CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA
ETEC JOÃO GOMES DE ARAÚJO
CURSO: ENSINO MÉDIO COM HABILITAÇÃO PROFISSIONAL DE TÉCNICO EM DESENVOLVIMENTO DE SISTEMAS

FROG:

SISTEMA DE GERENCIAMENTO DE ITENS EM ESTOQUE

DAVI CUNHA DA SILVA
GABRIELE BRAGA DE FARIAS SILVA
JOÃO PEDRO DE MELLO SILVA
MIGUEL CARMO VIEIRA AZOCAR
NICOLAS HENRIQUE CORRÊA DA SILVA

PINDAMONHANGABA
2025

DAVI CUNHA DA SILVA
GABRIELE BRAGA DE FARIAS SILVA
JOÃO PEDRO DE MELLO SILVA
MIGUEL CARMO VIEIRA AZOCAR
NICOLAS HENRIQUE CORRÊA DA SILVA

FROG :

SISTEMA DE GERENCIAMENTO DE ITENS EM ESTOQUE

Relatório Técnico Científico apresentado como Trabalho de Conclusão de Curso do Curso Técnico em Desenvolvimento de Sistemas da ETEC João Gomes de Araújo, orientado pelo Prof. Me. Claudemir Santos Pinto, como requisito parcial para obtenção do título de Técnico em Desenvolvimento de Sistemas.

Pindamonhangaba
2025

RESUMO

Este trabalho de conclusão de curso apresenta o desenvolvimento de uma aplicação web voltada ao gerenciamento de estoque, concebida no âmbito do curso técnico em Desenvolvimento de Sistemas. O principal objetivo do projeto foi criar uma ferramenta prática e acessível para reduzir perdas, aprimorar a organização e otimizar o controle e a disposição das informações relacionadas ao estoque de pequenos empreendimentos. Para atingir esse objetivo foram priorizadas três funcionalidades centrais: controle de estoque (registro e atualização de itens), uma *dashboard* para visualização rápida de indicadores e um sistema de movimentações com fluxo intuitivo para entradas e saídas. A metodologia adotada baseou-se em pesquisa qualitativa aplicada: foram realizadas entrevistas com responsáveis pela gestão de microempreendimentos e profissionais que atuam diretamente no controle de estoque e nas vendas. Esse levantamento de requisitos orientou tanto a modelagem das telas quanto às regras de negócio implementadas, garantindo que a solução atendesse às necessidades reais dos usuários. O processo de desenvolvimento utilizou tecnologias modernas de *front-end*: HTML, CSS e TypeScript, com apoio do *framework* React e do *bundler* Vite; o versionamento do código foi realizado via Git/GitHub e a aplicação foi preparada para execução em ambientes conteinerizados com Docker. O sistema implementado contempla funcionalidades de CRUD para produtos, registro de movimentações (entradas e saídas), filtros de busca e um painel gráfico que sintetiza informações relevantes para a tomada de decisão, como saldo de itens e histórico de movimentações. A interface foi projetada para ser clara e acessível, priorizando a usabilidade para operadores com diferentes níveis de familiaridade técnica. Como considerações finais, o trabalho alcançou o objetivo de produzir uma aplicação funcional conforme o planejado e evidencia o potencial de sistemas digitais simples para melhorar a gestão operacional de microempreendimentos.

Palavras-chaves: Controle de estoque, Itens em estoque, Sistema web

ABSTRACT

This Final Course Project presents the development of a web application focused on inventory management, conceived within the scope of the technical course in Systems Development. The project's primary objective was to create a practical and accessible tool to reduce losses, improve organization, and optimize the control and presentation of information related to the inventory of small businesses. To achieve this objective, three core features were prioritized: inventory control (item registration and updating), a graphical dashboard for quick visualization of indicators, and a transactions system with an intuitive flow for incoming and outgoing movements. The methodology adopted was based on applied qualitative research: interviews were conducted with those responsible for managing microenterprises and with professionals who work directly in inventory control and sales. This requirements gathering guided both the screen design and the implemented business rules, ensuring that the solution met the real needs of users. The development process employed modern front-end technologies: HTML, CSS, and TypeScript, supported by the React framework and the Vite bundler; code versioning was managed via Git/GitHub, and the application was prepared for deployment in containerized environments using Docker. The implemented system includes CRUD functionality for products, recording of stock movements (incoming and outgoing), search filters, and a graphical panel that synthesizes information relevant to decision-making, such as item balances and movement history. The interface was designed to be clear and accessible, prioritizing usability for operators with varying levels of technical familiarity. In conclusion, the project achieved its objective of producing a functional application as planned and highlights the potential of simple digital systems to improve the operational management of microenterprises.

Keywords: Inventory control, Items in stock, Web system

SUMÁRIO

Sumário

1 INTRODUÇÃO

A gestão de estoque em empresas é um aspecto fundamental para que se garanta um bom funcionamento do negócio, destacando-se que em micro e pequenas empresas de diversos setores, sua eficiência operacional pode determinar seu índice de crescimento. No entanto, é possível observar que muitos desses empreendimentos ainda enfrentam desafios que prejudicam aspectos relacionados ao gerenciamento, análise e controle de estoque de seu inventário. Ainda que soluções manuais sejam feitas, a ineficácia e a falta de automação podem resultar em perdas significativas de faturamento, desperdício de recursos e falha no atendimento ao cliente. Diante desse cenário, o desenvolvimento de um sistema de gerenciamento de estoque automatizado de fácil acesso surge como uma solução promissora para otimizar os processos operacionais referidos e aumentar a performance empresarial.

1.1 Justificativa

O uso de sistemas digitais de gerenciamento de estoque desempenha um papel fundamental na eficiência operacional das empresas. Segundo Miranda (2007), tais sistemas contribuem significativamente para o melhor controle de vendas, redução de custos, crescimento de vendas, agilidade para adquirir informações e entregar produtos, otimização de estoques e redução de tempo e perdas. De acordo com Souza et al. (2024, p. 34) 97% dos questionados relatam uma melhora significativa na performance de estoque. Vedovato et al. (2015, p., 6) diz que os efeitos das empresas incluem: agilidade para tomada de decisões, garantia ao negócio de dados precisos para a gestão de custos, controle de estoques, entre outras funções.

Uma pesquisa realizada por Carneiro (2015), estudou a implementação de sistemas digitais de estoque em uma microempresa do ramo de concessionária. Os resultados desse estudo relataram uma melhora na visibilidade do fluxo de estoque, menor ocorrência de erros e aumento da precisão no planejamento de compras. Esses resultados reforçam que microempresas têm um benefício significativo ao adotarem o uso de sistemas digitais de estoque.

Vale ainda mencionar novamente que, segundo Vedovato et al. (2015, p. 7), os efeitos são vistos de forma instantânea após a aplicação do sistema, o que resulta em tomadas de decisões mais rápidas e precisas.

1.2 Problema

Após pesquisas realizadas, a equipe notou como a falta de organização e controle de estoque em pequenas e microempresas pode ser um problema recorrente, então será trabalhado o projeto acima desta problematização.

De acordo com Catarino et al. (2017) é possível observar que micro e pequenas empresas de diversos setores, numa tentativa de atenuar o problema da falta de organização e controle de estoque, utilizam de planilhas eletrônicas não automatizadas ou registros manuais:

> Entre todas as ferramentas computacionais de gerenciamento de dados ou análise, as planilhas eletrônicas são as que mais se destacam tornando-se um diferencial para as pequenas empresas. Elas são utilizadas para suprir as dificuldades relacionadas à implantação da tecnologia da informação(CATARINO et al., 2017).

Essa prática leva a frequentes redundâncias em relação ao lançamento de dados no sistema, comprometendo a confiabilidade dos dados e dificultando a contabilidade do saldo real em estoque. Além disso, a falta de monitoramento em tempo real impede que o gerenciamento do sistema seja eficiente, dificultando o alinhamento e harmonia da empresa como um todo (Target Erp, 2025).

Como consequência desses problemas operacionais, os proprietários e investidores do negócio sofrem com possíveis perdas significativas de faturamento, além de um desperdício de recursos devido a uma falta de análise de estoque (Agência Sebrae De Notícias, 2024). O funcionamento inconsistente do sistema também prejudica o atendimento ao cliente, que sofre com atrasos na entrega e venda de produtos, podendo até mesmo levar à desistência da compra por parte do cliente, o levando a insatisfação.

1.3 Objetivos

Este Trabalho de Conclusão de Curso tem como objetivo principal o desenvolvimento de um sistema automatizado como solução para a melhoria da execução e facilitação do gerenciamento de itens em estoque em pequenas empresas.

Objetivo Geral

O Objetivo Geral consiste na criação de um sistema simples, acessível e funcional que proporcione um controle eficiente, organizado e intuitivo para o usuário, fazendo assim uma contribuição para a redução de erros, possíveis desperdícios de produto e perdas de faturamento. A partir das observações das dificuldades que são encontradas por pequenos empreendedores no aspecto de gestão, pretende-se oferecer uma ferramenta que consiga otimizar o tempo e aumentar a precisão do *status* do estoque.

Objetivos Específicos

- Aplicação de questionário de validação
- Levantamento e análise de requisitos
- Modelagem do sistema
- Prototipação de telas
- Definição das regras de negócios
- Implementação do software
- Testes
- Documentação

1.4 Metodologia

Além da pesquisa qualitativa de caráter bibliográfico para obtenção de conhecimento teórico, a metodologia utilizada neste TCC também contou com a realização de entrevistas com dois microempresários que atuam no controle de estoque e nas atividades de venda. O resultado da pesquisa pode ser visto no

Apêndice A deste documento. O objetivo principal dessas entrevistas foi compreender as reais necessidades, dificuldades e rotinas enfrentadas no dia a dia desses negócios, possibilitando o desenvolvimento de um sistema web mais aderente à realidade dos usuários finais.

As entrevistas foram conduzidas de forma semiestruturada, o que proporcionou flexibilidade para aprofundar aspectos específicos conforme as respostas dos participantes, favorecendo uma compreensão mais completa do contexto estudado. A partir dos dados obtidos, foram definidos os elementos essenciais do sistema, que servirão como base para o seu planejamento, modelagem e posterior implementação.

1.5 Referencial Teórico

Nos dias atuais, a adoção de sistemas digitais para o gerenciamento de estoques configura-se como vetor essencial para a competitividade, especialmente em micro e pequenas empresas, cujos índices de crescimento muitas vezes são limitados pela eficiência operacional. O controle de estoque, conforme Queiroz (2023), “envolve inventário, registro e otimização com o objetivo de melhorar a eficiência e reduzir custos operacionais”, enquanto Carvalho (2023) ressalta que a gestão de estoques extrapola o simples monitoramento de quantidades, englobando o alinhamento entre fluxos de entrada e saída de materiais, o que impacta diretamente os gastos operacionais e a capacidade de resposta ao mercado. Erros nessa gestão levam tanto à escassez de produtos quanto ao excesso de insumos, provocando rupturas de vendas ou capital imobilizado e comprometendo o fluxo de caixa (Carvalho, 2023).

A literatura aponta que, embora modelos clássicos como o Lote Econômico de Compra (EOQ) e práticas Just-in-Time (JIT) ofereçam parâmetros consolidados para redução de custos e minimização de desperdícios, ainda persistem lacunas quanto à adoção dessas abordagens em microempreendimentos. Miranda (2007) documenta ganhos substanciais em controle de vendas, redução de custos e agilidade na tomada de decisão com o uso de sistemas digitais, enquanto Souza et al. (2024, p. 34) relatam que 97% dos gestores observaram melhoria significativa na performance de estoque

após sua implementação. Vedovato et al. (2015, p. 7) sublinham a rapidez com que esses ganhos são percebidos, e Carneiro (2015) ilustra, em estudo de caso em uma concessionária, a elevação da precisão no planejamento de compras e a redução de erros após a digitalização do fluxo de estoque. Contudo, Catarino et al. (2017) alertam para a dependência de planilhas eletrônicas em muitas micro e pequenas empresas, gerando redundâncias e comprometedora confiabilidade dos dados, enquanto relatórios da Target ERP (2025) e da Agência Sebrae de Notícias (2024) enfatizam a falta de monitoramento em tempo real como fator determinante para perdas de faturamento e insatisfação dos clientes.

Diante desse cenário, justifica-se o desenvolvimento de um sistema de gerenciamento de estoque automatizado, acessível e intuitivo, capaz de oferecer monitoramento em tempo real, reduzir erros de lançamento e integrar previsão de demanda. Ao suprir as lacunas identificadas - da escassez à imobilização de capital, da lentidão de planilhas à ausência de dados instantâneos, espera-se não apenas minimizar perdas operacionais, mas também potencializar a agilidade na tomada de decisões, a precisão no planejamento de compras e o nível de serviço ao cliente. Assim, este estudo propõe uma solução digital que alinhe micro e pequenas empresas às melhores práticas de gestão de itens em estoques, promovendo ganhos de eficiência e competitividade.

2 DESENVOLVIMENTO

Neste capítulo, serão detalhados os principais aspectos do desenvolvimento do projeto, incluindo todas as tecnologias escolhidas e utilizadas, a arquitetura do sistema e os processos para a implementação. O objetivo é proporcionar uma solução que proporcione aos usuários uma experiência facilitada, com uma otimização dos processos e melhoria no controle de estoque, atendendo às demandas de micros e pequenas empresas.

2.1 Regras de negócios e Requisitos Funcionais e Não Funcionais

Nesta seção, vamos apresentar as principais regras de negócio, requisitos funcionais e não funcionais do sistema em desenvolvimento. Estas regras e requisitos definem como o sistema deve operar e as funcionalidades que estarão disponíveis para os donos dos comércios cadastrados e usuários autorizados. O Quadro 1 descreve as regras de negócios para este projeto.

Quadro 1 – Regras de negócios

Código	**Descrição**
RN001	Não poderá haver venda caso o estoque esteja negativo ou nulo.
RN002	A venda de produtos deve permitir desconto por item ou no total, com registro do valor final.
RN003	Os relatórios devem apresentar dados consolidados de faturamento, entradas e saídas, permitindo análise por dia, semana e mês.
RN004	Cada produto deve ter um estoque mínimo.
RN005	Qualquer ajuste manual de estoque precisa ser registrado.

Fonte: elaborado pelos autores

O quadro 2 descreve os requisitos funcionais, que são as funcionalidades esperadas para o sistema.

Quadro 2 – Requisitos Funcionais

Código	Descrição
RF001	As permissões para acesso devem ser controladas por um sistema de login criptografado.
RF002	Permitir alteração das configurações do sistema: estoque mínimo padrão, bloqueio ou permissão de vendas com estoque negativo, dados da empresa, preferências como tema do sistema e notificações.
RF003	Exibir alertas visuais para produtos que atingirem ou estiverem abaixo do estoque mínimo definido.
RF004	Permitir cadastro, edição e exclusão de produtos com formulário que contenha campos para nome, categoria, medida, estoque mínimo, preço e quantidade.
RF005	Exibir Dashboard com resumo de métricas principais: produtos cadastrados, total de itens em estoque, valor total do estoque, faturamento e alertas de baixo estoque.
RF006	Disponibilizar buscador inteligente para localizar produtos por nome ou código, com sugestões em tempo real.
RF007	Exibir Dashboard com resumo de métricas principais: produtos cadastrados, total de itens em estoque, valor total do estoque, faturamento e alertas de baixo estoque.
RF008	Permitir registro de movimentações de saída (venda), com seleção de produtos, quantidade, aplicação de descontos e cálculo automático do total.
RF009	Registrar cada movimentação de estoque (entrada e saída), armazenando dados como produtos, quantidade, valores e data/hora.
RF010	Possibilitar cadastro, edição e exclusão de fornecedores, com formulário que inclua nome, CNPJ, endereço, telefone e e-mail.

RF011	O sistema deve controlar produtos com unidades de medida distintas (ex.: quilos, litros, unidades), permitindo cálculo de estoque e valores precisos.

Fonte: elaborado pelos autores

O quadro 3 exibe os requisitos não funcionais, que são restrições esperadas no contexto do sistema proposto.

Quadro 3 – Requisitos Não Funcionais

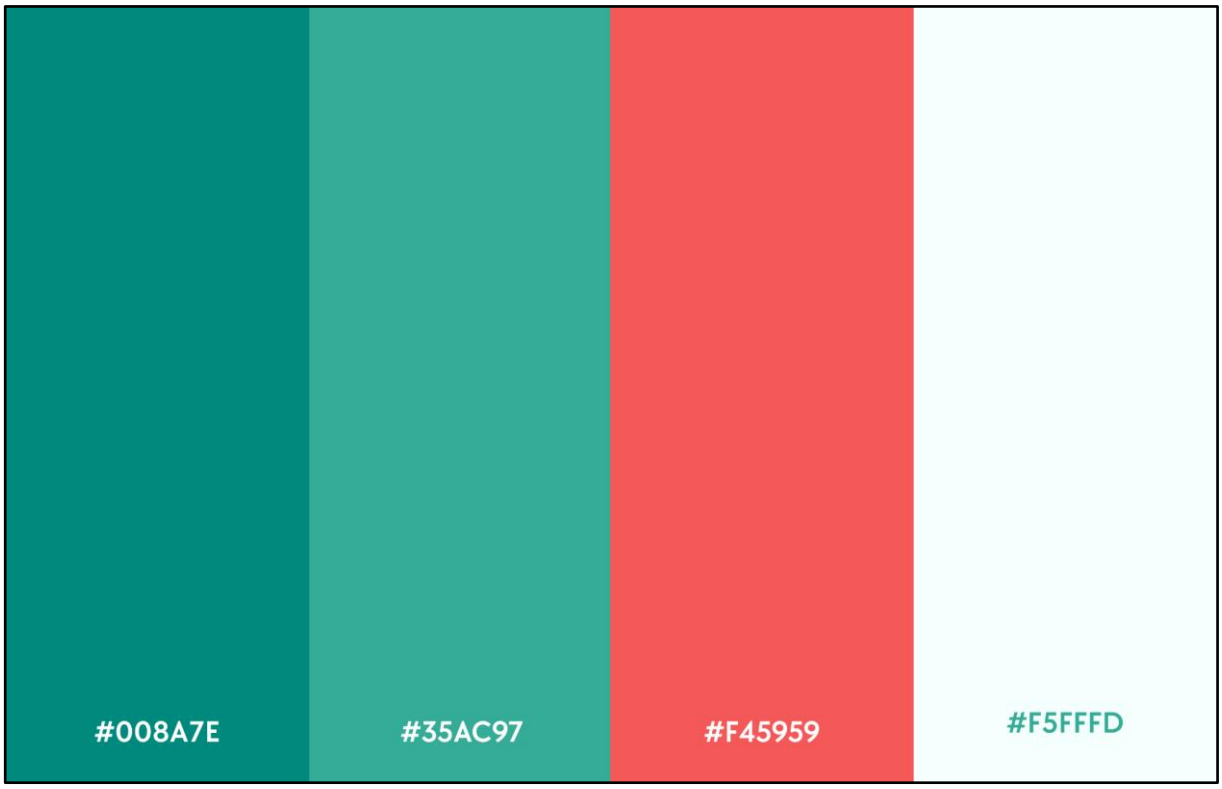
Código	**Descrição**
RNF001	Os carregamentos das telas devem ter no máximo uma duração de 10 segundos.
RNF002	Para o funcionamento do sistema, o dispositivo sendo utilizado deve estar em conexão com uma rede de internet.
RNF003	O carregamento da pesquisa sobre os itens devem ter no máximo uma duração de 5 segundos.
RNF004	O sistema de cadastro e de login devem ter uma resposta de 2 segundos com o banco de dados.
RNF005	O sistema deve suportar autenticação com hash de senhas e JWT
RNF006	O sistema deve ter mitigação contra XSS, CSRF, SQL Injection, deserialização insegura, etc.
RNF007	O sistema deve conter backups idealmente diários, ou semanais.

Fonte: elaborado pelos autores

2.2 Modelagem do sistema (Telas e Casos)

Nesta seção será especificada o processo de modelagem do sistema, contando com a escolha da paleta de cores do sistema, organização das telas, criação do protótipo das telas, e a organização dos diagramas de casos de uso. A figura 1 exibe a paleta de cores:

Figura 1 - Paleta de Cores

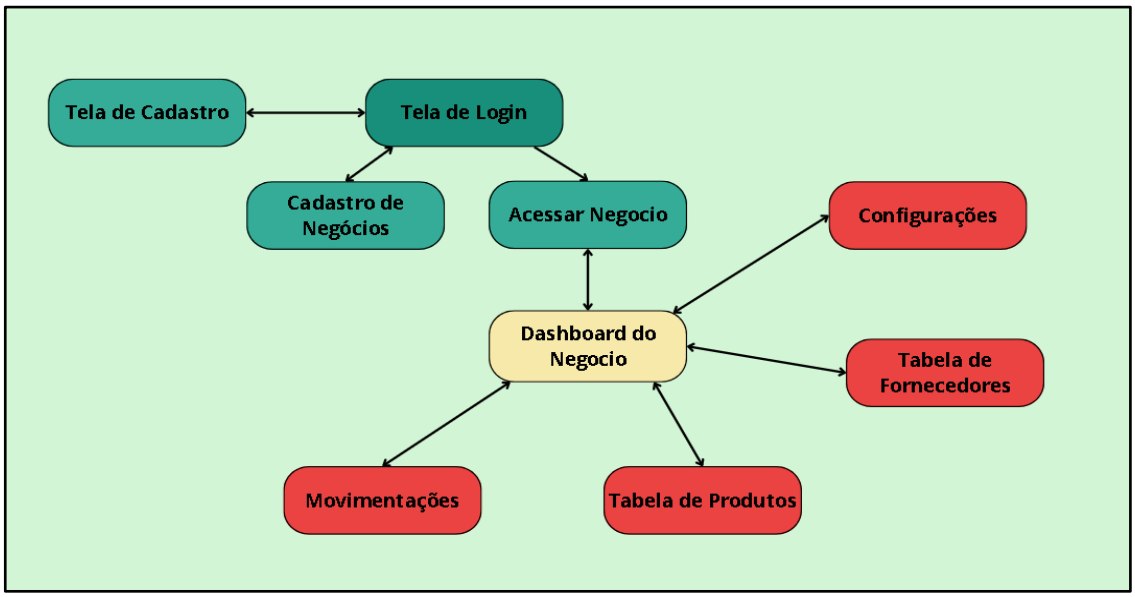


Fonte: elaborado pelos autores

A escolha da paleta de cores se deu devido ao equilíbrio na visão do usuário entre as cores frias, como o Verde-Azulado e o Verde-Água, e as cores quentes, como o Vermelho-Coral e o Branco-Ciano, assim criando um conforto visual e acolhedor para o usuário.

A figura 2 mostra o mapa do site, criado para dar visibilidade à organização das telas:

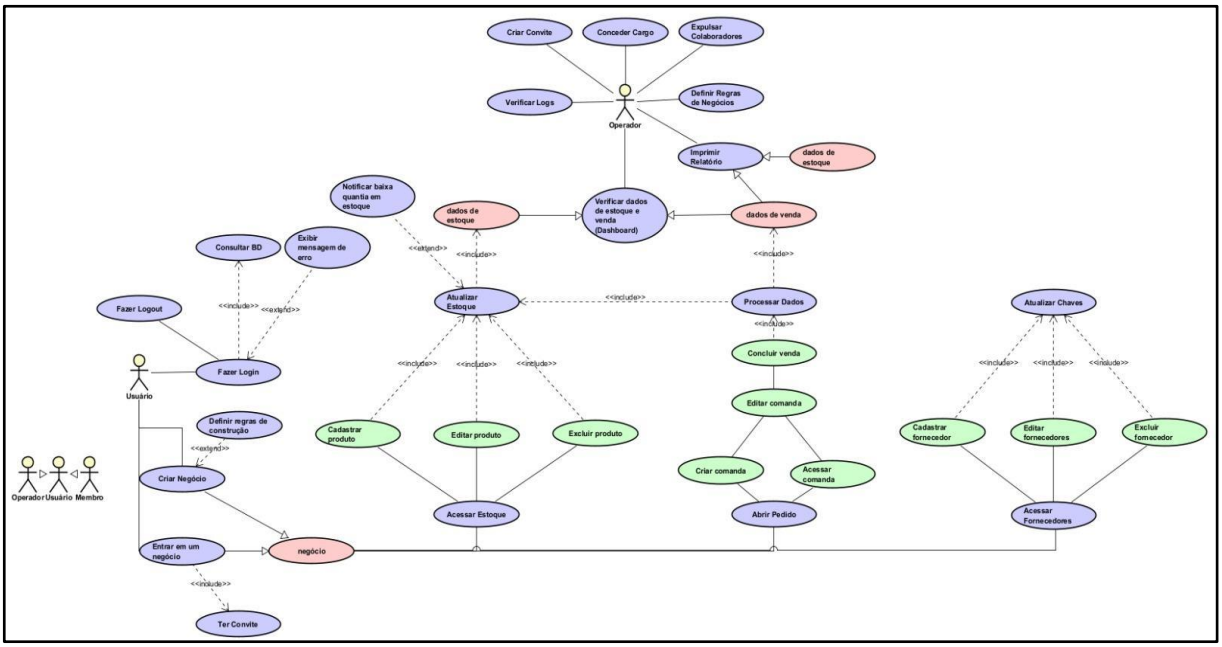
Figura 2 – Mapa do site



Fonte: elaborado pelos autores

A modelagem do sistema por meio de um Diagrama de Casos de Uso, conforme exibido na figura 3, foi um aspecto fundamental para se ter a visão do escopo do projeto. Cada funcionalidade e cada ator estão descritos nesse diagrama.

Figura 3 – Diagrama de Casos de Uso



Fonte: elaborado pelos autores

A modelagem do banco de dados exibida na figura 4 destaca as tabelas, campos, relacionamentos e cardinalidade, planejados para armazenar os dados do projeto.

Figura 4 – Modelagem do Banco de Dados

Fonte: elaborado pelos autores

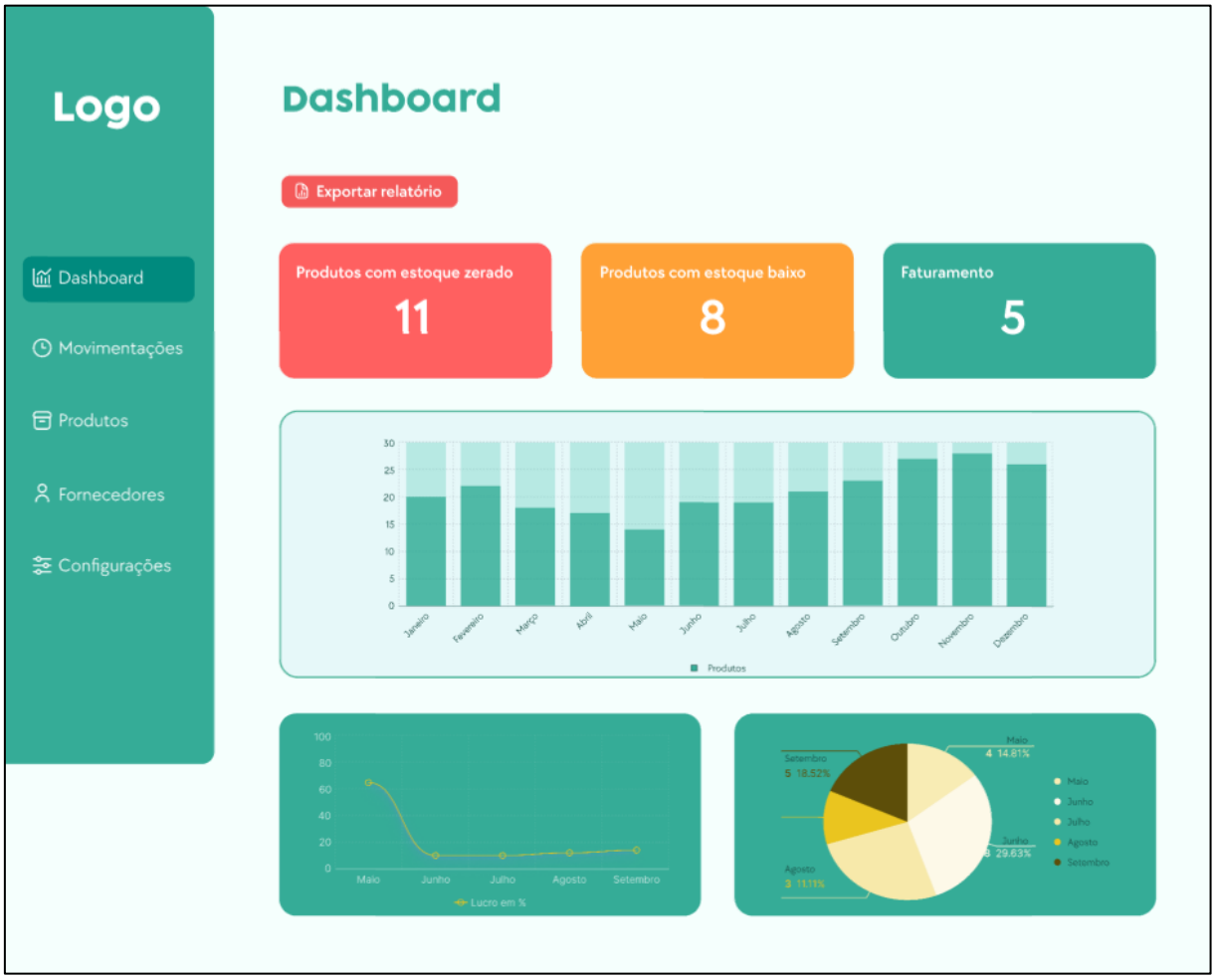
A partir da definição das funcionalidades e também banco de dados, partiu—se para o desenvolvimento das telas do sistema. A seguir são apresentadas as telas principais do sistema. A figura 5 mostra a tela de boas-vindas, onde o usuário pode fazer login ou criar uma nova conta.

Figura 5: Tela de Comércios

Fonte: elaborado pelos autores

A tela demonstrada na Figura 6 trata-se da Dashboard, um local onde o usuário vai poder acessar o resumo das principais informações do seu negócio. Essa é a tela principal do sistema, que servirá de base para as tomadas de decisões estratégicas do negócio, portanto as informações presentes numa Dashboard devem ser de muita precisão.

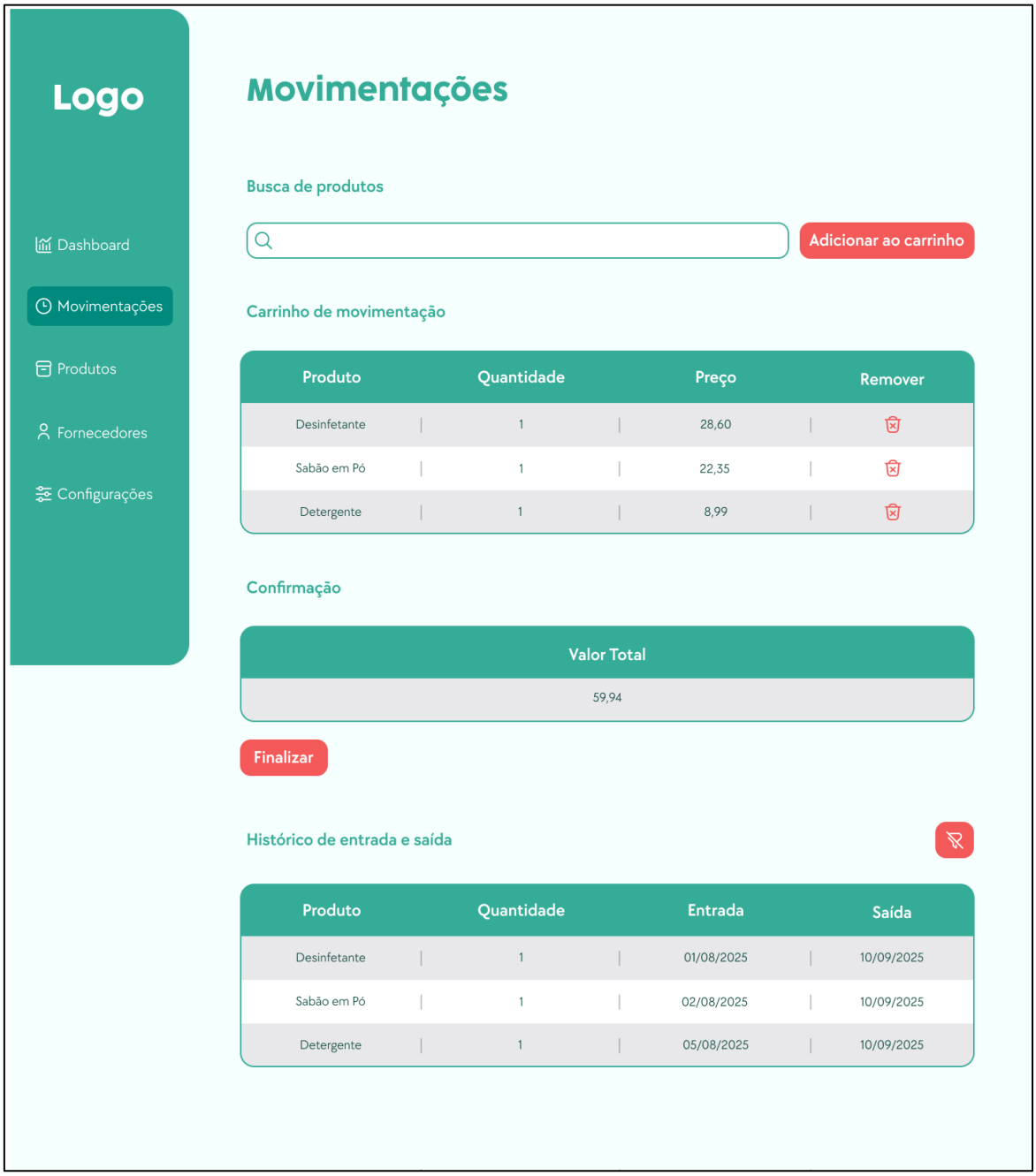
Figura 6: Dashboard do sistema



Fonte: elaborado pelos autores

A figura 7 apresenta a tela de lançamento das movimentações, onde o usuário após fazer a busca pelo produto, poderá adicionar produtos ao carrinho e ao clicar em finalizar, o sistema irá registrar a saída de produtos do estoque.

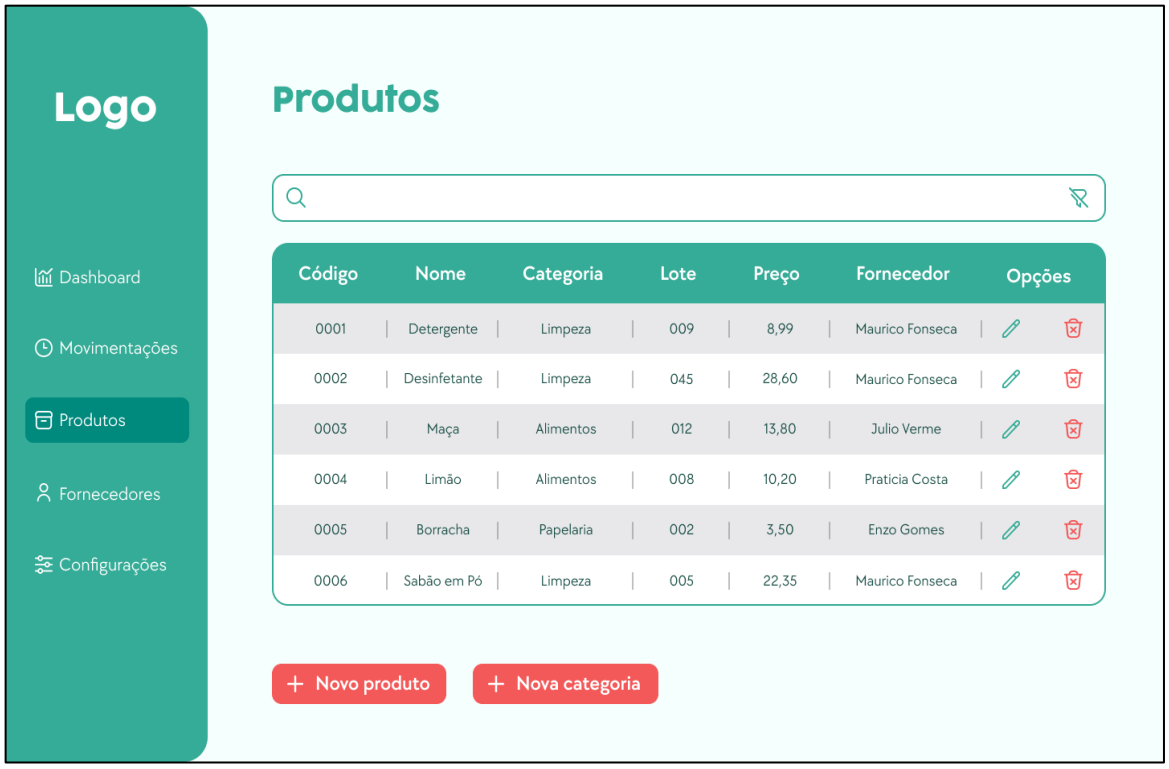
Figura 7: Tela de Movimentações



Fonte: elaborado pelos autores

A figura 8 apresenta a tela de Cadastro de Produtos, onde um novo produto poderá ser armazenado no banco de dados para utilização em movimentações futuras. A partir dessa tela também é possível registrar uma nova categoria.

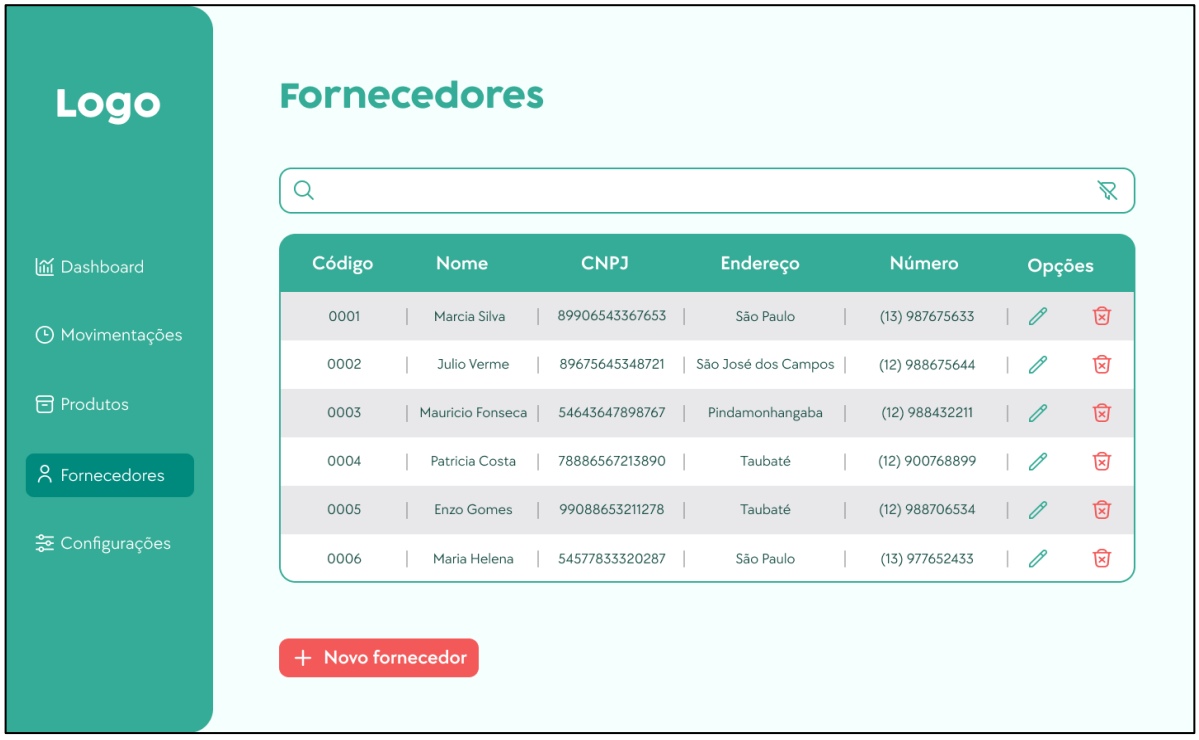
Figura 8: Cadastro de Produtos



Fonte: elaborado pelos autores

A figura 9 mostra a tela de consulta dos fornecedores.

Figura 9: Busca de Fornecedores



Fonte: elaborado pelos autores

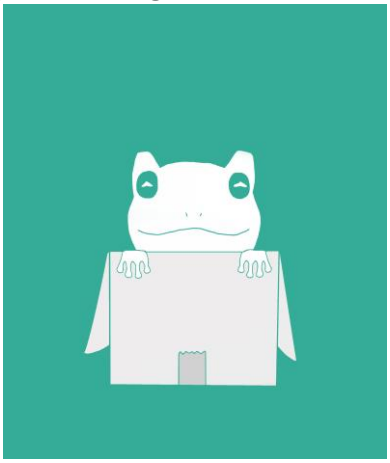
A figura 10 apresenta a Tela de Configurações do sistema, onde o usuário vai poder editar dados sobre a conta da empresa e também configurar parâmetros do estoque, como limite mínimo de estoque e unidade padrão, a ser utilizada para cada produto.

Figura 10: Configurações

Fonte: elaborado pelos autores

A Logomarca do sistema, exibida na figura 11, foi criada a partir da criatividade da equipe.

Figura 11: Logomarca do sistema



Fonte: elaborado pelos autores

A figura 12 mostra a Tipografia e a Cartela de Ícones utilizada.

Figura 12: Tipografia e ícones

Fonte: elaborado pelos autores

2.3 Escolha das Linguagens para Programação

Nesta seção demonstra-se a escolha das Linguagens para a programação e a explicação do porquê da escolha de cada uma delas para cada componente do projeto (Front-End, Back-End, Banco de Dados).

Front-End:

Na seção do desenvolvimento do Front-End, escolheu-se para a linguagem que seria utilizado o HTML, CSS, TypeScript, e com auxílio dos frameworks React e Vite.

- *Por que utilizar o HTML, CSS e TypeScript?*

O HTML é a linguagem de produção base para websites, dando a estrutura da aplicação web. O CSS é a linguagem que auxilia para o embelezamento da aplicação, utilizando de elementos como cores, botões e fontes. E o TypeScript serve para dar uma lógica mais complexa do site e mais interatividade para a aplicação web.

- *Por que utilizar o React e Vite?*

O React é um framework que se utiliza para fazer as UI da aplicação, de como a aplicação aparece e funciona para o usuário. E o Vite é o framework para otimizar o projeto em geral.

Back-End:

Na seção do desenvolvimento do Back-End, escolheu-se para a linguagem que seria utilizado o Python 3.11- slim e com auxílio do framework Flask.

- *Por que utilizar o Python 3.11-Slim?*

O Python é uma linguagem de programação de fácil implementação e com uma biblioteca bem completa incluso dentro da própria linguagem, além de haver uma estruturação de código mais limpa e organizada.

- *Por que utilizar o Flask?*

O Flask é um framework de Python de utilização minimalista, onde diferente de outros frameworks que são maiores como Django, ele vem o essencial, como roteamento e servidor, assim podendo adicionar o que for necessário na programação. Além de facilitar a criação de rotas dentro do site.

Banco de Dados:

Na seção de desenvolvimento do Banco de Dados, escolheu-se utilizar o PostgreSQL.

- *Por que utilizar o PostgreSQL?*

O PostgreSQL é um sistema de banco de dados gratuito e de código aberto com uma alta extensibilidade, em que por conta disso pode se definir seus próprios tipos de dados e criar funções personalizadas.

2.4 Configurações do Ambiente (Docker e GitHub)

Foi utilizado o Docker dentro do sistema para conseguir guardar suas linguagens em containers e padronizar o sistema para que ele funcionasse de forma igual em qualquer máquina, como Windows, Linux, MacOS e outros, como demonstrado na figura 13.

O GitHub foi utilizado para controle de versionamento entre o grupo, visto suas capacidades de guardar *branches* locais e remotas, permitindo um trabalho centralizado para todos.

Figura 13: Configurando o Docker

```yaml
services:
  db:
    image: postgres:16
    restart: always
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: "123"
      POSTGRES_DB: frog
    volumes:
      - ./db/init:/docker-entrypoint-initdb.d
    ports:
      - "5433:5432"
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U postgres -d frog"]
      interval: 5s
      timeout: 5s
      retries: 5

  backend:
    build: ./backend
    depends_on:
      - db
    ports:
      - "3001:3001"
    environment:
      DATABASE_URL: postgresql://postgres:123@db:5432/frog
      DB_HOST: db
      DB_PORT: 5432
      DB_USER: postgres
      DB_PASS: "123"
      PORT: 3001
    volumes:
      - ./backend:/app
    entrypoint: ["/app/entrypoint.sh"]
```

Fonte: elaborado pelos autores

2.5 Programação das Telas

Na programação do login foi utilizado de um formulário simples para ser enviado as informações como *e-mail* e senha para autenticação, e foi utilizado de *cookies* para conseguir armazenar as informações do usuário a partir de pequenos arquivos de texto que são enviados do servidor ao navegador que o guarda e envia automaticamente em caso de requisição.

Na programação do Back-End demonstrada nas figuras a seguir, se cria um sistema de autenticação em Flask usando JWT, com 3 rotas base, a rota de login – demonstrada nas figuras 14 e figura 15 (continuação do código) - verifica no banco o e-mail e senha recebidos e cria tokens de acesso (15 minutos) e refresh (7 dias e guardado como cookie HttpOnly).

Figura 14 – Rota de login

```python
from flask import Blueprint, make_response, request, jsonify, g
from app.database.database import get_db
from ..services.usuarios_service import get_usuario_por_email, get_usuario_por_id
from passlib.hash import bcrypt
import jwt
from datetime import datetime, timezone, timedelta
import os
import uuid

auth = Blueprint("auth", __name__)
SECRET_KEY = os.getenv("SECRET_KEY", "muda_esse_segredo")

# === utilitário para criar tokens (timezone-aware) ===
def _make_jwt(payload: dict, expire_minutes: int = 60) -> str:
    now = datetime.now(timezone.utc)
    claims = payload.copy()
    claims.update({
        "exp": now + timedelta(minutes=expire_minutes),
        "iat": now,
        "jti": str(uuid.uuid4())
    })
    token = jwt.encode(claims, SECRET_KEY, algorithm="HS256")
    if isinstance(token, bytes):
        token = token.decode()
    return token


# === rota de login ===
@auth.route('/login', methods=['POST'])
def login():
    data = request.get_json() or {}
    email = data.get('email')
    senha = data.get('senha')
```

Fonte: elaborado pelos autores

Figura 15 – Rota de login - continuação

```
    if not email or not senha:
        return jsonify({"mensagem": "Email e senha são obrigatórios"}), 400

    db_gen = get_db()
    db = next(db_gen)

    try:
        usuario = get_usuario_por_email(db, email)
        if not usuario:
            return jsonify({"mensagem": "Credenciais inválidas"}), 401

        if not bcrypt.verify(senha, usuario.senha_hash):
            return jsonify({"mensagem": "Credenciais inválidas"}), 401

        # access token de curta duração (recomendado: 15 minutos)
        access = _make_jwt(
            {"usuario_id": usuario.usuario_id, "email": usuario.email},
            expire_minutes=15
        )
        # refresh token mais longo, guardado apenas como cookie HttpOnly
        refresh = _make_jwt({"usuario_id": usuario.usuario_id, "type": "refresh"},
                            expire_minutes=60 * 24 * 7)

        resp = make_response(jsonify({
            "usuario": {
                "usuario_id": usuario.usuario_id,
                "email": usuario.email,
                "nome": usuario.nome_completo
            },
            "access_token": access,
            "token_type": "Bearer",
            "expires_in": 15 * 60
```

Fonte: elaborado pelos autores

A figura 16 mostra a rota *refresh* que serve para gerar um novo *acess token* a partir do *refresh token* armazenado no *cookie*.

Figura 16 – Rota refresh

```
        secure_flag = os.getenv("FLASK_ENV") == "production"
        resp.set_cookie(
            "refresh_token",
            refresh,
            httponly=True,
            samesite='Lax',
            secure=secure_flag,
        )

        return resp, 200
    finally:
        db_gen.close()


# === rota para refresh ===
@auth.route('/refresh', methods=['POST'])
def refresh():
    refresh_token = request.cookies.get('refresh_token')
    if not refresh_token:
        return jsonify({'mensagem': 'Refresh token ausente'}), 401

    try:
        dados = jwt.decode(refresh_token, SECRET_KEY, algorithms=["HS256"])
        if dados.get('type') != 'refresh':
            return jsonify({'mensagem': 'Token inválido'}), 401

        new_access = _make_jwt({"usuario_id": dados['usuario_id']}, expire_minutes=15)
        resp = make_response(jsonify({'mensagem': 'ok', 'access_token': new_access}))
        return resp, 200
    except jwt.ExpiredSignatureError:
        return jsonify({'mensagem': 'Refresh token expirado'}), 401
    except jwt.InvalidTokenError:
```

Fonte: elaborado pelos autores

A figura 17 mostra a rota *logout* que serve para apagar o *refresh token* assim finalizando a sessão do usuário.

Figura 17 – Rota logout

```
interface AuthContextType {
  user: User;
  token?: string | null;
  loading: boolean;
  comercios?: number[] | null;
  checkAuth: () => Promise<boolean>;
  login: (email: string, password: string) => Promise<boolean>;
  logout: () => Promise<void>;
  setUser: (u: User) => void;
  setToken?: (t: string | null) => void;
}
```

```
        return jsonify({'mensagem': 'Token inválido'}), 401


# === rota de logout ===
@auth.route('/logout', methods=['POST'])
def logout():
    resp = make_response(jsonify({'mensagem': 'Deslogado'}))
    resp.delete_cookie('refresh_token')
    return resp
```

Fonte: elaborado pelos autores

A figura 18 que está dividida em partes sequenciais logo a seguir, mostra a programação do Front-end, onde foi feita uma programação de autenticação geral em React, em que o navegador consegue guardar o usuário e o token criado, também permitindo fazer login e logout do sistema e com uma verificação se o usuário está autenticado, e quando a página carrega restaure o login automaticamente.

Figura 18 – Programação do front-end

```
const AuthContext = createContext<AuthContextType | undefined>(undefined);

export function useAuth() {
  const ctx = useContext(AuthContext);
  if (!ctx) throw new Error('useAuth must be used within AuthProvider');
  return ctx;
}

export function AuthProvider({ children }: { children: ReactNode }) {
  const [user, setUser] = useState<User>(null);
  const [token, setToken] = useState<string | null>(null);
  const [loading, setLoading] = useState(true);
  const mountedRef = useRef(true);

  async function checkAuth(): Promise<boolean> {
    if (!mountedRef.current) return false;
    if (mountedRef.current) setLoading(true);
    try {
      const data = await authServices.fetchCurrentUser();
      if (!mountedRef.current) return false;
      const u = data?.usuario ?? null;
      setUser(u);
      return !!u;
    } catch (err) {
      if (mountedRef.current) setUser(null);
      return false;
    } finally {
      if (mountedRef.current) setLoading(false);
    }
  }
```

```
  async function login(email: string, password: string): Promise<boolean> {
    try {
      const res = await api.post(LOGIN, { email, senha: password });
      if (res.data?.access_token) {
        const newToken = res.data.access_token as string;
        authServices.setAccessToken(newToken);
        setToken(newToken);
        await checkAuth();
        return true;
      }
      return false;
    } catch (err) {
      return false;
    }
  }

  async function logout() {
    try {
      await authServices.logoutServer();
    } finally {
      setUser(null);
      authServices.setAccessToken(null);
      setToken(null);
    }
  }

useEffect(() => {
  mountedRef.current = true;
  (async () => {
   try {
     // tenta obter token do authServices (se existir) ou do localStorage como fallback
     const existingToken = (authServices as any).getAccessToken?.() ?? localStorage.getItem("access_token");

     if (existingToken) {
       setToken(existingToken as string);
       authServices.setAccessToken(existingToken as string);
     }
     else {
       try {
         const newToken = await (authServices as any).refresh?.()
         if (newToken) setToken(newToken);
       }
       catch(_)
       {/*sem token? ok, ficará deslogado*/}
       await checkAuth();
     }
   }
```

```
      finally {
        if(mountedRef.current) setLoading(false)
      }
    })();
    return () => { mountedRef.current = false; };
  }, []);

  return (
    <AuthContext.Provider value={{ user, token, setUser, loading, checkAuth, login, logout, setToken }}>
      {children}
    </AuthContext.Provider>
  );
}
```

Fonte: elaborado pelos autores

- Tela de Cadastro:

Na programação de cadastro demonstrado na figura 19 - dividida em partes para melhor organização - foi utilizado também um formulário simples para ser enviado as informações cadastradas para o banco de dados, e com um script para confirmação de e-mail e senha.

Figura 19 – Programação do Cadastro

```
import { useState } from "react";
import "./formCriarConta.css";
import Step1 from "./multistepform/Step1";
import Step2 from "./multistepform/Step2";
import { FormProvider, useForm, type SubmitHandler } from "react-hook-form";
import { zodResolver } from "@hookform/resolvers/zod";
import type { FormData } from "./schemas.ts"
import { formSchema } from "./schemas";
import api from "src/api/axios.ts";
import { CADASTRO, LOGIN } from "src/api/enpoints.ts";
import { useNavigate } from "react-router";
import { useAuth } from "src/api/auth/AuthProvider.tsx";
import { FaArrowLeft, FaArrowRight } from "react-icons/fa";

export default function FormCadastrarUsuario() {
    const navigate = useNavigate();
    const { checkAuth, login } = useAuth();

    const methods = useForm<FormData>({
        resolver: zodResolver(formSchema),
        reValidateMode: "onChange",
        defaultValues: {
            nome: "",
            email: "",
            senha: "",
            confirmarSenha: ""
        }
    })

    const stepFields: Record<number, (keyof FormData)[]> ={
        0: ["email", "nome"],
        1: ["senha", "confirmarSenha"],
        2: ["senha", "confirmarSenha"]
    };
```

```
const handleLogin = async (data: any) => {
    try{
        if(await login(data?.email, data?.senha))
            navigate("/entrar", { replace: true });
    }
    catch (error) {
        console.error("Erro ao acessar a conta:", error);
        alert("Erro ao acessar a conta. Tente novamente.");
    }
}

const onSubmit: SubmitHandler<FormData> = async (data) => {
    api.post(CADASTRO, data)
        .then((res) => {
            alert("Cadastro com sucesso!");
            console.log("Usuário cadastrado:", res.data);
            methods.reset();
            handleLogin({ email: data.email, senha: data.senha });
        })
        .catch((error) => {
            console.error("Erro ao cadastrar usuário:", error);
            alert("Erro ao cadastrar usuário. Tente novamente.");
        });
}
```

```
const [step, setStep] = useState(0);
const previousStep = () => (setStep(prev => (prev > 0 ? prev - 1 : prev)));
const nextStep = async () => {
    let valid = false;
    if (step<2)
    {valid = await methods.trigger(stepFields[step]);}
    if (valid && step < 1)
        {setStep(step + 1); (document.getElementById("senha1") as HTMLInputElement).focus();}
};

return (<>
<FormProvider {...methods}>
    <form onSubmit={methods.handleSubmit(onSubmit)} className="formWrapper">
        <h2>Cadastrar</h2>
        <div
              className="slider"
              style={{ transform: `translateX(-${step * 100}%)` }}
            >
              <fieldset className="slide" disabled={step !== 0}><Step1 wrapperClassName="stepWrapper" /></fieldset>
              <fieldset className="slide" disabled={step !== 1}><Step2 wrapperClassName="stepWrapper" /></fieldset>
            </div>
        <div className="controls">
            <div>
                {step > 0 && <button type="button" onClick={previousStep}>
                    <FaArrowLeft/>
                </button>}
            </div>
```

```
                <div>
                    {step < 1 && <button type="button" onClick={nextStep}>
                        <FaArrowRight/>
                    </button>}
                    {step === 1 && <button type="submit">
                        <div className="finalizeButton">
                            <p>Cadastrar</p>
                        </div>
                    </button>}
                </div>
            </div>
        </form>
    </FormProvider>
     </>)
}
```

Fonte: elaborado pelos autores

Na figura 20 fica demonstrada a programação que detecta se o email é válido, caso não seja retorna uma mensagem com o que deve ter no Email, e a validação da senha, em que caso a segunda senha esteja diferente retorna uma mensagem de que a senha deve ser igual.

Figura 20 – Verificação de e-mail

```
import { z } from "zod";

export const step1Schema = z.object({
    email: z.string().max(150, "Email deve ter no máximo 150 caracteres").regex(/^[^\s@]+@[^\s@]+\.[^\s@]+$/, "Email inválido"),
    nome: z.string().min(1, "Nome do Proprietário é obrigatório").max(150, "Nome do Proprietário deve ter no máximo 150 caracteres"),
})

export const step2Schema = z.object({
    senha: z.string().min(6, "Senha deve ter pelo menos 6 caracteres").max(255, "Senha deve ter no máximo 255 caracteres"),
    confirmarSenha: z.string().min(6, "Confirmação de Senha é obrigatória").max(255, "Confirmação de Senha deve ter no máximo 255 caracteres"),
})
 export const formSchema = z.
     object({
         ...step1Schema.shape,
         ...step2Schema.shape
     })
     .superRefine((data, ctx) => {
         if (data.senha !== data.confirmarSenha) {
             ctx.addIssue({
                 path: ["confirmarSenha"],
                 code: "custom",
                 message: "As senhas devem ser iguais",
         });
             ctx.addIssue({
                 path: ["senha"],
                 code: "custom",
                 message: "As senhas devem ser iguais",
             });
         }
     });

 export type FormData = z.infer<typeof formSchema>;
```

Fonte: elaborado pelos autores

A figura 21 mostra a codificação da Tela de comércios, que foi programada para que criasse um comércio com suas predefinições como unidade de medida padrão e visualizasse os comércios já cadastrados do usuário.

Figura 21 – Tela de comércios

```
function MeusComercios() {
  const { user } = useAuth();
  const [isModalOpen, setIsModalOpen] = useState(false);
  const [comercios, setComercios] = useState<Comercio[]>([]);
  const [isRefreshing, setIsRefreshing] = useState<boolean>(false);
  const [error, setError] = useState<string>("");
  const isMountedRef = useRef(false);

  const fetchComercios =  useCallback(async () => {
    setError("");
    setIsRefreshing(true);

    try {
      const res = await api.get<Response>(`${ME}/comercios`);
      const data = res.data;

      if (!isMountedRef.current) {return;};
      setComercios(Array.isArray(data?.comercios) ? data.comercios : []);
    } catch (err) {
      const axiosErr = err as AxiosError<any>;
      const serverMessage =
        axiosErr?.response?.data?.message ??
        axiosErr?.message ??
        "Erro inesperado na requisição";
      if (isMountedRef.current) setError(serverMessage);
    } finally {
      if (isMountedRef.current) setIsRefreshing(false);
    }
  }, []);
```

Fonte: elaborado pelos autores

Esta foi a programação das telas principais do sistema.

3 CONSIDERAÇÕES FINAIS

Este trabalho culminou no desenvolvimento e apresentação de um protótipo funcional que integra as camadas essenciais de uma aplicação de gestão digital. As funcionalidades cruciais previstas no escopo foram implementadas e demonstradas, permitindo validar os fluxos básicos de cadastro, persistência de dados e operações centrais da aplicação. A entrega confirmou a viabilidade técnica da solução proposta e a sua aptidão como prova de conceito para evoluções futuras. Contudo, é necessário explicitar as limitações encontradas para orientar as próximas etapas de desenvolvimento.

Em razão do cronograma restrito, a profundidade do trabalho foi reduzida, limitando a validação em ambientes reais e a possibilidade de refinamentos iterativos no design. Requisitos não-funcionais importantes não foram plenamente contemplados, entre eles aspectos de acessibilidade e experiência do usuário, o sistema de pesquisa rápida e o módulo de hierarquia de itens e papéis. Além dessas lacunas, o sistema de relatórios relacionado ao controle de estoque não foi implementado, impedindo a extração de informações gerenciais essenciais. Nota-se que essas deficiências não prejudicam o funcionamento da aplicação no papel em que se propõe ocupar, embora as implementações dessas funcionalidades consolidariam o programa no cenário comercial.

Diante desses limites, recomenda-se priorizar a finalização dos requisitos não-funcionais críticos nas próximas etapas de desenvolvimento. Também é importante conduzir testes de carga e otimizações de desempenho para garantir escalabilidade e revisar aspectos de segurança como controle de acesso, criptografia e políticas de backup. A produção de documentação técnica e manuais de usuário deve acompanhar essas implementações para facilitar a adoção e a manutenção do sistema.

Recomenda-se ainda planejar integrações com ferramentas do mercado, tais como leitores de código de barras, sistemas contábeis e marketplaces, para ampliar o valor agregado da solução. A realização de estudos de implantação em ambientes reais permitirá coletar métricas de uso e impacto operacional que orientem a

priorização de funcionalidades futuras. De tal forma, seria possível aprimorar o faturamento do negócio e otimizar a gestão e a análise do estoque.

Com a execução das ações propostas será possível transformar o Frog de uma prova de conceito funcional em uma solução mais madura, confiável e alinhada às necessidades dos usuários. Essas evoluções contribuirão para que a ferramenta seja adotada de forma sustentável e gere valor prático em contextos reais.

4 REFERÊNCIAS

AGÊNCIA SEBRAE DE NOTÍCIAS – TOCANTINS. **Como organizar o estoque de pequenos negócios com eficiência**. 17 dez. 2024. Disponível em: https://to.agenciasebrae.com.br/arquivo/como-organizar-o-estoque-de-pequenos-negocios-com-eficiencia/. Acesso em: 23 maio 2025.

CARNEIRO, Isadora; VERONA, Gisele; COSTA, Vanessa**. Estudo de caso sobre o uso de sistemas de controle de estoque em uma microempresa do setor automotivo**. 2015. Trabalho de Conclusão de Curso (Curso de Tecnologia em Logística) – Pontifícia Universidade Católica do Paraná, Curitiba, 2015.

CARVALHO, Antônia Valéria Veras. **Estoque fora da caixa.** São Paulo: Editora Antônia Valéria Veras Carvalho, 2023.

CATARINO, Flávia Ramielle Silva; SANTOS, Marina Antunes dos; GONTIJO, Tiago Silveira; RODRIGUES, Alexandre de Cássio. **Gestão de estoque em uma microempresa do ramo alimentício: comparação entre a curva ABC e o método XYZ.** *Revista Caribeña de Ciencias Sociales*, n. 4, 2017. Disponível em: https://www.eumed.net/rev/caribe/2017/04/abcxyz.html. Acesso em: 23 maio 2025.

DINIZ VELOSO, Thamiris; FERNANDES DA FONSECA, Cassio. **CONTROLE E GESTÃO DE ESTOQUES: ESTUDO DE CASO EM UMA MICROEMPRESA.** Revista Latino-Americana de Inovação e Engenharia de Produção, *[S. l.]*, v. 6, n. 9, p. 189–201, 2018. DOI: 10.5380/relainep.v6i9.57517. Disponível em: https://revistas.ufpr.br/relainep/article/view/57517. Acesso em: 2 jun. 2025.

MIRANDA, Ângelo. Revista Solução Sama: Tecnologia da informação. **Organização e informatização**. São Paulo, v.18, n. 18, p. 14, 2007. Acesso em: 30 de maio 2025.

SOUZA, Danubia de Oliveira; JESUS, Jéssica Natalia de; SILVA, Sergio Luiz Rosa da; SILVA, Tatiana Santos Barbosa da. **A importância do gerenciamento de estoque através de planilhas eletrônicas para redução de perda em empresas do ramo alimentício**. São Paulo: ETEC Cidade Tiradentes – Extensão Céu Alto Alegre, 2024. Disponível em: https://ric.cps.sp.gov.br/handle/123456789/21328

TARGET ERP. **Gestão manual: o perigo invisível nas pequenas empresas**. Disponível em: https://targeterp.com.br/gestao-manual-o-perigo-invisivel-nas-pequenas-empresas/. Acesso em: 23 maio 2025.

QUEIROZ, Igor Moreira; CRUZ, Sérgio dos Santos. **A importância do controle de estoque: um estudo de caso em uma loja de autopeças em Ilhéus-BA.** *Revista Ibero-Americana de Humanidades, Ciências e Educação*, v. 9, n. 10, p. 3464-3477, out. 2023. Disponível em: https://periodicorease.pro.br/rease/article/view/11993. Acesso em: 30 maio 2025.

VEDOVATO, Ademir Junior; BRITO, Amanda Caroline de; SANTANA, Jessycka Brandão; CASTRO, Tainara Rigotti de. **A utilização da Tecnologia de Informação**

na gestão de estoques em uma empresa varejista. In: ENCONTRO DE ENGENHARIA DE PRODUÇÃO AGROINDUSTRIAL, 9., 2015, Campo Mourão. *Anais...* Campo Mourão: UNESPAR – Campus de Campo Mourão, 2015.

5 APÊNDICE A – ENTREVISTA PARA LEVANTAMENTO DE DADOS

Sobre o Negócio

1. Qual é o nome e o ramo da sua empresa?

Pesqueiro Espelho d'Água; restaurante.

2. Poderia descrever brevemente sua empresa/negócio?

Um pequeno negócio de entrada e saída de bebidas, porções e outros produtos, do qual ocasionalmente há eventos com descontos e maior consumo dos produtos. Trabalha-se principalmente com cerveja e porções.

3. Quantos funcionários lidam com o estoque ou vendas atualmente?

Dois.

4. Qual é a sua maior dor de cabeça hoje com o controle de vendas e estoque?

Fluxo de clientes inesperado: enquanto há dias em que há muitas compras e poucas vendas em relação aos produtos em estoque, o contrário também acontece, gerando perda de faturamento e correria para reabastecer o estoque.

5. Como você descreveria o problema citado anteriormente? De que modo ele afeta seu negócio?

Um problema causado principalmente pela falta de compreensão do fluxo do período. Afeta o faturamento de forma negativa.

6. De que forma o problema é contornado atualmente? Há alguma solução em mente para resolvê-lo de forma definitiva?

Não é contornado, os produtos simplesmente não são vendidos. Compreendendo melhor o movimento de um período, assim planejando e comprando a quantidade mais aproximada dos produtos.

Sobre o Estoque

1. Como você controla seu estoque hoje (planilhas, papel, outro sistema)?

O estoque é gerido de acordo com a necessidade (se algo falta, alguém sai e busca). Em um geral, não há controle e o gerenciamento é realizado pela necessidade do dia. O pedido referente a reposição de estoque é anotado no Whatsapp. Para localizar o que falta, os itens são contados manualmente.

2. Você trabalha com data de validade? Lotes? (caso de produtos perecíveis)

Apenas validade para os congelados.

3. Os produtos têm unidades diferentes (ex: kg, litros, unidades)?

Sim, a cerveja é contada em caixas, as carnes são contadas em quilo e as bebidas quentes são contadas em litros.

4. Precisa de relatórios de movimentação ou histórico de entrada/saída?

Sim, é importante.

Sobre as Vendas

1. Como você faz as vendas hoje? (caixa, anotação manual, outro sistema)

Anota-se o pedido no papel e realiza-se o pagamento durante a saída. Para a questão do cálculo do faturamento, os pedidos são anotados no Whatsapp e posteriormente são processados.

2. Os preços dos produtos mudam com frequência?

Não.

3. Precisa aplicar descontos por item ou no total da venda?

Sim, desconto por item.

4. Precisa gerar algum tipo de comprovante da venda (recibo, nota simples)?

Opcional, mas não necessário.

5. Como você gostaria que fosse o processo de selecionar e somar produtos na venda?

Consulta através de um indexador de busca com listagem dos produtos. Após a seleção, é desejado um elemento para alterar a quantidade de um mesmo produto de forma eficiente.

Sobre Relatórios e Dashboard

1. Quais informações você gostaria de ver logo ao entrar no sistema?

Alertas de estoque baixo com valor pré-determinado (mas editável) na criação do produto.

2. Você se interessa por gráficos (vendas por período, produtos mais vendidos)?

Há uma preferência em focar-se em um relatório bem estruturado, mas sim.

Sobre Configurações e Acesso

1. Você será o único a usar o sistema ou terá outros usuários?

Haverá outros.

2. Precisa cadastrar dados da empresa no sistema (ex: logotipo, CNPJ)?

Sim (cnpj, endereço, telefone…).

3. Há alguma exigência de segurança? (ex: criptografia, controle de acesso, backup automático…)

A princípio, apenas a tela de login será suficiente.

Sobre Necessidades Técnicas

1. Vai usar o sistema em computador, celular ou ambos?

Em ambos.

2. Precisa ser acessado apenas internamente ou deve funcionar na nuvem

Na nuvem.

3. Há alguma função que você sempre quis num sistema, mas nunca viu funcionando direito?

A princípio, não.

4. Tem alguma consideração a se fazer sobre o futuro sistema?

É desejado que o design do site seja nítido e ergonômico.