INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE RONDÔNIA - CAMPUS ARIQUEMES

JOÃO LUCAS VIANA PERUZZO

APLICATIVO DE AUDITORIA DE PRODUTOS EM FILIAIS NOVALAR

ARIQUEMES/RO
2025

JOÃO LUCAS VIANA PERUZZO

APLICATIVO DE AUDITORIA DE PRODUTOS EM FILIAIS NOVALAR

Relatório técnico apresentado como requisito para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia de Rondônia, campus Ariquemes.
Orientador: Andrey Alencar Quadros

ARIQUEMES/RO
2025

Ficha catalográfica elaborada pelo Sistema Gerador de Ficha Catalográfica do IFRO.

Peruzzo, João Lucas Viana.
Aplicativo de auditoria de produtos em filiais novalar / João Lucas Viana Peruzzo. - Ariquemes, 2025.
34 f. : il.

Orientador(a): Prof. Me. Andrey Alencar Quadros.

Trabalho de Conclusão de Curso (Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) – Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO, Ariquemes, 2025.

1. *Flutter*. 2. *GetX*. 3. Auditoria de estoque. 4. Aplicativo móvel. 5. Multiplataforma. I. Quadros, Andrey Alencar (orient.). II. Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO. III. Título.

: P471a

Bibliotecário(a) Responsável: Renilce Silva Morais, CRB-11/906

JOÃO LUCAS VIANA PERUZZO

APLICATIVO DE AUDITORIA DE PRODUTOS EM FILIAIS NOVALAR

Trabalho de Conclusão de Curso julgado adequado para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas e aprovado em sua forma final pelo Instituto Federal de Educação, Ciência e Tecnologia de Rondônia – Campus Ariquemes.

Ariquemes, 17 de dezembro de 2025.

Banca Examinadora:

gov.br
Documento assinado digitalmente
ANDREY ALENCAR QUADROS
Data: 17/12/2025 19:06:23-0300
Verifique em https://validar.iti.gov.br

Prof. Mestre Andrey Alencar Quadros
Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO – Campus Ariquemes)

gov.br
Documento assinado digitalmente
LUCIANO TOPOLNIAK
Data: 17/12/2025 18:11:55-0300
Verifique em https://validar.iti.gov.br

Prof. Mestre Luciano Topolniak
Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO – Campus Ariquemes)

gov.br
Documento assinado digitalmente
MARCOS ALVES FAINO
Data: 17/12/2025 18:52:00-0300
Verifique em https://validar.iti.gov.br

Prof. Especialista Marcos Alves Faino
Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO – Campus Ariquemes)

AGRADECIMENTOS

Agradeço primeiramente a Deus, pela saúde, pela força diária e por me sustentar em todos os momentos desta caminhada. Sem Sua presença, nada disso teria sido possível.

À minha família, deixo minha profunda gratidão pelo suporte incondicional, pelo carinho, pela paciência e por acreditarem em mim mesmo quando eu duvidei. Cada palavra de incentivo foi essencial para que eu chegasse até aqui.

Ao meu orientador, Prof. Andrey Alencar Quadros, agradeço pela orientação, disponibilidade, paciência e contribuições técnicas ao longo de todo o desenvolvimento deste trabalho. Suas observações e direcionamentos foram fundamentais para o amadurecimento do projeto e para a consolidação dos resultados apresentados.

Aos meus professores, agradeço pelo aprendizado, pela dedicação e por compartilharem conhecimento com tanta generosidade. Cada orientação contribuiu diretamente para a construção deste trabalho e para minha formação pessoal e profissional.

E, por fim, mas não menos importante, agradeço aos videogames, que foram meu refúgio e meu equilíbrio ao longo de toda a jornada. Nos momentos mais difíceis, foram eles que me ajudaram a manter a sanidade, a aliviar o estresse e a recarregar as energias para seguir em frente.

A todos, meu muito obrigado.

"Não se trata de quão forte pode bater,
se trata do quanto consegue apanhar
e continuar seguindo em frente."

— Rocky Balboa

RESUMO

O presente trabalho tem como objetivo apresentar o desenvolvimento de um aplicativo móvel voltado à auditoria de estoque das filiais da empresa Novalar, atuante no setor de móveis e eletrodomésticos. O sistema foi concebido para otimizar o processo de conferência de produtos e reduzir inconsistências operacionais nas rotinas de controle de estoque. Para tal, foi utilizada a tecnologia *Flutter*, um *framework* multiplataforma que possibilita o desenvolvimento de interfaces nativas para Android e iOS a partir de um único código-fonte. A arquitetura da aplicação foi estruturada com o auxílio do *GetX*, ferramenta que simplifica o gerenciamento de estado, a injeção de dependências e a navegação entre telas. O aplicativo permite a autenticação segura de usuários, o acesso às informações de produtos via integração com uma *API REST*, a leitura de códigos de barras por meio da câmera do dispositivo e a exportação dos dados auditados. Tais funcionalidades visam substituir processos manuais e planilhas, proporcionando agilidade e confiabilidade na coleta de informações. A implementação seguiu boas práticas de programação e design de software, priorizando modularização, segurança de dados e usabilidade. Com base nos testes realizados nas filiais da empresa, o aplicativo demonstrou desempenho satisfatório, reduzindo significativamente o tempo médio das auditorias e eliminando erros comuns oriundos de anotações em papel. O projeto evidencia a viabilidade e o impacto positivo do uso de tecnologias multiplataforma no contexto corporativo, especialmente em processos de controle e gestão de estoques.

Palavras-chave: *Flutter*; *GetX*; Auditoria de estoque; Aplicativo móvel; Multiplataforma.

ABSTRACT

This work presents the development of a mobile application designed for stock auditing in the branches of Novalar, a company operating in the furniture and home appliances sector. The system was conceived to optimize product verification processes and minimize operational inconsistencies in inventory control routines. The *Flutter* framework was used for implementation, enabling cross-platform development for Android and iOS using a single codebase. The application's architecture was structured with *GetX*, which simplifies state management, dependency injection, and navigation between views. The application provides secure user authentication, access to product data through *REST API* integration, barcode scanning using the device camera, and data export functionalities. These features aim to replace manual processes and spreadsheets, offering greater speed and reliability in data collection. The implementation followed software engineering best practices, emphasizing modularization, data security, and user experience. Tests conducted in the company's branches showed that the application significantly reduced the average time required for audits and eliminated frequent human errors associated with manual processes. The project demonstrates the feasibility and positive impact of cross-platform technologies in corporate environments, particularly in inventory control and management systems.

Keywords: *Flutter*; *GetX*; Stock audit; Mobile application; Cross-platform.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

API	:	Application Programming Interface;
APP	:	Aplicativo;
CD	:	Centro de Distribuição;
Dio	:	Biblioteca HTTP para Flutter;
HTTP	:	Hypertext Transfer Protocol;
HTTPS	:	Hypertext Transfer Protocol Secure;
JSON	:	JavaScript Object Notation;
JWT	:	JSON Web Token;
REST	:	Representational State Transfer;
SDK	:	Software Development Kit;
SKU	:	Stock Keeping Unit;
UI	:	User Interface;
UX	:	User Experience;
XLSX	:	Formato de planilha Microsoft Excel;
PDF	:	Portable Document Format;
API REST	:	API baseada no estilo REST;

SUMÁRIO

1 INTRODUÇÃO

A gestão eficaz de estoques é um dos pilares fundamentais para o bom funcionamento das empresas do setor varejista, especialmente aquelas que atuam com bens de consumo duráveis, como móveis e eletrodomésticos. Em um cenário competitivo e dinâmico, manter o controle preciso do estoque é essencial para evitar perdas, garantir o atendimento ao cliente e otimizar a operação logística.

Nesse contexto, a auditoria de estoques se apresenta como uma ferramenta imprescindível de controle e verificação. Segundo Crepaldi, a auditoria de estoques visa assegurar que os registros contábeis representem, de forma fidedigna, os saldos reais dos bens existentes, verificando a existência física e avaliando a adequação dos controles internos (CREPALDI Silvio A.; CREPALDI, 2016). Esse processo, além de garantir a conformidade contábil, contribui diretamente para a eficiência operacional da empresa.

A realização periódica da contagem física, também conhecida como *inventário*, permite identificar divergências entre o estoque real e o estoque registrado, promovendo ajustes e prevenindo fraudes. Como afirma Attie:

> "Não importa quão eficiente é o sistema de registro contábil dos estoques empregado pela companhia, a contagem física faz-se necessária para assegurar que o registro contábil reflita com propriedade a existência física." (ATTIE, 2018, p. 454)

No entanto, muitas empresas ainda realizam esse processo de forma manual, o que aumenta o risco de erros, retrabalho e inconsistências nos dados. Diante dessa realidade, o desenvolvimento de soluções tecnológicas pode representar um diferencial estratégico. Aplicativos voltados à contagem de estoque em tempo real permitem maior agilidade, precisão e integração das informações, facilitando o trabalho das equipes e fortalecendo os controles internos.

Com base nesse cenário, este trabalho apresenta o desenvolvimento de um aplicativo voltado à contagem de estoque nas filiais da Novalar, empresa do setor de móveis e eletrodomésticos. A proposta tem como objetivo aprimorar o processo de inventário físico, oferecendo uma ferramenta prática, eficiente e alinhada às necessidades operacionais da organização. A solução digitaliza a coleta de dados, reduzindo significativamente o tempo gasto em cada auditoria, eliminando erros de transcrição manual e proporcionando aos gestores acesso imediato às informações coletadas, o que possibilita decisões mais rápidas e assertivas. O uso do *framework Flutter* permite o desenvolvimento de uma única aplicação compatível com dispositivos Android e iOS, otimizando custos e acelerando o processo de desenvolvimento.

2 OBJETIVOS

2.1 OBJETIVOS GERAIS

Desenvolver um aplicativo móvel multiplataforma para auditoria de estoque das filiais da empresa Novalar.

2.2 OBJETIVOS ESPECÍFICOS

- Analisar o processo atual de auditoria de estoque da empresa Novalar e identificar suas principais limitações.
- Projetar uma solução tecnológica que substitua processos manuais e planilhas físicas por registros digitais integrados.
- Implementar uma aplicação com autenticação segura, integração a *API REST* e leitura de códigos de barras.
- Avaliar o desempenho e a usabilidade do aplicativo em cenários reais de auditoria nas filiais da empresa.
- Documentar as etapas de desenvolvimento e validar os resultados com base em métricas de eficiência e confiabilidade.

3 JUSTIFICATIVA

O controle de estoque é um dos processos mais críticos em empresas do setor varejista, pois influencia diretamente a lucratividade, a tomada de decisão e o nível de satisfação do cliente. No caso da Novalar, que atua com um vasto portfólio de móveis e eletrodomésticos, a acurácia das informações de estoque é fundamental para garantir que os produtos estejam disponíveis nas filiais e que as movimentações sejam devidamente registradas.

O método tradicional utilizado pela empresa baseava-se em planilhas manuais e formulários impressos, preenchidos pelos colaboradores durante as auditorias. Essa abordagem, embora funcional em contextos menores, revelou-se ineficiente à medida que o volume de produtos e a quantidade de filiais aumentaram. Entre os principais problemas identificados estavam erros de digitação, duplicidade de registros, demora na consolidação dos dados e falta de padronização entre as auditorias realizadas.

Essas dificuldades impactavam diretamente na tomada de decisão gerencial, pois as informações obtidas durante as conferências de estoque nem sempre refletiam a realidade operacional. Consequentemente, havia prejuízos financeiros, atraso no reabastecimento de produtos e falhas no controle de *inventário*.

Diante desse cenário, o desenvolvimento de um aplicativo móvel dedicado à auditoria de estoque surge como uma solução estratégica. O uso de dispositivos móveis oferece mobilidade, rapidez e precisão, reduzindo a dependência de registros manuais e facilitando o acompanhamento das informações em tempo real.

Além disso, a escolha do *framework Flutter* garante a criação de uma aplicação moderna, com alto desempenho e compatibilidade multiplataforma, atendendo tanto dispositivos Android quanto iOS. A adoção do *GetX* como gerenciador de estado contribui para uma arquitetura de software organizada, escalável e de fácil manutenção.

A justificativa técnica também se apoia no potencial de redução de custos operacionais e melhoria da produtividade. Ao automatizar o processo de auditoria, o aplicativo elimina redundâncias, padroniza a coleta de dados e aumenta a confiabilidade das informações. Assim, o projeto se mostra relevante tanto sob o ponto de vista tecnológico quanto empresarial, representando uma contribuição prática e acadêmica para o campo da Análise e Desenvolvimento de Sistemas.

4 METODOLOGIA

A metodologia adotada neste trabalho fundamenta-se em princípios da engenharia de software, seguindo um processo iterativo e incremental de desenvolvimento. As etapas compreendem análise de requisitos, modelagem do sistema, implementação, testes e validação. O enfoque principal foi o desenvolvimento ágil, priorizando entregas funcionais curtas e testes constantes para assegurar a qualidade do produto final.

4.1 TIPO DE PESQUISA

Trata-se de uma pesquisa aplicada, pois busca resolver um problema concreto enfrentado pela empresa Novalar. Segundo Gil, a pesquisa aplicada caracteriza-se por seu foco prático, voltado à solução de problemas reais e imediatos dentro de organizações ou contextos específicos (Gil, 2024). O trabalho tem natureza tecnológica e experimental, uma vez que envolve a criação de uma ferramenta digital inédita para o contexto da organização. Quanto à abordagem, é qualitativa e quantitativa, pois combina a observação direta do processo de auditoria com a mensuração de ganhos operacionais após a implementação do aplicativo.

4.2 FERRAMENTAS E TECNOLOGIAS

Foram utilizadas as seguintes tecnologias e ferramentas:

Flutter: *framework* de código aberto, desenvolvido pelo Google, utilizado para a criação de interfaces nativas multiplataforma com a linguagem *Dart* (Flutter, 2025a);

Dart: linguagem principal do projeto, com tipagem forte e foco em desempenho (Dart, 2025);

GetX: biblioteca para gerenciamento de estado, injeção de dependências e navegação (Borges, 2025);

Dio: cliente HTTP responsável pela comunicação com a *API REST* da empresa (Flutter, 2025b);

Flutter Secure Storage: biblioteca usada para armazenamento seguro de tokens e credenciais (Steenbakker, 2025a);

Mobile Scanner: integração com a câmera do dispositivo para leitura de códigos de barras (Steenbakker, 2025b);

Flutter Local Notifications: envio de notificações locais para feedback do usuário (Bui, 2025);

Git e *GitHub*: controle de versão e repositório remoto do projeto (Git, 2025) (GitHub, 2025);

Visual Studio Code: ambiente de desenvolvimento integrado (IDE) utilizado (Microsoft, 2025);

Android Studio: ferramenta de compilação, emulador e geração de *builds* para publicação (Google, 2025).

4.3 PROCESSO DE DESENVOLVIMENTO

O desenvolvimento foi dividido em módulos funcionais correspondentes às principais etapas do fluxo de auditoria:

Autenticação de usuários: implementação do login com integração ao *backend* e armazenamento seguro do token *JWT*.

Listagem de filiais e histórico de auditorias: sincronização com a *API* para exibir as unidades disponíveis e as auditorias realizadas.

Registro e leitura de produtos: escaneamento de códigos de barras, busca de informações e atualização do status dos itens.

Exportação de resultados: geração e envio de relatórios de auditoria com notificações locais.

Cada módulo foi testado isoladamente e em conjunto, validando fluxos de navegação e integridade de dados. A comunicação com o *backend* foi monitorada com *logs* estruturados e tratamento de exceções via *Either*, um construto funcional que representa explicitamente resultados de sucesso ou falha, permitindo o controle seguro de erros sem o uso de exceções tradicionais, assegurando a consistência das respostas da *API*.

5 FUNDAMENTAÇÃO TEÓRICA

A fundamentação teórica aborda os conceitos e tecnologias que sustentam o desenvolvimento do aplicativo, articulando a literatura técnica e acadêmica à prática implementada.

5.1 AUDITORIA DE ESTOQUE E GESTÃO EMPRESARIAL

A auditoria de estoque é um processo sistemático destinado a verificar a correspondência entre o estoque físico e o estoque registrado nos sistemas de controle. Segundo Corrêa, o objetivo principal é assegurar que as quantidades registradas representem com fidelidade os bens disponíveis, permitindo uma gestão eficiente e decisões assertivas sobre compras, reposição e vendas (CORRÊA, 2017).

Em empresas de médio e grande porte, a precisão no controle de estoque é vital para manter o equilíbrio entre oferta e demanda, reduzir perdas e evitar rupturas. A aplicação de tecnologia da informação nesse contexto representa um diferencial competitivo, uma vez que permite monitoramento contínuo, rastreabilidade e automação das etapas de conferência.

5.2 APLICATIVOS MÓVEIS CORPORATIVOS

Com a popularização dos *smartphones*, as organizações têm adotado soluções móveis para aprimorar a comunicação interna, o atendimento ao cliente e a gestão operacional. De acordo com Pressman, o desenvolvimento de aplicativos corporativos deve considerar critérios de segurança, usabilidade e integração com sistemas legados (PRESSMAN, 2021).

Os aplicativos móveis para uso interno empresarial possibilitam a execução de tarefas críticas de forma ágil, mesmo fora do ambiente de escritório. No caso da auditoria de estoque, o uso de dispositivos móveis reduz o tempo de conferência e aumenta a precisão, pois a coleta de dados ocorre diretamente na origem da informação, o próprio produto físico.

5.3 DESENVOLVIMENTO MULTIPLATAFORMA COM FLUTTER

O *Flutter*, lançado oficialmente pelo Google em 2018, é um *framework* que permite o desenvolvimento de aplicações nativas para Android, iOS, Web e Desktop com uma

única base de código. Seu principal diferencial é o uso do motor gráfico *Skia*, que renderiza os componentes diretamente, garantindo alto desempenho e consistência visual.

A principal vantagem do *Flutter* está na produtividade, uma vez que elimina a necessidade de manter projetos separados por plataforma. Além disso, sua linguagem *Dart* é moderna, de tipagem estática e orientada a objetos, o que facilita a manutenção e a escalabilidade do código.

Para o desenvolvimento mobile, o *Flutter* oferece uma abordagem moderna e completa. Ele utiliza a linguagem *Dart* e possui um mecanismo de renderização próprio, que permite total controle sobre a interface do usuário, garantindo consistência visual e desempenho nativo em qualquer dispositivo. Além disso, o extenso conjunto de *widgets* personalizáveis oferece suporte aos padrões de design tanto do Android (*Material Design*) quanto do iOS (*Cupertino*), o que permite criar aplicativos visualmente integrados a cada plataforma sem perder a produtividade.

O recurso de *hot reload* é especialmente valioso no ambiente mobile, pois permite testar rapidamente alterações no código e no layout, acelerando significativamente o ciclo de desenvolvimento. Esse ganho de produtividade é essencial em projetos com prazos curtos ou equipes enxutas.

No contexto empresarial, o *Flutter* vem sendo amplamente adotado por *startups*, pequenas empresas e grandes corporações que buscam agilidade e redução de custos no desenvolvimento de aplicativos móveis. Ao permitir o uso de uma única base de código para múltiplas plataformas, o *Flutter* reduz drasticamente o esforço necessário para manter e evoluir o aplicativo, além de facilitar a integração com sistemas legados e *APIs* corporativas por meio dos *platform channels*, mecanismo que possibilita a comunicação direta entre o código *Dart* e funcionalidades nativas específicas de cada plataforma, como *Android* e *iOS*.

Empresas valorizam também a escalabilidade e a facilidade de manutenção proporcionadas pelo *Flutter*. O suporte à modularização do código, testes automatizados e boas práticas arquiteturais contribui para projetos mais robustos e sustentáveis a longo prazo. Além disso, a comunidade ativa e o ecossistema crescente de pacotes prontos reduzem o tempo de desenvolvimento e promovem inovação contínua.

Grandes marcas como Google, Alibaba, BMW, Nubank e iFood já utilizam *Flutter* em seus produtos, evidenciando sua maturidade e confiabilidade no ambiente corporativo.

Em resumo, o *Flutter* se destaca como uma solução poderosa e estratégica para o desenvolvimento de aplicativos móveis empresariais, combinando desempenho nativo, rapidez no desenvolvimento e redução de custos operacionais.(Flutter, 2025a)

5.4 GERENCIAMENTO DE ESTADO COM GETX

O *GetX* é uma biblioteca popular na comunidade *Flutter* por oferecer uma abordagem simples e eficiente para o gerenciamento de estado reativo. Ela combina três pilares principais: *State Management*, *Dependency Injection* e *Route Management*. Essa combinação permite que os controladores mantenham a lógica de negócios separada da camada de apresentação, favorecendo a modularidade e o baixo acoplamento, conforme descrito por Borges (Borges, 2025).

O uso de bibliotecas de gerenciamento de estado é fundamental em aplicações que apresentam múltiplas telas e dependem de dados dinâmicos, pois evitam inconsistências e facilita a testabilidade.

O *GetX* se mostrou essencial no aplicativo da Novalar para gerenciar de forma eficiente a sincronização dos estados de carregamento, autenticação e atualização das listas de produtos auditados. A biblioteca foi escolhida pela sua simplicidade, baixa curva de aprendizado e facilidade de manutenção do código, permitindo um controle de estado eficaz sem a necessidade de estruturas complexas para a comunicação entre *widgets*. Esses benefícios contribuíram diretamente para um desenvolvimento mais ágil e organizado, resultando em uma aplicação consistente e de fácil manutenção, aspectos fundamentais para sistemas corporativos de uso intensivo, como o sistema de contagem de estoque (Borges, 2025).

5.5 CONSUMO DE APIS REST COM DIO

A comunicação entre o aplicativo e o *backend* é realizada via *API REST*, padrão amplamente utilizado por sua simplicidade e interoperabilidade. O *Dio* foi escolhido como cliente HTTP por oferecer suporte a *interceptors*, que permitem interceptar e modificar requisições e respostas para fins de autenticação, *logging* e tratamento de erros; *formdata*, utilizado no envio de dados estruturados e arquivos por meio de requisições HTTP; e *timeout* configurável, que define um limite máximo de tempo de espera para evitar bloqueios em operações de rede. Além disso, a biblioteca oferece cancelamento de requisições, tratamento centralizado de erros e suporte a operações assíncronas, o que garante maior controle sobre as comunicações. Adicionalmente, o *Dio* facilita a integração com o *GetX*, permitindo que os dados recebidos das *APIs* sejam processados e refletidos em tempo real nas telas da aplicação. Essa combinação assegura uma comunicação estável, segura e eficiente com o *backend*, reduzindo falhas de sincronização e proporcionando uma melhor experiência ao usuário final (Flutter, 2025b).

De acordo com Fielding, a arquitetura *REST* é baseada em princípios de recursos identificáveis, operações padronizadas (*GET*, *POST*, *PUT*, *DELETE*) e transferência de representações em *JSON* ou *XML* (FIELDING, 2000). O uso dessa arquitetura no aplicativo da Novalar permite que as informações de estoque sejam atualizadas em tempo real, garantindo consistência entre os dados locais e os sistemas corporativos.

5.6 SEGURANÇA E ARMAZENAMENTO LOCAL

Em aplicativos corporativos, a segurança da informação é um aspecto crítico. O armazenamento de credenciais e tokens de autenticação deve ser feito de forma criptografada, impedindo o acesso não autorizado. O *Flutter Secure Storage* cumpre esse papel ao utilizar as *APIs* nativas do sistema operacional (*Keychain* no iOS e *Keystore* no Android) (Steenbakker, 2025a).

Além disso, o projeto adota boas práticas de proteção de dados, como a remoção de permissões excessivas e o uso de autenticação baseada em tokens. Essas medidas estão em conformidade com os princípios da Lei Geral de Proteção de Dados (LGPD), garantindo privacidade e integridade das informações coletadas.

6 DESENVOLVIMENTO DO SISTEMA

O desenvolvimento do aplicativo de auditoria de estoque para a empresa Novalar foi conduzido de forma modular e incremental, assegurando que cada componente funcional fosse concebido, implementado e validado antes da integração total do sistema. Nesta seção são apresentados o processo de análise, a arquitetura utilizada, as tecnologias aplicadas e as funcionalidades desenvolvidas.

6.1 ANÁLISE DO PROBLEMA

Durante o diagnóstico inicial, constatou-se que o processo manual de auditoria de estoque apresentava gargalos significativos. As filiais utilizavam planilhas impressas ou arquivos em formato *.xlsx*, preenchidos por colaboradores em campo. As informações eram posteriormente digitadas no sistema corporativo, o que gerava retrabalho e inconsistências de dados.

Além disso, a ausência de integração direta com o sistema principal impedia que os gestores tivessem visibilidade em tempo real dos resultados das auditorias. O processo era demorado, sujeito a falhas humanas e com baixa rastreabilidade.

Esses fatores motivaram a criação de um aplicativo que automatizasse as principais etapas do processo: autenticação do auditor, seleção da filial, leitura dos produtos via código de barras, registro das quantidades e exportação dos resultados consolidados.

6.2 ARQUITETURA E TECNOLOGIAS UTILIZADAS

A arquitetura adotada segue o padrão modular com separação por camadas, típica do ecossistema *Flutter* com *GetX*. Essa estrutura facilita a manutenção e o isolamento das responsabilidades de cada módulo, além de permitir expansões futuras sem comprometer a base do código.

A seguir, a visão geral das camadas do sistema:

- **Camada de Apresentação (UI):** composta por páginas, *widgets* e temas personalizados;
- **Camada de Controle (Controllers):** gerencia o estado da aplicação, as regras de negócio e a interação com os serviços;

- **Camada de Serviço (Services):** responsável pela comunicação com a *API REST* e persistência de dados locais;
- **Camada de Modelo (Models):** define as classes de dados (como *Filial*, *Item*, *Grupo*, *UserData*) e seus mapeamentos *fromMap*/*toMap*.

Essa organização reflete o princípio *Clean Code*, que busca reduzir acoplamento e aumentar a coesão entre os componentes, conforme defendido por Martin, ao enfatizar que um código limpo deve ser simples, legível e estruturalmente organizado para facilitar sua manutenção (Martin, 2009). Entre as principais tecnologias empregadas estão:

- **Flutter 3.19+**: *framework* principal;
- **Dart 3**: linguagem de programação;
- **GetX**: gerenciamento de estado e rotas;
- **Dio**: comunicação com *API REST*;
- **Flutter Secure Storage**: autenticação segura;
- **Mobile Scanner**: leitura de código de barras;
- **Flutter Local Notifications**: alertas e notificações locais;
- **Path Provider / Open File**: exportação e manipulação de arquivos.

6.3 ESTRUTURA DE PASTAS E MÓDULOS

A estrutura de diretórios do projeto foi organizada conforme boas práticas de desenvolvimento em *Flutter*, garantindo clareza e escalabilidade:

Cada módulo contém seus próprios *controllers*, *bindings*, *pages* e *services*, conforme a convenção do *GetX*. Essa estrutura modulariza as responsabilidades e permite que cada parte seja desenvolvida e testada independentemente.

6.4 FUNCIONALIDADES IMPLEMENTADAS

O aplicativo foi desenvolvido com foco em simplicidade, desempenho e confiabilidade. As principais funcionalidades incluem:

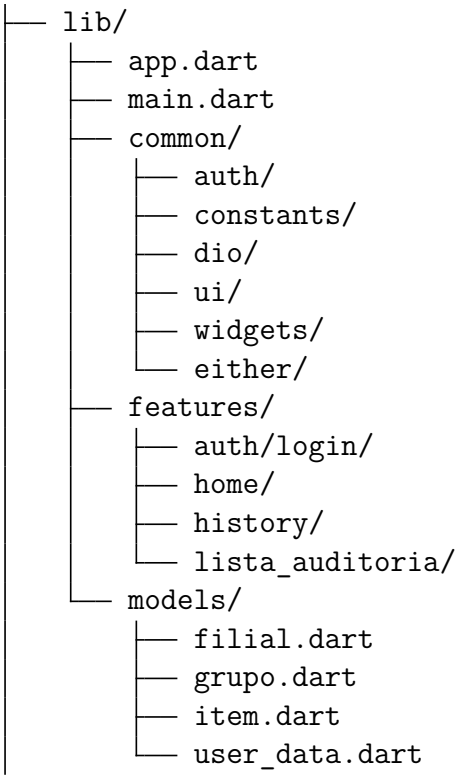


Figura 6.1: Estrutura de diretórios do projeto.

- **Autenticação de Usuário:** O fluxo de login utiliza credenciais corporativas integradas à *API* da Novalar. Após autenticação bem-sucedida, o token *JWT* é armazenado localmente por meio do *Flutter Secure Storage*, garantindo persistência segura da sessão.(Steenbakker, 2025a)

- **Listagem de Filiais:** Após o login, o usuário visualiza uma lista de filiais vinculadas à sua conta. Essa listagem é obtida via requisição HTTP utilizando o *Dio*, e os dados são carregados de forma assíncrona com estado controlado por *GetX*.

- **Histórico e Seleção de Grupos:** A tela de histórico exibe auditorias anteriores e permite selecionar grupos de produtos para uma nova verificação. Essa funcionalidade promove rastreabilidade e reutilização de configurações.

- **Auditoria de Produtos:** Na tela principal da auditoria, o aplicativo utiliza o pacote *Mobile Scanner* para ler códigos de barras. Ao detectar um código, o sistema realiza uma busca no *backend* para obter informações do produto, permitindo que o auditor atualize quantidades e observações diretamente no dispositivo.(Steenbakker, 2025b)

- **Exportação de Dados:** Após a finalização da auditoria, o aplicativo gera um

relatório consolidado em formato de arquivo e envia uma notificação local informando a conclusão do processo. O arquivo pode ser aberto diretamente pelo usuário ou compartilhado por outros canais corporativos.

6.5 FLUXO DE NAVEGAÇÃO

O fluxo de navegação segue o padrão *GetX* com rotas nomeadas. O usuário percorre as seguintes etapas:

1. **Tela de Login:** Autenticação e verificação de permissões;
2. **Tela de Filiais:** Seleção da unidade de auditoria;
3. **Tela de Histórico:** Exibição de auditorias anteriores;
4. **Tela de Auditoria de Itens:** Escaneamento e registro de produtos;
5. **Exportação:** Geração de arquivo e notificação de sucesso.

Esse fluxo foi projetado para reduzir o número de interações necessárias, mantendo uma experiência fluida e eficiente para o auditor em campo.

6.6 INTEGRAÇÃO COM API E AUTENTICAÇÃO

A integração entre o aplicativo e o servidor segue o padrão *RESTful*. O *Dio* é configurado em um arquivo central, que define o *baseUrl*, o tempo limite e os *interceptors*. As principais rotas de comunicação incluem:

- **/login**: autenticação de usuário;
- **/filiais**: listagem das unidades;
- **/grupos**: histórico e agrupamento de auditorias;
- **/itens**: consulta e atualização de produtos;
- **/export**: geração e download dos relatórios.

Cada requisição é encapsulada em um *Service*, e os controladores do *GetX* tratam as respostas utilizando a classe genérica *Either<Failure, Success>*, o que permite manipular resultados e erros de forma elegante.

6.7 EXPORTAÇÃO E NOTIFICAÇÕES

A exportação dos resultados é realizada no formato de arquivo, salvo em diretório interno do aplicativo por meio da biblioteca `path_provider`. Após a geração, o sistema utiliza o `Flutter Local Notifications` para informar o usuário sobre a conclusão da auditoria. Essa etapa foi pensada para substituir mensagens visuais transitórias (como `SnackBars`) por alertas persistentes e rastreáveis.(Bui, 2025)(Flutter, 2025c)

A notificação oferece a opção de abrir o arquivo diretamente via `open_file`, garantindo praticidade ao auditor.

6.8 APRESENTAÇÃO DAS TELAS DO APLICATIVO

Figura 6.2: Tela de login

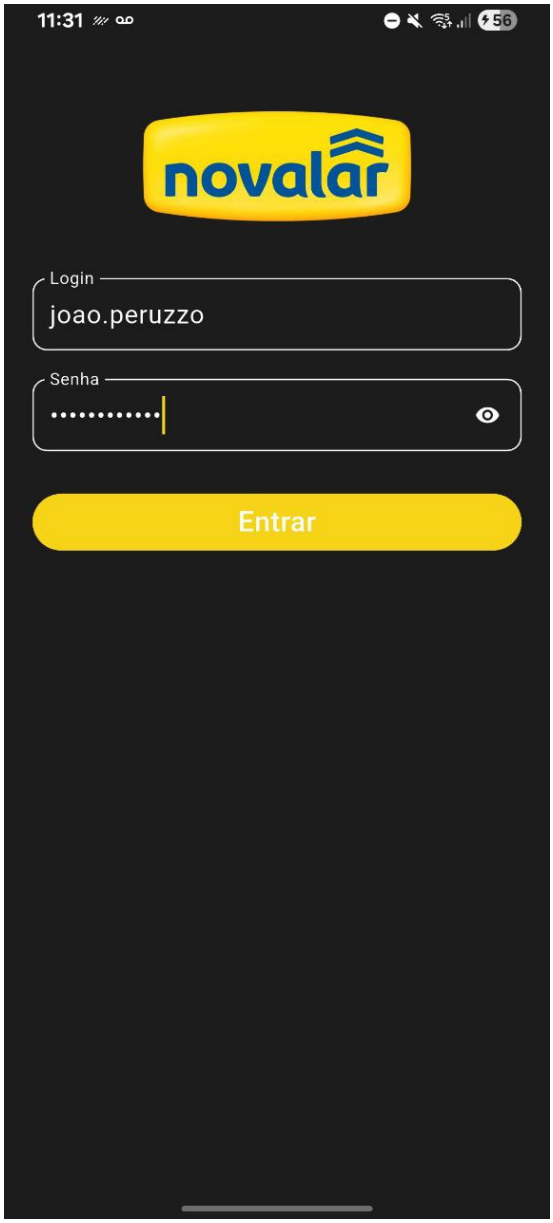
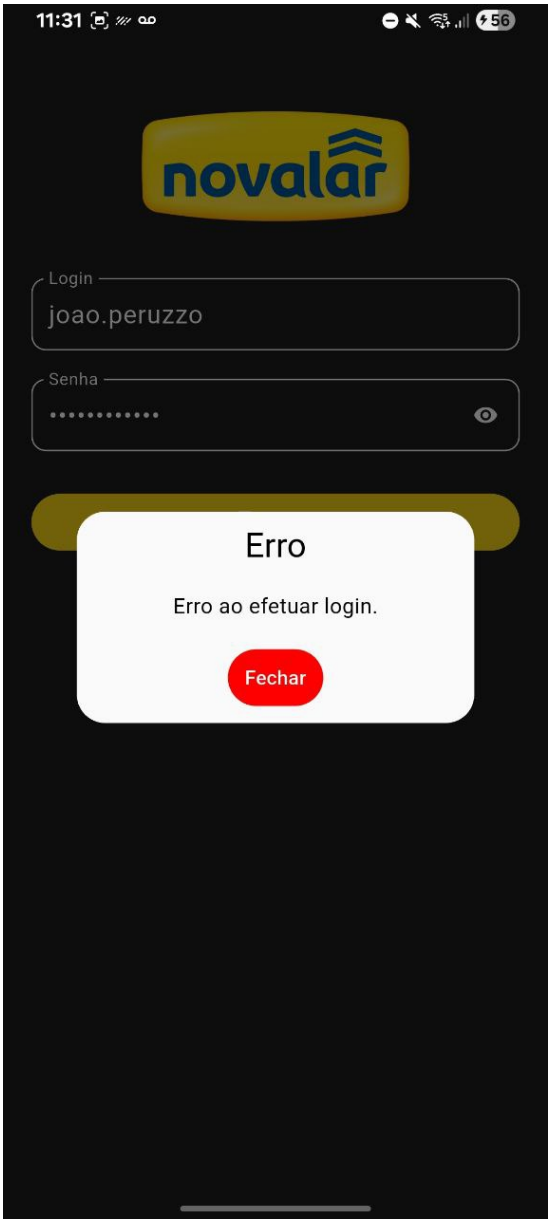


Figura 6.3: Mensagem de erro

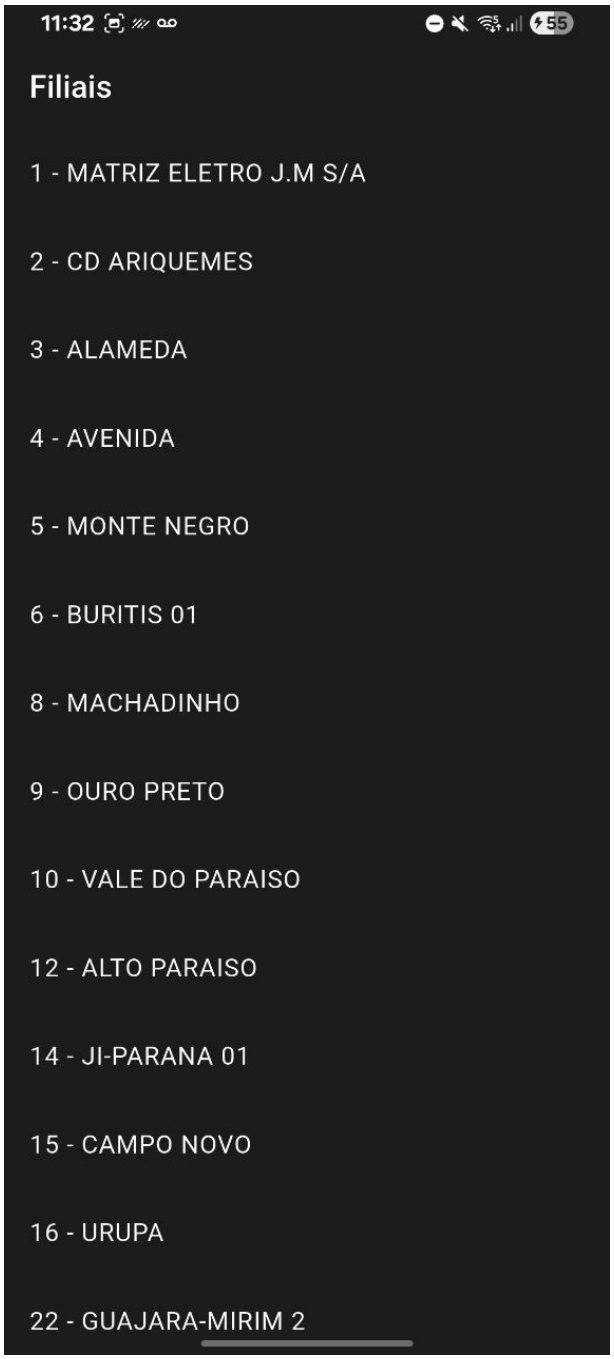


Fonte: Elaborado pelo autor, 2025.

A Figura 6.2 apresenta a tela inicial do aplicativo, responsável pela autenticação do usuário. Nela, o colaborador insere suas credenciais de acesso, validadas por meio de uma *API* segura. O design segue a identidade visual da empresa, utilizando o logotipo da Novalar e a paleta de cores amarela e azul característica da marca. A interface é minimalista, priorizando clareza e usabilidade. A autenticação é um ponto crítico do sistema, garantindo que apenas usuários autorizados acessem as informações sensíveis

de estoque. O aplicativo realiza a comunicação com o *backend* utilizando o protocolo *HTTPS* e, após a validação, armazena temporariamente o token de sessão no dispositivo via *Flutter Secure Storage*. Em caso de falha de login, o sistema exibe uma mensagem de erro (Figura 6.3).

Figura 6.4: Tela de seleção da filial



Fonte: Elaborado pelo autor, 2025.

A Figura 6.4 mostra a tela de seleção das filiais. Nesta etapa, o auditor escolhe qual unidade da empresa será auditada, dentre uma lista que exibe o nome e o identificador

de cada filial. A listagem segue um formato simples e limpo, utilizando rolagem vertical e tipografia clara para facilitar a leitura. Essa abordagem melhora a experiência em dispositivos móveis, permitindo navegação fluida mesmo em listas extensas. Essa tela é diretamente alimentada pelos dados provenientes da *API*, garantindo que as informações estejam sempre atualizadas conforme as filiais cadastradas no sistema central.

Figura 6.5: Tela de histórico de listas da filial selecionada

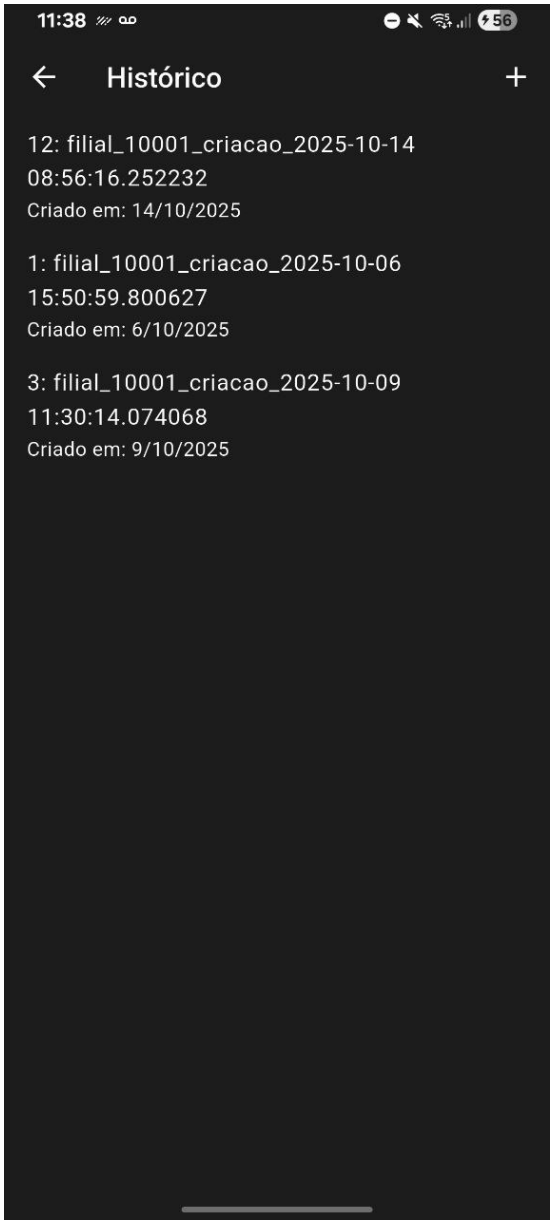
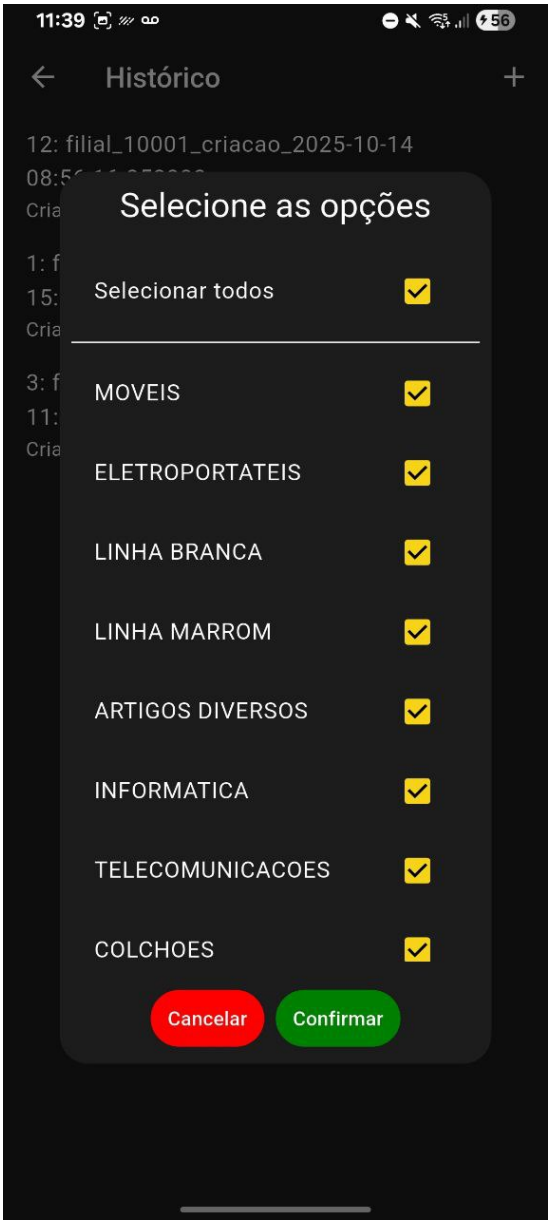


Figura 6.6: Seleção das linhas de produtos para gerar a lista



Fonte: Elaborado pelo autor, 2025.

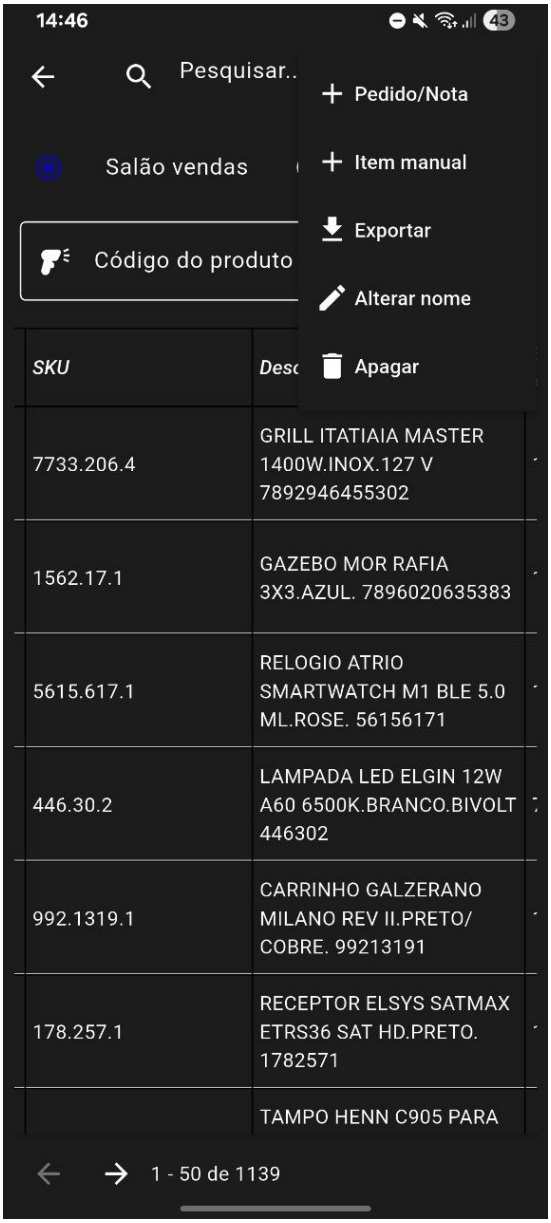
A Figura 6.5 apresenta o histórico de auditorias realizadas. Cada item contém o código da filial, a data de criação da auditoria e um identificador único de sessão. Esse

recurso foi implementado para garantir rastreabilidade e controle das operações realizadas pelos auditores, possibilitando que cada ação seja auditável posteriormente. O botão "+", localizado no canto superior direito, permite a criação de uma nova sessão de auditoria, conduzindo o usuário ao processo de seleção de setores de produtos.

A Figura 6.6 exibe a interface de seleção dos setores de produtos. O usuário pode marcar individualmente os segmentos que serão incluídos na auditoria, como "Móveis", "Eletroportáteis", "Linha Branca", "Informática" e outros, ou optar por selecionar todos simultaneamente. A estrutura visual adota caixas de seleção com indicadores coloridos, botões de ação ("Cancelar" em vermelho e "Confirmar" em verde) e disposição vertical dos itens, favorecendo o uso em campo com apenas uma mão. Essa etapa do fluxo é importante para personalizar a auditoria de acordo com o escopo de cada filial, evitando a contagem de itens irrelevantes.

Figura 6.7: Tela principal da auditoria de produtos

Figura 6.8: Opções adicionais disponíveis



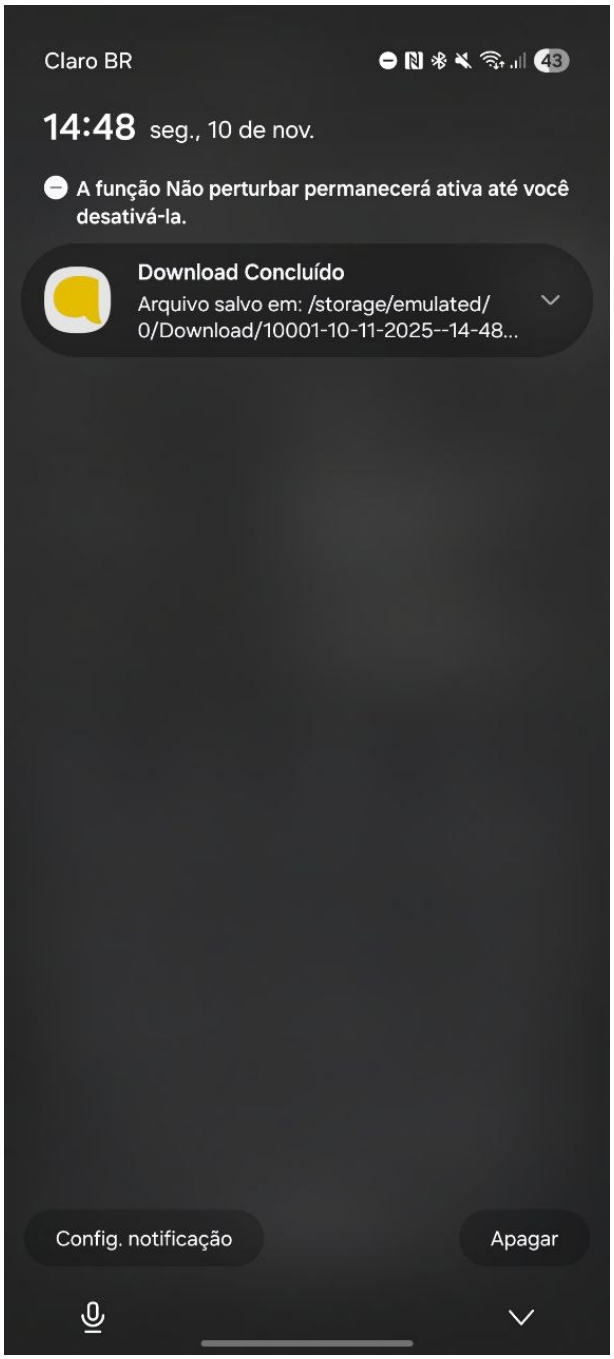
Fonte: Elaborado pelo autor, 2025.

A Figura 6.7 mostra a principal tela funcional do sistema, a de auditoria. Nela, o auditor realiza a contagem dos produtos disponíveis na filial. Na parte superior, há um campo de busca com suporte à leitura de código de barras, representado por um ícone de scanner, além da opção de utilizar a câmera do smartphone para ser realizado o processo de leitura. Abaixo, a listagem dos produtos é organizada em formato tabular, contendo as colunas: “Total contado”, “Salão vendas”, “Depósito”, “SKU”, “Descrição”, “Saldo Atual”, “Saldo Pendente”, “Avaria” e “Observação”. O usuário pode alternar

entre as opções “Salão Vendas” e “Depósito” através de botões circulares, o que permite segmentar a auditoria conforme a localização física dos produtos. O uso de tabelas com cores contrastantes e linhas delimitadas aumenta a legibilidade das informações.

A Figura 6.8 apresenta o menu lateral, no qual estão disponíveis as opções de adicionar itens provenientes de pedidos e notas fiscais, incluir itens manualmente quando ausentes na lista original, exportar a lista em formato *.xlsx*, renomeá-la ou excluí-la.

Figura 6.9: Notificação ao exportar lista



Fonte: Elaborado pelo autor, 2025.

A Figura 6.9 demonstra a notificação gerada ao exportar uma lista de contagem, onde é informado o caminho onde o arquivo foi salvo.

Figura 6.10: Primeira parte da planilha exportada

	A	B	C
1	SKU	Descrição Produto	Diferença
18	7511.30.3	AR CONDICIONADO SPLIT TCL TAC-12CTG2 INVERTER 12000BTUS .BRANCO.220 V 7908293001243 (2/2)	1
19	5098.440.1	ARMARIO BRIZ B14 MULTIUSO .RUSTICO. 7890022046932	0
20	4677.792.1	ARMARIO ITATIAIA NEW JAZZ ANGULO 2 PORTAS.FREIJO/OFF WHITE. 7892946409060	-1
21	185.3.4	ASPIRADOR DE PO E AGUA ELECTROLUX GT30N 20 LITROS.AMARELO/PRETO.127 V 7896347144193	0
22	7517.2.4	ASPIRADOR DE PO E AGUA KARCHER WDL1S 12 LITROS.AMARELO.127 V 7891374302158	0
23	6259.264.4	ASPIRADOR DE PO ELECTROLUX LIT31 FILTRO HEPA.PRETO/BRANCO.127 V 7896347158862	0
24	7261.257.5	AUTO RADIO AIWA BLUETOOTH 100W AWS-CA-D-01.PRETO.12 V 7898578153708	0
25	5103.150.1	BALCAO HENN BELIZE C509 2 PORTAS 800MM.CINZA. 51031501	0
26	6803.30.1	BALCAO ITATIAIA 120 NEW PREMIUM 3 PORTAS 2 GAVETAS SEM TAMPO.BRANCO. 7892946439241	0
27	4679.792.1	BALCAO ITATIAIA NEW JAZZ ANGULO 1 PORTA.FREIJO/OFF WHITE. 7892946409145	-1
28	6334.30.1	BALCAO TELASUL DIAMANTE SMART NEW 818134 3 PORTAS 1 GAVETAS COM TAMPO.BRANCO. 7890724094347 (1/2)	0
29	6334.30.1	BALCAO TELASUL DIAMANTE SMART NEW 818134 3 PORTAS 1 GAVETAS COM TAMPO.BRANCO. 7890724094347 (2/2)	0
30	7441.1219.1	BANCADA BALCAO POLITORNO GOURMET	1
31	5049.1260.1	BANCADA MADETEC LANA SUSPENSA 180.TITANIO/CARVALHO. 7899762727125 (1/2)	0
32	5049.1260.1	BANCADA MADETEC LANA SUSPENSA 180.TITANIO/CARVALHO. 7899762727125 (2/2)	0
33	3335.30.1	BANHEIRA GALZERANO STANDARD II PANDA.BRANCO. 7908200100939	0
34	4591.1212.1	BANQUETA FAMAIS FORTS ARTESANAL .ARGILA/MARROM. 7898617927451	0
35	1374.257.1	BANQUETA TRAMONTINA NITEROI.PRETO. 7898187046835	0
36	1026.17.1	BARRACA MOR IGLU 3 PESSOAS.AZUL. 7898937674349	0
37	4148.257.4	BATEDEIRA MONDIAL PLANETARIA BP-03-B 700W.PRETO.127 V 7899882308525	0
38	4149.317.4	BATEDEIRA MONDIAL PLANETARIA BP-03-R 700W.VERMELHO.127 V 7899882308549	0

Fonte: Elaborado pelo autor, 2025.

Figura 6.11: Segunda parte da planilha exportada

	B	C	D	E	F	G	H	I	J	K	L	M
1		Diferença	Saldo	Saldo T	Salão Venda	Depósito	Pedido v	Transferência	Total contagem	Observação	Avaria	
18		1	2	3	3	0	0	0	3			
19		0	1	1	1	0	0	0	1			
20		-1	1	0	0	0	0	0	0			
21		0	2	2	2	0	0	0	2			
22		0	1	1	1	0	0	0	1			
23		0	1	1	1	0	0	0	1			
24		0	1	1	1	0	0	0	1			
25		0	1	1	1	0	0	0	1			
26		0	1	1	1	0	0	0	1			
27		-1	1	0	0	0	0	0	0			
28		0	1	1	1	0	0	0	1			
29		0	1	1	1	0	0	0	1			
30		1	0	1	1	0	0	0	1			
31		0	1	1	1	0	0	0	1			
32		0	1	1	1	0	0	0	1			
33		0	1	1	0	1	0	0	1			
34		0	1	1	1	0	0	0	1			
35		0	11	11	11	0	0	0	11		Base quebrada	
36		0	2	2	2	0	0	0	2			
37		0	3	3	3	0	0	0	3			
38		0	1	1	1	0	0	0	1			

Fonte: Elaborado pelo autor, 2025.

Conforme mostra a figura 6.10 e 6.11, a planilha exportada pelo sistema contém colunas estruturadas para representar tanto os dados registrados no estoque corporativo quanto os valores coletados durante a conferência física. Entre os campos exportados estão: “SKU”, que identifica unicamente cada produto; “Descrição Produto”, que auxilia na conferência visual; “Saldo”, representando o estoque teórico; e “Saldo T”, valor calculado a partir das quantidades contadas e possíveis ajustes operacionais. As colunas “Salão Venda” e “Depósito” correspondem às contagens físicas realizadas em cada ambiente. O campo “Total contagem” é gerado automaticamente para consolidar a soma dessas quantidades, enquanto a coluna “Diferença” evidencia a variação entre o saldo registrado e o saldo contado. Completam a estrutura os campos “Pedido v”, “Transferência”, “Observação”

e “Avaria”, garantindo capacidade de detalhamento das situações encontradas durante a auditoria.

Tabela 6.1: Descrição das colunas geradas na planilha de auditoria

Coluna	**Descrição**
SKU	Identificador único do produto no sistema corporativo, utilizado como chave primária para consulta e controle de estoque.
Descrição Produto	Nome completo do item, permitindo a identificação visual durante a conferência física.
Diferença	Valor calculado automaticamente, representando a diferença entre o saldo total contado e o saldo registrado. Fórmula: *Diferença = Saldo T – Saldo.*
Saldo	Quantidade teórica registrada no sistema antes da auditoria, correspondente ao estoque esperado na filial.
Saldo T	Saldo total ajustado após a contagem física. Calculado considerando Salão de vendas, Depósito, Pedido de venda e Transferência. Fórmula: *Saldo T = Salão Venda + Depósito – Pedido v – Transferência.*
Salão Venda	Quantidade de produtos localizada no salão de vendas da filial durante a auditoria.
Depósito	Quantidade de itens localizada no depósito da unidade auditada.
Pedido v	Quantidade vinculada a pedidos de venda pendentes.
Transferência	Quantidade de produtos em movimentação entre unidades.
Total contagem	Soma das quantidades contadas no salão e no depósito. Representa a contagem física realizada pelo auditor. Fórmula: *Total contagem = Salão Venda + Depósito.*
Observação	Campo textual destinado a anotações específicas relacionadas ao produto, como avarias, embalagens danificadas ou observações operacionais.
Avaria	Quantidade de itens avariados identificados no processo de auditoria.

Fonte: Elaborado pelo autor, 2025.

7 CONCLUSÃO

O desenvolvimento do aplicativo de auditoria de estoque da empresa Novalar atingiu os objetivos propostos, demonstrando que é possível construir uma solução multiplataforma, eficiente e segura para apoiar processos corporativos críticos.

A utilização do *Flutter* e do *GetX* proporcionou uma arquitetura limpa, modular e escalável, capaz de sustentar futuras expansões. O uso de *Dio*, *Secure Storage* e *Local Notifications* consolidou um ecossistema moderno, alinhado às boas práticas de engenharia de software e segurança da informação.

Como resultado, o aplicativo proporcionou uma redução expressiva no tempo de auditoria, aumento na confiabilidade dos dados e melhor aproveitamento dos recursos humanos. A adoção dessa tecnologia p ela Novalar simboliza um passo importante rumo à maturidade digital da empresa.

REFERÊNCIAS

ATTIE, William. **Auditoria Conceitos e Aplicações**. 7. ed. Rio de Janeiro: Atlas, 2018.

BORGES, Jonatas. **get | Flutter package**. [S.l.*:* s.n.], 2025. https://pub.dev/packages/get. [Acesso em: 12-11-2025].

BUI, Michael. **flutter_local_notifications | Flutter package**. [S.l.*:* s.n.], 2025. https://pub.dev/packages/flutter_local_notifications. [Acesso em: 14-11-2025].

CORRÊA, Henrique L. **Administração de Produção e Operações: Manufatura e Serviços – Uma Abordagem Estratégica**. 3. ed. São Paulo: Atlas, 2017.

CREPALDI SILVIO A.; CREPALDI, Guilherme S. **Auditoria Contábil: Teoria e Prática**. 10. ed. São Paulo: Atlas, 2016.

DART. **Dart Programming Language**. [S.l.*:* s.n.], 2025. https://dart.dev/. [Acesso em: 24-11-2025].

FIELDING, Roy Thomas. **Architectural Styles and the Design of Network-based Software Architectures**. [S.l.]: University of California, 2000.

FLUTTER. **Build apps for any screen**. [S.l.*:* s.n.], 2025. https://flutter.dev. [Acesso em: 05-11-2025].

FLUTTER. **dio | Flutter package**. [S.l.*:* s.n.], 2025. https://pub.dev/packages/dio. [Acesso em: 12-11-2025].

FLUTTER. **path_provider | Flutter package**. [S.l.*:* s.n.], 2025. https://pub.dev/packages/path_provider. [Acesso em: 15-11-2025].

GIL, Antonio Carlos. **Métodos e Técnicas de Pesquisa Social**. 7. ed. São Paulo: Atlas, 2024.

GIT. **Git – Distributed Version Control System**. [S.l.*:* s.n.], 2025. https://git-scm.com/. [Acesso em: 24-11-2025].

GITHUB. **GitHub – Software Development Platform**. [S.l.*:* s.n.], 2025. https://github.com/. [Acesso em: 24-11-2025].

GOOGLE. **Android Studio**. [S.l.*:* s.n.], 2025. https://developer.android.com/studio. [Acesso em: 24-11-2025].

MARTIN, Robert C. **Clean Code: A Handbook of Agile Software Craftsmanship**. Upper Saddle River: Prentice Hall, 2009.

MICROSOFT. **Visual Studio Code**. [S.l.*:* s.n.], 2025. https://code.visualstudio.com/. [Acesso em: 24-11-2025].

PRESSMAN, Roger S. **Engenharia de Software: uma abordagem profissional**. 9. ed. Porto Alegre: AMGH, 2021.

STEENBAKKER, Julian. **flutter_secure_storage | Flutter package**. [S.l.*:* s.n.], 2025. https://pub.dev/packages/flutter_secure_storage. [Acesso em: 12-11-2025].

STEENBAKKER, Julian. **mobile_scanner | Flutter package**. [S.l.*:* s.n.], 2025. https://pub.dev/packages/mobile_scanner. [Acesso em: 14-11-2025].