



GESTÃO E MONITORAMENTO NO AGRONEGÓCIO: UMA PLATAFORMA PARA EFICIÊNCIA OPERACIONAL

MANAGEMENT AND MONITORING IN AGRIBUSINESS: A PLATFORM FOR OPERATIONAL EFFICIENCY

GESTIÓN Y MONITORAMIENTO NO AGRONEGÓCIO: UNA PLATAFORMA PARA EFICIENCIA OPERACIONAL

Gabriel Matias da Silva¹, Felipe Diniz Dallilo², Fabiana Florian³

<https://doi.org/10.47820/recima21.v6i1.7040>

PUBLICADO: 11/2025

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema integrado para a gestão de propriedades rurais, abrangendo controle financeiro, gerenciamento de insumos, rotinas fiscais, folha de pagamento e geração de relatórios estratégicos. O objetivo é centralizar informações e otimizar as atividades da fazenda, promovendo uma administração mais eficiente. A ferramenta foi implementada com Flutter no *front-end* e Flask no *back-end*, com interface baseada em componentes e arquitetura escalável. A metodologia envolveu prototipação e validação técnica em ambiente controlado, com cenários operacionais simulados representativos da rotina de propriedades rurais; a conformidade foi verificada com *checklist* de requisitos e testes funcionais e de integração. Os resultados indicaram atendimento dos módulos (insumos, maquinários, folha e relatórios) aos requisitos definidos. Conclui-se pela viabilidade técnica da solução para apoiar a gestão rural, com potencial de ganhos em controle e eficiência pela centralização e rastreabilidade dos registros quando aplicada a dados reais, recomendando-se avaliações com usuários em etapas futuras.

PALAVRAS-CHAVE: Administração Rural. Agronegócio. Flask. Flutter. Sistema Mobile.

ABSTRACT

This study presents the development and technical feasibility assessment of an integrated system for farm management, encompassing financial control, input management, tax routines, payroll, and the generation of strategic reports. The objective is to centralize information and optimize farm activities, fostering more efficient administration. The solution was implemented with Flutter on the front end and Flask on the back end, featuring a component-based interface and a scalable architecture. The methodology comprised prototyping and technical validation in a controlled environment with simulated operational scenarios representative of rural properties; conformance was verified using a requirements checklist and functional and integration tests. The results indicate that the modules (financial, inputs, tax, payroll, and reports) met the defined requirements. The study concludes that the solution is technically feasible to support rural management, with potential gains in control and efficiency through the centralization and traceability of records when applied to real data and recommends subsequent evaluations with end users in future stages.

KEYWORDS: Farm Management. Agribusiness. Flask. Flutter. Mobile System.

¹ Graduando do Curso de Sistemas de Informação da Universidade de Araraquara - UNIARA. Araraquara-SP.

² Orientador: Docente do curso de Sistemas de Informação da Universidade de Araraquara - UNIARA. Araraquara-SP.

³ Coorientador: Docente do curso de Sistemas de Informação da Universidade de Araraquara - UNIARA. Araraquara-SP.

RESUMEN

Este trabajo presenta el desarrollo de un sistema integrado para la gestión de propiedades rurales, abarcando el control financiero, la gestión de insumos, las rutinas fiscales, la nómina de pago y la generación de informes estratégicos. El objetivo es centralizar la información y optimizar las actividades de la finca, promoviendo una administración más eficiente. La herramienta fue implementada con Flutter en el front-end y Flask en el back-end, con una interfaz basada en componentes y una arquitectura escalable. La metodología incluyó la creación de prototipos y la validación técnica en un entorno controlado, con escenarios operativos simulados que representan la rutina de propiedades rurales; la conformidad se verificó mediante una lista de requisitos y pruebas funcionales y de integración. Los resultados indicaron que los módulos (insumos, maquinarias, nómina e informes) cumplen con los requisitos establecidos. Se concluye la viabilidad técnica de la solución para apoyar la gestión rural, con potencial de mejoras en control y eficiencia gracias a la centralización y la trazabilidad de los registros cuando se aplique a datos reales, recomendándose evaluaciones con usuarios en etapas futuras.

PALABRAS CLAVE: Administración rural. Agronegocio. Flask. Flutter. Sistema móvil.

1. INTRODUÇÃO

A gestão operacional de propriedades rurais refere-se ao conjunto de práticas voltadas ao planejamento, execução e controle das atividades produtivas no campo, incluindo o manejo de insumos, mão de obra, maquinário, processos financeiros e logísticos (Batalha, 2021). Diante da complexidade crescente das atividades agropecuárias, a adoção de soluções tecnológicas é essencial para uma administração mais eficiente e baseada em dados. A digitalização possibilita melhor controle de insumos, fluxo de caixa, processos fiscais e relatórios estratégicos, reduzindo desperdícios.

Nesse contexto, soluções integradas capazes de consolidar dados dispersos e suportar a análise gerencial são estratégicas. Contudo, ainda são limitadas as plataformas que unificam informações financeiras, administrativas e operacionais em um único ambiente (Revista Cultivar, 2021), especialmente com foco mobile e operação *offline* e de baixa conectividade, o que evidencia uma lacuna prática na área de Sistemas de Informação para o agronegócio.

Este trabalho teve por objetivo desenvolver um sistema multiplataforma, com funcionalidades de registro de produtividade, controle de insumos, gestão financeira e relatórios estratégicos de forma centralizada para apoiar a tomada de decisão no campo, objetivo que, em validação técnica realizada em ambiente controlado, mostrou-se alcançável. A aplicação foi capaz de centralizar dados operacionais e apoiar a tomada de decisão.

O tema foi escolhido porque, além da relevância do setor agropecuário nacional, há o interesse de empresas e fazendas em possuir aplicativos integrados para melhor tomada de decisão gerencial com dados consolidados, reduzindo desperdícios e elevando a eficiência operacional.

A implementação técnica utiliza Flutter para o *front-end*, Flask para o *back-end* e MongoDB como banco de dados, priorizando segurança, escalabilidade e acesso multiplataforma. Observa-se, na prática, a dificuldade recorrente de consolidar dados provenientes de rotinas distintas; parte-se, assim, da hipótese de que uma solução digital centralizada pode otimizar esse processo e elevar a qualidade das informações para suporte gerencial. A metodologia contempla prototipação e validação técnica em ambiente controlado, com cenários operacionais simulados representativos da rotina de propriedades rurais, por meio de testes funcionais e de integração para verificar a aderência aos requisitos.

2. REVISÃO BIBLIOGRÁFICA

Esta seção apresenta os principais conceitos relacionados à gestão operacional rural, abordando os processos envolvidos, tecnologias utilizadas e a relevância da transformação digital no campo.

2.1. Gestão Operacional no Agronegócio

A gestão eficiente de propriedades rurais vai além do controle de insumos e fluxos financeiros; ela envolve a liderança de equipes e um planejamento estratégico e adaptação às novas demandas do setor. Segundo Jaramillo (2023), um dos principais desafios do agronegócio é a profissionalização da gestão, garantindo que as operações sejam conduzidas com base em processos estruturados e na análise de dados confiáveis. Nesse sentido, a ausência de um sistema que possua a integração de dados e processos pode resultar em falhas que acarretam recursos mal aproveitados e uma produtividade limitada, comprometendo em longo prazo o faturamento e qualidade das entregas do que é produzido nas fazendas.

2.2. Digitalização no ambiente Agrícola

De acordo com os dados da Embrapa (2023), o número de incubadoras do agro focadas em inovação agrícola aumentou 224%, evidenciando a crescente necessidade do setor em buscar novas soluções que maximizem o potencial produtivo e o avanço constante rumo à digitalização, com a criação de um ambiente tecnológico onde o impacto dos gestores agrícolas seja mais explorado.

Uma fazenda que adota um ambiente digital para sua gestão pode incorporar diversas melhorias operacionais. De acordo com o artigo da Aegro (2022), algumas das principais são:

- Planejamento de safra: Permite um planejamento mais preciso e otimizado das atividades agrícolas, considerando dados climáticos e históricos.
- Controle de estoque através de aplicativos: Facilita o monitoramento em tempo real dos insumos, evitando desperdícios e garantindo reposição eficiente.
- Balanço financeiro e gestão de pessoas automatizados: Integra dados financeiros e de recursos humanos, promovendo uma gestão mais eficiente e controlada.

2.3. Tecnologias e suas utilizações no Aplicativo

As tecnologias escolhidas para o projeto foram estrategicamente selecionadas com base na realidade atual do Brasil. Segundo o IBGE (2024), 87,2% das pessoas com 10 anos ou mais utilizaram a internet em 2023, indicador que sustenta a adoção de soluções digitais para gestão no campo. Embora as taxas em áreas rurais sejam inferiores às urbanas, a oferta de um aplicativo com suporte a dispositivos móveis e operação *offline* amplia o alcance e a utilidade para produtores, facilitando o registro e o acompanhamento das atividades agrícolas em diferentes condições de conectividade.

2.3.1. Utilização do Flutter

Considerando a crescente adoção dos dispositivos móveis, o Flutter foi escolhido para a construção das interfaces utilizadas pelos usuários, devido à sua flexibilidade e capacidade de entregar experiência nativa para diversas plataformas. Segundo Alberto (2022), destaca-se a sua natureza de código aberto e comunidade ativa o que facilita a integração via pacotes prontos,

tornando-o uma solução eficiente e otimizada para a gestão agrícola.

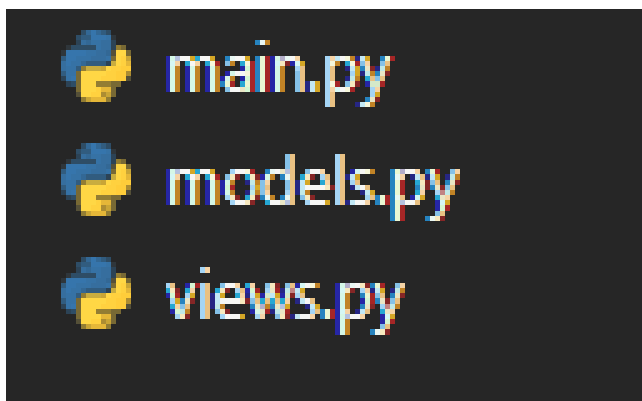
2.3.2. Conceitos de Programação no Flutter

O Flutter é um *framework* desenvolvido com a linguagem de programação *Dart*, que, segundo Alberto (2024), é de tipagem forte e apresenta uma sintaxe familiar, facilitando o aprendizado para quem já tem experiência com outras linguagens mais verbosas. A partir desse conceito, destacam-se dois elementos fundamentais na construção de aplicações: os *Widgets*, que são responsáveis pela criação e estruturação da interface e o gerenciamento de estado (*state management*), que gerencia o estado da aplicação, controlando como os dados são atualizados e refletidos na interface do usuário.

2.3.3. Utilização do Flask no Backend

Segundo a Hashtag Treinamentos (2024), o Flask é um *framework* baseado em Python que permite flexibilidade e escalabilidade, sendo altamente modular. Quando aplicado no desenvolvimento de *APIs RESTful*, em linha com os princípios discutidos por Andrade (2023), o Flask se destaca pela facilidade de construção de suas estruturas, proporcionando uma comunicação eficiente entre o *front-end* e o *back-end*. Sua simplicidade e capacidade de personalização tornam-no uma escolha ideal para a criação de sistemas robustos e ágeis, atendendo de forma eficaz às necessidades de aplicações dinâmicas (Figura 1).

Figura 1. Estrutura de API Flask



Fonte: Hashtag Treinamentos (2024)

2.3.4. Utilização do MongoDB

O MongoDB é um banco de dados não relacional amplamente utilizado em aplicações que exigem escalabilidade e flexibilidade no armazenamento de dados. De acordo com Boaglio (2015), sua arquitetura permite que ele se adapte facilmente a modelos de dados dinâmicos, tornando-o adequado para sistemas que lidam com grandes volumes de dados. Quando integrado ao Flask, o MongoDB facilita a comunicação por meio de bibliotecas como PyMongo, o que simplifica a realização de operações como a adição, remoção e atualização de dados de colaboradores, dados financeiros e informações contábeis nas fazendas.

2.4. Benefícios da centralização digital das informações

A centralização das informações no contexto agrícola significa reunir dados cruciais, como inventário de insumos, registros financeiros e monitoramento de produção em um único sistema digital. Isso facilita o acesso e a análise desses dados, permitindo aos gestores uma visão mais

clara e estratégica das operações. Segundo Alves (2024), a centralização digital não só melhora a organização interna, mas também promove maior precisão na tomada de decisão, ao eliminar a dispersão de informações e reduzir erros humanos.

2.5. Conceitos de Programação com Blueprints

De acordo com Lipi (2018), Blueprints no Flask são um mecanismo de modularização que facilita a separação lógica de funcionalidades em aplicações, permitindo organizar rotas, *views* e demais componentes por domínio de negócio. Cumprindo esse papel na *API*, os *blueprints collaborators*, *machines* e insumos implementam uma *API REST* padronizada com *CORS*, paginação, ordenação e índices de busca para performance em redes móveis, assegurando compatibilidade e estabilidade, além de *endpoint* de monitoramento (*/health*) para verificação de disponibilidade e estabilidade.

2.6. Interface no contexto Rural

Segundo Rodrigues (2023), uma boa interface vai além da estética, oferecendo uma experiência clara e consistente que favorece o uso contínuo. No meio rural, isso implica simplicidade de fluxos, legibilidade e acessibilidade para usuários com menor familiaridade tecnológica e em cenários de baixa conectividade. No aplicativo, optou-se por navegação direta, contraste elevado, botões com área de toque ampliada e ícones acompanhados de rótulos, facilitando o acesso às funções essenciais, como registro de produtividade e controle de insumos. Esses cuidados de design reduzem erros, tornam os fluxos aprendíveis e favorecem a adoção prática da tecnologia, contribuindo para uma gestão rural mais eficiente.

2.6.1. Usabilidade e componentes da interface

Segundo Rangel (2024), o Flutter organiza seus aplicativos por meio de uma arquitetura dividida entre *Embedder*, *Engine*, *Framework* e *Widgets*, sendo estes últimos os blocos fundamentais para a construção de interfaces. Os *widgets* são altamente reutilizáveis e personalizáveis, e permitem criar telas responsivas e interativas com foco em performance e usabilidade.

No sistema, os componentes foram escolhidos para garantir uma navegação fluida, com foco no público rural e suas necessidades práticas. A seguir, são apresentados os principais *widgets* e estruturas utilizadas na construção da interface da aplicação:

- **MaterialApp:** É o *widget* raiz do app, responsável por definir o tema, as rotas principais e os padrões visuais do aplicativo. Ele organiza toda a hierarquia de navegação e aplica consistência à navegação entre as telas (Flutter, s.d.). (Figura 2).

Figura 2. MaterialApp

```

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    const String appTitle = 'Flutter layout demo';
    return MaterialApp(
      title: appTitle,
      home: Scaffold(
        appBar: AppBar(title: const Text(appTitle)),
        body: const Center(
          child: Text('Hello World'),
        ),
      ),
    );
  }
}

```

Fonte: Flutter s.d.)

- Scaffold: É o *widget* que fornece a estrutura principal de cada tela da aplicação. No caso da tela de *Dashboard*, o Scaffold define uma cor de fundo padronizada, inclui uma AppBar personalizada e organiza o conteúdo central por meio de um *CustomScrollView*. Este, por sua vez, utiliza *SliverPadding* e *SliverToBoxAdapter* para garantir o posicionamento adequado dos painéis (Figura 3), conforme a definição de sua estrutura (ou esqueleto) base para aplicativos Material Design (Flutter, s.d).

Figura 3. Estrutura do widget Scaffold

```

6 void main() => runApp(const ScaffoldExampleApp());
7
8 class ScaffoldExampleApp extends StatelessWidget {
9   const ScaffoldExampleApp({super.key});
10
11   @override
12   Widget build(BuildContext context) {
13     return const MaterialApp(home: ScaffoldExample());
14   }
15 }
16
17 class ScaffoldExample extends StatefulWidget {
18   const ScaffoldExample({super.key});
19
20   @override
21   State<ScaffoldExample> createState() =>
22     _ScaffoldExampleState();
23 }

```

Fonte: Flutter s.d.)

TextField: os campos existentes foram implementados com TextField, encapsulados no componente InputField para manter estilo padronizado. O controller gerencia o texto, obscureText oculta a senha e o InputDecoration define rótulos, dicas e bordas com realce

em verde, garantindo consistência visual e melhor usabilidade (Figura 4), seguindo as diretrizes de interface de usuário e layout (Flutter, s.d).

Figura 4. TextField

```

24  @override
25  Widget build(BuildContext context) {
26    return Column(
27      crossAxisAlignment: CrossAxisAlignment.start,
28      children: <Widget>[
29        const Padding(
30          padding: EdgeInsets.symmetric(horizontal: 8, vertical: 16),
31          child: TextField(
32            decoration: InputDecoration(
33              border: OutlineInputBorder(),
34              hintText: 'Enter a search term',
35            ),
36          ),
37        ),

```

Fonte: Flutter s.d.)

- **ElevatedButton:** No aplicativo, o **ElevatedButton** é utilizado como botão de ação principal para a autenticação do usuário. Ele aparece na tela de *login* do sistema com fundo verde, texto branco e bordas levemente arredondadas, destacando-se como elemento central da interface. Essa configuração visual reforça sua função primária (“*Log in*”) e assegura que o botão se integre de forma harmônica com o estilo do sistema, ao mesmo tempo em que guia a navegação do usuário (Figura 5), em conformidade com os princípios de design do *framework* (Flutter, s.d).

Figura 5. ElevatedButton

```

24  @override
25  Widget build(BuildContext context) {
26    return Column(
27      crossAxisAlignment: CrossAxisAlignment.start,
28      children: <Widget>[
29        const Padding(
30          padding: EdgeInsets.symmetric(horizontal: 8, vertical: 16),
31          child: TextField(
32            decoration: InputDecoration(
33              border: OutlineInputBorder(),
34              hintText: 'Enter a search term',
35            ),
36          ),
37        ),

```

Fonte: Flutter s.d.)

- **SliverGrid:** O *SliverGrid* cria uma grade que permite construção de colunas e linhas com responsividade em telas amplas e dispositivos menores. Essa abordagem melhora a legibilidade, otimiza o uso do espaço e proporciona uma navegação mais fluida por meio dos cartões de métricas e navegação (Figura 6), sendo uma peça-chave na construção de layouts adaptáveis do *framework* (Flutter, s.d).

Figura 6. SliverGrid

```
SliverGrid(
  gridDelegate: const SliverGridDelegateWithMaxCrossAxisExtent(
    maxCrossAxisExtent: 200.0,
    mainAxisSpacing: 10.0,
    crossAxisSpacing: 10.0,
    childAspectRatio: 4.0,
  ),
  delegate: SliverChildBuilderDelegate(
    (BuildContext context, int index) {
      return Container(
        alignment: Alignment.center,
        color: Colors.teal[100 * (index % 9)],
        child: Text('grid item $index'),
      );
    },
    childCount: 20,
  ),
)
```

Fonte: Flutter s.d.)

- Container: É essencial em um app de gestão agrônômica, pois permite a personalização visual e organização da *interface*. Ele é usado para aplicar margens, bordas, sombras e fundo com imagens ou gradientes, ajudando a criar layouts limpos e intuitivos. Esse *widget* contribui para a clareza na apresentação de informações, como dados de produtividade ou áreas de cultivo, melhorando a usabilidade e a navegação no app (Figura 7), sendo uma ferramenta primária de composição visual do *framework* (Flutter, s. d).

Figura 7. Container

```
final stars = Row(
  mainAxisAlignment: MainAxisAlignment.min,
  children: [
    Icon(Icons.star, color: Colors.green[500]),
    Icon(Icons.star, color: Colors.green[500]),
    Icon(Icons.star, color: Colors.green[500]),
    const Icon(Icons.star, color: Colors.black),
    const Icon(Icons.star, color: Colors.black),
  ],
);

final ratings = Container(
  padding: const EdgeInsets.all(20),
  child: Row(
    mainAxisAlignment: MainAxisAlignment.spaceEvenly,
    children: [
      stars,
      const Text(
        '170 Reviews',
        style: TextStyle(
          color: Colors.black,
          fontWeight: FontWeight.w800,
          fontFamily: 'Roboto',
          letterSpacing: 0.5,
          fontSize: 20,
        ),
      ),
    ],
  ),
);
```

Fonte: Flutter s.d.)

3. DESENVOLVIMENTO

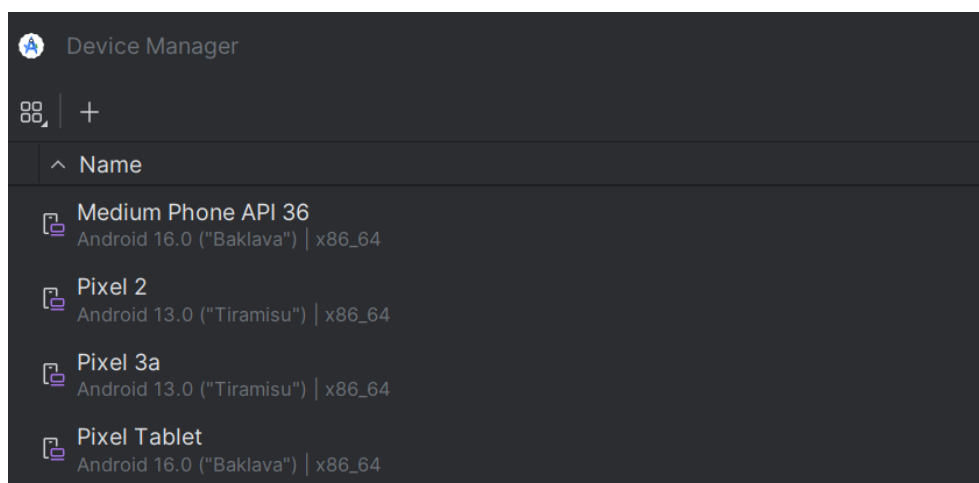
Nesta seção, são apresentados os objetivos, a metodologia e o desenvolvimento da criação do sistema, desde a sua programação até a estrutura final da aplicação desenvolvida. Aqui, observamos as etapas descritas de configuração do ambiente de desenvolvimento, definição da arquitetura, escolha dos principais componentes visuais e funcionais, bem como as decisões técnicas que orientaram a usabilidade da solução.

3.1. Configuração do Ambiente

O desenvolvimento do sistema foi realizado utilizando o Visual Studio Code (VS Code), um editor de código-fonte leve, gratuito e altamente personalizável, amplamente adotado por desenvolvedores Flutter, sendo descrito como um editor de código aberto que, segundo Providello (2023), pode ser baixado na página oficial.

Além do VS Code, o ambiente foi configurado com o Android SDK, necessário para compilar e executar o aplicativo em dispositivos Android. Utilizou-se o Android Emulator – Pixel 3a (API 34), configurado diretamente pelo Android Studio, mas controlado via VS Code, para testes de responsividade e visualização em tempo real. O terminal integrado foi amplamente utilizado para rodar comandos Flutter, executar *builds*, realizar *debug* e gerenciar o controle de versão com Git, garantindo um fluxo de trabalho centralizado e produtivo (Figura 8).

Figura 8. Android Device Manager

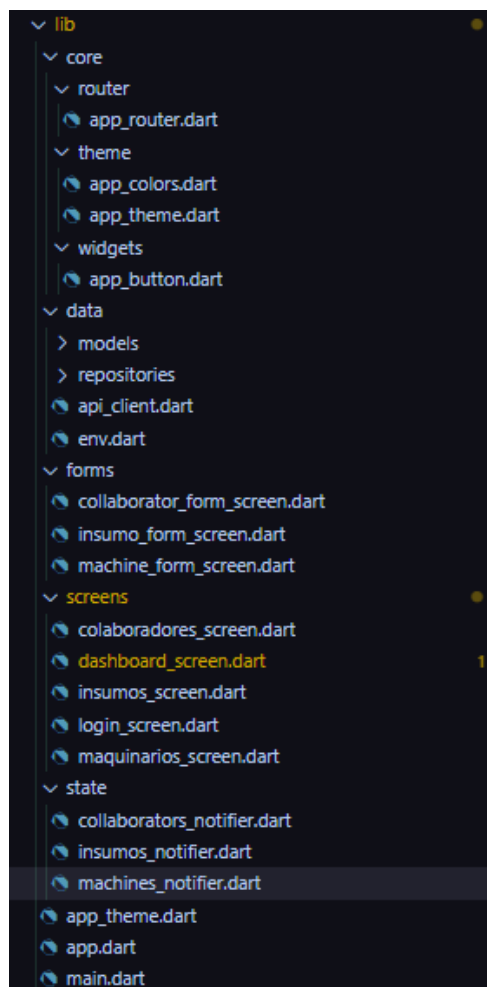


Fonte: O Autor, 2025

3.2. Estrutura do Projeto

A estrutura do projeto foi organizada dentro da pasta *lib/*, onde todo o código-fonte da aplicação está centralizado. Foram criadas subpastas como *core*, *data*, *forms*, *screens* e *state*, garantindo uma separação clara entre dados, formulários, telas e gerenciamento de estado.

O arquivo *main.dart* é o ponto de entrada da aplicação, responsável por inicializar o app e delegar temas/rotas aos respectivos módulos. A arquitetura adotada facilita a manutenção do código, permitindo que os *widgets* sejam reutilizados e que a lógica de estado, feita com *Provider* e *ChangeNotifier*, mantenha a interface sempre sincronizada com os dados do sistema (Figura 9).

Figura 9. Estrutura dos arquivos do projeto

Fonte: O Autor, 2025

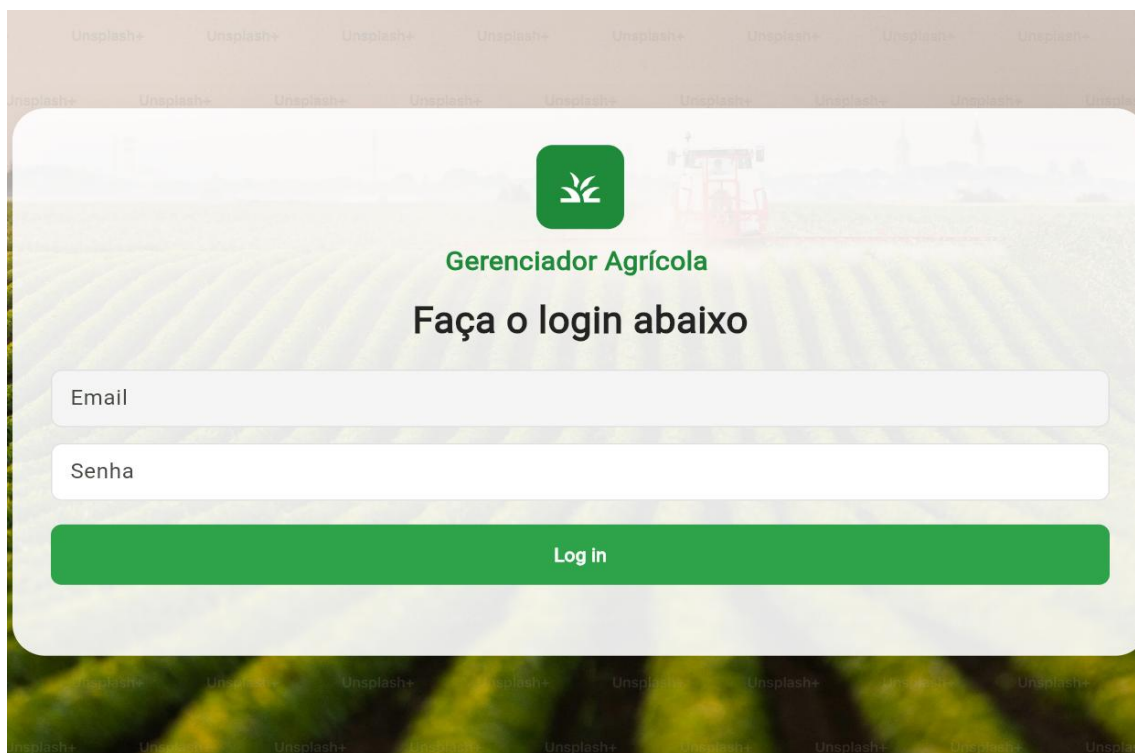
3.3. Funcionalidades Implementadas

O sistema desenvolvido contempla funcionalidades essenciais para a gestão agrícola, estruturadas de forma modular para garantir praticidade, clareza e adaptação ao dia a dia do produtor. As funcionalidades estão organizadas em três frentes principais: autenticação, painel de controle e cadastros. A autenticação é realizada por meio de uma tela de *login* intuitiva, que direciona o usuário para o *dashboard* principal. No *dashboard*, o usuário encontra atalhos visuais e modernos para os módulos de Insumos, Maquinários e Colaboradores.

3.3.1. Tela de *Login* e Autenticação

A tela de *login* utiliza campos validados de e-mail e senha, com um *layout* limpo e objetivo, sendo os campos implementados com o *TextField* seguidos do botão principal que leva o usuário a tela inicial, na qual o próprio foi implementado com a utilização de um *ElevatedButton* (Figura 10).

Figura 10. Tela de Login

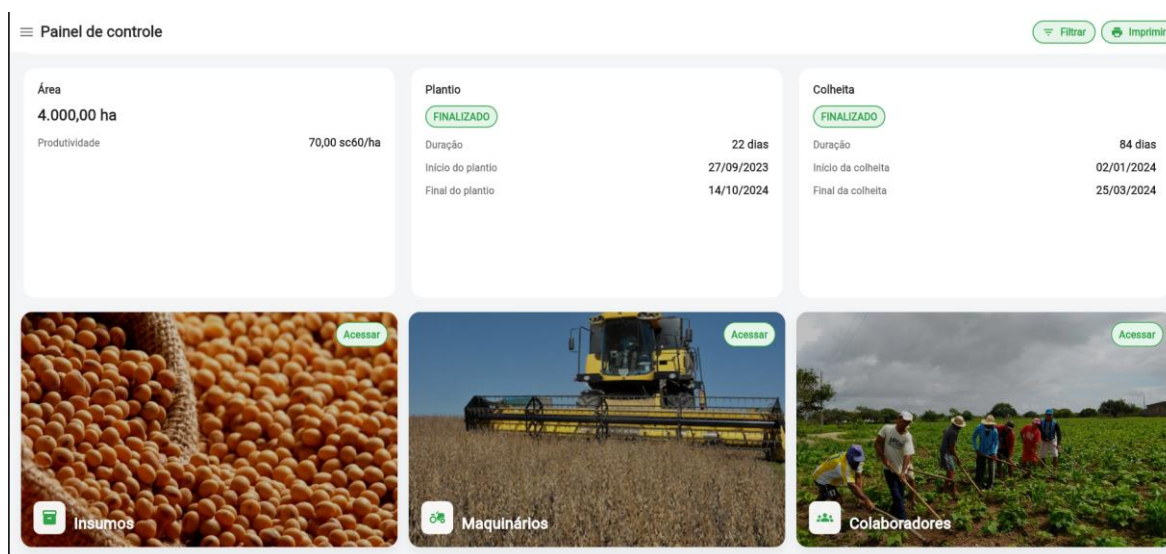


Fonte: O Autor, 2025

3.3.2. Dashboard de Acesso

A tela principal exibe métricas gerais como área cultivada, produtividade, *status* do plantio e da colheita, apresentadas em cartões de fácil leitura e disponibiliza atalhos visuais para os módulos principais de utilização como Insumos, Maquinários e Colaboradores, que por meio de cartões ilustrados que reforçam a identidade do sistema e facilitam a interação (Figura 11).

Figura 11. Dashboard de Acesso



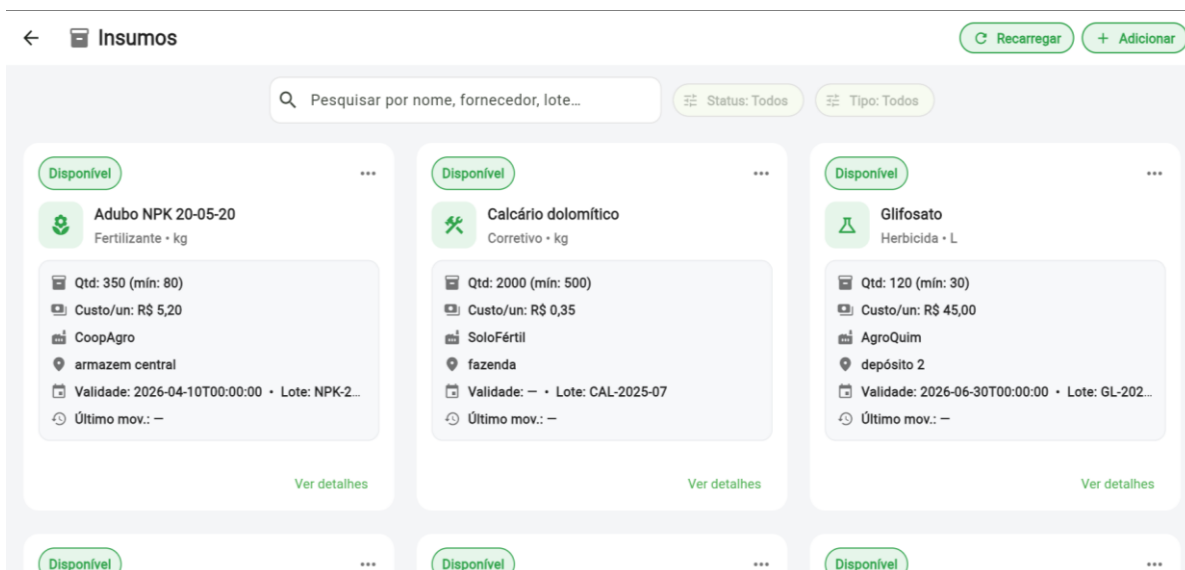
Fonte: O Autor, 2025

3.3.3. Tela de Insumos

A tela de insumos tem como sua finalidade gerenciar recursos agrícolas utilizados nas produções, como fertilizantes, corretivos e herbicidas. Nela, o usuário pode consultar, filtrar e

cadastrar novos insumos, acompanhando informações detalhadas de estoque, custo e validade. Esse controle é necessário para evitar a má gestão dos recursos, e manter a rastreabilidade de cada item (Figura 12).

Figura 12. Tela de insumos

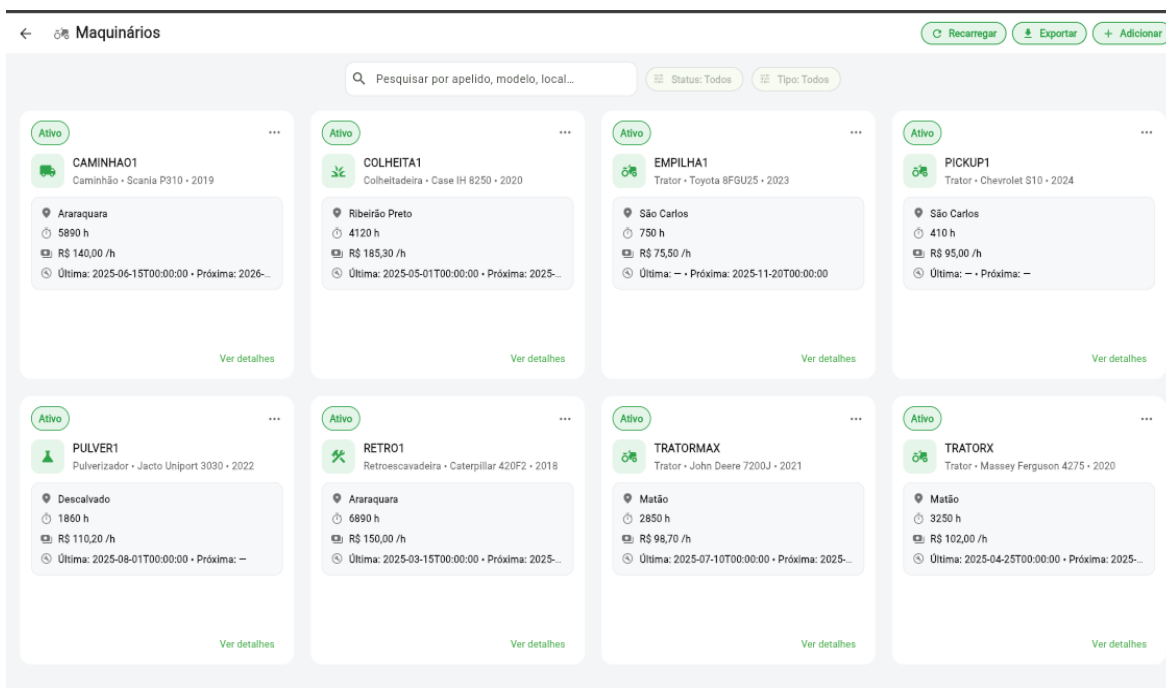


Fonte: O Autor, 2025

3.3.4. Tela de Maquinários

A tela de Maquinários centraliza a gestão dos veículos e equipamentos utilizados nas operações agrícolas, permitindo ao usuário acompanhar de forma clara informações essenciais sobre cada máquina. A interface também possibilita o registro e consulta de manutenções realizadas e previstas, a pesquisa por apelido, modelo ou local, bem como o uso de filtros por tipo ou *status* (Figura 13).

Figura 13. Tela de maquinários

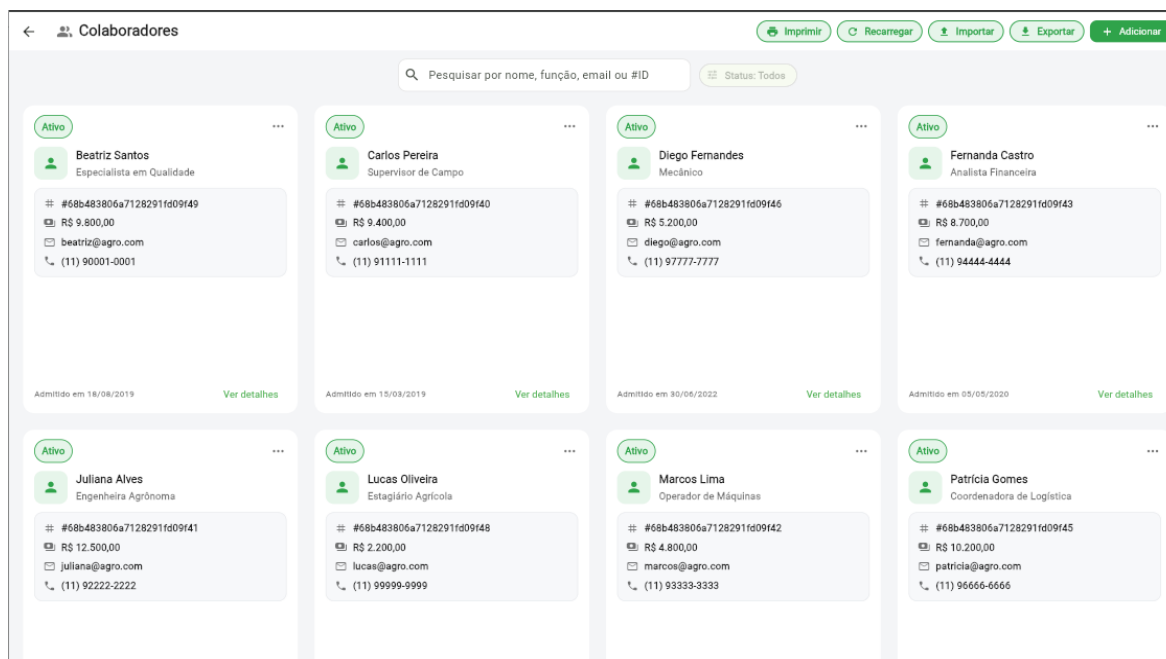


Fonte: O Autor, 2025

3.3.5. Tela de Colaboradores

A tela de Colaboradores fornece a gestão da equipe envolvida nas atividades agrícolas, permitindo o cadastro, consulta e acompanhamento de informações essenciais sobre cada funcionário. Entre suas funcionalidades estão a exibição de dados pessoais e profissionais, como nome, função, e-mail, telefone, salário e data de admissão, além do *status* de atividade (ativo ou inativo). A interface oferece recursos de busca e filtros para facilitar a localização de colaboradores por nome, função, e-mail ou identificador, bem como opções para importar, exportar, recarregar ou adicionar novos registros (Figura 14).

Figura 14. Tela de colaboradores

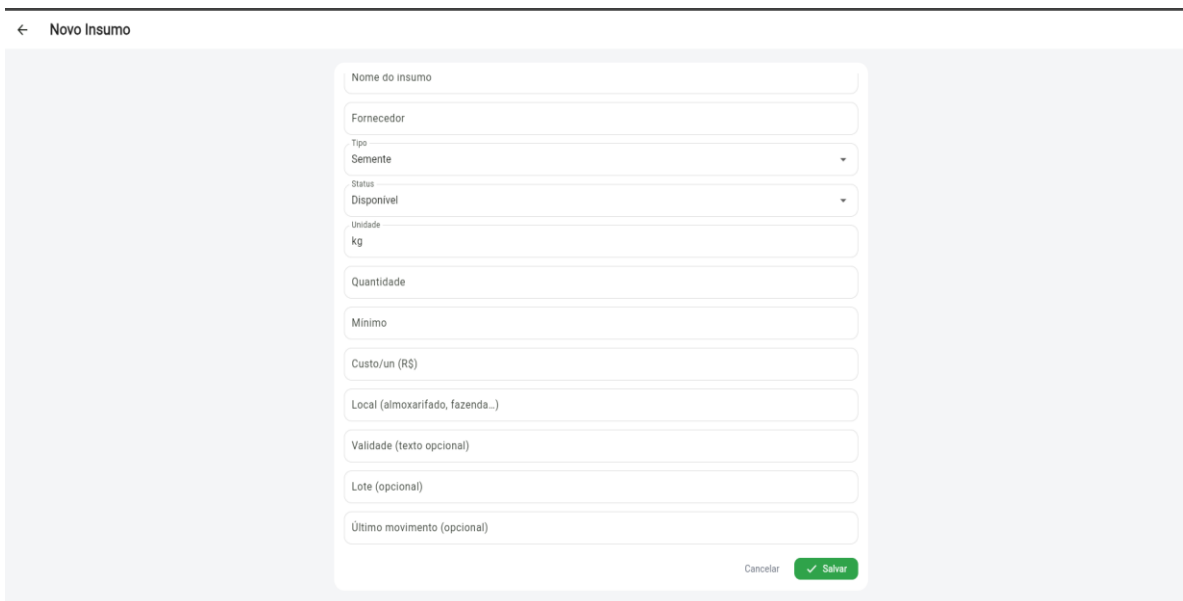


Fonte: O Autor, 2025

3.4. Formulário de Insumos

O módulo de gestão de insumos oferece o controle completo dos produtos agrícolas utilizados nas operações, permitindo o cadastro, consulta e acompanhamento de informações essenciais de cada item. Suas funcionalidades incluem a exibição de dados como tipo, unidade de medida, quantidade disponível, custo unitário, fornecedor, validade e lote. Adicionalmente, o sistema possibilita o controle do *status* dos insumos (ativo ou inativo) (Figura 15).

Figura 15. Formulário de insumos



← Novo Insumo

Nome do insumo

Fornecedor

Tipo
Semente

Status
Disponível

Unidade
kg

Quantidade

Mínimo

Custo/un (R\$)

Local (almoxarifado, fazenda...)

Validade (texto opcional)

Lote (opcional)

Último movimento (opcional)

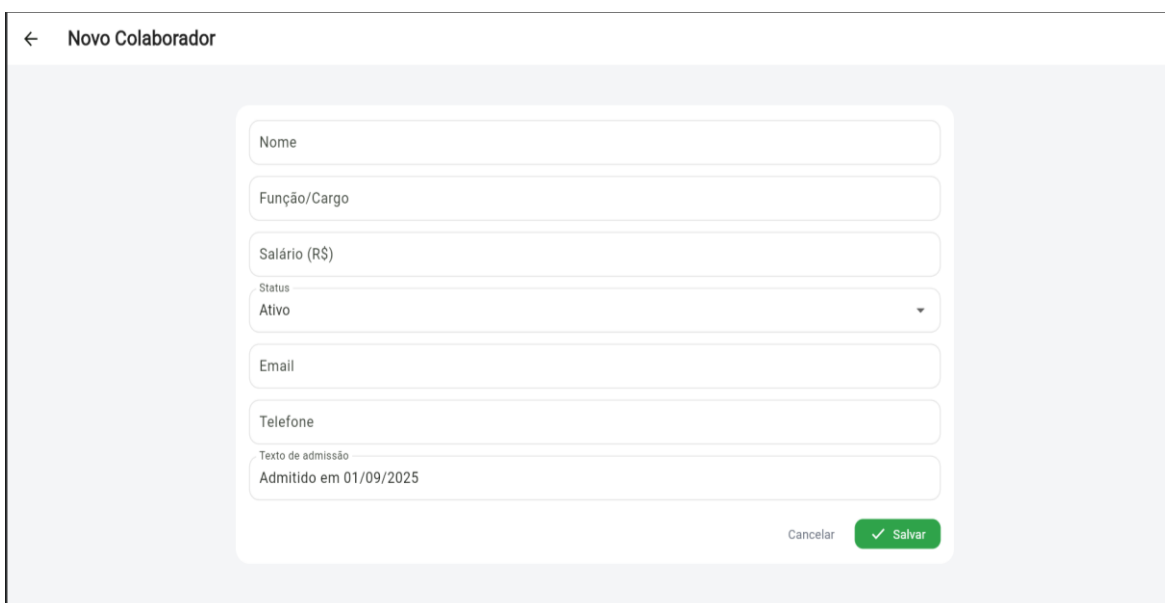
Cancelar Salvar

Fonte: O Autor, 2025

3.4.1. Formulário de Colaboradores

O módulo de gestão de colaboradores fornece as ferramentas para o gerenciamento completo da equipe, permitindo o cadastro, a consulta e o acompanhamento das informações de cada funcionário. As funcionalidades do sistema contemplam o registro de dados profissionais e de contato, como nome completo, função/cargo, salário, e-mail e telefone. Adicionalmente, o módulo controla a data de admissão e o *status* de atividade do colaborador na empresa (ativo ou inativo). A fim de otimizar a administração da equipe, a interface oferece recursos de busca e filtros para localizar registros por nome ou cargo, além de permitir a adição de novos colaboradores (Figura 16).

Figura 16. Formulário de colaboradores



← Novo Colaborador

Nome

Função/Cargo

Salário (R\$)

Status
Ativo

Email

Telefone

Texto de admissão
Admitido em 01/09/2025

Cancelar Salvar

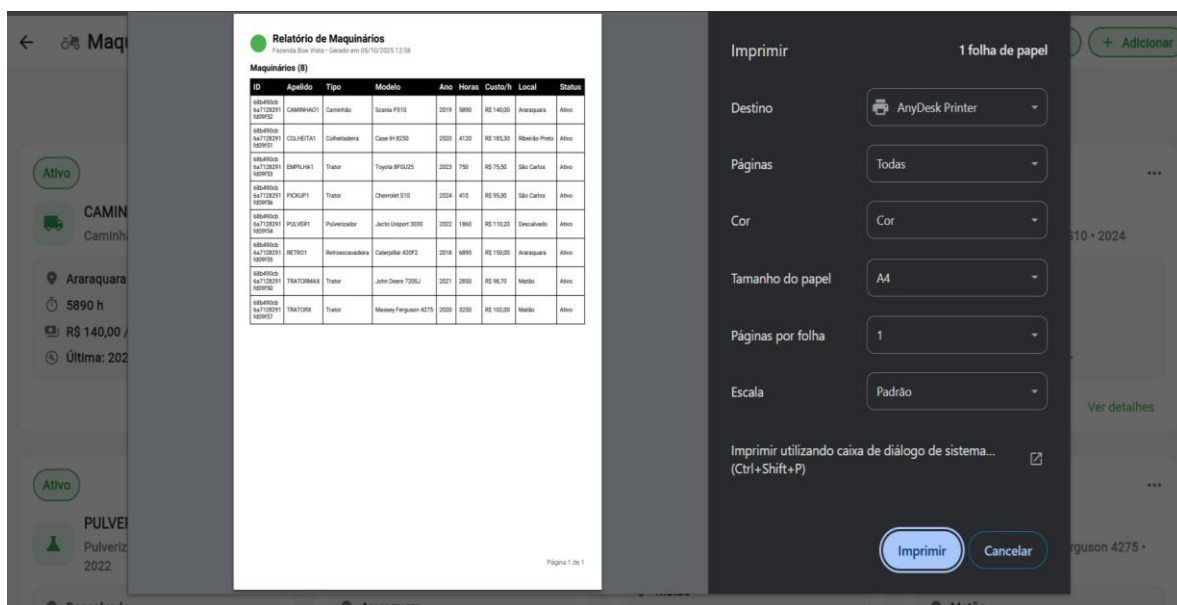
Fonte: O Autor, 2025

3.5. Função de exportar PDF para Maquinários

A funcionalidade de exportação para PDF do módulo de maquinários permite gerar

relatórios padronizados contendo informações como ID, apelido, tipo, modelo, ano, horas trabalhadas, custo por hora, local e *status*. Desenvolvido com a biblioteca *Printing* do Flutter, o sistema converte automaticamente os dados em um documento digital organizado e pronto para impressão. O recurso assegura padronização visual, clareza e integridade das informações, auxiliando no controle e na gestão da frota agrícola (Figura 17).

Figura 17. Função de exportar relatórios de maquinários

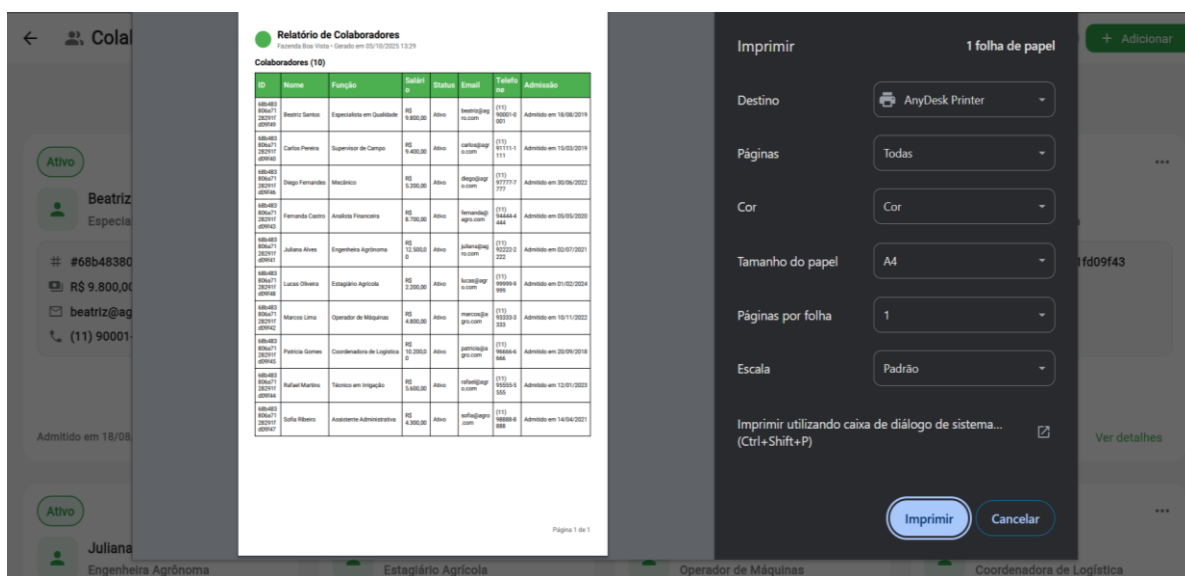


Fonte: O Autor, 2025

3.5.1. Função de exportar PDF para colaboradores

A funcionalidade de exportação para PDF do módulo de colaboradores gera relatórios padronizados com informações como ID, nome, cargo, salário, *status*, e-mail, telefone e data de admissão. Utilizando a biblioteca *Printing* do Flutter, o sistema converte automaticamente os dados exibidos em um documento digital formatado e pronto para impressão. O recurso garante padronização visual, clareza e integridade das informações, facilitando a gestão e o arquivamento dos registros (Figura 18).

Figura 18. Função de exportar relatórios de colaboradores

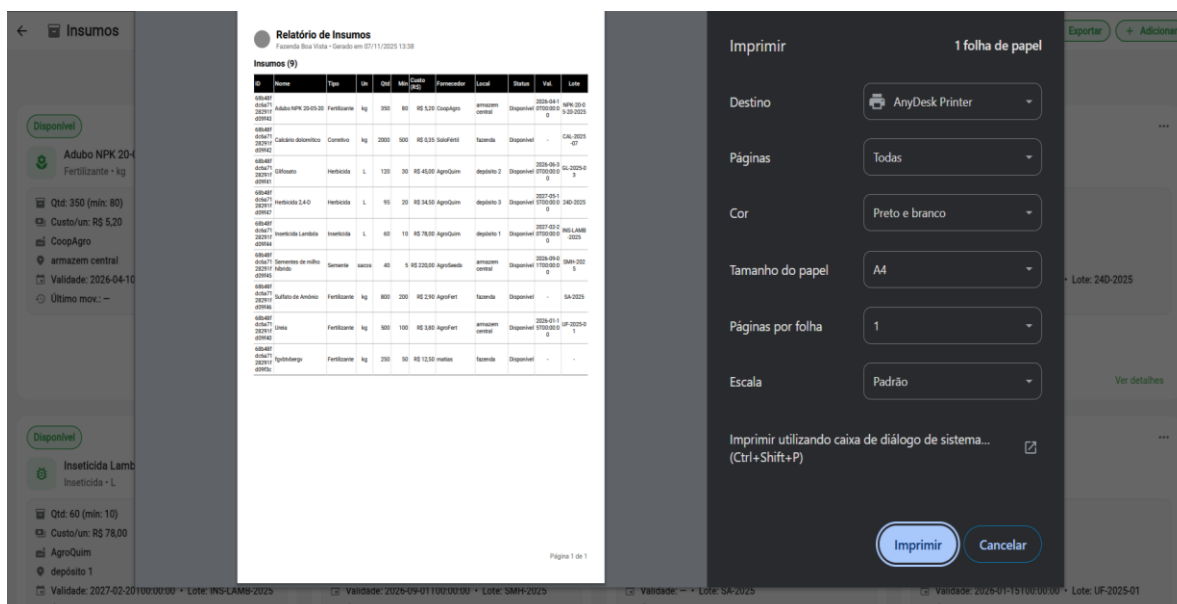


Fonte: O Autor, 2025

3.5.2. Função de exportar PDF para insumos

A funcionalidade de exportação para PDF do módulo de Insumos gera relatórios padronizados com campos como ID, nome, tipo, unidade, quantidade, mínimo, custo unitário (R\$), fornecedor, local, *status*, validade e lote. Utilizando a biblioteca *Printing* do Flutter, o sistema converte automaticamente os dados de estoque em um documento digital formatado e pronto para impressão, preservando padronização visual, clareza e integridade das informações. O recurso facilita o controle de estoque, rastreabilidade por lote/validade e o arquivamento dos registros, apoiando auditorias e a gestão operacional do almoxarifado (Figura 19).

Figura 19. Função de exportar relatórios de insumos



Fonte: O Autor, 2025

4. CONSIDERAÇÕES

Este trabalho apresentou o desenvolvimento de uma plataforma integrada para gestão e monitoramento no agronegócio, com o objetivo de centralizar informações e otimizar as operações rurais. Ao consolidar dados de diferentes frentes operacionais em um único ambiente, o estudo atingiu o objetivo proposto de apoiar a decisão gerencial por meio de informações organizadas e acessíveis.

As funcionalidades implementadas, como o controle de insumos, maquinários e colaboradores, além da geração de relatórios em PDF, mostraram-se eficazes para organizar processos e contribuir com a tomada de decisões. Observou-se maior controle sobre recursos, indícios de redução de desperdícios e melhoria da produtividade operacional, a partir da padronização de registros e do acompanhamento sistemático das rotinas.

Com a centralização digital das informações, os gestores rurais passam a contar com dados mais precisos e disponíveis, favorecendo uma administração contemporânea e sustentável, com uso racional de recursos. A aplicação evidencia o papel da tecnologia na evolução do campo e aproxima a realidade agrícola de práticas de gestão alinhadas a melhores procedimentos.

Como limitações, destacam-se o recorte do contexto de avaliação e a ausência de mensurações quantitativas de longo prazo. Há a recomendação em trabalhos futuros, a adoção de métricas objetivas (por exemplo, ROI, taxa de perdas/desperdícios) e a ampliação do escopo de validação em diferentes perfis de propriedade. A solução mostra-se aplicável em cenários com

conectividade mínima e equipe treinada para registro padronizado dos dados.

Conclui-se que a digitalização é essencial para a competitividade do agronegócio. A plataforma desenvolvida cumpre seu papel ao integrar tecnologia e gestão, servindo como base para futuras expansões, como novos módulos e recursos de inteligência artificial voltados à análise preditiva e eficiência operacional.

REFERÊNCIAS

AEGRO. Fazendas digitais: como a tecnologia pode revolucionar a gestão agrícola. **Aegro**, 2022. Disponível em: <https://blog.aegro.com.br/fazendas-digitais>. Acesso em: 28 mar. 2025.

ALBERTO, Matheus. Dart: O que é? Como começar a estudar? Para quê serve? **Alura**, 2024. Disponível em: <https://www.alura.com.br/artigos/dart>. Acesso em: 28 mar. 2025.

ALBERTO, Matheus. Flutter: o que é e tudo sobre o framework. **Alura**, 6 set. 2022. Disponível em: <https://www.alura.com.br/artigos/flutter>. Acesso em: 28 mar. 2025.

ALVES, Lynn; LOPES, David (org.). **Educação e plataformas digitais**: popularizando saberes, potencialidades e controvérsia. Salvador: EDUFBA, 2024. Disponível em: <https://repositorio.ufba.br/bitstream/ri/39372/3/Educa%C3%A7%C3%A3o%20e%20plataformas%20digitais-digital.pdf>. Acesso em: 5 out. 2025.

ANDRADE, Emmanuel. Entendendo as diferenças entre APIs REST e RESTful. **DIO**, 2023. Disponível em: <https://www.dio.me/articles/entendendo-as-diferencas-entre-apis-rest-e-restful>. Acesso em: 5 out. 2025.

BATALHA, Mário Otávio (coord.). **Gestão agroindustrial**: volume único. 4. ed. São Paulo: Atlas, 2021.

BOAGLIO, Fernando. **MongoDB**: Construa novas aplicações com novas tecnologias. São Paulo: Casa do Código, 2015.

EMBRAPA. **Número de incubadoras do agro cresceu 224% e de aceleradoras 90%, registra o Radar AgTech Brasil**. [S. l.]: Embrapa, 2023. Disponível em: <https://www.embrapa.br/cultivos/busca-de-noticias/-/noticia/99266318/numero-de-incubadoras-do-agro-cresceu-224-e-de-aceleradoras-90-registra-o-radar-agtech-brasil>. Acesso em: 28 mar. 2025.

FLUTTER. **Biblioteca de Widgets**. [S. l.: s. n.], s. d. Disponível em: <https://api.flutter.dev/flutter/widgets>. Acesso em: 6 out. 2025.

FLUTTER. **Layouts no Flutter**. [S. l.: s. n.], s. d. Disponível em: <https://docs.flutter.dev/ui>. Acesso em: 6 out. 2025.

HASHTAG TREINAMENTOS. O que é Flask no Python: Micro-Framework de Desenvolvimento Web no Python. **Hashtag Treinamentos**, 2024. Disponível em: <https://www.hashtagtreinamentos.com/o-que-e-flask-python>. Acesso em: 28 mar. 2025.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA (IBGE). Em 2023, 87,2% das pessoas com 10 anos ou mais utilizaram internet. **Agência de Notícias IBGE**, 2024. Disponível em: <https://agenciadenoticias.ibge.gov.br/agencia-noticias/2012-agencia-de-noticias/noticias/41026-em-2023-87-2-das-pessoas-com-10-anos-ou-mais-utilizaram-internet>. Acesso em: 28 mar. 2025.

JARAMILLO, Santiago Franco. **Liderança e Gestão de Pessoas no Agronegócio**: Como a Gestão Focada em Pessoas Pode Trazer Resultados Extraordinários Para sua Empresa. Rio de Janeiro: Alta Books Editora, 2023.

LIPÍ, Renato. Flask Blueprints em miúdos. **Training Center – Medium**, 16 fev. 2018. Disponível em: <https://medium.com/trainingcenter/flask-blueprints-em-miúdos-41c1da8e020a>. Acesso em: 5 out. 2025.

PROVIDELLO, Walakys. Visual Studio Code: a importância de uma poderosa ferramenta de desenvolvimento. **DIO**, 23 jul. 2023. Disponível em: <https://www.dio.me/articles/visual-studio-code-a-importancia-de-uma-poderosa-ferramenta-de-desenvolvimento>. Acesso em: 5 out. 2025.

RANGEL, Marcos. Flutter Framework e Arquitetura. **DIO**, 28 mar. 2024. Disponível em: <https://www.dio.me/articles/flutter-framework-e-arquitetura>. Acesso em: 5 out. 2025.

REVISTA CULTIVAR. Desafios da tecnologia digital no agro. **Revista Cultivar**, 15 jan. 2021. Disponível em: <https://revistacultivar.com.br/noticias/desafios-da-tecnologia-digital-no-agro>. Acesso em: 6 out. 2025.

RODRIGUES, Kauê. Guia para interface do usuário: Aprimorando a Experiência do Usuário. **Blog Cubos Academy**, 2023. Disponível em: <https://blog.cubos.academy/design-guia-para-interface-do-usuario>. Acesso em: 5 out. 2025.