

**UNIVERSIDADE DE SOROCABA
PRÓ-REITORIA DE GRADUAÇÃO E ASSUNTOS ESTUDANTIS
ENGENHARIA DA COMPUTAÇÃO**

**Gabriel Claro de Almeida
Gabriel Ramon Ribeiro Ramos
José Antônio Moraes de Oliveira
Yosaf Marques Machado**

FLUXO – FERRAMENTA DE LOGÍSTICA UNIFICADA E OTIMIZADA

**Sorocaba/SP
2025**

RESUMO

Este relatório apresenta o desenvolvimento do FLUXO – Ferramenta de Logística Unificada e Otimizada, projeto integrador realizado na Universidade de Sorocaba (UNISO), com o objetivo integrar e automatizar o controle de estoque, minimizando custos e falhas operacionais, por meio de uma interface web responsiva e um aplicativo móvel híbrido. O desenvolvimento seguiu seis fases: (1) fundamentação teórica, embasada em usabilidade e design de informação; (2) definição de identidade visual; (3) prototipagem de telas de login, consulta, cadastro e administração; (4) implementação front-end com HTML5, CSS3 e JavaScript; (5) construção de API RESTful em Spring Boot/Java com autenticação JWT e persistência via PostgreSQL; e (6) desenvolvimento mobile em Flutter com WebView para garantia de consistência entre plataformas. Os testes demonstraram usabilidade intuitiva e desempenho satisfatório, conferindo ao FLUXO aplicabilidade em cenários acadêmicos e em pequenas e médias empresas.

Palavras-chave: Gestão inteligente de estoque; Controle de estoque; Otimização Logística; Automação de Processos; Inovação Tecnológica.

ABSTRACT

This report presents the development of FLUXO – Unified and Optimized Logistics Tool, an Integrator Project carried out at the University of Sorocaba (UNISO), aimed to integrate and automate inventory management, reducing costs and operational errors through a responsive web interface and a hybrid mobile app. Development proceeded in six stages: (1) theoretical foundation grounded in usability and information design; (2) visual identity definition; (3) prototyping of login, search, registration, and administration screens; (4) front-end implementation using HTML5, CSS3, and JavaScript; (5) RESTful API development with Spring Boot/Java, JWT authentication, and PostgreSQL persistence; and (6) mobile development in Flutter with WebView to ensure cross-platform consistency. Testing confirmed intuitive usability and adequate performance, demonstrating FLUXO's applicability in both academic environments and small- to medium-sized enterprises.

Keywords: Smart Inventory Management; Inventory Control; Logistics Optimization; Process Automation; Technological Innovation.

LISTA DE ILUSTRAÇÕES

Figura 1 - Logotipo FLUXO	15
Figura 2 - Protótipo da tela de login	16
Figura 3 - Protótipo da tela de consulta de produtos (lista)	17
Figura 4 - Protótipo da tela de cadastro de produtos	18
Figura 5 - Protótipo da tela de visualização detalhada (dados gerais)	18
Figura 6 - Protótipo da tela de visualização detalhada (dados de estoque)	19
Figura 7 - Protótipo da tela de gestão de estoque.....	20
Figura 8 - Protótipo da tela de gestão de estoque (incluir lançamento)	20
Figura 9 - Protótipo da tela de listagem de usuários	21
Figura 10 - Protótipo da tela de listagem de usuários (incluir usuário)	22
Figura 11 - Protótipo da tela de listagem de usuários (gerenciar usuário)	22
Figura 12 - Tela de login finalizada	23
Figura 13 - Tela de consulta de produtos finalizada	23
Figura 14 - Tela de cadastro de produtos finalizada	24
Figura 15 - Tela de visualização detalhada finalizada (dados gerais)	24
Figura 16 - Tela de visualização detalhada finalizada (dados de estoque)	24
Figura 17 - Tela de gestão de estoque finalizada.....	25
Figura 18 - Tela de gestão de estoque finalizada (incluir lançamento)	25
Figura 19 - Tela de listagem de usuários finalizada	25
Figura 20 - Tela de listagem de usuários finalizada (incluir usuário)	26
Figura 21 - Tela de listagem de usuários finalizada (gerenciar usuário)	26
Figura 22 – Overview do Projeto no Render	30

LISTA DE SIGLAS E ABREVIATURAS

SIGLA	Significado (digitar a SIGLA e pressionar a tecla TAB)
FLUXO	Ferramenta de Logística Unificada e Otimizada (software de gestão de estoque)
APP	Application
API	Application Programming Interface
JWT	JSON Web Token
ORM	Object Relational Mapping
DTO	Data Transfer Object
UI	User Interface

SUMÁRIO

1	INTRODUÇÃO.....	6
2	TERMO DE ABERTURA DE PROJETO	7
3	OBJETIVOS	12
4	CONTEXTUALIZAÇÃO DO PROJETO	13
5	DESENVOLVIMENTO	14
5.1	Fundamentação Teórica.....	14
5.2	Etapas do Projeto	14
5.2.1	Etapa 1: Identidade Visual	14
5.2.2	Etapa 2: Funcionalidades Gerais do Sistema.	15
5.2.3	Etapa 3: Prototipagem das Telas.....	15
5.2.4	Desenvolvimento Frontend (HTML5, CSS3 e JavaScript)	23
5.2.5	API RESTful e Banco de Dados	26
5.2.6	Desenvolvimento Mobile	30
6	CONSIDERAÇÕES FINAIS.....	32
	REFERÊNCIAS.....	33

1 INTRODUÇÃO

A **Ferramenta de Logística Unificada e Otimizada “Fluxo”** é um software de controle de estoque desenvolvido para simplificar processos logísticos por meio da integração entre programação, gestão e inovação. Ao resolver problemas como registros inconsistentes e monitoramento ineficiente de insumos, o projeto busca consolidar competências técnicas e operacionais, unindo teoria e prática para preparar os estudantes para desafios do mercado, como a otimização de recursos e redução de custos. A iniciativa demonstra como soluções tecnológicas aplicadas à educação podem oferecer respostas concretas a problemas reais da área logística.

2 TERMO DE ABERTURA DE PROJETO

TERMO DE ABERTURA DO PROJETO

1 - Nome do Projeto	2 - Código
<i>FLUXO - FERRAMENTA DE LOGISTICA UNIFICADA E OTIMIZADA</i>	<i>003</i>
3 - Líder do Projeto	3.1 - Área de lotação
<i>José Antonio Moraes de Oliveira</i>	<i>Responsável</i>
3.2 - E-mail	3.3 - Telefone
<i>jose.joz46@gmail.com</i>	<i>(15) 99734-4986</i>
4 - Gestores do Projeto	4.1 - Área de lotação
<i>José Antonio Moraes de Oliveira</i>	<i>Gerência</i>
4.2 - E-mail	4.3 - Telefone
<i>jose.joz46@gmail.com</i>	<i>(15) 99734-4986</i>
5. Objetivo do Documento	
<p>Este documento tem como objetivo autorizar formalmente o início do projeto e estabelecer as diretrizes essenciais para seu entendimento, incluindo a definição macro do serviço a ser desenvolvido, o contexto estratégico com justificativas e benefícios esperados, a estrutura de governança com papéis, responsabilidades e recursos alocados, além de uma síntese dos requisitos e entregas prioritárias para garantir aderência às expectativas das partes envolvidas.</p>	

6 - Histórico de Mudança

Versão	Data	Descrição	Autor
0.1	17/02/2025	Definição do Projeto, objetivo, justificativa, parte interessada e equipe.	José Antonio Moraes de Oliveira
0.2	24/03/2025	Alteração da composição da equipe do projeto para inclusão de novo membro.	José Antonio Moraes de Oliveira
0.3	07/04/2025	Adicionar as etapas já prontas.	José Antonio Moraes de Oliveira
0.4	15/04/2025	Adequação do Cronograma, Revisão de Funções, Atualização do Escopo e Não-Escopo.	José Antonio Moraes de Oliveira

7 - Objetivo do Projeto

Desenvolver um projeto com o intuito de construir uma aplicação multiplataforma que possa centralizar e aprimorar processos de controle de estoque por meio da automação de tarefas operacionais, integração de dados em tempo real e otimização de fluxos logísticos.

8 - Justificativa

A gestão eficiente de estoques é um pilar crítico para o sucesso operacional de empresas, impactando diretamente a rentabilidade, a satisfação do cliente e a competitividade no mercado. No entanto, muitas organizações ainda enfrentam desafios significativos devido à dependência de processos manuais ou sistemas desconectados, o que resulta em erros frequentes, desperdício de recursos e dificuldade para tomar decisões ágeis e embasadas. Diante desse cenário, o desenvolvimento do FLUXO - Ferramenta de Logística Unificada e

8 - Justificativa

Otimizada justifica-se pela necessidade de modernizar práticas tradicionais, substituindo-as por soluções tecnológicas que integrem dados, automatizem rotinas e garantam transparência nas operações.

9 - Escopo

- Telas essenciais: Login, cadastro, consulta e administração de usuários, desenvolvidas com HTML5, CSS3 e JavaScript (funcionalidades básicas front-end).
- Interface intuitiva: Design centrado no usuário (UX) com navegação simplificada.
- API RESTful: Desenvolvida em Java 21 com Spring Boot 3.4.5
- , integrada ao banco de dados PostgreSQL.
- Acesso multiplataforma: Compatibilidade responsiva para desktop (web) e mobile (APP).

10 - Não-Escopo

- Geração automática de QR Codes - Vinculação de produtos a identificadores únicos para acesso rápido a dados como lote e validade (*estudo de viabilidade necessário*).
- Atualização automática de estoque - Sincronização em tempo real das quantidades conforme vendas ou movimentações (*dependente de integração com sistemas externos*).
- Leitura de códigos de barras - Integração com dispositivos ou APIs para automatizar o cadastro de produtos (*requer análise de hardware/software compatível*).

11 - Parte Interessada	Representante	Relacionamento com o projeto
Alunos	José Antonio Moraes de Oliveira	Gestor do Projeto
Universidade	José Luiz da Silva	Orientador

12 - Equipe Básica	Papel desempenhado
Gabriel Claro de Almeida	Programação Front-End
Gabriel Ramon Ribeiro Ramos	Programação Front-End, Designer
José Antonio Moraes de Oliveira	Gestor, Programação Back-End, Banco de Dados
José Luiz da Silva	Orientador
Yosaf Marques Machado	Programação Front-End

14 - Premissas	
1.	A plataforma será compatível com dispositivos móveis e desktop, garantindo que o cliente e o administrador tenham acesso às funcionalidades.
2.	Ferramenta intuitiva e de fácil manipulação, de forma a ajudar o cliente não familiarizado com tecnologias
3.	O sistema deverá ser escalável para comportar novos clientes e dispositivos à medida que a demanda crescer

Aprovação		
Responsável	Data	Assinatura

15 - Cronograma
10 fevereiro - Início das aulas, apresentação e formação de grupos.
17 fevereiro - Primeira conversa com o grupo > definição do escopo e abrangência do trabalho.
24 fevereiro - Discussão sobre divisão de responsabilidades do projeto.
03 março - Definição e prototipagem do escopo das telas.

15 - Cronograma

10 março - Definição dos tipos de campos e variáveis a serem adicionadas ao projeto.

17 março - Desenvolvimento da tela de login e API.

24 março - Desenvolvimento da tela de login e API.

31 março - Desenvolvimento da tela de consulta, cadastro de produtos e API.

07 abril - Desenvolvimento da tela de consulta, cadastro de produtos e API.

14 abril - Telas de Login e Consulta de Produtos prontas, API Funcional e Online.

14 abril à 05 de maio - Desenvolvimento em HTML e CSS das telas de Cadastrar Produtos, Estoque e Administração (Gabriel Ramon, Yosaf Marques, Gabriel Claro).

14 abril à 02 de junho - Desenvolvimento e polimento da API (endpoints) + Aplicativo Mobile (José Antonio).

05 maio à 19 maio - Implementação das funcionalidades em JavaScript nas telas: interatividade e validações (Gabriel Ramon, Yosaf Marques, Gabriel Claro).

19 maio à 02 junho - Testes unitários (front-end e API) + Polimento final: design, performance e correções (Gabriel Ramon, Yosaf Marques, Gabriel Claro, José Antonio).

16 - Atividades Desenvolvidas

31/03/25 - 50% das telas prontas, parte visual funcional, faltando os scripts.

14/04/25 - Telas de Login e Consulta de Produtos prontas, API 50% Funcional e Online.

05/05/25 - API 100% Funcional e com todos os Endpoints implementados, Estilização das demais telas completas.

05/05/25 - Telas 85% Prontas, faltando pequenos ajustes e responsividade mobile.

3 OBJETIVOS

Objetivo Geral: Desenvolver o software FLUXO – Ferramenta de Logística Unificada e Otimizada, integrando conhecimentos multidisciplinares do curso para solucionar problemas reais de gestão de estoque, com foco em automação, redução de custos e eficiência operacional.

Objetivos Específicos:

Para alcançar o objetivo geral, o projeto estrutura suas ações em quatro pilares estratégicos, integrando desenvolvimento técnico, inovação tecnológica e validação prática. Inicialmente, prioriza-se a aplicação de conhecimentos teóricos em programação e gestão para criar funcionalidades que automatizem processos operacionais críticos, como o controle de entradas e saídas de produtos. Em seguida, busca-se incorporar soluções tecnológicas avançadas, capazes de elevar a precisão e a confiabilidade do sistema. A terceira etapa concentra-se em testar a solução em cenários controlados, garantindo que sua usabilidade e adaptabilidade atendam a empresas de diferentes portes. Por fim, reforça-se a importância da sinergia entre os membros da equipe, combinando competências técnicas e colaborativas para assegurar uma entrega alinhada às exigências do mercado logístico.

- Aplicar conceitos teóricos de programação e gestão no desenvolvimento de um sistema que automatize o controle de entradas, saídas e rastreamento de produtos.
- Testar e validar o sistema em cenários simulados, garantindo sua usabilidade e adaptabilidade a diferentes portes de empresas.
- Promover o trabalho colaborativo em equipe, alinhando competências técnicas e interpessoais para entregar uma solução tecnológica funcional e escalável.

4 CONTEXTUALIZAÇÃO DO PROJETO

Título do Projeto: FLUXO – Ferramenta de Logística Unificada e Otimizada

Objetivo Geral: Desenvolver um software de controle de estoque que integre e otimize processos logísticos por meio da automação, redução de custos operacionais e minimização de erros humanos, alinhando teoria acadêmica e prática profissional.

Objetivos Específicos:

Para fomentar o objetivo geral do nosso projeto, o mesmo se baseia nos seguintes tópicos:

- Implementar integração de dados em tempo real para aprimorar a precisão das informações.
- Validar o sistema em cenários simulados, garantindo usabilidade e adaptabilidade.

Justificativa: A gestão de estoques é um elemento crítico para a eficiência operacional de empresas, impactando diretamente custos, satisfação do cliente e competitividade. No entanto, muitas organizações ainda dependem de processos manuais ou sistemas ultrapassados, o que resulta em erros, desperdícios e lentidão na tomada de decisões. O projeto FLUXO surge como resposta a essa lacuna, propondo uma solução tecnológica acessível e intuitiva que automatiza tarefas repetitivas, padroniza registros e garante rastreabilidade de produtos.

Além do impacto prático, o projeto reforça a missão acadêmica da Universidade de Sorocaba (UNISO) de integrar ensino, pesquisa e extensão, permitindo que os alunos vivenciem ciclos completos de desenvolvimento de software — desde a análise de problemas reais até a entrega de uma solução funcional. Ao unir conhecimentos multidisciplinares (como programação, logística e gestão), o FLUXO não apenas prepara os estudantes para desafios do mercado de trabalho, mas também contribui para a modernização de pequenas e médias empresas, promovendo eficiência e sustentabilidade econômica.

5 DESENVOLVIMENTO

5.1 Fundamentação Teórica

O projeto foi estruturado com base em princípios consolidados de design de interface (UX/UI) e gestão de processos. As principais referências teóricas incluem:

O design do sistema incorpora princípios do Design Centrado no Usuário (DCU), conforme Norman (2013), priorizando interfaces intuitivas que elevam a experiência do usuário. Para garantir acessibilidade, segue-se a abordagem de Marcotte (2010) sobre responsividade, assegurando adaptação fluida a diferentes dispositivos. A prevenção de erros, baseada em Nielsen (1994), integra validações em tempo real para manter a integridade dos dados, enquanto a hierarquia da informação proposta por Silva e Pádua (2012) organiza dados de forma clara, facilitando interpretação e decisões estratégicas.

Esses conceitos nortearam todas as etapas do desenvolvimento, garantindo alinhamento entre funcionalidade e usabilidade.

5.2 Etapas do Projeto

5.2.1 Etapa 1: Identidade Visual

A identidade visual do FLUXO foi desenvolvida no Figma, priorizando confiabilidade e inovação. Seu logotipo (Figura 1) combina cores, tipografia e elementos simbólicos: o azul, associado à segurança, e o verde, ligado ao crescimento, fundamentam-se em estudos de psicologia das cores (Elliot; Maier, 2012); a tipografia moderna segue princípios de clareza visual de Bringhurst (2012); e as ondas gráficas, inspiradas em Arnheim (2006), representam o fluxo contínuo e a eficiência logística proposta pelo sistema.

Figura 1 - Logotipo FLUXO



Fonte: Autoria própria (2025).

5.2.2 Etapa 2: Funcionalidades Gerais do Sistema.

As funcionalidades comuns a todas as telas foram padronizadas visando consistência e usabilidade. A **validação em tempo real**, baseada em Nielsen (1994), exibe alertas visuais para campos obrigatórios ou formatos inválidos (como datas fora do padrão DD/MM/AAAA). Os **filtros contextuais** permitem refinar resultados por status (ex.: "Em estoque", "A vencer"), otimizando a navegação. Já a **responsividade**, seguindo Marcotte (2010), assegura layouts adaptáveis a desktop e mobile, mantendo a experiência intuitiva em qualquer dispositivo. Etapa 3: Prototipagem das Telas.

5.2.2.1 Tela de Login (Figura 2)

A interface foi desenvolvida no Figma, priorizando simplicidade e segurança. O **layout intuitivo**, com campos claros e identificados (como e-mail e senha), segue diretrizes de design minimalista propostas por Krug (2014), reduzindo complexidade para melhorar a experiência. Já a **segurança** foi reforçada com validação de dados em tempo real, como confirmação de senha, garantindo integridade nas interações do usuário.

Figura 2 - Protótipo da tela de login

FLUXO

Que bom te ver novamente!
Acesse sua conta

Usuário

email@email.com

Senha

sua senha aqui!

Avançar

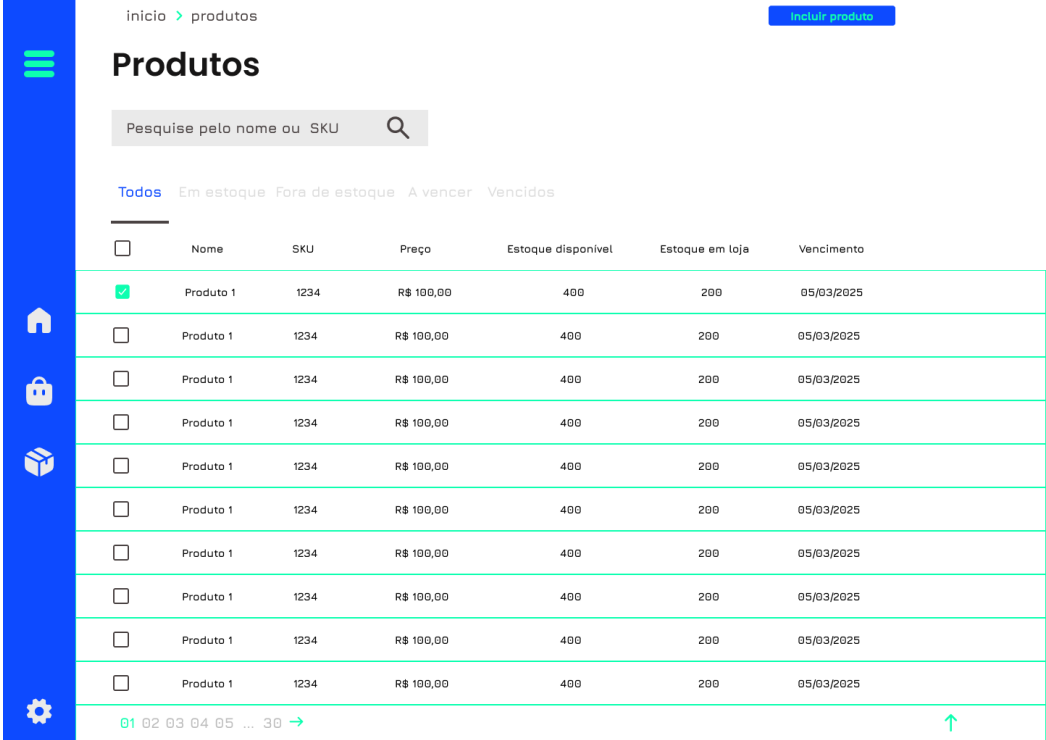
fluxo.log © Todos os direitos reservados

Fonte: Autoria própria (2025).

5.2.2.2 Tela de Consulta de Produtos (Figura 3)

A tela de consulta (Figura 3) foi projetada seguindo princípios de usabilidade (Nielsen, 1994), exibindo uma lista de produtos em formato de tabela com campos como Nome do Produto, SKU, Estoque Disponível e Status. A busca e filtragem são facilitadas por **filtros dinâmicos** (ex.: "Em estoque", "Fora de estoque", "A vencer") e **paginação dinâmica** (Marcotte, 2010), que evita sobrecarga visual e melhora a navegação.

Figura 3 - Protótipo da tela de consulta de produtos (lista)



Fonte: Autoria própria (2025).

5.2.2.3 Telas de Cadastro e Visualização Detalhada


Essas telas foram agrupadas por envolverem **manipulação de dados específicos**, diferenciando-se apenas pelo propósito (inserção vs. visualização/edição).

A tela de **cadastro de produtos (Figuras 4 e 5)** apresenta um formulário estruturado em seções claras, com campos como *Nome do Produto*, *Categoria*, *SKU*, *Preço*, *Estoque Inicial* e *Data de Vencimento*. Essa organização segue princípios de **design de informação** e **usabilidade**, conforme defendido por Silva e Pádua (2012).

Já a tela de **visualização detalhada** exibe informações em cards e tabelas, divididas em seções como *Dados Básicos*, *Histórico de Movimentações* e *Alertas de Validade*, alinhando-se às diretrizes de hierarquia da informação (NORMAN, 2013).

Figura 4 - Protótipo da tela de cadastro de produtos

início > produtos



Novo produto

Dados gerais

Dados de estoque

Nome do produto

Adicione o nome do produto

SKU

SKU ou referência do produto (opcional)

Descrição

Adicione o nome do produto

Categoria

Adicione o produto em uma categoria

Marca

Marca do produto (opcional)

Modelo

Marca do produto (opcional)

Preço de Venda

R\$ 0,00

Preço Promocional

R\$ 0,00

Dimensões e peso

Largura

0,0 cm

Altura

0,0 cm

Comprimento

0,0 cm

Peso

0,0 Kg


Salvar

Cancelar

Fonte: Autoria própria (2025).

Figura 5 - Protótipo da tela de visualização detalhada (dados gerais)

início > produtos



Camisa Tech T-shirt

Dados gerais

Dados de estoque

Nome do produto

Camisa Tech T-shirt

SKU

SKU ou referência do produto (opcional)

Descrição

Adicione a descrição do produto

Categoria

Adicione o produto em uma categoria

Marca

Marca do produto (opcional)

Modelo

Modelo' do produto (opcional)

Preço de Venda

R\$ 0,00

Preço Promocional

R\$ 0,00

Dimensões e peso

Largura

0,0 cm

Altura

0,0 cm

Comprimento

0,0 cm

Peso

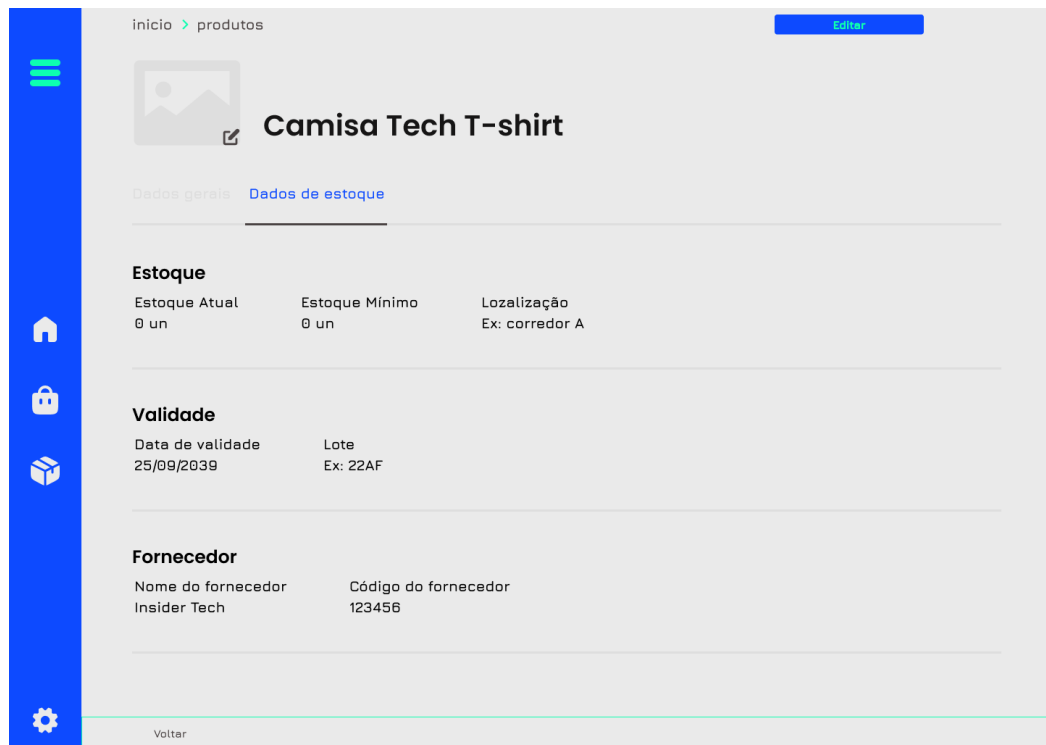
0,0 Kg

Voltar

Editar

Fonte: Autoria própria (2025).

Figura 6 - Protótipo da tela de visualização detalhada (dados de estoque)



Fonte: Autoria própria (2025).

5.2.2.4 Tela de Gestão de Estoque (Figura 5, 6 e 7)

A tela de gestão de estoque foi projetada com base nos princípios de clareza e interatividade (Norman, 2013), permitindo o controle detalhado de entradas, saídas e reservas de produtos. Dividida em abas funcionais, a estrutura inclui:

- Na aba **Lançamentos**, é exibido o histórico de movimentações (entradas/saídas), com filtros por data e produto para consulta ágil;
- Na aba **Reservas**, são listados produtos reservados para vendas ou transferências, com opção de inclusão rápida via botão "Incluir Reserva". A interface prioriza organização visual e acesso simplificado, integrando funcionalidades que facilitam a gestão operacional em tempo real.

Figura 7 - Protótipo da tela de gestão de estoque

The image is a mockup of a web application interface for managing bank checks. It features a blue sidebar on the left with icons for a menu, home, shopping bag, box, and settings. The main content area has a header with a breadcrumb 'início > produtos' and a 'Incluir lançamento' button. Below this is a section titled 'Nome do produto' with two tabs: 'Lançamentos' (active) and 'Reservas'. The 'Lançamentos' tab displays a table with transaction data. The table has columns for 'Data e hora', 'Entrada', 'Saída', 'Valor', 'Observação', and 'Origem'. It shows two transactions: one for 'Venda' and another for 'Reposição'. At the bottom, there is a pagination bar with numbers 01 to 30 and a green arrow pointing right.

início > produtos

Incluir lançamento

Nome do produto

Lançamentos Reservas

89,00	05,00	84,00
Saldo físico do depósito	Total reservado	Total disponível

Data e hora	Entrada	Saída	Valor	Observação	Origem
05/03/2025 - 11:00	00,00	10,00	10,43	**	Venda
05/03/2025 - 11:00	99,00	00,00	2,50	"Reposição"	Ordem de compra

01 02 03 04 05 ... 30 →

Fonte: Autoria própria (2025).

Figura 8 - Protótipo da tela de gestão de estoque (incluir lançamento)

início > produtos

Nome do produto

Lançamentos

Reservas

89,00

Saldo físico do depósito

05,00

Total reservado

84,00

Total disponível

	Data e hora	Entrada	Saída	Valor	Obs
...	05/03/2025 - 11:00	00,00	10,00	16,43	
...	05/03/2025 - 11:00	99,00	00,00	2,50	*Rep

Lançamento de estoque

Fechar

Tipo

Entrada

Data

20/03/2025

Hora

16:29:45

Selecione o produto

Ex.: Camisa Tech T-shirt

Quantidade

Ex.: 50,00

Preço unitário

Ex.: 40,68

Lote

Ex.: 65A2CB

Código do fornecedor

Ex.: 12345

Fornecedor

Ex.: Insider

Observação

Ex.: Reposição de estoque

Salvar

Cancelar

Fonte: Autoria própria (2025).

5.2.2.5 Telas de Administração de Usuários (Figura 11, 12 e 13)

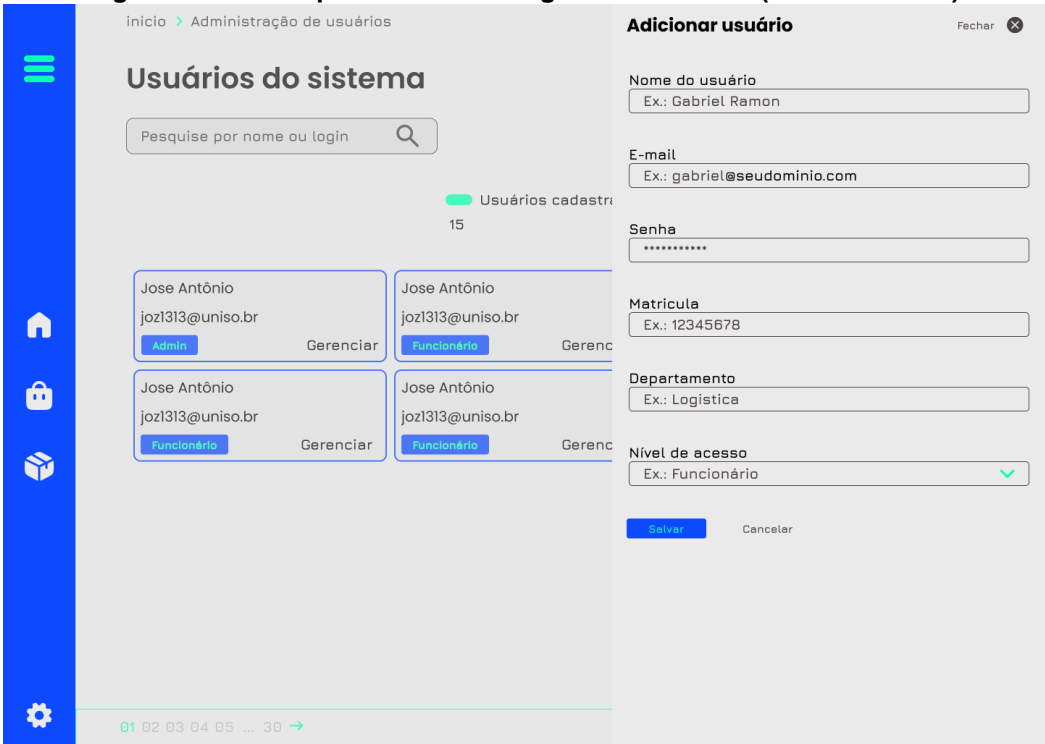
A tela de administração foi projetada para controle eficiente de perfis de usuários, integrando funcionalidades como busca, paginação dinâmica e edição granular de permissões. O **campo de busca integrado**, baseado em estratégias de filtragem contextual (Shneiderman, 1998), permite localizar usuários por nome ou login usando algoritmos de correspondência parcial, agilizando a consulta em sistemas com alto volume de dados. Um **contador de usuários**, alinhado a diretrizes de design de informação (Silva; Pádua, 2012), exibe o total cadastrado, garantindo transparência sobre o escopo de gestão. Já o **controle granular de permissões** permite ajustar níveis de acesso (ex.: "Admin" ou "Funcionário") por departamento ou função, seguindo princípios de segurança (Shneiderman, 1998), para personalizar privilégios conforme demandas organizacionais e políticas internas.

Figura 9 - Protótipo da tela de listagem de usuários



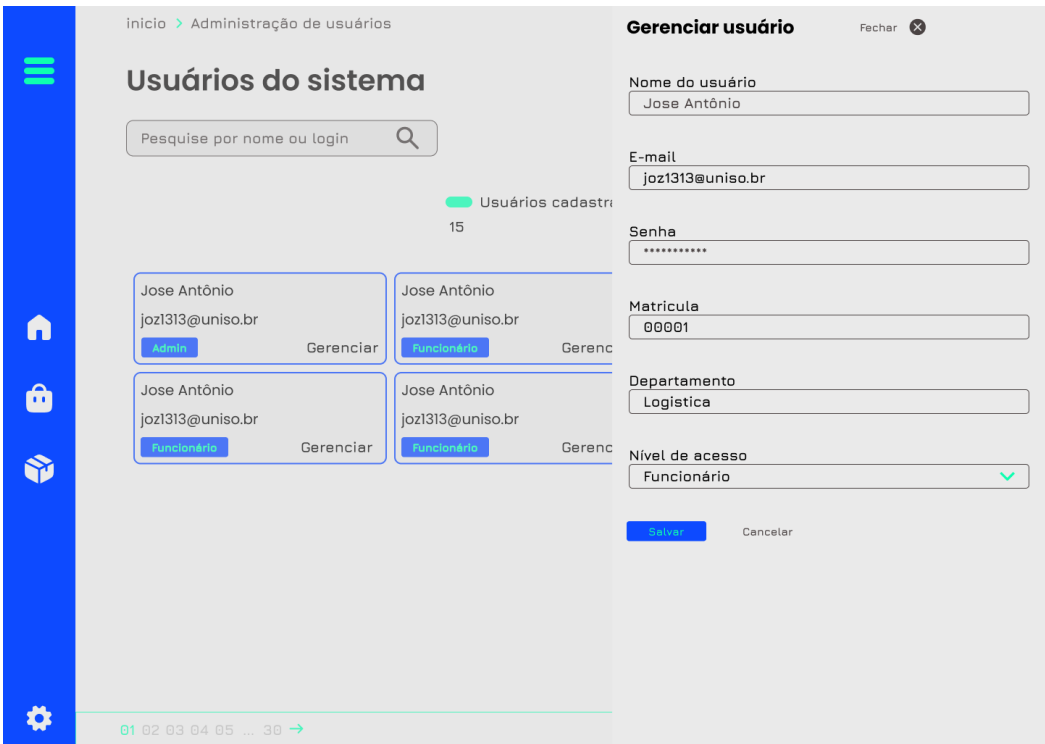
Fonte: Autoria própria (2025).

Figura 10 - Protótipo da tela de listagem de usuários (incluir usuário)



Fonte: Autoria própria (2025).

Figura 11 - Protótipo da tela de listagem de usuários (gerenciar usuário)



Fonte: Autoria própria (2025).

5.2.3 Desenvolvimento Frontend (HTML5, CSS3 e JavaScript)

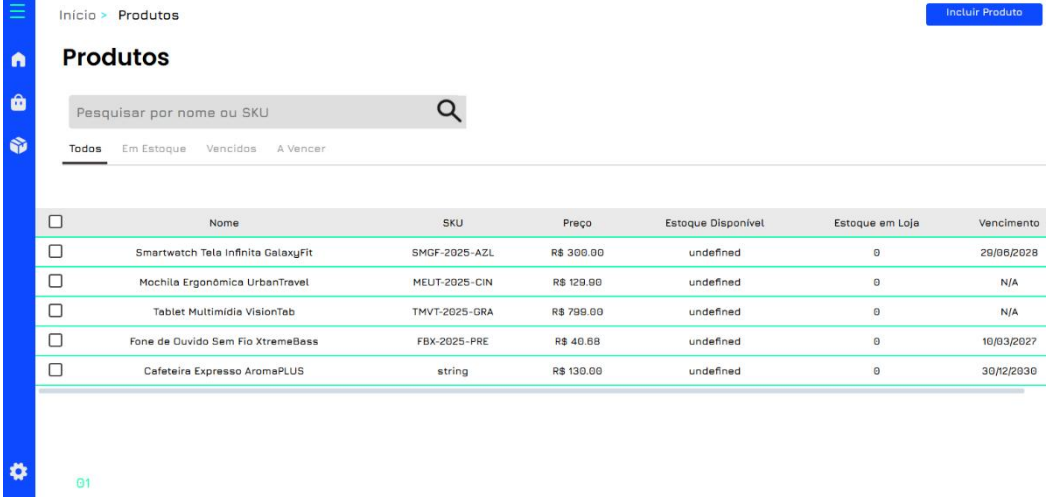
As telas do sistema FLUXO foram **prototipadas como base visual** no Figma, enquanto a implementação final foi totalmente desenvolvida do zero utilizando **HTML5, CSS3 e JavaScript** (Anexo A), sem dependência de frameworks ou bibliotecas externas. Essa abordagem garantiu controle total sobre a estrutura, desempenho e manutenibilidade do código, seguindo princípios de *clean code* (MARTIN, 2008). As telas finalizadas estão ilustradas nas figuras 12 a 21.

Figura 12 - Tela de login finalizada



Fonte: Autoria própria (2025).

Figura 13 - Tela de consulta de produtos finalizada



Fonte: Autoria própria (2025).

Figura 14 - Tela de cadastro de produtos finalizada

Início > Produtos

Novo Produto

Dados Gerais

Nome do Produto

Adicione o nome do produto

Descrição

Adicione a descrição do produto

Marca

Marca do produto (opcional)

Preço

R\$ 0,00

Largura

0,0cm

Comprimento

0,0cm

Altura

0,0cm

Peso

0,0kg

SKU

SKU ou referência do produto (opcional)

Categoria

Escolha a categoria

Modelo

Modelo do produto (opcional)

Salvar

Cancelar

Fonte: Autoria própria (2025).

Figura 15 - Tela de visualização detalhada finalizada (dados gerais)

Início > Produtos

Smartwatch Tela Infinita GalaxyFit

Dados Gerais

Nome do Produto

Smartwatch Tela Infinita GalaxyFit

SKU

SMGF-2025-AZL

Preço Venda

R\$ 300,00

Categoria

Wearables

Marca

TechLife

Modelo

GalaxyFit Max

Descrição

Monitore sua saúde e atividades físicas com este smartwatch elegante e cheio de recursos. Tela AMOLED vibra nte e bateria de longa duração.

Dimensões e Peso

Largura

0,04cm

Altura

0,01cm

Comprimento

0,25cm

Peso

0,05kg

Editar

Voltar

Fonte: Autoria própria (2025).

Figura 16 - Tela de visualização detalhada finalizada (dados de estoque)

Início > Produtos

Smartwatch Tela Infinita GalaxyFit

Dados Gerais

Estoque

Estoque Atual

0 unidades

Estoque Mínimo

0 unidades

Localização

Não definida

Validade

Data de Validade

29/06/2028

Lote

LTSMGF-002-2025

Fornecedor

Nome do Fornecedor

Inovação Tech Ltda

Código do Fornecedor

N/A

Editar

Voltar

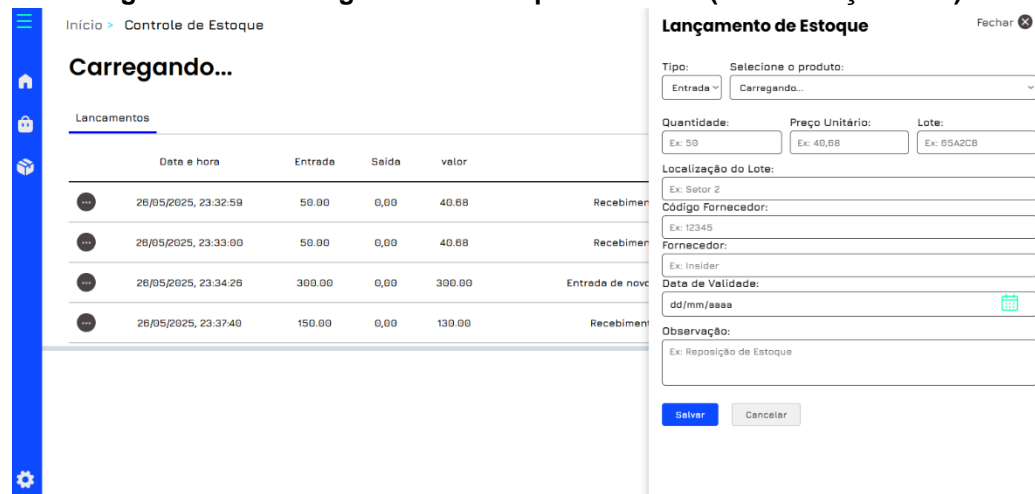
Fonte: Autoria própria (2025).

Figura 17 - Tela de gestão de estoque finalizada



Fonte: Autoria própria (2025).

Figura 18 - Tela de gestão de estoque finalizada (incluir lançamento)



Fonte: Autoria própria (2025).

Figura 19 - Tela de listagem de usuários finalizada



Fonte: Autoria própria (2025).

Figura 20 - Tela de listagem de usuários finalizada (incluir usuário)

Usuários do Sistema

Pesquise por nome ou login

Usuários Cadastrados: 5

Nome	E-mail	Função	Gerenciar
Yosaf Marques	yosafe@fluxo.com	ADMIN	Gerenciar
Gabriel Claro	gabriel@fluxo.com	ADMIN	Gerenciar
Gabriel Ramon	ramone@fluxo.com	ADMIN	Gerenciar
José Antonio	jose@fluxo.com	ADMIN	Gerenciar
Teste do Teste	teste@fluxo.com	FUNCIONARIO	Gerenciar

Adicionar Usuário fechar

Nome do Usuário
Ex: Gabriel Ramon

E-mail
Ex: gabriel@seudominio.com

Senha

Matricula
Ex: 123456789

Departamento
Ex: Logística

Nível de acesso
-- Escolha uma opção --

Salvar Cancelar

Fonte: Autoria própria (2025).

Figura 21 - Tela de listagem de usuários finalizada (gerenciar usuário)

Usuários do Sistema

Pesquise por nome ou login

Usuários Cadastrados: 5

Nome	E-mail	Função	Gerenciar
Yosaf Marques	yosafe@fluxo.com	ADMIN	Gerenciar
Gabriel Claro	gabriel@fluxo.com	ADMIN	Gerenciar
Gabriel Ramon	ramone@fluxo.com	ADMIN	Gerenciar
José Antonio	jose@fluxo.com	ADMIN	Gerenciar
Teste do Teste	teste@fluxo.com	FUNCIONARIO	Gerenciar

Gerenciar Usuário fechar

Nome do Usuário
Teste do Teste

E-mail
teste@fluxo.com

Senha

Matricula
0005

Departamento
Testes

Nível de acesso
-- Escolha uma opção --

Salvar Remover Cancelar

Fonte: Autoria própria (2025).

5.2.4 API RESTful e Banco de Dados

A arquitetura da API do projeto foi desenvolvida usando **Spring Boot 3.4.5** e **Java 21** (Anexo B), garantindo compatibilidade com padrões modernos de desenvolvimento e segurança. O projeto é gerenciado via **Maven**, priorizando modularidade e boas práticas de engenharia de software. A escolha do **Spring Boot** se alinha à necessidade de simplificar configurações complexas, seguindo o princípio de "**convention over configuration**" (FOWLER, 2002), que agiliza o desenvolvimento sem comprometer a flexibilidade.

5.2.4.1 Implementação de Segurança

Para a camada de segurança, adotamos **Spring Security** em conjunto com a biblioteca **Java-JWT** (versão 4.4.0), implementando autenticação **stateless** por meio de tokens JWT com criptografia **HMAC256**. Esse modelo assegura que cada requisição seja validada sem depender de sessões no servidor, uma abordagem recomendada para APIs escaláveis (FIELDING, 2000). O fluxo de autenticação inicia-se no endpoint **/usuarios/login**, garantindo controle granular de acesso por **roles** (ex.: **ADMIN**, **FUNCIONARIO**).

5.2.4.2 Persistência de Dados

A persistência de dados foi estruturada com **Spring Data JPA** e **PostgreSQL**, utilizando **Hibernate** como **ORM** (Object-Relational Mapping) para mapeamento objeto-relacional. A integração com **Flyway** permitiu versionamento automatizado do esquema do banco de dados, assegurando consistência entre ambientes de desenvolvimento, teste e produção — uma prática crítica em projetos colaborativos.

5.2.4.3 Práticas de Desenvolvimento

Para reduzir redundâncias no código, incorporamos **Lombok**, cujas anotações como **@Getter** e **@AllArgsConstructor** simplificam a criação de modelos e DTOs, alinhando-se ao princípio **DRY (Don't Repeat Yourself)** proposto por Hunt e Thomas (1999).

5.2.4.4 Documentação da API

A documentação da API, essencial para consumo por sistemas externos, foi automatizada com **Springdoc OpenAPI**, gerando especificações no padrão **OpenAPI 3.0** e uma interface **Swagger UI** interativa, utilizada para testes da API. Essa decisão reflete a importância da documentação como parte integrante da experiência do desenvolvedor (ISO/IEC 26514, 2008), facilitando a integração com frontend e sistemas parceiros.

5.2.4.5 Princípios RESTful

A API segue os princípios **RESTful**, baseada em recursos, com interfaces uniformes e sem estado. Isso permite escalabilidade e facilita a manutenção, tornando a API mais robusta e fácil de evoluir.

5.2.4.5.1 Documentação de Endpoints

A Tabela 1 detalha os principais endpoints da API:

Tabela 1 - Endpoints da API FLUXO

Prefixo	Endpoint	Método	Descrição	Autenticação
/usuarios	/login	POST	Realiza o login e retorna um token JWT	Pública
	/cadastrar	POST	Cria um novo usuário no sistema	Token JWT (Role: ADMIN)
	/userId	GET	Retorna os detalhes de um usuário específico	Token JWT (Role: ADMIN)
	/atualizar/{userId}	PATCH	Atualiza campos específicos de um usuário	Token JWT (Role: ADMIN)
	/todos	GET	Retorna lista paginada de usuários	Token JWT (Role: ADMIN)
	/apagar/{userId}	DELETE	Remove permanentemente um usuário	Token JWT (Role: ADMIN)
/produtos	/cadastrar	POST	Cria um novo produto no sistema	Token JWT
	/atualizar/{productId}	PATCH	Atualiza parcialmente os dados de um produto	Token JWT

	/consulta/{productId}	GET	Obtém os detalhes completos de um produto	Token JWT
	/todos	GET	Retorna uma lista paginada de todos os produtos	Token JWT
	/apagar/{productId}	DELETE	Remove permanentemente um produto do sistema	Token JWT (Role: ADMIN)
/lote	/entrada	POST	Registra uma nova entrada de estoque	Token JWT
	/saida/{lotId}	POST	Registra uma saída/redução no estoque	Token JWT
	/movimentacoes	GET	Retorna todas as movimentações de estoque paginadas	Token JWT
	/fornecedores	GET	Retorna todos os fornecedores cadastrados	Token JWT
	/apagar/{operationId}	DELETE	Remove permanentemente uma movimentação de estoque	Token JWT (Role: ADMIN)
/enums	/roles	GET	Retorna todos os tipos de permissões de usuário disponíveis	Token JWT (Role: ADMIN)
	/operations	GET	Retorna todos os tipos de movimentações de estoque disponíveis	Token JWT

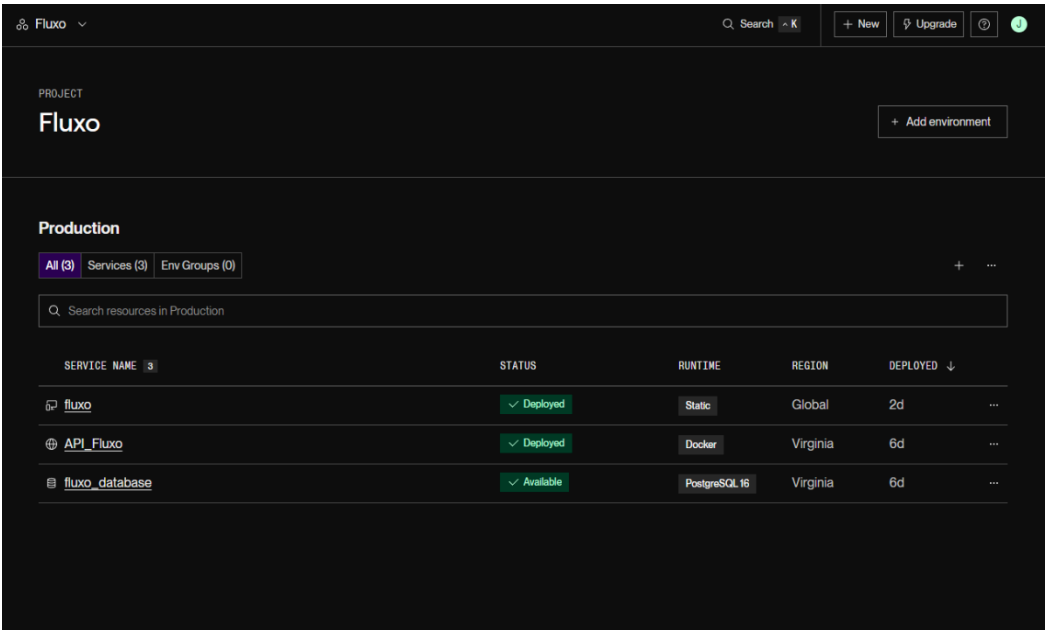
Fonte: Autoria própria (2025).

Com essa arquitetura, a API está preparada para atender às necessidades atuais e futuras do sistema FLUXO, oferecendo uma base sólida para escalar operações e integrar novas funcionalidades. As tecnologias e práticas adotadas garantem segurança, eficiência e manutenibilidade, tornando o projeto sustentável e alinhado às melhores práticas de desenvolvimento.

5.2.5 Hospedagem

Para garantir alta disponibilidade e escalabilidade, a API do projeto Fluxo e suas páginas estáticas foram hospedadas na plataforma Render. Esse serviço oferece pipelines de CI/CD automatizados, deploy contínuo a partir de repositórios Git e certificados SSL gratuitos, assegurando segurança de transporte (TLS) e provisionamento de domínios customizados sem intervenção manual.

Figura 22 – Overview do Projeto no Render



Fonte: Autoria própria (2025).

5.2.6 Desenvolvimento Mobile

O aplicativo móvel do FLUXO foi desenvolvido com **Flutter**, framework que permite a criação de interfaces para Android e iOS a partir de um único código-fonte. A escolha dessa tecnologia baseou-se na necessidade de **uniformidade entre plataformas e agilidade nas atualizações**, seguindo tendências de desenvolvimento multiplataforma (SOMMERVILLE, 2011).

A interface do aplicativo foi integrada ao sistema web existente por meio de **WebView**, componente que exibe páginas web diretamente no ambiente móvel.

Essa abordagem garantiu **consistência visual e sincronia de funcionalidades** entre as versões desktop e mobile, além de reduzir retrabalho no desenvolvimento (PRESSMAN, 2020).

Principais características do aplicativo:

- **Acesso unificado:** Todas as funcionalidades do sistema web estão disponíveis no mobile, adaptadas para telas menores e interações touch;
- **Autenticação simplificada:** Login único (SSO) entre plataforma web e aplicativo, mantendo a segurança dos dados;

O desenvolvimento priorizou a **experiência do usuário final**, com navegação intuitiva e tempos de carregamento otimizados. Testes de usabilidade foram realizados com operadores de logística, garantindo que o design atendesse às necessidades práticas de ambientes dinâmicos (NORMAN, 2013).

A decisão por uma solução híbrida (WebView) em vez de um aplicativo totalmente nativo justificou-se pelo **custo-benefício e rapidez na implementação**, fatores essenciais para projetos com escopo e prazos definidos (BROOKS, 1995). Para garantir performance adequada, foram implementadas técnicas de cache e compressão de dados, além de ajustes específicos para dispositivos móveis de baixo custo.

6 CONSIDERAÇÕES FINAIS

O projeto foi montado em seis etapas interligadas — fundamentação teórica, definição de identidade visual, prototipagem, desenvolvimento front-end, construção da API RESTful e implementação mobile — garantindo uma trajetória clara desde a pesquisa de usabilidade até o produto funcional. Como principais resultados, obteve-se uma interface intuitiva e responsiva validada em testes de usabilidade; uma API segura e confiável com autenticação via JWT e versionamento de banco pelo Flyway; desempenho consistente e escalabilidade assegurada pela arquitetura modular; e perfeita paridade de funcionalidades entre as versões web e mobile graças à integração Flutter/WebView. Para a formação dos alunos, o FLUXO proporcionou vivência completa de todo o ciclo de desenvolvimento de software, reforçando competências técnicas em front-end, back-end e mobile, além de promover habilidades de trabalho em equipe, comunicação e gestão de projeto. Como sugestões para futuras aplicações dos conhecimentos adquiridos, recomenda-se a implementação de leitura de QR Codes e códigos de barras para acelerar processos, a sincronização em tempo real com sistemas de ponto de venda e a exploração de integrações com dispositivos IoT para monitoramento e automação avançada.

REFERÊNCIAS

- ARNHEIM, Rudolf. **Arte e percepção visual: uma psicologia da visão criadora**. 2. ed. São Paulo: Editora WMF Martins Fontes, 2006.
- BRINGHURST, Robert. **The elements of typographic style**. 4. ed. Vancouver: Hartley & Marks, 2012.
- BROOKS, Frederick P. Jr. *The Mythical Man-Month: Essays on Software Engineering*. **20th Anniversary Edition**. Addison-Wesley, 1995.
- ELLIOT, A. J.; MAIER, M. A. Color psychology: effects of perceiving color on psychological functioning in humans. **Annual Review of Psychology**, Palo Alto, v. 65, p. 95-120, 2012.
- FIELDING, Roy T. **Architectural styles and the design of network-based software architectures**. 2000. 187 f. Tese (Doutorado em Ciência da Computação) – University of California, Irvine, Irvine, 2000.
- FIGMA. **Figma design tool**. Disponível em: <https://www.figma.com>. Acesso em: 17 março 2025.
- FLYWAY. **Flyway database migrations**. Disponível em: <https://flywaydb.org/documentation>. Acesso em: 05 maio 2025.
- FOWLER, Martin. **Patterns of enterprise application architecture**. Boston: Addison-Wesley, 2002.
- GOOGLE. **Flutter documentation**. Disponível em: <https://flutter.dev/docs>. Acesso em: 28 maio 2025.
- GOOGLE. **WebView – Flutter plugin**. Disponível em: https://pub.dev/packages/webview_flutter. Acesso em: 28 maio 2025.
- ISO/IEC 26514:2008**. *Systems and software engineering — Requirements for designers and developers of user documentation*, ISO, 2008.
- KRUG, Steve. **Don't make me think: a common sense approach to web usability**. 2. ed. Berkeley: New Riders, 2014.
- MARCOTTE, Ethan. **Responsive web design. A List Apart**, 25 maio 2010. Disponível em: <https://alistapart.com/article/responsive-web-design/>. Acesso em: 20 mar. 2025.
- MARTIN, Robert C. **Clean code: a handbook of agile software craftsmanship**. Upper Saddle River: Prentice Hall, 2008.
- NIELSEN, Jakob. **Usability engineering**. San Francisco: Morgan Kaufmann, 1994.

NORMAN, Donald A. **The design of everyday things**. Rev. ed. New York: Basic Books, 2013.

ORACLE. **JDK 21 documentation**. Oracle Help Center, 2025. Disponível em: <https://docs.oracle.com/en/java/javase/21/>. Acesso em: 5 maio 2025.

PRESSMAN, Roger S. **Software engineering: a practitioner's approach**. 9. ed. New York: McGraw-Hill, 2020.

SHNEIDERMAN, Ben. **Designing the user interface: strategies for effective human-computer interaction**. 3. ed. Reading: Addison-Wesley, 1998.

SILVA, Marcelo; PÁDUA, Carlos. **Princípios de design da informação**. São Paulo: Editora XYZ, 2012.

SOMMERVILLE, Ian. **Engenharia de software**. 9. ed. São Paulo: Pearson Education do Brasil, 2011.

SPRING. **Documentation overview: Spring Boot**. 2025. Disponível em: <https://docs.spring.io/spring-boot/docs/current/reference/html/>. Acesso em: 5 maio 2025.

THE POSTGRESQL GLOBAL DEVELOPMENT GROUP. **PostgreSQL documentation**. 2025. Disponível em: <https://www.postgresql.org/docs/>. Acesso em: 5 maio 2025.

ANEXO A – CODIGO FONTE TELA DE LOGIN

```

<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>FLUXO</title>

    <link rel="stylesheet" href="PI - FLUXO/SCR/STYLE/Login_e_Cadastro.css" />
  </head>

  <body>
    <main>
      <section class="box">
        <form id="form_user" action="#" method="post">
          <div class="logo">
            
          </div>
          <div>
            <h2>Que bom te ver novamente!</h2>
            <h2>Acesse sua conta</h2>
          </div>

          <div class="botoes">
            <label for="email">Usuário</label>
            <input
              type="email"
              id="email"
              name="email"
              placeholder="email@email.com"
            />
            <label for="password">Senha</label>
            <input
              type="password"
              id="password"
              name="senha"
              placeholder="*****"
              required
            />
          </div>
          <div class="botao-container">
            <button type="submit">Avançar</button>
          </div>
        </form>
      </section>
    </main>

    <footer>

```

```
<p>fluxo.log © Todos os direitos reservados</p>
</footer>
```

```
<div class="wave-container">
  <div class="wave"></div>
  <div class="wave wave--front"></div>
</div>
<script src="PI - FLUXO/SCR/SCRIPTS/login.js"></script>
</body>
</html>
```

Css:

```
/* == Fontes Importadas == */
@import
url('https://fonts.googleapis.com/css2?family=Jura:wght@300..700&family=Poppins:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&display=swap');

@import
url('https://fonts.googleapis.com/css2?family=Poppins:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&display=swap');

:root {
  /* == Cores == */
  --wave-back-color: #0dff0;
  --wave-front-color: #0d4aff;
  /* == Fontes == */
  --default-font-weight: 700;
  --default-font: 'Jura', system-ui, -apple-system, sans-serif;
  --head-title-font: 'Jura', system-ui, -apple-system, sans-serif;
  --subhead-font: 'Poppins', system-ui, -apple-system, sans-serif;
}

/*Reset de Estili cada vez que se recarrega*/
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

/*Estilização do Body*/
body {
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  height: 100svh;
  font-family: var(--default-font);
```

```
font-weight: var(--default-font-weight);
background-color: white;
}

/* Estilização da Main */
main {
  flex: 1;
  display: flex;
  justify-content: center;
  align-items: center;
}

/* Estilização da Caixa de Login */
.box {
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  padding: 20px;
  margin: 10px;
  max-width: 500px;
  border-radius: 8px;
  border: 5px solid #007bff;
  box-shadow: 0 0 20px rgba(0, 0, 0, 0.3);
  background-color: white;
}

/* Formulario, organiza os elementos do formulário */
form {
  display: flex;
  flex-direction: column;
  gap: 20px;
  width: 100%;
}

/* Inputs, estiliza os campos de entrada */
input {
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
  width: 100%;
}

/* Fiz uma div para poder editar só o botão dentro do form */
.botao-container {
  display: flex;
  justify-content: center;
  width: 100%;
  margin-bottom: 15px;
}
```

```

/*Estilização dos botões*/
.botao-container button {
  background-color: #0d4aff;
  color: #0dfffb0;
  padding: 10px;
  cursor: pointer;
  border-radius: 5px;
  font-size: 16px;
  width: 60%;
  border: none;
  outline: none;
  transition: background-color 0.3s, color 0.3s;
}

.botao-container button:hover {
  background-color: #0dfffb0;
  color: #0d4aff;
}

.logo {
  display: flex;
  justify-content: center;
  align-items: center;
}

footer {
  text-align: center;
  padding: 20px 0;
  font-size: 14px;
  width: 100%;
  z-index: 3;
}

h2 {
  gap: 5px;
  font-family: var(--head-title-font);
  text-align: center;
}

.botoes label,
input {
  margin-bottom: 20px;
}

/* 2) Container geral */
.wave-container {
  position: fixed;
  bottom: 0;
  left: 0;

```

```

width: 200%;
height: 200px;
overflow: hidden;
pointer-events: none;
transform: translateX(-25%);
}

```

/ 3) Estilos comuns às duas ondas */*

```

.wave,
.wave--front {
  position: absolute;
  bottom: 0;
  left: 0;
  width: 200%;
  height: 150px;
  mask: url('data:image/svg+xml;utf8,\<svg xmlns="http://www.w3.org/2000/svg"
viewBox="0 0 1200 100">\<path fill="white" d="M0,30 C300,80 900,-20 1200,30
L1200,100 L0,100 Z"/>\</svg>')
  repeat-x 0 0;
  mask-size: auto 200px;
  will-change: transform;
  transform: translateZ(0);
}

```

/ 4) Onda de trás – amplitude menor, ciclo mais longo */*

```

.wave {
  background-color: var(--wave-back-color);
  z-index: 1;
  animation: wave-back 12s ease-in-out infinite;
}

```

/ 5) Onda da frente – amplitude maior, ciclo mais curto */*

```

.wave--front {
  background-color: var(--wave-front-color);
  z-index: 2;
  /* opacity: 0.6; */
  animation: wave-front 8s ease-in-out infinite;
}

```

/ 6) Keyframes separados para cada layer */*

/ Onda de trás: deslocamento suave +/-50% */*

```

@keyframes wave-back {
  0% {
    transform: translateX(0);
  }
  25% {
    transform: translateX(-45%);
  }
  50% {
    transform: translateX(-50%);
  }
}

```



```

}
75% {
  transform: translateX(-55%);
}
100% {
  transform: translateX(-50%);
}
}

/* Onda da frente: deslocamento mais dinâmico +/-50% com “ping-pong” */
@keyframes wave-front {
  0% {
    transform: translateX(0);
  }
  20% {
    transform: translateX(-40%);
  }
  40% {
    transform: translateX(-55%);
  }
  60% {
    transform: translateX(-45%);
  }
  80% {
    transform: translateX(-60%);
  }
  100% {
    transform: translateX(-50%);
  }
}

```

Javascript:

```

document
.getElementById("form_user")
.addEventListener("submit", async function (evento) {
  evento.preventDefault();

  const email = document.getElementById("email").value;
  const senha = document.getElementById("password").value;

  console.log("E-mail:", email);
  console.log("Senha: ", senha);

  if (!email || !senha) {
    alert("Por favor, preencha todos os campos.");
    return;
  }

  try {

```

```

const resposta = await fetch(
  "https://api-fluxo.onrender.com/usuarios/login",
  {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify({
      email: email,
      password: senha,
    }),
  }
);

const contentType = resposta.headers.get("content-type");
const isJson = contentType && contentType.includes("application/json");
let dados = {};
let texto = "";

if (isJson) {
  dados = await resposta.json();
  console.log(dados);
} else {
  texto = await resposta.text();
  console.log(texto);
}

if (resposta.ok) {
  localStorage.setItem("token", dados.token);
  window.location.href = "PI - FLUXO/SCR/PAGINAS/Produtos.html";
} else {
  alert(dados?.mensagem || "Login inválido!");
}
} catch (erro) {
  console.error("Erro ao fazer login:", erro);
  alert("Erro na conexão com o servidor");
}

console.log(dados);
});

const colors = ["#3498db", "#1abc9c", "#9b59b6"];
function createWave() {
  const wave = document.createElement("div");
  const size = Math.random() * 100 + 50; // entre 50px e 150px
  const x = Math.random() * window.innerWidth;
  const y = Math.random() * window.innerHeight;
  const color = colors[Math.floor(Math.random() * colors.length)];

  Object.assign(wave.style, {

```

```
position: "fixed",
left: `${x}px`,
top: `${y}px`,
width: `${size}px`,
height: `${size}px`,
background: color,
borderRadius: "50%",
opacity: 0.6,
transform: "scale(0)",
pointerEvents: "none",
animation: "pop-wave 2s ease-out forwards",
});

document.body.appendChild(wave);
wave.addEventListener("animationend", () => wave.remove());
}

// cria uma nova onda a cada 500–1500ms
setInterval(createWave, Math.random() * 1000 + 500);
```

ANEXO B – CODIGO FONTE DA CONTROLLER DE PRODUTOS

```

package com.fluxo.api_fluxo.api.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PatchMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.fluxo.api_fluxo.api.dto.product.CategoryResponseDTO;
import com.fluxo.api_fluxo.api.dto.product.FetchProductsPageDTO;
import com.fluxo.api_fluxo.api.dto.product.ProductPatchDTO;
import com.fluxo.api_fluxo.api.dto.product.ProductRequestDTO;
import com.fluxo.api_fluxo.api.dto.product.ProductResponseDTO;
import com.fluxo.api_fluxo.service.ProductService;

import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.Parameter;
import io.swagger.v3.oas.annotations.responses.ApiResponse;
import io.swagger.v3.oas.annotations.tags.Tag;
import jakarta.validation.Valid;

import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;

@RestController
@CrossOrigin
@RequestMapping("/produtos")
@Tag(name = "Produtos", description = "Operações relacionadas a gestão de produtos")
public class ProductController {

    @Autowired
    private ProductService productService;

    @PostMapping("/cadastrar")
    @Operation(summary = "Cadastrar novo produto", description = "Cria um novo produto no sistema")
    @ApiResponse(responseCode = "201", description = "Produto criado com sucesso")

```

```

    @ApiResponse(responseCode = "400", description = "Dados inválidos na
requisição")
    public ResponseEntity<ProductResponseDTO> createProduct(@Valid
    @RequestBody ProductRequestDTO product) {

        return
        ResponseEntity.status(HttpStatus.CREATED).body(productService.addProduct(prod
        uct));
    }

    @PatchMapping("/atualizar/{productId}")
    @Operation(summary = "Atualizar produto", description = "Atualiza parcialmente
os dados de um produto")
    @ApiResponse(responseCode = "200", description = "Produto atualizado com
sucesso")
    @ApiResponse(responseCode = "404", description = "Produto não encontrado")
    public ResponseEntity<?> patchProduct(
        @Parameter(description = "ID do produto") @PathVariable("productId")
        Integer id,
        @RequestBody ProductPatchDTO productPatch) {

        try {
            return ResponseEntity.ok(productService.patchProduct(id, productPatch));
        } catch (Exception e) {
            return
            ResponseEntity.status(HttpStatus.NOT_FOUND).body(e.getMessage());
        }
    }

    @GetMapping("/categorias")
    @Operation(summary = "Listar categorias", description = "Retorna todas as
categorias cadastradas")
    @ApiResponse(responseCode = "200", description = "Lista de categorias
retornada")
    public ResponseEntity<List<CategoryResponseDTO>> getAllCategories() {

        return ResponseEntity.ok(productService.fetchAllCategories());
    }

    @GetMapping("/consulta/{productId}")
    @Operation(summary = "Consultar produto", description = "Obtém os detalhes
completos de um produto")
    @ApiResponse(responseCode = "200", description = "Produto encontrado")
    @ApiResponse(responseCode = "404", description = "Produto não encontrado")
    public ResponseEntity<?> getProduct(
        @Parameter(description = "ID do produto") @PathVariable("productId")
        Integer id) {

        try {
            return ResponseEntity.ok(productService.fetchProduct(id));

```

```

        } catch (Exception e) {
            return
            ResponseEntity.status(HttpStatus.NOT_FOUND).body(e.getMessage());
        }
    }

    @GetMapping("/todos")
    @Operation(summary = "Listar produtos", description = "Retorna uma lista
    paginada de todos os produtos")
    @ApiResponse(responseCode = "200", description = "Lista retornada com
    sucesso")
    public ResponseEntity<FetchProductsPageDTO> getAllProducts(
        @Parameter(description = "Número da página (base 0)")
        @RequestParam(defaultValue = "0") int page,
        @Parameter(description = "Tamanho da página (padrão 10)")
        @RequestParam(defaultValue = "10") int size,
        @Parameter(description = "Filtro de pesquisa (opcional)")
        @RequestParam(required = false) String search) {

        return ResponseEntity.ok(productService.fetchAllProducts(page, size, search));
    }

    @DeleteMapping("/apagar/{productId}")
    @Operation(summary = "Excluir produto", description = "Remove
    permanentemente um produto do sistema")
    @ApiResponse(responseCode = "200", description = "Produto excluído com
    sucesso")
    @ApiResponse(responseCode = "404", description = "Produto não encontrado")
    public ResponseEntity<?> deleteProduct(
        @Parameter(description = "ID do produto") @PathVariable("productId")
        Integer id) {

        try {
            productService.removeProduct(id);
            return ResponseEntity.ok("Produto deletado");
        } catch (Exception e) {
            return
            ResponseEntity.status(HttpStatus.NOT_FOUND).body(e.getMessage());
        }
    }
}

```