteres para nomes Gurobi (máximo de 255
caracteres)
      p_idx = {p: i for i, p in enumerate(produtos)}
      m_idx = {m: i for i, m in enumerate(meses)}

      p_short = [f"p{i}" for i in p_idx.values()]
      m_short = [f"m{i}" for i in m_idx.values()]

      # Mapas inversos do nome dos produtos
      p_long = {f"p{i}": p for p, i in p_idx.items()}
      m_long = {f"m{i}": m for m, i in m_idx.items()}

      # Demandas (dict para acesso rápido)
      demanda = {
          (f"p{p_idx[p]}", f"m{m_idx[meses[m_idx_val]]}"):
int(pivot.loc[p, meses_raw[m_idx_val]])
          for p in produtos for m_idx_val in range(len(meses))
      }



      # Custos fixos por produto
      ## Lista de custos fixos dos produtos (na ordem p0 … p16)
      custos_fixos = [
          37.04, 43.25, 38.89, 74.04, 66.70, 17.15, 15.45, 12.25,
          11.25, 20.22, 18.20, 19.78, 17.81, 14.12, 12.71, 22.75,
          20.46
      ]
      ## Garantia de consistência
      assert len(produtos) == len(custos_fixos), (
```

```
          f"N.° de produtos ({len(produtos)}) ≠ N.° de custos
({len(custos_fixos)})"
      )
      custo_pedido = {f"p{i}": v for i,v in enumerate(custos_fixos)}


      # Custo de manutenção no estoque
      custo_manut = {f"p{p_idx[p]}": (12 if p_idx[p] < 5 else 10) for p
in produtos}

      # 2. Conjuntos e índices úteis
      keys = [(p_s, m_s) for p_s in p_short for m_s in m_short]
      # --- Capacidade de armazém (20 % acima do pico mensal agregado)
---------------
      demanda_mensal = {m_s: sum(demanda[p_s, m_s] for p_s in p_short)
for m_s in m_short}
      capacidade  = int(1.2 * max(demanda_mensal.values()))
      #print(f"Capacidade agregada do armazém: {capacidade} unidades")

      # --------- orçamento mensal ----------
      #lista_orcamento = [400.0, 350.0, 450.0, 380.0, 300.0, 500.0,
420.0, 400.0]
      lista_orcamento = [40000000.0]*len(meses)
      assert len(lista_orcamento) == len(meses), "Revise vetor de
orçamento!"
      orcamento = {m: lista_orcamento[i] for i, m in enumerate(m_short)}


      # --- Parâmetros auxiliares
----------------------------------------
      L_MIN = 100         # lote mínimo - 100 camisetas por caixa
      T_LIMIT = 300      # 5 min
      MIP_GAP = 0.01     # 1 %
```

Apêndice 12 - Resolução do modelo com variável Y binária e quantidade Q

```
      # Resolução do Modelo com variável y binária e quantidade q

      # 3. Modelo
      m = Model("Reposicao_Camisetas")

      # 3.1 Variáveis
      y = m.addVars(keys,vtype=GRB.BINARY,name="pedido_num_caixas")
# pedido de caixas com 100 camisetas
      q =
m.addVars(keys,vtype=GRB.INTEGER,lb=0,name="qtde_camisetas_pedida") #
quantidade
      I = m.addVars(keys,
vtype=GRB.CONTINUOUS,lb=0,name="estoque_camisetas") # estoque

      # 3.2 Função-objetivo
      m.setObjective(
          quicksum(custo_pedido[k[0]] * q[k] for k in keys) +
          quicksum(custo_manut[k[0]] * I[k] for k in keys),
          GRB.MINIMIZE
      )

      # 3.3 Restrição-indicadora (y = 0 ⇒ q = 0)
      for k in keys:
          m.addGenConstrIndicator(y[k], 0, q[k] == 0,
name=f"ind_{k[0]}_{k[1]}")

      # 3.4 Lote mínimo
      for k in keys:
          m.addConstr(q[k] >= L_MIN * y[k],
                              name=f"lote_min_{k[0]}_{k[1]}")

      # 3.5 Balanço de estoque
      for p_s in p_short:
          for t_idx, m_s in enumerate(m_short):
              dem = demanda[(p_s, m_s)]
              if t_idx == 0:
                  # Estoque inicial = 0  (pode definir outro valor)
                  m.addConstr(I[p_s, m_s] == q[p_s, m_s] - dem,
                              name=f"bal_{p_s}_{m_s}")
              else:
                  m_prev = m_short[t_idx-1]
```

```
                    m.addConstr(I[p_s, m_s] == I[p_s, m_prev] + q[p_s,
m_s] - dem,
                                name=f"bal_{p_s}_{m_s}")


      # 3.6 Capacidade
      for m_s in m_short:
        m.addConstr(
            quicksum(I[p_s, m_s] for p_s in p_short) <= capacidade,
            name=f"cap_{m_s}"
        )

      # 3.7 Restrição de orçamento mensal
      for m_s in m_short:
          m.addConstr(
              gp.quicksum(custo_pedido[p_s] * q[p_s, m_s] for p_s in
p_short)
              <= orcamento[m_s],
              name=f"orcamento_{m_s}"
          )


      # 3.8 Parâmetros de otimização
      m.Params.TimeLimit = T_LIMIT
      m.Params.MIPGap    = MIP_GAP
      #m.Params.Threads   = 1      # ajustar ao hardware
      #m.Params.MIPFocus  = 1      # opcional: focar no primal
```

Apêndice 13 - Solução e relatório resumido

```
      #
-----------------------------------------------------------------
      # 4. Solução
      m.optimize()

      #
-----------------------------------------------------------------
      # 5. Relatório resumido
      if m.SolCount:
          print(f"\nCusto ótimo: {m.ObjVal:,.2f}\n")
          for p_s in p_short:
              p = p_long[p_s]
              print(f"Produto: {p}")
              for m_s in m_short:
                  mes = m_long[m_s]
                  if y[p_s, m_s].X > 0.5:      # pedido emitido
                      print(f"  {mes}: y = {y[p_s, m_s].X:.0f}, q =
{q[p_s, m_s].X:.0f}, I = {I[p_s, m_s].X:.0f}")
```