CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA “PAULA SOUZA”
FACULDADE DE TECNOLOGIA DE BEBEDOURO
TECNOLOGIA EM BIG DATA NO AGRONEGÓCIO

SISTEMA DIGITAL DE ESCRITURAÇÃO RURAL: UMA FERRAMENTA PARA GESTÃO E ANÁLISE DOCUMENTAL DO PRODUTOR RURAL

AUTOR: ROGÉRIO RICARDO DOS SANTOS TOLEDO

ORIENTADOR: PROF. ME. PAULO EDUARDO CARDOSO ANDRADE

BEBEDOURO

2025

ROGÉRIO RICARDO DOS SANTOS TOLEDO

SISTEMA DIGITAL DE ESCRITURAÇÃO RURAL: UMA FERRAMENTA PARA GESTÃO E ANÁLISE DOCUMENTAL DO PRODUTOR RURAL

Monografia apresentada à Faculdade de Tecnologia de Bebedouro, como parte dos requisitos para a obtenção do título de Tecnólogo em Big Data no Agronegócio.

Orientador: Prof. Me. Paulo Eduardo Cardoso Andrade

BEBEDOURO

2025

TOLEDO, ROGERIO. **Sistema Digital de Escrituração Rural: Uma Ferramenta Para Gestão e Análise Documental do Produtor Rural.** Trabalho de Graduação (Monografia). Centro Estadual de Educação Tecnológica "Paula Souza". Faculdade de Tecnologia de Bebedouro. 88 p. 2025.

RESUMO

O presente trabalho desenvolve um Sistema Digital de Escrituração Rural orientado ao processamento, organização e análise de dados financeiros provenientes dos extratos bancários do produtor rural. A ferramenta foi construída em Python, utilizando bibliotecas para extração de informações estruturadas a partir de documentos PDF, transformações e normalização de dados, com interface interativa desenvolvida em Streamlit e armazenamento em banco de dados MySQL. O sistema se fundamenta na lógica de um pipeline de dados que compreende a coleta (ingestão de extratos bancários), a transformação (tratamento, padronização e classificação), a carga (persistência no banco) e a disponibilização (relatórios e visualização estruturada). Esse fluxo reduz significativamente o retrabalho manual e as inconsistências típicas dos processos tradicionais, permitindo obter dados confiáveis, limpos e prontos para análises contábeis, gerenciais e fiscais. A solução respeita o modelo clássico de livro-caixa rural, preservando sua metodologia, mas eleva o processo à lógica de um sistema analítico, capaz de subsidiar decisões e aprimorar a gestão operacional. Os resultados mostram ganhos expressivos em eficiência, rastreabilidade e organização documental, evidenciando como técnicas de automação, extração de dados e arquitetura modular pode modernizar atividades de escrituração financeira e contábil rural, apoiando o profissional na preparação das informações destinadas ao fisco, como na elaboração das informações para a Declaração do Imposto de Renda Pessoa Física (DIRPF). O sistema também permite evoluções futuras, como integração com ferramentas de Business Intelligence, uso de OCR para leitura de PDFs escaneados e aplicação de modelos preditivos, ampliando o uso estratégico de dados na gestão rural. O trabalho corrobora a convergência entre tecnologia, agronegócio e contabilidade, apresentando uma solução prática e acessível que demonstra como a área de Big Data pode transformar rotinas tradicionais.

Palavras-chave: *Sistemas de informação no agronegócio. Automação de processos. Processamento de dados.*

TOLEDO, ROGERIO. **Sistema Digital de Escrituração Rural: Uma Ferramenta Para Gestão e Análise Documental do Produtor Rural.** Trabalho de Graduação (Monografia). Centro Estadual de Educação Tecnológica “Paula Souza”. Faculdade de Tecnologia de Bebedouro. 88 p. 2025.

ABSTRACT

This study develops a Digital Rural Bookkeeping System designed for the processing, organization, and analysis of financial data extracted from rural producers’ bank statements. The tool was built in Python, using libraries for structured data extraction from PDF documents, data transformation, and normalization, with an interactive interface developed in Streamlit and storage in a MySQL database. The system is based on a data-pipeline architecture comprising ingestion (import of bank statements), transformation (cleaning, standardization, and classification), loading (database persistence), and delivery (reports and structured visualization). This workflow significantly reduces manual rework and the inconsistencies typically found in traditional bookkeeping processes, enabling reliable and clean data suitable for accounting, managerial, and tax analyses. Although it preserves the methodology of the classical rural cash-book model, the solution elevates the process to an analytical system capable of supporting decision-making and improving operational management. The results demonstrate substantial gains in efficiency, traceability, and document organization, showing how automation techniques, data extraction, and modular architecture can modernize rural financial and accounting bookkeeping activities, assisting professionals in preparing information required by tax authorities, such as the Individual Income Tax Return (DIRPF). The system also enables future enhancements, including integration with Business Intelligence tools, the use of OCR for scanned PDFs, and the application of predictive models, expanding the strategic use of data in rural management. The study reinforces the convergence between technology, agribusiness, and accounting by presenting a practical and accessible solution that illustrates how Big Data can transform traditional routines.

Keywords: Information systems in agribusiness. Process automation. Data processing.

SUMÁRIO

1 INTRODUÇÃO

A atividade rural permanece como um dos pilares da economia brasileira, figurando entre os setores mais dinâmicos e estratégicos do país. Segundo o Centro de Estudos Avançados em Economia Aplicada (CEPEA, 2025), o Produto Interno Bruto (PIB) do agronegócio cresceu 6,49% no primeiro trimestre de 2025, com projeções de alcançar 29,4% do PIB nacional ao final do ano, contra 23,5% em 2024.

Apesar desse desempenho expressivo, pequenos e médios produtores ainda enfrentam dificuldades na organização e controle das informações financeiras de suas atividades. Essa deficiência compromete a gestão, dificulta o acompanhamento econômico e impacta diretamente o cumprimento de obrigações fiscais, como a Declaração de Ajuste Anual do Imposto de Renda da Pessoa Física (IRPF) (RECEITA FEDERAL, 2024).

O avanço dos sistemas de informação tem modernizado o agronegócio, proporcionando maior eficiência na centralização de dados e na automação de rotinas (TURBAN et al., 2018). Entretanto, estudos da EMBRAPA (2021) mostram que grande parte dos produtores ainda utiliza métodos manuais, cadernos, anotações soltas e planilhas básicas, prática que limita a padronização e reduz a confiabilidade dos registros. A falta de ferramentas acessíveis e adequadas à realidade do campo evidencia a necessidade de soluções simples e integradas.

No âmbito da contabilidade aplicada ao produtor rural pessoa física, a escrituração financeira baseada em extratos bancários é uma das rotinas mais críticas, exigindo organização, precisão e grande volume de digitação (RECEITA FEDERAL, 2024). Esses desafios se intensificam no período que antecede o prazo do IRPF, quando muitos extratos chegam ao contador de forma tardia e despadronizada, ampliando riscos de inconsistência e retrabalho (SANTOS, 2023).

Nesse contexto, torna-se relevante o desenvolvimento de ferramentas que auxiliem o profissional contábil no tratamento e padronização dessas informações. Diferente de sistemas rurais robustos, muitas vezes complexos ou de custo elevado, a solução proposta neste trabalho busca simplicidade e eficiência, automatizando a leitura de extratos bancários em PDF, estruturando os dados e gerando relatórios compatíveis com a escrituração interna do produtor rural.

O sistema desenvolvido não substitui obrigações fiscais oficiais, mas atua como apoio operacional, permitindo a classificação e o armazenamento de informações em banco de dados

estruturado. Com isso, contribui para reduzir o tempo de processamento, aumentar a confiabilidade dos registros e melhorar a organização financeira do produtor rural.

A motivação deste estudo surge da experiência prática em escritório contábil e empresas no ramo do agronegócio, onde a digitação manual de extratos é frequentemente identificada como uma tarefa repetitiva, demorada e suscetível a falhas. A automação desse processo representa ganho de eficiência, redução de custos e maior segurança nas informações.

1.1 Objetivo Geral

Desenvolver um Sistema Digital de Escrituração Rural capaz de automatizar a leitura, tratamento, padronização, classificação e armazenamento das informações financeiras provenientes de extratos bancários, oferecendo suporte ao processo de escrituração interna do produtor rural pessoa física e facilitando a preparação das informações destinadas ao IRPF.

1.2 Objetivos Específicos

1. Automatizar a leitura de extratos bancários digitalizados em PDF, extraindo dados estruturados como data, histórico, documento e valor.
2. Padronizar e organizar as informações financeiras, reduzindo inconsistências e retrabalho no processo de escrituração contábil.
3. Classificar receitas, despesas, custos e investimentos de acordo com a estrutura da atividade rural e com as categorias utilizadas na apuração do resultado rural.
4. Armazenar as informações tratadas em um banco de dados MySQL, garantindo segurança, integridade e rastreabilidade.
5. Gerar relatórios operacionais e gerenciais que auxiliem o contador na revisão, conferência e preparação dos dados para a Declaração de Ajuste Anual do IRPF.
6. Propor uma solução acessível, prática e alinhada à realidade de pequenos e médios produtores rurais que necessitam de maior organização financeira.

1.3 Estrutura do Trabalho

Para atender a esses objetivos, este trabalho está estruturado da seguinte forma: o Capítulo 1 apresenta a introdução, contextualizando a relevância do agronegócio, o problema enfrentado na escrituração financeira e a justificativa do estudo. O Capítulo 2 reúne a fundamentação teórica, abordando os temas de gestão documental, escrituração de dados rural, sistemas de informação e

digitalização no agronegócio. O Capítulo 3 descreve a metodologia utilizada no desenvolvimento do sistema. O Capítulo 4 detalha o processo de construção da ferramenta, incluindo leitura dos extratos, tratamento dos dados, estrutura do banco e geração de relatórios. O Capítulo 5 apresenta os resultados e discussões obtidos com a aplicação do sistema. Por fim, o Capítulo 6 traz as conclusões e considerações finais, destacando as contribuições do estudo e possíveis desdobramentos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

A fundamentação teórica deste trabalho tem por objetivo apresentar os conceitos que sustentam o desenvolvimento de sistemas digitais aplicados à escrituração e a gestão documental no meio rural. Esse contexto é caracterizado pelo elevado volume de documentos gerados ao longo da atividade produtiva, pelas exigênciafiscais cada vez mais rigorosas e pela necessidade de rapidez e confiabilidade no processo de tomada de decisão por parte do produtor rural, fatores que demandam informações organizadas, tradas e confiáveis.

Nesse sentido, este capitulo aborda a importância da gestão documental, o papel do livro caixa na atividade rural, a automação e ao tratamento de informações, bem como às tecnologias empregadas no desenvolvimento do sistema proposto. Dessa forma, estabelec-se o o embasamento conceitual necessário para a compreensão das escolhas metodológicas e técnicas adotadas ao longo deste estudo.

2.1 Gestão Documental Rural

A gestão documental constitui um dos pilares da organização administrativa, especialmente no âmbito contábil e fiscal, sendo responsável por assegurar a produção, a organização, o uso, a avaliação, a preservação e oarquivamento das informações ao longo do ciclo de vida dos documentos.

No agronegócio, a gestão documental assume papel estratégico e indispensável devido a demanda de documentos gerados antes, durante e no decorrer das safras, exemplos de documentos financeiros, fiscais e contábeis: os boletos, os comprovantes de pagamentos, os extratos bancários, as notas fiscais, as declarações, as certidões, entre outros.

É comum que pequenos e médios produtores rurais adotem práticas informais de organização documental, mantendo registros dispersos. A falta de uma rotina sistemática de organização documental acaba elevando o risco de erros, retrabalho e inconsistências nos dados utilizados para fins administrativos, contábeis e fiscais.

Nesse contexto, a gestão documental digital surge como alternativa para superar as limitações dos controles manuais ainda predominantes no meio rural. A aplicação da tecnologia da informação permite automatizar processos de entrada, classificação, armazenamento e recuperação dos documentos, reduzindo falhas humanas e promovendo maior padronização dos registros. Morais, Mussi e Lima (2021) ressaltam que a tecnologia da informação contribui diretamente para

o desempenho da gestão documental, ao melhorar a qualidade dos processos, otimizar recursos e ampliar a eficiência operacional das organizações.

A adoção de sistemas informatizados de gestão documental também favorece a recuperação ágil das informações, aspecto essencial para a gestão financeira do produtor rural. O acesso rápido e estruturado aos dados financeiros permite maior controle de gastos, melhor acompanhamento às exigências fiscais. Além disso, a digitalização e o armazenamento ordenado dos documentos fortalecem a preservação da memória financeira do produtor, assegurando a guarda dos registros pelo prazo legal e facilitando sua apresentação em eventuais fiscalizações.

Sob essa perspectiva, a gestão documental deixa de ser apenas uma atividade de arquivamento e passa a desempenhar função estratégica no tratamento das informações. Estudos desenvolvidos na área da Ciência da Informação reforçam que a ausência de metodologias formais de gestão documental tende a gerar desorganização, dificuldades de recuperação e fragilidade nos processos administrativos, enquanto a adoção de práticas sistematizadas contribui para maior confiabilidade e eficiência.

Dessa forma, a gestão documental rural digitalizada configura-se como fundamento essencial para o sistema proposto neste trabalho, ao promover padronização, organização, preservação e confiabilidade das informações financeiras, elementos indispensáveis ao bom desempenho da atividade contábil aplicada ao produtor rural.

2.2 O Livro Caixa na Atividade Rural

O livro-caixa constitui um dos principais instrumentos de controle financeiro utilizados pelo produtor rural pessoa física, bem como pelo profissional contábil responsável por sua escrituração. Trata-se de um registro fundamentado no regime de caixa, no qual as receitas e despesas são reconhecidas no momento em que ocorre a efetiva entrada ou saída de recursos financeiros. Autores clássicos da contabilidade rural, como Crepaldi (2016), destacam que o livro-caixa permite acompanhar a movimentação financeira da atividade, possibilitando o controle do patrimônio e a apuração do resultado econômico da exploração rural.

No âmbito do Imposto de Renda da Pessoa Física, o livro-caixa assume papel central, pois funciona como base para demonstração das receitas oriundas da produção agropecuária, bem como das despesas relacionadas ao custeio, investimentos, mão de obra e insumos. A legislação tributária brasileira estabelece que o produtor rural deve manter o controle próprio de suas operações e

conservar a documentação comprobatória pelo prazo legal, conforme disposto na Lei nº 8.023 de abril de 1990, garantindo sua apresentação à Receita Federal quando solicitado.

Com o avanço da informatização e das exigências, o Livro Caixa passou a incorporar formatos digitais, destacando-se o Livro Caixa do Produtor Rural (LCDPR). Estudos recentes reforçam a relevância desse instrumento no contexto contemporâneo da atividade rural. Maciel et al. (2025) apontam que o livro-caixa, especialmente em sua versão digital, é essencial para registrar a forma detalhada de todas receitas e despesas da atividade, permitindo a apuração precisa do resultado tributável, o adequado planejamento do Imposto de Renda e maior transparência das informações financeiras.

Além de atender às obrigações legais, o livro-caixa desempenha função estratégica na gestão da atividade rural. A correta escrituração possibilita o controle de custos, a separação entre despesas pessoais e produtivas e acompanhamento do desempenho econômico da propriedade.

A manutenção adequada do livro-caixa, sustentada por documentação idônea, como notas fiscais, recibos e extratos bancários, contribui para a credibilidade do produtor rural perante instituições financeiras e órgãos fiscalizadores. O autor reforça que o controle financeiro eficiente é indispensável para qualquer atividade rural, independentemente de seu porte, sendo o livro caixa um dos principais instrumentos para esse controle.

Dessa forma, o livro-caixa, aliado à atuação profissional do contador e ao uso de ferramentas digitais, configura-se como instrumento indispensável para a gestão financeira, o planejamento tributário e a segurança fiscal do produtor rural, contribuindo para um avanço na organização e na confiabilidade da escrituração rural.

2.3 Automação e Tratamento de Informações

A automação tem se consolidado como um caminho natural para modernizar rotinas administrativas e contábeis, especialmente em ambientes que dependem de grande volume de registros repetitivos, caso típico da escrituração do livro-caixa de pequenos produtores rurais. Embora a atividade rural possua forte tradição manual na organização de documentos, a crescente complexidade das operações bancárias e das exigências fiscais torna a automação um recurso estratégico para melhorar a qualidade das informações.

Segundo Manzueto (2016), softwares de automação transformam rotinas organizacionais ao “aumentar a agilidade e o controle das atividades, reduzindo custos e integrando processos”.

Essa observação se encaixa diretamente na realidade rural, onde o produtor e o contador lidam com extratos de diferentes bancos, nomenclaturas variadas e falta de padronização, fatores que tornam o processo vulnerável a erros manuais.

A literatura também destaca que a automação não trata apenas de substituir trabalho humano, mas de reorganizar a forma de registrar e interpretar informações. Venkatraman (1994), citado por Manzueto, lembra que o uso da tecnologia só produz resultados quando acompanhado de mudanças na forma de trabalhar e de gerenciar processos, pois “benefícios da TI são marginais quando apenas impostos sobre rotinas existentes”. No contexto rural, isso significa que a simples digitalização de documentos não resolve o problema: é necessário estruturar as informações financeiras de maneira padronizada, confiável e auditável.

Outro ponto importante é que, conforme o estudo mostra, rotinas dependem de múltiplos atores, pessoas, documentos e tecnologias, e a automação funciona justamente como elo entre essas partes. Manzueto menciona que a integração tecnológica permite “reduzir o risco operacional presente em controles paralelos baseados em planilhas e anotações manuais”. Esse cenário é idêntico ao do pequeno produtor rural que registra suas movimentações em cadernos, memórias ou planilhas simples.

A utilização de técnicas de ETL (Extract, Transform, Load) fortalece esse processo ao permitir que dados brutos (como extratos bancários em PDF) sejam transformados em informações estruturadas. A combinação tecnológica viabiliza a execução de processos mais eficientes, exatamente o que ocorre ao transformar extratos em lançamentos classificados para o livro caixa e demais relatórios.

A automação é empregada como ferramenta de apoio à gestão documental rural, permitindo:

- Redução de erros de digitação, especialmente em lançamentos bancários de alto volume;
- Padronização das informações financeiras, facilitando a apuração do resultado rural;
- Agilidade na conciliação, reduzindo o tempo operacional gasto pelo produtor e pelo contador;
- Melhor rastreabilidade documental, essencial para eventual fiscalização da Receita Federal;
- Organização interna, servindo como base para o livro caixa.

Assim, a automação não substitui a responsabilidade contábil, mas amplia a qualidade das informações registradas, apoia a tradição do livro-caixa rural e fortalece práticas de gestão financeira nas propriedades, especialmente entre pequenos e médios produtores que ainda dependem fortemente de registros manuais.

2.4 Linguagem Python Aplicada ao Processamento de Dados

A linguagem Python destaca-se no contexto da tecnologia da informação por sua ampla aplicação em automação de processos e processamento de dados. Sua sintaxe simples e elevada legibilidade fovorecem o desenvolvimento de soluções eficientes, organizadas e de fácil manutenção, características amplamente reconhecidas pela literatura acadêmica (SILVA; SILVA, 2019).

No âmbito do processamento de dados, Python tem sido amplamente utilizado em aplicações que envolvem extração, transformação e estruturação de informações provenientes de diferentes fontes. Estudos recentes demonstram sua eficácia na automação da coleta e organização de grandes volumes de dados, reduzindo significativamente o tempo operacional e a dependência de intervenções manuais (GOBI; ZUCCHI; PACHECO, 2024).

Essas características tornam a linguagem especialmente adequada para sistemas que lidam com dados financeiros não estruturados, como extratos bancários em formado PDF. A combinação de bibliotecas específicas permite transformar dados brutos em informações estruturadas, confiáveis e prontas para análise, viabilizando a aplicação de rotinas de classifição, padronização e armazenamento em bancos de dados relacionais.

No sistema proposto neste trabalho, Python foi empregado como tecnologia central para automatizar a leitura de extratos bancários, realizar o tratamento e a normalização das informações extraídas e integrar os dados ao banco MySQL. Essa abordagem reduz erros manuais, aumenta a confiabilidade da escrituração rural e proporciona maior eficiência no processo de organização documental.

Dessa forma, a utilização da linguagem Python sustenta tecnicamente a proposta do sistema, evidenciando sua adequação para aplicações voltadas à automação, ao processamento de dados e à gestão da informação no contexto do agronegócio.

2.5 Banco de Dados MySQL

O armazenamento estruturado e seguro das informações constitui um dos alicerces fundamentais de qualquer sistema digital. No contexto dos sistemas de informação voltados ao controle financeiro e escrituração contábil, os bancos de dados relacionais desempenham papel fundamental ao permitir o armazenamento estruturado e consistente de grandes volumes de informações, como lançamentos financeiros, saldos e históricos de movimentação.

Segundo Korth, Silberschatz e Sudarshan (2012), um banco de dados pode ser compreendido como uma coleção de dados inter-relacionados, definição que se aplica diretamente a sistemas que necessitam organizar informações financeiras de forma lógica, segura e acessível para fins de controle e fiscalização.

Entre os sistemas gerenciadores de bancos de dados relacionais, o MySQL destaca-se por sua ampla utilização em sistemas de gestão financeira e administrativa, especialmente em aplicações que exigem confiabilidade, controle de integridade e facilidade de integração com linguagens de programação, como Python.

A literatura destaca que sua popularidade decorre da compatibilidade com múltiplas plataformas, facilidade de uso e integração simples com diferentes linguagens de programação. Conforme apresentado no material “Tipos de Bancos de Dados”, o MySQL se disseminou amplamente porque “diversos sistemas e linguagens de programação têm uma fácil e rápida conexão com ele”, especialmente linguagens de alto uso em desenvolvimento de sistemas, como PHP e Python.

Além disso, o MySQL oferece vantagens significativas, entre elas:

a) Suporte completo à linguagem SQL;
b) Alta performance em operações de leitura e escrita;
c) Estabilidade em ambientes com múltiplas conexões simultâneas;
d) Ampla comunidade e documentação robusta;
e) Compatibilidade com diversos sistemas operacionais;
f) Estrutura orientada à integridade referencial e transações seguras.

Os bancos relacionais, conforme reforçam Silberschatz, Korth e Sudarshan (2020), são altamente indicados em cenários que exigem consistência transacional, integridade dos dados e estruturação lógica das informações. Esses fatores explicam sua ampla adoção em sistemas que demandam confiabilidade, padronização e segurança no armazenamento e manipulação de dados.

2.6 Streamlit como Ferramenta de Interface Interativa

No desenvolvimento de sistemas baseados em processamento e análise de dados, a interface com o usuário desempenha papel fundamental na visualização, interpretação e utilização das informações geradas. Em aplicações que lidam com grande volume de dados, especialmente no contexto da gestão financeira e da escrituração contábil, torna-se essencial a adoção de ferramentas que possibilitem a construção de interfaces interativas de forma clara, acessível e eficiente, facilitando o apoio à tomada de decisão.

Nesse contexto, o Streamlit destaca-se como uma biblioteca da linguagem Python voltada ao desenvolvimento de aplicações web interativas, amplamente utilizada em projetos acadêmicos e científicos que envolvem análise e visualização de dados. Diferentemente de abordagens tradicionais de desenvolvimento web, o Streamlit permite a integração direta entre o processamento dos dados e sua apresentação visual, reduzindo a complexidade de implementação e favorecendo a rápida prototipação de sistemas baseados em dados.

Estudos recentes demonstram a aplicabilidade do Streamlit em ambientes acadêmicos, especialmente em sistemas que demandam interatividade, visualização dinâmica e integração com rotinas computacionais desenvolvidas em Python. Paulino (2024), ao empregar o Streamlit no desenvolvimento de uma aplicação web baseada em dados, destaca que a ferramenta facilita a construção de interfaces intuitivas, permitindo ao usuário final acessar e interpretar os resultados de forma simplificada, sem a necessidade de conhecimentos técnicos aprofundados. De forma semelhante, Silva e Santos (2024) evidenciam que o uso do Streamlit contribui para maior eficiência na apresentação de dados, ao integrar recursos de visualização com bibliotecas de análise de dados amplamente utilizadas na linguagem Python.

A literatura também aponta que ferramentas desse tipo são especialmente adequadas para aplicações que exigem atualização dinâmica das informações e interação constante do usuário com os dados apresentados. A possibilidade de integração direta com bibliotecas de processamento e análise de dados torna o Streamlit uma alternativa viável para sistemas que necessitam transformar dados brutos em informações compreensíveis e organizadas, favorecendo a usabilidade e a clareza na apresentação dos resultados.

No sistema proposto neste trabalho, o Streamlit foi adotado como ferramenta de interface interativa para possibilitar a visualização das informações financeiras processadas a partir dos extratos bancários. Sua utilização permitiu a construção de telas que apresentam os dados de forma

estruturada, favorecendo o acompanhamento das movimentações financeiras, a organização das informações e a interação do usuário com os resultados gerados pelo sistema.

Dessa forma, a escolha do Streamlit justifica-se por sua integração nativa com a linguagem Python, simplicidade de implementação e adequação ao desenvolvimento de interfaces interativas voltadas à visualização e análise de dados. A ferramenta contribui diretamente para a usabilidade do sistema, reforçando seu papel como apoio à gestão documental e à escrituração contábil no contexto da atividade rural.

2.7 Visual Studio Code (VS Code) como Ambiente de Desenvolvimento

Visual Studio Code, desenvolvido pela Microsoft, é um dos editores de código mais utilizados no mundo. Segundo o Stack Overflow Developer Survey (2022), mais de 70% dos desenvolvedores utilizam VS Code como ferramenta principal devido à sua leveza, extensões, integração com Git e suporte a múltiplas linguagens.

No desenvolvimento do sistema aqui apresentado, o VS Code foi essencial por oferecer:

a) Ambiente leve e rápido;
b) Suporte direto à linguagem Python;
c) Terminal integrado;
d) Extensões para MySQL e Streamlit;
e) Depuração eficiente;
f) Excelente navegação entre arquivos.

3. METODOLOGIA

A metodologia adotada neste trabalho foi estruturada com o objetivo de desenvolver um sistema capaz de automatizar a importação dos dados e o tratamento das informações financeiras do produtor rural, integrando leitura de extratos bancários em PDF, padronização de dados, classificação das movimentações e armazenamento em banco de dados. Para isso, foram utilizadas tecnologias modernas, mas acessíveis.

A seguir apresentam-se as etapas metodológicas que nortearam o desenvolvimento da ferramenta.

3.1 Abordagem geral do desenvolvimento

Este estudo segue uma abordagem aplicada e experimental, uma vez que visa a construção de uma solução tecnológica para um problema prático identificado na rotina da escrituração das atividades rural. O desenvolvimento ocorreu de maneira incremental, permitindo ajustes contínuos à medida que as necessidades da ferramenta eram identificadas e testadas.

Seguindo o princípio da prototipação evolutiva, cada módulo, leitura de extrato, tratamento das linhas, classificação automática, edição manual, geração de relatórios e armazenamento em banco de dados, foi desenvolvido separadamente e integrado ao final. Esse método permitiu maior controle sobre erros e maior flexibilidade na implementação de melhorias.

3.2 Arquitetura geral da solução

A arquitetura do sistema foi organizada em três camadas principais:

a) Camada de Entrada de Dados: Responsável por receber arquivos PDF contendo extratos bancários. Nesta etapa, é aplicada a extração de texto utilizando a biblioteca *pdfplumber*, que permite capturar linhas, colunas e padrões visuais dos extratos.

b) Camada de Processamento e Normalização: Aqui ocorre o coração metodológico do sistema:

- Remoção de linhas irrelevantes (como saldo anterior);
- Identificação de datas, documentos, histórico e valores;
- Padronização dos campos (transformações numéricas, limpeza de caracteres);
- Tratamento das variações de layout entre bancos (Bradesco, Sicoob, Itaú, Santander, etc.);

- Classificação das movimentações entre receitas, despesas, custos e investimentos operacionais.

Essa etapa combina regras contábeis e rotinas de manipulação de dados em Python, garantindo confiabilidade e consistência para a escrituração rural.

c) Camada de Armazenamento e Consulta: Após tratadas, as informações são enviadas automaticamente ao banco de dados MySQL, estruturado previamente com campos específicos para atender às necessidades da escrituração interna. Essa camada também contempla rotinas de consulta, edição, exclusão e geração de relatórios.

3.3 Tecnologias e Ferramentas utilizadas

a) **Python como linguagem principal:** Python foi adotado devido à sua simplicidade, legibilidade e ampla disponibilidade de bibliotecas para manipulação de dados, automação e criação de interfaces. Sua sintaxe permite que o contador que está migrando para um perfil híbrido compreenda melhor os processos internos da aplicação.

b) **Streamlit para a interface interativa:** O framework Streamlit foi empregado para criar telas funcionais e intuitivas, permitindo que o usuário:

1. Envie PDFs;
2. Visualize tabelas tratadas;
3. Edite classificações;
4. Gere relatórios operacionais;
5. Interaja com o banco de dados.

Sua integração direta com Python reduz a complexidade do desenvolvimento e facilita a manutenção do sistema.

c) **MySQL como banco de dados:** O banco de dados MySQL foi escolhido por ser robusto, gratuito e amplamente utilizado em sistemas empresariais. Sua estrutura relacional permite organizar as informações financeiras de forma confiável, evitando redundâncias e garantindo integridade referencial.

d) **Visual Studio Code como ambiente de desenvolvimento:** O VS Code foi utilizado para programação, versionamento, testes e organização dos módulos do projeto, oferecendo terminal integrado, extensões específicas e um fluxo de trabalho ágil.

3.4 Fluxo Metodológico do Sistema

O funcionamento da ferramenta segue um fluxo de processamento definido em etapas:

1. **Upload do extrato em PDF pelo contador:** O sistema recebe o arquivo bancário por meio da interface Streamlit.
2. **Extração do texto:** O módulo *pdfplumber* identifica todas as linhas, separa colunas e interpreta padrões.
3. **Tratamento das linhas brutas:** O código filtra informações irrelevantes, limpa caracteres especiais e normaliza datas.
4. **Padronização dos dados:** Todos os bancos são convertidos para um layout único e homogêneo.
5. **Classificação automática das movimentações:** Regras contábeis definidas no projeto determinam o tipo de cada lançamento.
6. **Revisão manual pelo profissional contábil:** A interface permite ajustes, correções e reclassificações.
7. **Armazenamento no MySQL:** Os dados tratados são gravados no banco para uso posterior.
8. **Geração de relatórios e análise operacional:** O sistema sintetiza: receitas, despesas, custos, investimentos, visão da atividade rural.

3.5 Critérios de Validação da Ferramenta

Para validar a eficácia da solução, foram aplicados os seguintes critérios:

1. **Precisão da extração de dados:** Comparação entre o extrato original e os valores importados pelo sistema.
2. **Coerência das classificações:** Avaliação das regras de classificação automática conforme padrões contábeis rurais.
3. **Confiabilidade do banco de dados:** Testes de inserção, edição, exclusão e consistência dos dados armazenados.
4. **Usabilidade da interface:** Verificação da facilidade de navegação, clareza das telas e velocidade de resposta.
5. **Aplicação prática no ambiente contábil:** Simulação de uso real, escrituração, classificação, conferência das informações e geração de relatórios.

3.6 Limitações e Considerações

Embora eficiente, o sistema apresenta limitações inerentes, como as variações extremas de layout entre bancos podem exigir ajustes futuros; PDF de baixa qualidade pode comprometer a extração textual; o sistema não substitui obrigações fiscais oficiais, atuando apenas como apoio interno; as classificações dependem de constantes aperfeiçoamentos.

3.7 Síntese da Metodologia

A metodologia adotada combina técnicas modernas de automação com práticas de escrituração específicas do produtor rural. A abordagem incremental permitiu construir um sistema robusto, funcional e alinhado às necessidades reais dos profissionais e escritórios contábeis, entregando rapidez, organização e precisão ao processo de escrituração interna rural.

4. DESENVOLVIMENTO DO SISTEMA

O desenvolvimento do Sistema Digital de Escrituração Rural foi realizado com base em uma arquitetura modular, buscando separar claramente as responsabilidades de cada componente e garantindo organização, manutenibilidade e transparência do código. O sistema foi estruturado em três arquivos principais: *app.py*, *funcoes.py e main.py.*

Responsáveis, respectivamente, pela interface do usuário, processamento dos extratos bancários e criação/manutenção do esquema de banco de dados.

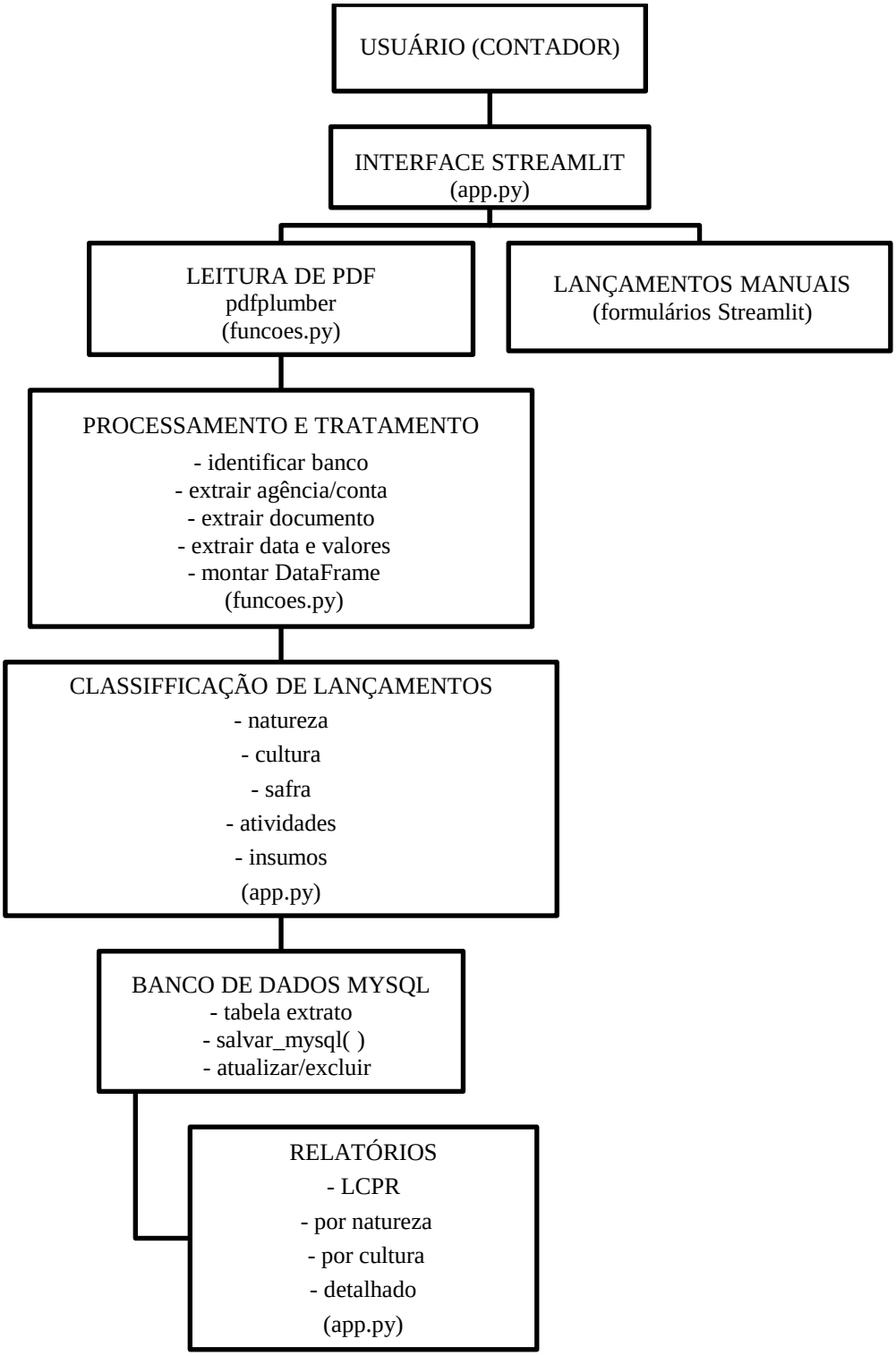
A seguir, apresenta-se uma descrição detalhada do funcionamento interno de cada parte do sistema, explicando a lógica aplicada e relacionando diretamente com o processo de escrituração rural.

4.1 Estrutura Geral do Sistema

O sistema foi projetado em três camadas:

1. **Interface e Navegação (app.py):** Responsável pelas telas, formulários, menus, classificação dos lançamentos, relatórios e integração com o banco de dados.
2. **Processamento e Extração de Dados (funcoes.py):** Módulo que contém as funções críticas para leitura de PDFs, identificação do banco, captura de agência/conta, tratamento das linhas e comunicação com o MySQL.
3. **Infraestrutura e Banco de Dados (main.py):** Arquivo auxiliar que garante a criação e atualização da tabela *extrato*, assegurando compatibilidade estrutural.

Figura 1 - Arquitetura Geral do Sistema



Fonte: Autor (2025)

Essa organização modular permite evolução contínua, correções específicas e maior facilidade para auditoria e documentação contábil.

4.2 Interface e Lógica de Navegação – app.py

O arquivo *app.py* constitui o núcleo visual do sistema, utilizando a biblioteca **Streamlit** para criar uma interface intuitiva, responsiva e alinhada à rotina do contador rural.

4.2.1 Login e Autenticação

O sistema utiliza hash SHA-256 para autenticação:

- Função `_hash_pwd()` gera o hash da senha;
- Usuários pré-configurados são validados no dicionário `USUARIOS`;
- Após login, o usuário é armazenado em `st.session_state`.

Isso garante segurança básica e controle de acesso para os módulos internos.

Figura 2 - Tela de Login do Sistema

Fonte: Autor (2025)

4.2.2 Menu Lateral e Organização das Páginas

O menu lateral oferece:

1. Leitura de extratos em PDF;
2. Lançamentos manuais;

3. Classificação linha a linha ou em tabela;
4. Edição e exclusão de dados já gravados;
5. Relatórios em vários formatos (CSV, Excel e PDF).

A navegação é gerenciada pela função `menu_lateral()`, que altera a página exibida dinamicamente conforme a seleção do usuário.

Figura 3 - Tela Inicial do Sistema

Fonte: Autor (2025)

4.2.3 Leitura de Extratos (PDF → Tabela)

A página "Extrato Bancário (PDF)" executa:

1. Upload do arquivo PDF;
2. Envio do PDF para as funções ler_pdf() e tratar_linhas();
3. Montagem do DataFrame base;
4. Padronização automática dos campos.

Isso permite que o contador visualize imediatamente as movimentações bancárias de forma estruturada.

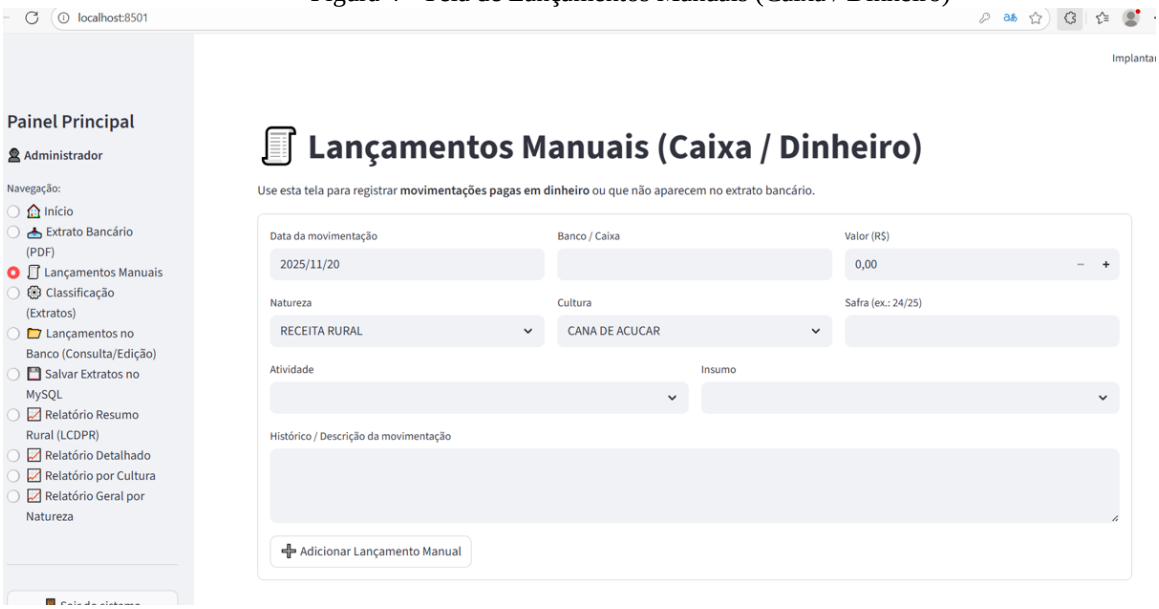
4.2.4 Lançamentos Manuais

A tela de lançamentos manuais foi projetada para registrar operações fora do extrato:

- Pagamentos em dinheiro,

- Movimentações via PIX pessoal,
- Uso de dinheiro em espécie,
- Ajustes manuais.

Figura 4 - Tela de Lançamentos Manuais (Caixa / Dinheiro)



Fonte: Autor (2025)

Os lançamentos seguem a mesma estrutura dos extratos, garantindo padronização.

4.2.5 Classificação dos Lançamentos

O módulo mais operacional do sistema:

a) Classificação individual (modo cartão);

b) Classificação em lote;

c) Classificação em grade (editor de tabela interativo).

Campos classificáveis:

a) Natureza (ex.: Receita Rural, Custo Rural etc.);

b) Cultura (ex.: soja, cana, milho etc.);

c) Safra (ex.: 23/24);

d) Atividades (ex.: salários, gratificação, alimentação);

e) Insumos (óleo diesel, calcário agrícola, adubo NPK).

O sistema destaca automaticamente linhas com natureza pendente, usando estilização da função `styler_pendencias()`.

4.2.6 Salvamento no MySQL

O salvamento final utiliza:

- Comandos SQL via **pymysql**;
- Modos: “pular duplicados” ou “sobrepor duplicados”;
- Normalização dos textos antes da gravação.

O backend valida cada linha e grava com consistência.

4.3 Processamento Interno dos Extratos – funcoes.py

O arquivo *funcoes.py:* concentra a inteligência do sistema, sendo o responsável por interpretar extratos bancários, extrair informações relevantes e transformar texto bruto em dados estruturados.

4.3.1 Leitura do PDF – `ler_pdf()`

Utiliza **pdfplumber** para:

- Abrir o PDF página por página;
- Extrair o texto integral;
- Concatenar as páginas em uma string única.

Pdfplumber permite alta precisão em tabelas e colunas, superando o PyPDF2 para este tipo de extrato.

4.3.2 Identificação Automática do Banco – `identificar_banco()`

O sistema analisa padrões textuais para identificar o banco:

a) Banco do Brasil;
b) Sicoob / Credicitrus;
c) Bradesco;
d) Santander;
e) Itaú;

A identificação do banco permite aplicar regras específicas para capturar documento e agência/conta.

4.3.3 Extração de Agência e Conta – `extrair_agencia_conta()`

Inclui padrões distintos para: Sicoob (COOP. e CONTA), Banco do Brasil e Bradesco; demais instituições.

Isso garante que relatórios gerenciais identifiquem corretamente a origem das movimentações.

4.3.4 Função Principal: `tratar_linhas()`

Essa função é a essência do sistema pois:

1. Percorre linha a linha do texto;
2. Identifica datas;
3. Detecta valores;
4. Captura documentos bancários (Bradesco e Sicoob possuem padrões diferentes);
5. Monta descrições completas;
6. Ignora cabeçalhos e rodapés do PDF;
7. Cria um registro estruturado com: `DATA, BANCO, AGENCIA_CONTA, DOCUMENTO, HISTORICO, VALOR.`

Após isso, converte em DataFrame Pandas pronto para classificação.

4.3.5 Salvamento no MySQL – `salvar_mysql()`

A função recebe um DataFrame já tratado e grava cada linha na tabela *extrato*, aplicando:

- Validação de formato de data;
- `INSERT ... ON DUPLICATE KEY UPDATE`
- Contagem de inseridos, atualizados e erros.

O log final retorna ao usuário relatórios de inserção, facilitando conferências contábeis.

4.4 Estrutura do Banco de Dados – main.py

O arquivo *main.py:* assegura que a tabela extrato esteja sempre criada com o seguinte esquema:

```
id INT PK
```

```
data DATE
banco VARCHAR(50)
agencia_conta VARCHAR(50)
documento VARCHAR(50)
historico TEXT
valor DECIMAL(15,2)
natureza VARCHAR(20)
cultura VARCHAR(30)
safra VARCHAR(20)
atividades VARCHAR(80)
insumos VARCHAR(120)
```

A tabela foi construída pensando nas exigências da escrituração rural e nas necessidades dos relatórios:

a) LCPR;

b) Relatórios gerenciais por cultura;

c) Relatórios detalhados por natureza;

d) Visão contábil da atividade rural.

4.5 Relatórios Gerados pelo Sistema

O sistema possui quatro tipos de relatórios:

1. Resumo Livro Caixa Rural (LCPR);
2. Relatório Detalhado (movimentações com filtros);
3. Relatório por Cultura;
4. Relatório Geral por Natureza.

Todos podem ser exportados em: CSV, Excel (via .csv), PDF (via ReportLab)

Os relatórios contemplam:

a) Receitas rurais;

b) Despesas rurais;

c) Custos de produção;

d) Investimentos;

e) Análise de safra;

f) Análise por cultura;

g) Comparativos por mês.

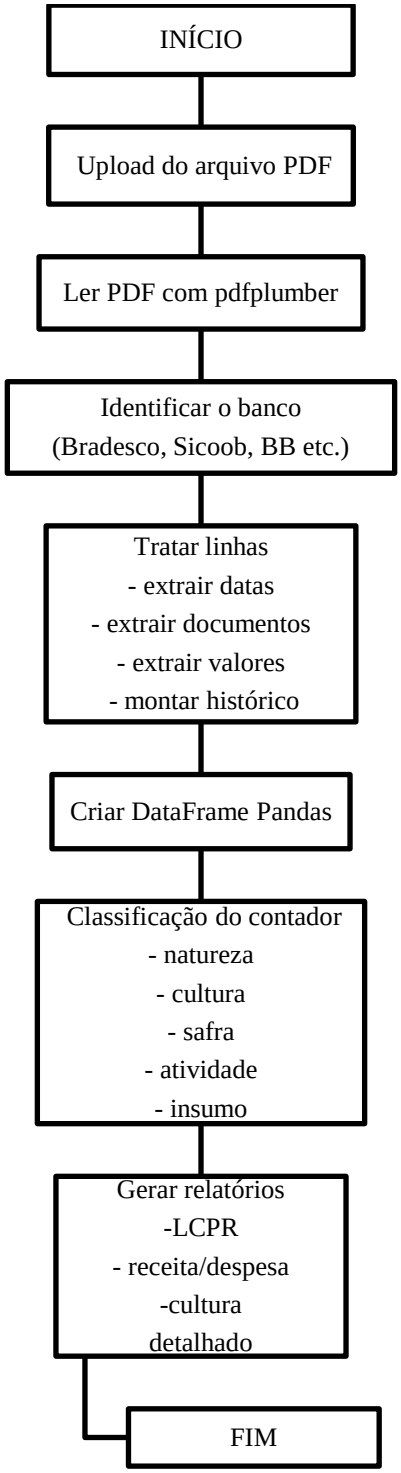
Figura 5 - Tela do Relatório – Resumo LCPR

Fonte: Autor (2025)

4.6 Fluxo de Processamento do Extrato

1. Upload do PDF: Usuário envia o extrato.
2. Leitura e extração do texto: pdfplumber extrai o conteúdo.
3. Identificação do banco: Define o padrão de interpretação.
4. Interpretação linha a linha: Captura data, documento, histórico e valor.
5. Montagem do DataFrame: Tabela padronizada pronta para classificação.
6. Classificação: Natureza, cultura, safra, atividade, insumo.
7. Salvamento no MySQL: Gravação com validação e retorno de status.
8. Relatórios e análises: Geração de visões contábeis e gerenciais.

Figura 6 - Fluxograma do Processamento do Extrato Bancário



Fonte: Autor (2025)

5 RESULTADOS E DISCUSSÃO

O desenvolvimento do Sistema Digital de Escrituração Rural permitiu avaliar, na prática, sua eficiência na automação de processos de escrituração aplicados ao produtor rural pessoa física. Os resultados obtidos demonstram uma evolução significativa na organização e no tratamento das informações provenientes dos extratos bancários, etapa essencial para a elaboração da escrituração interna e para o preenchimento do Livro Caixa da Atividade Rural, (LCPR) e da Declaração de Ajuste Anual do Imposto de Renda da Pessoa Física (DIRPF).

Do ponto de vista operacional, observou-se que o sistema reduziu de forma expressiva o tempo gasto com digitação manual dos extratos, resultado que confirma a importância da automação documental discutida no Capítulo 2. Conforme abordado na fundamentação teórica, a substituição de rotinas manuais por processos automatizados contribui para maior eficiência, padronização e redução de erros na escrituração financeira.

Além disso, a classificação estruturada dos lançamentos, por natureza, cultura, safra, atividade e insumo, permitiu organizar os dados de forma alinhada às necessidades da escrituração rural. Essa estrutura padronizada facilitou não apenas a geração dos relatórios, mas também a análise de desempenho da atividade agrícola e pecuária, especialmente quando agrupada por safra e cultura. A utilização de um banco de dados relacional (MySQL) reforçou a integridade das informações, garantindo que todos os registros permanecessem armazenados de forma segura, auditável e coerente.

Outro resultado relevante foi a melhoria na gestão documental. O sistema consolidou em um único ambiente todas as movimentações financeiras tratadas, substituindo controles dispersos em cadernos, planilhas ou anotações informais. Para o contador, essa centralização reduz retrabalho, simplifica conferências e melhora a rastreabilidade dos dados, requisito importante tanto para fins fiscais quanto gerenciais.

Os relatórios gerados pelo sistema apresentaram-se claros, objetivos e alinhados à rotina do profissional contábil. O Relatório Resumo Rural, por exemplo, permite ao usuário visualizar de forma mensal a relação entre receitas brutas, despesas rurais, custos de produção e investimentos, facilitando o acompanhamento da atividade e a preparação da escrituração anual. Já os relatórios por cultura e por natureza oferecem uma perspectiva detalhada dos gastos agrícolas, contribuindo para a tomada de decisão e para o planejamento produtivo.

Do ponto de vista da usabilidade, o sistema mostrou-se simples, intuitivo e aderente às necessidades práticas do contador. A interface construída com Streamlit foi fundamental para tornar o processo acessível, permitindo que usuários sem conhecimento técnico avançado pudesse navegar pelas funcionalidades com facilidade. A presença de recursos como menus laterais, formulários simplificados, opções de exportação de relatórios e feedback visual das operações auxiliou na condução natural do fluxo de trabalho.

Os testes realizados demonstraram que o sistema é capaz de lidar com extratos de diferentes instituições financeiras, identificando padrões específicos de cada banco e adaptando-se às suas particularidades. Essa flexibilidade amplia a aplicabilidade da ferramenta para diversos perfis de produtores rurais que utilizam bancos distintos em suas operações.

Por fim, a modularidade do projeto mostrou-se um diferencial importante. A separação entre interface, processamento e banco de dados possibilita futuras melhorias sem comprometer a estrutura já construída.

A modularidade do projeto mostrou-se um diferencial importante durante os testes realizados, uma vez que a separação entre interface, processamento e banco de dados permitiu identificar pontos de possível expansão do sistema. Observou-se, por exemplo, que a estrutura atual favorece a futura integração com ferramentas de Business Intelligence, como o Power BI, possibilitando análises mais visuais e comparativas a partir dos dados já armazenados.

Além disso, verificou-se que a leitura de extratos bancários em formato PDF poderia ser ampliada para documentos digitalizados, o que indica a viabilidade de adoção futura de técnicas de reconhecimento óptico de caracteres (OCR). Da mesma forma, a recorrência de padrões nos históricos bancários sugere potencial para o uso de classificadores baseados em machine learning, com o objetivo de apoiar a classificação automática dos lançamentos e reduzir ainda mais a intervenção manual do contador.

De modo geral, os resultados obtidos demonstram que o Sistema Digital de Escrituração Rural atingiu os objetivos propostos, ao reduzir significativamente o retrabalho manual, melhorar a organização das informações financeiras e oferecer maior confiabilidade à escrituração interna do produtor rural. A ferramenta mostrou-se adequada à rotina contábil, especialmente diante do volume crescente de movimentações financeiras, evidenciando seu potencial como apoio prático e eficiente à atividade profissional do contador no agronegócio.

6 CONCLUSÃO

O presente trabalho teve como objetivo desenvolver um Sistema Digital de Escrituração Rural capaz de automatizar a leitura, organização, classificação e armazenamento das informações financeiras provenientes de extratos bancários do produtor rural pessoa física. A proposta nasceu da vivência prática em ambiente contábil, onde o volume crescente de documentos, aliado à complexidade das operações rurais, torna a escrituração manual cada vez mais demorada, suscetível a erros e operacionalmente desgastante.

A solução apresentada demonstrou eficiência no tratamento dos extratos bancários, convertendo arquivos em PDF em dados estruturados e padronizados, prontos para classificação e posterior utilização na escrituração interna e na elaboração da Declaração de Ajuste Anual do Imposto de Renda da Pessoa Física (IRPF). A implementação em Python, associada ao uso de bibliotecas como *pdfplumber* e *pandas*, permitiu automatizar etapas críticas do processo, enquanto o uso de um banco de dados MySQL garantiu integridade, segurança e persistência das informações.

O sistema também se destacou pela facilidade de uso, graças à interface desenvolvida em Streamlit, que possibilitou uma navegação simples e intuitiva, adequada para o contador rural que precisa lidar com operações complexas sem, necessariamente, ter familiaridade com programação ou sistemas avançados. A modularidade do projeto contribuiu para sua organização interna e abre caminho para futuras expansões, demonstrando que a ferramenta pode evoluir de acordo com as necessidades do escritório contábil e da atividade rural.

Os resultados apresentados evidenciam que o sistema atinge seu propósito central: modernizar e agilizar a escrituração rural, reduzir o risco de inconsistências, aumentar a confiabilidade dos dados e oferecer uma visão clara e objetiva da movimentação financeira do produtor. Além disso, a ferramenta proporciona melhorias significativas na gestão documental, contribuindo para práticas contábeis mais precisas, organizadas e alinhadas às exigências fiscais.

Embora não substitua softwares de gestão rural completos, o sistema desenvolvido se posiciona como uma ferramenta prática, acessível e eficiente para o contador que busca otimizar sua rotina, reduzir retrabalho e proporcionar melhores análises aos seus clientes. O projeto apresenta potencial para futuras melhorias, como integração com ferramentas de Business Intelligence, implementação de OCR para documentos digitalizados e uso de técnicas de *machine learning* para classificação automática dos lançamentos.

Assim, conclui-se que o Sistema Digital de Escrituração Rural representa uma contribuição relevante para o campo da contabilidade aplicada ao agronegócio, reforçando o papel da tecnologia como aliada na transformação da rotina contábil. O trabalho demonstra que soluções simples, bem estruturadas e orientadas às necessidades reais do profissional podem gerar avanços significativos na precisão, na organização e na eficiência do processo de escrituração rural.

7 REFERÊNCIAS

ARAÚJO Lorena Cristina Silva. **Gestão Documental: Gerenciamento Eletrônico De Documentos, Sistema Que Torna Os Processos Eficientes E Que Facilita A Administração Dos Arquivos**. Universidade Federal Do Rio Grande Do Norte, Natal/RN, 2021. Disponível em: https://repositorio.ufrn.br/server/api/core/bitstreams/85651069-6580-43ef-aba0-06df193cd566/content. Acesso em: 16 dez. 2025**.**

BARBOZA, Fabrício Felipe Meleto. **Modelagem e desenvolvimento de banco de dados: tipos de bancos de dados.** [S.l.: s.n.], 2018. Disponível em: https://www.ouka.com.br/carol/e-book/database/tipos-de-bancos-de-dados.pdf. Acesso em: 19 nov. 2025.

BRASIL. **Lei nº 8.023, de 12 de abril de 1990.** Altera a legislação do Imposto de Renda sobre o resultado da atividade rural e dá outras providências. Diário Oficial da União, Brasília, DF, 13 abr. 1990. Disponível em: https://www.planalto.gov.br/ccivil_03/leis/L8023.htm. Acesso em: 16 dez. 2025.

CEPEA – Centro de Estudos Avançados em Economia Aplicada. **PIB do agronegócio cresce 6,49% no 1º trimestre de 2025.** Piracicaba, 2025. Disponível em: https://www.cepea.esalq.usp.br/. Acesso em: 18 nov. 2025.

CREPALDI, Sílvio Aparecido. **Contabilidade rural: agrícola, pecuária e imposto de renda.** 8. ed. São Paulo: Atlas, 2016.

CREPALDI, Sílvio Aparecido**. Contabilidade rural: uma abordagem decisorial**. 8. ed. São Paulo: Atlas, 2016.

DATE, C. J. **Introdução a sistemas de banco de dados**. 8. ed. Rio de Janeiro: Elsevier, 2004. Disponível em: https://books.google.com.br/books?id=xBeO9LSlK7UC. Acesso em: 19 nov. 2025.

EMBRAPA. **Agricultura Digital no Brasil: relatório técnico 2021.** Brasília: Embrapa, 2021. Disponível em: https://www.embrapa.br. Acesso em: 18 nov. 2025.

GOBI, Guilherme; ZUCCHI, Maurício; PACHECO, Felipe. **Desenvolvimento de um programa em Python para extração automatizada de dados na Biblioteca Cochrane: um estudo descritivo.** Journal of Assistance in Pharmacy and Pharmacoeconomics, 2024. Disponível em: https://www.ojs.jaff.org.br/ojs/index.php/jaff/article/view/890/837. Acesso em: 17 dez. 2025.

KORTH, H. F.; SILBERSCHATZ, A.; SUDARSHAN, S. **Sistema de banco de dados.** 6. ed. Rio de Janeiro: Campus, 2012.

LOPES, Gesiel; et al. **Introdução à análise exploratória de dados com Python.** *[S.l.: s.n.]*, 2019. Disponível em: https://www.researchgate.net/publication/336778766_Introducao_a_Analise_Exploratoria_de_Dados_com_Python. Acesso em: 17 dez. 2025.

MANZUETO, Mauricio Santos. **Automação de processos: a influência dos softwares de automação de processos nas rotinas organizacionais.** 2016. 57 f. Dissertação (Mestrado em Administração de Empresas) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Administração, Rio de Janeiro, 2016. Disponível em: https://www.maxwell.vrac.puc-rio.br/28434/28434.PDF. Acesso em: 16 dez. 2025.

MARION, José Carlos. **Contabilidade rural: agrícola, pecuária e imposto de renda.** 14. ed. São Paulo: Atlas, 2014.

MICROSOFT. **Visual Studio Code Documentation.** Disponível em: https://code.visualstudio.com/docs. Acesso em: 18 nov. 2025.

MORAIS, Silvia Cintra Borges; MUSSI, Clarissa Carneiro; LIMA, Mauricio Andrade. **Tecnologia da informação e desempenho da gestão documental: uma estrutura conceitual**. Revista Brasileira de Preservação Digital, Campinas, v. 2, e021004, 2021. Disponível em: https://econtents.sbu.unicamp.br/inpec/index.php/rebpred/article/view/15659/10699. Acesso em: 16 dez. 2025.

MORAIS, Silvia Cintra Borges; MUSSI, Clarissa Carneiro; LIMA, Mauricio Andrade. **Tecnologia da informação e desempenho da gestão documental: uma estrutura conceitual.** Revista Brasileira de Preservação Digital, Campinas, v. 2, e021004, 2021. Disponível em: https://ojs.focopublicacoes.com.br/foco/article/view/10491/7377. Acesso em: 16 dez. 2025.

OLIVEIRA, Natanael Silva; MATTOS, Enrique Vieira; REBOITA, Michelle Simões. **Dashboard interativo de dados de queimadas utilizando o framework Streamlit do Python**. In: Anais Do XXI Simpósio Brasileiro De Sensoriamento Remoto, 2025, Salvador. São José dos Campos: Instituto Nacional de Pesquisas Espaciais (INPE), 2025. Disponível em: http://marte2.sid.inpe.br/col/sid.inpe.br/marte2/2025/08.16.17.46.30/doc/@individualPDF.pdf. Acesso em: 17 dez. 2025.

PAN, Franco Brunetta. **Desenvolvimento de extensão de VS Code para a análise de complexidade de algoritmos.** 2024. 65 f. Monografia (Bacharelado em Ciência da Computação) – Universidade de Caxias do Sul, Caxias do Sul, 2024. Disponível em: https://repositorio.ucs.br/bitstream/handle/11338/14592/TCC%20Franco%20Brunetta%20Pan.pdf. Acesso em: 18 nov. 2025.

PAULINO, Marcos Antônio Baccin. **Machine learning para classificação de problemas cardíacos: proposta de uma aplicação.** Morrinhos: Instituto Federal Goiano, 2024. Disponível em: https://repositorio.ifgoiano.edu.br/bitstream/prefix/4941/1/tcc_Marcos%20Paulino.pdf Acesso em: 18 nov. 2025.

PYTHON SOFTWARE FOUNDATION. **Python Documentation.** Disponível em: https://docs.python.org/. Acesso em: 17 nov. 2025.

RECEITA FEDERAL DO BRASIL. **Manual de Preenchimento do IRPF – Exercício de 2024.** Brasília: RFB, 2024. Disponível em: https://www.funprespjud.com.br/wp-content/uploads/2024/03/Manual-IRPF-2024.pdf. Acesso em: 18 nov. 2025.

RECEITA FEDERAL DO BRASIL. **Manual de Preenchimento do Livro Caixa Digital do Produtor Rural – LCDPR: leiaute 1.3.** Brasília: RFB, 2020. Disponível em: https://www.gov.br/receitafederal/pt-br/assuntos/orientacao-tributaria/declaracoes-e-demonstrativos/lcdpr-livro-caixa-digital-do-produtor-rural/manual-de-preenchimento-do-lcdpr-1-3. Acesso em: 18 nov. 2025.

SANTOS, Juliana Cardoso dos; VALENTIM, Marta Lígia Pomim. **Gestão documental e gestão da informação como ferramentas da memória organizacional: foco na memória repositório.**

Ágora: Arquivologia em Debate, Florianópolis, v. 31, n. 62, p. 1–25, jan./jun. 2021. Disponível em: https://agora.emnuvens.com.br/ra/article/view/957/922. Acesso em: 16 dez. 2025.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S**. Database System Concepts. *7.*** ed. New York: McGraw-Hill, 2020.

SILVA, Diogo Henrique; SANTOS, Paulo César.**ZENOAI: otimização de suporte técnico e treinamento corporativo com RAG e LLM**. Anais do JOSIF – Jornada de Sistemas de Informação, 2024. Disponível em: https://josif.ifsuldeminas.edu.br/ojs/index.php/anais/article/view/1958/1852. Acesso em: 17 dez. 2025.

STACK OVERFLOW. **Stack Overflow Developer Survey 2022.** Disponível em: https://survey.stackoverflow.co/2022/. Acesso em: 18 nov. 2025.

STREAMLIT. **Streamlit Documentation.** Disponível em: https://docs.streamlit.io/. Acesso em: 18 nov. 2025.

TASK. **O que é um framework?** Disponível em: https://www.task.com.br/blog/o-que-e-um-framework/. Acesso em: 18 nov. 2025.

TURBAN, Efraim; VOLONINO, Linda; WOOD, Gregory. **Tecnologia da informação para gestão. *9.*** ed. São Paulo: Atlas, 2018.

UNSTRACT. **Python libraries to extract tables from PDF: a comparison.** Disponível em: https://unstract.com/blog/extract-tables-from-pdf-python/. Acesso em: 18 nov. 2025.

APÊNDICE A – Código fonte do arquivo app.py

```
import os
import math
import hashlib
from datetime import datetime
import streamlit as st
import pandas as pd
import pymysql
from funcoes import DB_CONFIG

import streamlit as st
import pandas as pd
import pymysql
import matplotlib.pyplot as plt

# Apenas não mostrar aqueles rastros gigantes de erro na tela
st.set_option("client.showErrorDetails", False)

from funcoes import (
    ler_pdf,
    tratar_linhas,
    salvar_mysql,
    remover_acentos,
    DB_CONFIG,
)



# ======================== CONFIGURAÇÕES GERAIS ========================
LOGO_ICONE = "82da52ba-a153-43f2-9fd8-fcb95ab0e8dc.png"
LOGO_COMPLETO = "e1e53166-86cc-4dbc-a971-fb4f09e5efa2.png"

st.set_page_config(
    page_title="RT - Sistema de Escrituração Rural",
    layout="wide",
    initial_sidebar_state="expanded",
    page_icon=LOGO_ICONE
)
```

```
# ======================== LOGIN E AUTENTICAÇÃO ========================
def _hash_pwd(pwd: str) -> str:
    return hashlib.sha256(pwd.encode("utf-8")).hexdigest()

USUARIOS = {
    "rogerio": {"hash": _hash_pwd("rogerio2025"), "nome": "Rogério"},
    "admin":   {"hash": _hash_pwd("admin123"),   "nome": "Administrador"},
}

def autenticar(usuario: str, senha: str) -> bool:
    u = USUARIOS.get(usuario.strip().lower())
    return bool(u and _hash_pwd(senha) == u["hash"])

# ======================== TELA DE LOGIN ========================
def tela_login():
    st.markdown(
        """
        <style>
            .stApp {
                background-image: url('fundo_login.png');
                background-size: cover;
                background-position: center;
                background-attachment: fixed;
                background-repeat: no-repeat;
            }
            .stTextInput > div > div > input {
                background-color: rgba(255, 255, 255, 0.9);
                border: 1px solid #ccc;
                border-radius: 6px;
            }
            .stForm {
                background-color: rgba(255, 255, 255, 0.90);
                padding: 25px;
                border-radius: 10px;
                box-shadow: 0 0 15px rgba(0,0,0,0.3);
                width: 420px;
                margin: auto;
            }
```

```
            .stButton > button {
                background-color: #2e7d32;
                color: white;
                border: none;
                border-radius: 8px;
                padding: 0.5rem 1rem;
                font-weight: bold;
                transition: 0.3s;
            }
            .stButton > button:hover {
                background-color: #1b5e20;
            }
            h1, h3, p {
                text-shadow: 1px 1px 2px rgba(255,255,255,0.7);
            }
        </style>
        """,
        unsafe_allow_html=True
    )

    st.markdown("<div style='text-align:center;'>", unsafe_allow_html=True)
    if os.path.exists(LOGO_COMPLETO):
        st.image(LOGO_COMPLETO, width=360)
    else:
        st.markdown("<h1 style='color:#2e7d32;'>🌱 RT – Sistema de
Escrituração Rural</h1>", unsafe_allow_html=True)
    st.markdown("<h3>Sistema de Lançamentos e Escrituração Rural</h3>",
unsafe_allow_html=True)
    st.markdown("</div>", unsafe_allow_html=True)
    st.markdown("---")

    st.markdown("<h4 style='text-align:center;'>🔐 Acesso Restrito</h4>",
unsafe_allow_html=True)
    st.caption("Informe seu usuário e senha para continuar.")

    with st.form("form_login"):
        col1, col2 = st.columns(2)
        user = col1.text_input("Usuário", placeholder="Ex.: rogerio")
```

```
        pwd = col2.text_input("Senha", type="password", placeholder="Digite
sua senha")
        ok = st.form_submit_button("Entrar", width="stretch"
)

        if ok:
            if autenticar(user, pwd):
                st.session_state["auth"] = True
                st.session_state["usuario"] = user.strip().lower()
                st.session_state["nome_exibicao"] =
USUARIOS[user.strip().lower()]["nome"]
                st.success("✓ Acesso concedido.")
                st.rerun()
            else:
                st.error("Usuário ou senha incorretos.")

# ======================== LISTAS PADRÕES ========================
NATUREZAS = ["RECEITA RURAL","DESPESA RURAL","CUSTOS RURAL","INVESTIMENTO
RURAL", "RECEITAS - NAO RURAL", "DESPESAS - NAO RURAL", "OUTRAS"]
CULTURAS  = ["CANA DE
ACUCAR","SOJA","MILHO","SORGO","AMENDOIM","PECUARIA","OUTRAS"]

ATIVIDADES_LIST = [
    "SALARIOS","HORAS EXTRAS","PERICULOSIDADE","INSALUBRIDADE","ADICIONAL
NOTURNO","GRATIFICACAO",
    "ALIMENTACAO","FERIAS","13O SALARIO","AUXILIO
ALIMENTACAO","EPIS","FGTS","ASSISTENCIA MEDICA E SOCIAL",
    "OUTRAS ENTIDADES (SEST/SENAT)","ENCARGOS SOCIAIS
(RETENCOES)","COMISSAO","SERVICOS TERCEIRIZADOS",
    "TRIBUTOS","OUTRAS"
]
ATIVIDADES_OPCOES = [""] + ATIVIDADES_LIST

INSUMOS_OPCOES = ["",
    "OLEO DIESEL","COMBUSTIVEL E LUBRIFICANTES","CALCARIO AGRICOLA","OXIDO DE
CALCIO",
    "GESSO AGRICOLA","HERBICIDA PRE EMERGENTE FOLHA FINA","HERBICIDA PRE
EMERGENTE FOLHA LARGA",
```

```
    "HERBICIDA POS EMERGENTE FOLHA FINA","HERBICIDA POS EMERGENTE FOLHA
LARGA","MICRONUTRINTE DE SOLO",
    "ADUBO
NPK","MUDAS","SEMENTES","INOCULANTE","ENRAIZANTE","NEMATICIDA","ESTIMULANTE",
    "ADJUVANTE / ESTABILIZADOR DE
CALDA","MICRONUTRIENTE","INSETICIDA","FUNGICIDA",
    "VERMIFUGO E CARRAPATICIDA","VACINA","ANTIBIOTICO","ANTI
INFLAMATORIO","DESINTOXICANTE",
    "NUTRIENTES","SAL MINERAL","SAL PROTEINADO","SAL
ENERGETICO","RACAO","SEMEN",
    "MATERIAIS PARA INSEMINACAO","OUTROS"
]

# ======================== FUNÇÕES AUXILIARES ========================
def limpar_ui(s: str) -> str:
    return remover_acentos(str(s)).upper().strip()

def formatar_data_br(df: pd.DataFrame, col: str = "DATA") -> pd.DataFrame:
    if col in df.columns:
        try:
            df[col] = pd.to_datetime(df[col], dayfirst=True,
errors="coerce").dt.strftime("%d/%m/%Y")
        except Exception:
            pass
    return df

def pend_mask(df: pd.DataFrame):
    if "NATUREZA" not in df.columns:
        return pd.Series([True] * len(df), index=df.index)
    return df["NATUREZA"].astype(str).str.strip().eq("")

def styler_pendencias(df: pd.DataFrame):
    def _highlight(row):
        pendente = str(row.get("NATUREZA", "")).strip() == ""
        if pendente:
            return ["background-color: #ffe6e6; color: #900;"] * len(row)
        else:
            return [""] * len(row)
```

```
    styler = df.style.apply(_highlight, axis=1)
    if "VALOR" in df.columns:
        styler = styler.format({"VALOR": "{:,.2f}"})
    return styler

# ✅ Correção: função preparar_df unificada e segura
def preparar_df(df: pd.DataFrame) -> pd.DataFrame:
    df = df.copy()
    df.columns = [c.strip().upper() for c in df.columns]
    for col in ["NATUREZA", "CULTURA", "SAFRA", "ATIVIDADES", "INSUMOS"]:
        if col not in df.columns:
            df[col] = ""
    if "DATA" in df.columns:
        df["DATA"] = pd.to_datetime(df["DATA"], errors="coerce")
    if "VALOR" in df.columns:
        df["VALOR"] = pd.to_numeric(df["VALOR"], errors="coerce").fillna(0.0)
    df = df.fillna("").reset_index(drop=True)
    return df



# ======================== MENU LATERAL ========================
def menu_lateral():
    st.sidebar.title("Painel Principal")
    nome = st.session_state.get("nome_exibicao", "Usuário")
    st.sidebar.markdown(f"👤 **{nome}**")

    escolha = st.sidebar.radio(
        "Navegação:",
        [
        "🏠 Início",
        "📥 Extrato Bancário (PDF)",
        "📝 Lançamentos Manuais",
        "⚙️📝 Classificação (Extratos)",
        "🗃️ Lançamentos no Banco (Consulta/Edição)",
        "💾 Salvar Extratos no MySQL",
        "📈 Relatório Resumo Rural (LCDPR)",
```

```
            "📈 Relatório Detalhado",
            "📈 Relatório por Cultura",
            "📈 Relatório Geral por Natureza"
            ]
        )

        st.sidebar.divider()
        if st.sidebar.button("🚪 Sair do sistema", use_container_width=True):
            keys = list(st.session_state.keys())
            for k in keys:
                del st.session_state[k]
            st.success("Sessão encerrada.")
            st.rerun()

        return escolha


# ======================== PÁGINA: INÍCIO ========================
def pagina_inicio():
    # Exibe logotipo e introdução
    if os.path.exists(LOGO_COMPLETO):
        st.image(LOGO_COMPLETO, width=360)
    else:
        st.title("🌱 RT – Sistema de Escrituração Rural")

    st.markdown("---")
    st.markdown("""
    ### 🧾 Descrição Geral
    O **Sistema de Lançamentos e Escrituração Rural** foi desenvolvido para
**facilitar a escrituração do Livro Caixa** dos produtores rurais,
    permitindo o registro de **entradas e saídas de recursos financeiros** de
forma **simples, organizada e segura**.

    Ele trabalha com dois caminhos:
    - Movimentações via **extrato bancário (PDF)**;
    - **Lançamentos manuais** (pagos em dinheiro, PIX pessoal, cartão etc.).
```

```
    ---

    ### ⚙ Funcionalidades
    - 📥 Leitura de extratos bancários em PDF;
    - 📝 Registro manual de receitas e despesas;
    - 📊 Classificação por natureza, cultura, safra, atividades e insumos;
    - 💾 Salvamento em banco MySQL;
    - 📈 Base pronta para relatórios e dashboards (ex.: Power BI).

    ---

    ### 👨‍🌾 Público-Alvo
    - Produtores rurais;
    - Profissionais contábeis;
    - Gestores administrativos do agronegócio.

    O foco é **facilitar a rotina contábil e fiscal**, mantendo uma visão clara do caixa rural.
    """)

# (as demais funções de extrato, lançamentos, classificação, salvar, editar permanecem as mesmas do seu código anterior)
# Basta colar o restante abaixo, pois não há necessidade de alteração estrutural nelas.


# ======================== PÁGINA: EXTRATO (PDF) ========================
def pagina_extrato():
    st.title("📥 Leitor de Extrato Bancário (PDF → Tabela)")
    if "df_base" not in st.session_state:
        st.session_state.df_base = None
    if "df_edit" not in st.session_state:
        st.session_state.df_edit = None

    arq = st.file_uploader("Anexe um extrato bancário em PDF", type=["pdf"], key="upload_extrato")
```

```
    if arq is None:
        if st.session_state.df_edit is not None:
            st.info("Já existe um extrato carregado. Você pode ir em
'Classificação (Extratos)' para continuar.")
            df_vis =
st.session_state.df_edit[["DATA","BANCO","AGENCIA_CONTA","DOCUMENTO","HISTORIC
O","VALOR"]].copy()
            df_vis = formatar_data_br(df_vis, "DATA")
            st.dataframe(
                df_vis.style.format({"VALOR":"{:,.2f}"}),
                use_container_width=True, height=420
            )
        else:
            st.info("Anexe um arquivo PDF para iniciar a leitura do extrato.",
icon="📎")
        return

    tmp = f"temp_{os.path.basename(arq.name)}"
    with open(tmp, "wb") as f:
        f.write(arq.getbuffer())

    texto = ler_pdf(tmp)
    df = tratar_linhas(texto)
    if df.empty:
        st.warning("⚠️ Formato não reconhecido ou extrato sem
movimentações.")
        return

    st.session_state.df_base = df.copy()
    st.session_state.df_edit = preparar_df(df)

    st.success("✅ Extrato lido e preparado com sucesso! Agora vá para
'Classificação (Extratos)'.")
    df_vis =
st.session_state.df_edit[["DATA","BANCO","AGENCIA_CONTA","DOCUMENTO","HISTORIC
O","VALOR"]].copy()
    df_vis = formatar_data_br(df_vis, "DATA")
```

```
    st.dataframe(
        df_vis.style.format({"VALOR":"{:,.2f}"}),
        use_container_width=True, height=420
    )

# ======================== PÁGINA: LANÇAMENTOS MANUAIS (SEPARADO)
========================
def pagina_lancamentos_manuais():
    st.title("□ Lançamentos Manuais (Caixa / Dinheiro)")
    st.markdown("Use esta tela para registrar **movimentações pagas em
dinheiro** ou que não aparecem no extrato bancário.")

    if "df_manual" not in st.session_state:
        st.session_state.df_manual = pd.DataFrame(columns=[
            "DATA","BANCO","AGENCIA_CONTA","DOCUMENTO","HISTORICO",
            "VALOR","NATUREZA","CULTURA","SAFRA","ATIVIDADES","INSUMOS"
        ])

    with st.form("form_manual"):
        col1, col2, col3 = st.columns(3)
        data = col1.date_input("Data da movimentação", datetime.today())
        banco = col2.text_input("Banco / Caixa", "CAIXA")
        valor = col3.number_input("Valor (R$)", min_value=0.0, step=0.01)

        col4, col5, col6 = st.columns(3)
        natureza = col4.selectbox("Natureza", NATUREZAS)
        cultura  = col5.selectbox("Cultura", CULTURAS)
        safra    = col6.text_input("Safra (ex.: 24/25)", "")

        col7, col8 = st.columns(2)
        atividade = col7.selectbox("Atividade", ATIVIDADES_OPCOES)
        insumo    = col8.selectbox("Insumo", INSUMOS_OPCOES)

        historico = st.text_area("Histórico / Descrição da movimentação")

        enviar = st.form_submit_button("➕ Adicionar Lançamento Manual")

        if enviar:
```

```
            novo = pd.DataFrame([{
                "DATA": data.strftime("%d/%m/%Y"),  # DD/MM/AAAA
                "BANCO": limpar_ui(banco),
                "AGENCIA_CONTA": "-",
                "DOCUMENTO": "-",
                "HISTORICO": limpar_ui(historico),
                "VALOR": valor,
                "NATUREZA": limpar_ui(natureza),
                "CULTURA": limpar_ui(cultura),
                "SAFRA": limpar_ui(safra),
                "ATIVIDADES": limpar_ui(atividade),
                "INSUMOS": limpar_ui(insumo)
            }])
            st.session_state.df_manual = pd.concat(
                [st.session_state.df_manual, novo], ignore_index=True
            )
            st.success("✅ Lançamento manual adicionado ao lote.")

    if not st.session_state.df_manual.empty:
        st.subheader("📋 Lançamentos Manuais em Lote (prontos para salvar)")
        dfm_vis = st.session_state.df_manual.copy()
        dfm_vis = formatar_data_br(dfm_vis, "DATA")
        st.dataframe(
            dfm_vis.style.format({"VALOR":"{:,.2f}"}),
            use_container_width=True, height=320
        )

        modo = st.radio("Modo de gravação (lançamentos manuais)",
                        ["PULAR DUPLICADOS","SOBREPOR DUPLICADOS"],
                        index=1, horizontal=True)
        modo_interno = "pular" if modo == "PULAR DUPLICADOS" else "sobrepor"

        if st.button("💾 Salvar Lançamentos Manuais no MySQL"):
            df_save = st.session_state.df_manual.copy()
            for c in
["NATUREZA","CULTURA","SAFRA","ATIVIDADES","INSUMOS","BANCO","HISTORICO"]:
                df_save[c] = df_save[c].astype(str).map(limpar_ui)
```

```
            res = salvar_mysql(df_save, modo=modo_interno)
            st.success(
                f"LANÇAMENTOS MANUAIS → Inseridos: {res.get('inseridos',0)} |
"
                f"Atualizados: {res.get('atualizados',0)} | Ignorados:
{res.get('ignorados',0)} | "
                f"Erros: {res.get('erros',0)}"
            )
            # Se quiser limpar o lote depois:
            # st.session_state.df_manual =
st.session_state.df_manual.iloc[0:0]

# ======================== PÁGINA: CLASSIFICAÇÃO (EXTRATOS)
========================
def pagina_classificacao_extratos():
    st.title("⚙ Classificação e Lançamentos , EXTRATOS")

    # --- Verifica se o extrato está carregado ---
    if "df_edit" not in st.session_state or st.session_state.df_edit is None:
        st.warning("Nenhum extrato carregado. Use primeiro a opção '⬇ Extrato
Bancário (PDF)'.")
        return

    base = st.session_state.df_edit.copy()

    # --- Padroniza nomes das colunas (evita erro de maiúscula/minúscula e
espaços) ---
    base.columns = base.columns.str.strip().str.upper()

    # --- Garante que todas as colunas esperadas existam ---
    colunas_necessarias = [
        "DATA", "BANCO", "AGENCIA_CONTA", "DOCUMENTO", "HISTORICO", "VALOR",
        "NATUREZA", "CULTURA", "SAFRA", "ATIVIDADES", "INSUMOS"
    ]
    for c in colunas_necessarias:
        if c not in base.columns:
            base[c] = ""
```

```
    # --- Exibe visualização geral ---
    with st.expander("Visualização geral do extrato (somente leitura)",
expanded=False):
        df_vis = base[["DATA", "BANCO", "AGENCIA_CONTA", "DOCUMENTO",
"HISTORICO", "VALOR"]].copy()
        df_vis = formatar_data_br(df_vis, "DATA")
        st.dataframe(
            df_vis.style.format({"VALOR": "{:,.2f}"}),
            use_container_width=True,
            height=320
        )

    # --- Filtros e classificação ---
    st.subheader("Classificação , Natureza (obrigatório)")
    modo = st.radio("MODO DE EDIÇÃO", ["LINHA A LINHA (CLICAR)", "TABELA
(GRADE)"], index=0, horizontal=True)

    colf1, colf2, colf3 = st.columns([1.2, 0.9, 1.3])
    with colf1:
        bancos_disp = sorted(base["BANCO"].dropna().unique())
        bancos_sel = st.multiselect("FILTRAR POR BANCO", bancos_disp,
default=bancos_disp, key="flt_bancos")

    with colf2:
        nat_disp = ["PENDENTES", "RECEITA RURAL", "DESPESA RURAL", "CUSTOS
RURAL", "INVESTIMENTO RURAL" "RECEITAS - NAO RURAL", "DESPESAS - NAO RURAL",
"OUTRAS"]
        nat_sel = st.multiselect("FILTRAR POR NATUREZA", nat_disp,
default=nat_disp, key="flt_nat")

    with colf3:
        busca = limpar_ui(st.text_input("BUSCAR NO HISTORICO (CONTEM)", "",
key="flt_busca"))

    # --- Monta máscara de filtro ---
    nats_real = []
    if "PENDENTES" in nat_sel:
        nats_real.append("")
```

```
    nats_real += [c for c in nat_sel if c != "PENDENTES"]

    # Filtro com segurança
    mask_view = base["BANCO"].isin(bancos_sel)
    if "NATUREZA" in base.columns:
        mask_view &= base["NATUREZA"].isin(nats_real)
    else:
        st.warning("⚠ Coluna 'NATUREZA' não encontrada na base de dados.
Exibindo todas as linhas.")

    if busca:
        mask_view &= base["HISTORICO"].str.contains(busca, na=False,
case=False)

    view = base.loc[mask_view].copy()

    # --- Barra de progresso ---
    total = len(base)
    pend = int(pend_mask(base).sum()) if "pend_mask" in globals() else 0
    prog = int(100 * (total - pend) / max(1, total))
    st.progress(prog / 100.0, text=f"PROGRESSO: {total - pend}/{total} LINHAS
OK ({prog}%)")

    # --- Prévia com destaque ---
    st.markdown("**Prévia (linhas pendentes em vermelho):**")
    prev_cols = ["DATA", "BANCO", "AGENCIA_CONTA", "DOCUMENTO", "HISTORICO",
"VALOR",
                 "NATUREZA", "CULTURA", "SAFRA", "ATIVIDADES", "INSUMOS"]
    prev_vis = view[prev_cols].copy()
    st.dataframe(styler_pendencias(prev_vis), use_container_width=True,
height=260)

    # ============== AÇÕES EM LOTE ==============
        # ============== AÇÕES EM LOTE ==============
    with st.expander("⚡ Ações em lote , aplicar nas linhas filtradas",
expanded=False):
        escopo = st.radio("ESCOPO", ["TODAS AS FILTRADAS","APENAS PENDENTES"],
                          index=0, horizontal=True, key="lote_escopo")
```

```
        colb1, colb2, colb3 = st.columns([1,1,1])
        with colb1:
            natureza_lote = st.selectbox("NATUREZA (opcional)", NATUREZAS,
index=0, key="lote_nat")
        with colb2:
            cultura_lote  = st.selectbox("CULTURA (opcional)", CULTURAS,
index=0, key="lote_cult")
            safra_lote    = st.text_input("SAFRA (opcional, ex.: 24/25)", "",
key="lote_safra")
        with colb3:
            atv_lote  = st.selectbox("ATIVIDADES (opcional)",
ATIVIDADES_OPCOES, index=0, key="lote_atv")
            ins_lote  = st.selectbox("INSUMOS (opcional)", INSUMOS_OPCOES,
index=0, key="lote_ins")

        dfm = st.session_state.df_edit
        nats_real2 = []
        if "PENDENTES" in nat_sel:
            nats_real2.append("")
        nats_real2 += [c for c in nat_sel if c != "PENDENTES"]

        mask = (dfm["BANCO"].isin(bancos_sel)) &
(dfm["NATUREZA"].isin(nats_real2))
        if busca:
            mask &= dfm["HISTORICO"].str.contains(busca, na=False)
        if escopo == "APENAS PENDENTES":
            mask &= pend_mask(dfm)

        previa = dfm.loc[mask, prev_cols]
        st.info(f"Serão afetadas {len(previa)} linha(s).")
        st.dataframe(styler_pendencias(previa), use_container_width=True,
height=240)

        algo_escolhido = any([
            natureza_lote.strip(),
            cultura_lote.strip(),
            safra_lote.strip(),
```

```
            atv_lote.strip(),
            ins_lote.strip()
        ])

        if st.button("✓ Aplicar aos filtrados", use_container_width=True,
key="btn_aplicar_lote"):
            if len(dfm.loc[mask]) == 0:
                st.info("Nada a aplicar no filtro atual.")
            elif not algo_escolhido:
                st.warning("Selecione ao menos um campo para alterar em
lote.")
            else:
                if natureza_lote.strip() != "":
                    dfm.loc[mask, "NATUREZA"]  = limpar_ui(natureza_lote)
                if cultura_lote.strip() != "":
                    dfm.loc[mask, "CULTURA"]   = limpar_ui(cultura_lote)
                if safra_lote.strip() != "":
                    dfm.loc[mask, "SAFRA"]     = limpar_ui(safra_lote)
                if atv_lote.strip() != "":
                    dfm.loc[mask, "ATIVIDADES"] = limpar_ui(atv_lote)
                if ins_lote.strip() != "":
                    dfm.loc[mask, "INSUMOS"]    = limpar_ui(ins_lote)

                st.session_state.df_edit = dfm
                st.success(f"Aplicado em {int(mask.sum())} linha(s).")
                # ✓ sem rerun, sem set_query_params , o próprio clique já
reroda o script


    # ======== MODO CARTÕES (CLASSIFICAÇÃO INDIVIDUAL + SALVAR LINHA) ========
    if modo == "LINHA A LINHA (CLICAR)":
        colp1, colp2, colp3 = st.columns([1,1,2])
        with colp1:
            page_size = st.number_input("Itens por página", 5, 100, 15, 5,
key="card_page_size")
        total_rows = len(view)
        total_pages = max(1, math.ceil(total_rows / page_size))
        with colp2:
```

```
            page = st.number_input("Página", 1, total_pages, 1, 1,
key="card_page")
        with colp3:
            st.caption(f"Total filtrado: {total_rows} | Páginas:
{total_pages}")
        start = (page - 1) * page_size
        subset = view.iloc[start:start+page_size]

        st.divider()
        for idx, row in subset.iterrows():
            pendente = (row["NATUREZA"] == "")
            barra = "● PENDENTE" if pendente else "✓ OK"
            with st.container():
                st.markdown(f"**STATUS:** {barra}")
                c1, c2, c3, c4 = st.columns([1.3, 1, 1.2, 1.5])
                with c1:
                    st.write(f"DATA: {row['DATA']}")
                    st.write(f"BANCO: {row['BANCO']}")
                    st.write(f"HISTÓRICO: {row['HISTORICO'][:120]}")
                with c2:
                    st.write(f"AG/CONTA: {row['AGENCIA_CONTA']}")
                    st.write(f"DOC: {row['DOCUMENTO'] if row['DOCUMENTO'] else
'-'}")
                    st.write(f"VALOR: {row['VALOR']:.2f}")
                with c3:
                    natureza = st.selectbox("NATUREZA", NATUREZAS,
                        index=(NATUREZAS.index(row["NATUREZA"]) if
row["NATUREZA"] in NATUREZAS else 0),
                        key=f"card_nat_{idx}")
                    st.session_state.df_edit.at[idx, "NATUREZA"] =
limpar_ui(natureza or "")
                    cultura = st.selectbox("CULTURA", CULTURAS,
                        index=(CULTURAS.index(row["CULTURA"]) if
row["CULTURA"] in CULTURAS else 0),
                        key=f"card_cult_{idx}")
                    st.session_state.df_edit.at[idx, "CULTURA"] =
limpar_ui(cultura or "")
```

```
                    safra = st.text_input("SAFRA (ex.: 24/25)",
value=row.get("SAFRA",""), key=f"card_saf_{idx}")
                    st.session_state.df_edit.at[idx, "SAFRA"] =
limpar_ui(safra or "")
                with c4:
                    atv_atual = row["ATIVIDADES"] if row["ATIVIDADES"] in
ATIVIDADES_OPCOES else ""
                    atividade = st.selectbox("ATIVIDADES", ATIVIDADES_OPCOES,
                        index=(ATIVIDADES_OPCOES.index(atv_atual) if atv_atual
in ATIVIDADES_OPCOES else 0),
                        key=f"card_atv_{idx}")
                    st.session_state.df_edit.at[idx, "ATIVIDADES"] =
limpar_ui(atividade or "")
                    ins_atual = row["INSUMOS"] if row["INSUMOS"] in
INSUMOS_OPCOES else ""
                    ins = st.selectbox("INSUMOS", INSUMOS_OPCOES,
                        index=(INSUMOS_OPCOES.index(ins_atual) if ins_atual in
INSUMOS_OPCOES else 0),
                        key=f"card_ins_{idx}")
                    st.session_state.df_edit.at[idx, "INSUMOS"] =
limpar_ui(ins or "")

                # ===== BOTÃO PARA SALVAR APENAS ESTA LINHA =====
                collinha1, collinha2 = st.columns([1,3])
                with collinha1:
                    if st.button("💾 Salvar esta linha no MySQL",
key=f"btn_salvar_linha_{idx}"):
                        linha = st.session_state.df_edit.loc[[idx]].copy()
                        for c_nome in
["NATUREZA","CULTURA","SAFRA","ATIVIDADES","INSUMOS",

"BANCO","HISTORICO","AGENCIA_CONTA","DOCUMENTO"]:
                            if c_nome in linha.columns:
                                linha[c_nome] =
linha[c_nome].astype(str).map(limpar_ui)
                        res_linha = salvar_mysql(linha, modo="sobrepor")
                        st.success(
```

```
                            f"Linha salva/atualizada. Inseridos:
{res_linha.get('inseridos',0)} | "
                            f"Atualizados: {res_linha.get('atualizados',0)} |
"
                            f"Ignorados: {res_linha.get('ignorados',0)} | "
                            f"Erros: {res_linha.get('erros',0)}"
                        )

    # ======== MODO GRADE ========
    else:
        st.caption("Edite na grade. Obrigatório: NATUREZA. Opcionais:
CULTURA/SAFRA/ATIVIDADES/INSUMOS.")
        view_cols = ["DATA","BANCO","AGENCIA_CONTA","DOCUMENTO","HISTORICO",

"VALOR","NATUREZA","CULTURA","SAFRA","ATIVIDADES","INSUMOS"]

        grid = view[view_cols].copy()
        grid = formatar_data_br(grid, "DATA")
        mask_p = pend_mask(grid)
        grid.insert(0, "STATUS", mask_p.map({True:"PENDENTE", False:"OK"}))

        edited = st.data_editor(
            grid,
            use_container_width=True, height=520, num_rows="fixed",
hide_index=False,
            column_config={
                "VALOR":      st.column_config.NumberColumn("VALOR",
format="%.2f"),
                "NATUREZA":   st.column_config.SelectboxColumn("NATUREZA",
options=NATUREZAS, required=False),
                "CULTURA":    st.column_config.SelectboxColumn("CULTURA",
options=CULTURAS, required=False),
                "SAFRA":      st.column_config.TextColumn("SAFRA (ex.:
24/25)"),
                "ATIVIDADES": st.column_config.SelectboxColumn("ATIVIDADES",
options=ATIVIDADES_OPCOES, required=False),
                "INSUMOS":    st.column_config.SelectboxColumn("INSUMOS",
options=INSUMOS_OPCOES, required=False),
```

```
            },
            disabled=["STATUS"],
            key="grid_editor"
        )
        for c in ["NATUREZA","CULTURA","SAFRA","ATIVIDADES","INSUMOS"]:
            edited[c] = edited[c].astype(str).map(limpar_ui)
        st.session_state.df_edit.loc[edited.index,
["NATUREZA","CULTURA","SAFRA","ATIVIDADES","INSUMOS"]] = \
            edited[["NATUREZA","CULTURA","SAFRA","ATIVIDADES","INSUMOS"]]

    # ======================== SALVAMENTO PARCIAL EM LOTE
========================
    st.markdown("---")
    st.subheader("💾 Salvamento em andamento (parcial)")

    st.caption("""
    Use este botão para **salvar o lote inteiro como está agora** no banco
MySQL.
    Mesmo com linhas pendentes, você poderá voltar, ajustar e salvar novamente
em modo **SOBREPOR**.
    """)

    col_s1, col_s2 = st.columns([1,2])
    with col_s1:
        modo_parcial = st.radio(
            "Modo de salvamento parcial",
            ["SOBREPOR DUPLICADOS","PULAR DUPLICADOS"],
            index=0,
            horizontal=False,
            key="modo_parcial_extrato"
        )
    modo_parcial_int = "sobrepor" if modo_parcial == "SOBREPOR DUPLICADOS"
else "pular"

    with col_s2:
        if st.button("💾 Salvar andamento no MySQL (parcial)",
use_container_width=True, key="btn_salvar_parcial_extrato"):
            df_parcial = st.session_state.df_edit.copy()
```

```
            for c in ["NATUREZA","CULTURA","SAFRA","ATIVIDADES","INSUMOS",
                      "BANCO","HISTORICO","AGENCIA_CONTA","DOCUMENTO"]:
                if c in df_parcial.columns:
                    df_parcial[c] = df_parcial[c].astype(str).map(limpar_ui)
            res = salvar_mysql(df_parcial, modo=modo_parcial_int)
            st.success(
                f"PARCIAL SALVO → Inseridos: {res.get('inseridos',0)} | "
                f"Atualizados: {res.get('atualizados',0)} | Ignorados:
{res.get('ignorados',0)} | "
                f"Erros: {res.get('erros',0)}"
            )
            st.caption("Você pode continuar classificando com tranquilidade. O
que já foi feito está guardado no banco. ✅")



# ======================== PÁGINA: CONSULTAR / EDITAR / EXCLUIR NO BANCO
========================
def pagina_consultar_editar_banco():
    import streamlit as st
    import pandas as pd
    import pymysql
    from datetime import datetime
    from funcoes import DB_CONFIG

    st.title("🗃️ Lançamentos no Banco (Consulta / Edição / Exclusão)")

    # ------------------- FILTRO DE BUSCA -------------------
    st.subheader("🔍 Filtro de pesquisa")
    with st.expander("Clique para abrir o filtro"):
        col1, col2, col3, col4 = st.columns(4)
        data_ini = col1.date_input("Data inicial", datetime(2024, 1, 1))
        data_fim = col2.date_input("Data final", datetime.today())
        banco_f = col3.text_input("Banco (opcional)")
        nat_f = col4.text_input("Natureza (opcional)")
        hist_f = st.text_input("Trecho do histórico (opcional)")
```

```
if st.button("🔎 Buscar lançamentos"):
    try:
        conn = pymysql.connect(**DB_CONFIG)
        cur = conn.cursor()

        sql = """
            SELECT
                id,
                DATE_FORMAT(data,'%%d/%%m/%%Y') AS data,
                banco,
                agencia_conta,
                documento,
                historico,
                valor,
                natureza,
                cultura,
                safra,
                atividades,
                insumos
            FROM extrato
            WHERE data BETWEEN %s AND %s
        """
        params = [data_ini, data_fim]

        if banco_f:
            sql += " AND banco LIKE %s"
            params.append(f"%{banco_f}%")
        if nat_f:
            sql += " AND natureza LIKE %s"
            params.append(f"%{nat_f}%")
        if hist_f:
            sql += " AND historico LIKE %s"
            params.append(f"%{hist_f}%")

        sql += " ORDER BY data DESC"
        cur.execute(sql, params)
        rows = cur.fetchall()
        colnames = [d[0] for d in cur.description]
```

```
            cur.close()
            conn.close()

            if not rows:
                st.warning("Nenhum lançamento encontrado para o filtro
informado.")
                st.session_state.pop("df_db", None)
            else:
                df_db = pd.DataFrame(rows, columns=colnames)
                st.session_state["df_db"] = df_db

        except Exception as e:
            st.error(f"Erro ao consultar o banco: {e}")
            return

    # ------------------- EXIBIÇÃO E EDIÇÃO -------------------
    if "df_db" in st.session_state:
        st.divider()
        st.subheader("✏️ Resultados (clique nas células para editar)")
        df_db = st.session_state["df_db"]

        # Editor interativo
        edited_df = st.data_editor(
            df_db,
            num_rows="dynamic",
            key="edicao_banco",
            use_container_width=True,
        )

        # Botão para salvar alterações
        if st.button("💾 Salvar alterações"):
            try:
                conn = pymysql.connect(**DB_CONFIG)
                cur = conn.cursor()
                for _, row in edited_df.iterrows():
                    cur.execute("""
                        UPDATE extrato SET
                            data=%s, banco=%s, agencia_conta=%s, documento=%s,
```

```
                            historico=%s, valor=%s, natureza=%s, cultura=%s,
                            safra=%s, atividades=%s, insumos=%s
                        WHERE id=%s
                    """, (
                        datetime.strptime(row["data"], "%d/%m/%Y").date(),
                        row["banco"], row["agencia_conta"], row["documento"],
                        row["historico"], row["valor"], row["natureza"],
                        row["cultura"], row["safra"], row["atividades"],
                        row["insumos"], row["id"]
                    ))
                conn.commit()
                cur.close()
                conn.close()
                st.success("✅ Alterações salvas com sucesso!")
            except Exception as e:
                st.error(f"Erro ao salvar: {e}")

        # Botão para exclusão
        if st.button("🗑️ Excluir selecionados"):
            try:
                selecionados = edited_df.loc[edited_df.get("excluir", False)
== True]
                if not selecionados.empty:
                    conn = pymysql.connect(**DB_CONFIG)
                    cur = conn.cursor()
                    for _, row in selecionados.iterrows():
                        cur.execute("DELETE FROM extrato WHERE id=%s",
(row["id"],))
                    conn.commit()
                    cur.close()
                    conn.close()
                    st.warning(f"{len(selecionados)} lançamento(s)
excluído(s). Atualize a busca.")
                else:
                    st.info("Nenhum item marcado para exclusão.")
            except Exception as e:
                st.error(f"Erro ao excluir: {e}")
```

```
    # ------------------- EXIBIÇÃO E EDIÇÃO -------------------
    if "df_db" in st.session_state:
        st.subheader("□ Lançamentos encontrados no banco")

        df_db = st.session_state["df_db"].copy()
        df_db = formatar_data_br(df_db, "DATA")

        # Checkbox global
        marcar_todos = st.checkbox("Marcar todos os lançamentos para
EXCLUSÃO", key="chk_marcar_todos_excluir")
        if marcar_todos:
            df_db["EXCLUIR"] = True

        # --- EDITOR COM CHAVE DINÂMICA PARA EVITAR ERRO DE DOM ---
        editor_key = f"editor_db_{len(df_db)}_{int(datetime.now().timestamp()
* 1000)}"
        with st.container():
            edited = st.data_editor(
                df_db,
                width="stretch",           # substitui o antigo
use_container_width
                height=520,
                num_rows="fixed",
                column_config={
                    "id": st.column_config.NumberColumn("ID", disabled=True),
                    "VALOR": st.column_config.NumberColumn("VALOR",
format="%.2f"),
                    "EXCLUIR": st.column_config.CheckboxColumn("Excluir?",
default=False),
                    "NATUREZA": st.column_config.SelectboxColumn("NATUREZA",
options=NATUREZAS),
                    "CULTURA": st.column_config.SelectboxColumn("CULTURA",
options=CULTURAS),
                    "ATIVIDADES":
st.column_config.SelectboxColumn("ATIVIDADES", options=ATIVIDADES_OPCOES),
                    "INSUMOS": st.column_config.SelectboxColumn("INSUMOS",
options=INSUMOS_OPCOES),
                },
```

```
                key=editor_key
            )


        st.markdown("Marque **EXCLUIR** nas linhas que deseja apagar e/ou
altere os campos desejados.")
        colb1, colb2 = st.columns(2)

        # ------------------- EXCLUSÃO -------------------
                # ------------------- EXCLUSÃO -------------------
        with colb1:
            if st.button("🗑️ Excluir marcados do banco",
use_container_width=True):
                apagar = edited[edited["EXCLUIR"] == True]
                if apagar.empty:
                    st.info("Nenhuma linha marcada para exclusão.")
                else:
                    try:
                        conn = pymysql.connect(**DB_CONFIG)
                        cur = conn.cursor()
                        for _, linha in apagar.iterrows():
                            cur.execute("DELETE FROM extrato WHERE id = %s",
(int(linha["id"]),))
                        conn.commit(); cur.close(); conn.close()

                        st.success(f"{len(apagar)} linha(s) excluída(s) do
banco.")

                        # Atualiza a base local: tira do DataFrame as linhas
excluídas
                        st.session_state["df_db"] = edited[edited["EXCLUIR"]
== False].copy()

                    except Exception as e:
                        st.error(f"Erro ao excluir: {e}")


        # ------------------- ATUALIZAÇÃO -------------------
```

```
        with colb2:
            if st.button("💾 Aplicar alterações (UPDATE)",
use_container_width=True):
                manter = edited[edited["EXCLUIR"] == False].copy()
                if manter.empty:
                    st.info("Nenhuma linha para atualizar.")
                else:
                    try:
                        conn = pymysql.connect(**DB_CONFIG)
                        cur = conn.cursor()
                        sql_upd = """
                            UPDATE extrato
                               SET data=%s,
                                   banco=%s,
                                   agencia_conta=%s,
                                   documento=%s,
                                   historico=%s,
                                   valor=%s,
                                   natureza=%s,
                                   cultura=%s,
                                   safra=%s,
                                   atividades=%s,
                                   insumos=%s
                             WHERE id=%s
                        """
                        upd_ok = 0
                        for _, linha in manter.iterrows():
                            try:
                                data_fmt =
datetime.strptime(str(linha["DATA"]), "%d/%m/%Y").date()
                            except Exception:
                                continue
                            cur.execute(sql_upd, (
                                data_fmt,
                                limpar_ui(linha["BANCO"]),
                                limpar_ui(linha["AGENCIA_CONTA"]),
                                limpar_ui(linha.get("DOCUMENTO","")),
                                limpar_ui(linha["HISTORICO"]),
```

```
                                    float(linha["VALOR"]),
                                    limpar_ui(linha.get("NATUREZA","")),
                                    limpar_ui(linha.get("CULTURA","")),
                                    limpar_ui(linha.get("SAFRA","")),
                                    limpar_ui(linha.get("ATIVIDADES","")),
                                    limpar_ui(linha.get("INSUMOS","")),
                                    int(linha["id"]),
                                ))
                                upd_ok += 1
                            conn.commit(); cur.close(); conn.close()
                            st.success(f"Atualização concluída: {upd_ok} linha(s)
atualizada(s) no banco.")
                        except Exception as e:
                            st.error(f"Erro ao atualizar: {e}")
    # 🔄 Atualiza a página com segurança fora dos containers interativos
if st.session_state.get("refresh_consulta"):
    st.session_state.pop("refresh_consulta")
    st.experimental_set_query_params(updated=str(datetime.now().timestamp()))


# ======================== PÁGINA: SALVAR EXTRATOS (FECHAMENTO)
========================
def pagina_salvar_extratos():
    st.title("💾 Salvar EXTRATOS no MySQL")

    # Garante que há extrato carregado/classificado
    if "df_edit" not in st.session_state or st.session_state.df_edit is None:
        st.warning("Nenhum extrato carregado. Use primeiro '📥 Extrato
Bancário (PDF)' e a classificação.")
        return

    df_final = st.session_state.df_edit.copy()

    # Verifica se ainda existem linhas com NATUREZA pendente
    faltas = pend_mask(df_final)
    qtd_pend = int(faltas.sum())
    if qtd_pend > 0:
```

```
        st.error("Ainda existem linhas pendentes (NATUREZA vazia). Finalize a
classificação antes de salvar.")
        st.dataframe(
            df_final.loc[faltas,
["DATA","BANCO","AGENCIA_CONTA","DOCUMENTO","HISTORICO",

"VALOR","NATUREZA","CULTURA","SAFRA","ATIVIDADES","INSUMOS"]],
            use_container_width=True, height=280
        )
        return

    # Escolha do modo de inserção
    modo = st.radio("Modo de inserção dos EXTRATOS",
                    ["PULAR DUPLICADOS","SOBREPOR DUPLICADOS"],
                    index=0, horizontal=True)
    modo_interno = "pular" if modo == "PULAR DUPLICADOS" else "sobrepor"

    # Normaliza textos antes de salvar
    for c in
["NATUREZA","CULTURA","SAFRA","ATIVIDADES","INSUMOS","BANCO","HISTORICO","AGEN
CIA_CONTA","DOCUMENTO"]:
        if c in df_final.columns:
            df_final[c] = df_final[c].astype(str).map(limpar_ui)

    st.subheader("Prévia final antes de salvar")
    prev = df_final[["DATA","BANCO","AGENCIA_CONTA","DOCUMENTO","HISTORICO",

"VALOR","NATUREZA","CULTURA","SAFRA","ATIVIDADES","INSUMOS"]].copy()
    prev = formatar_data_br(prev, "DATA")
    st.dataframe(
        prev.style.format({"VALOR":"{:,.2f}"}),
        use_container_width=True, height=320
    )

    if st.button("💾 Salvar EXTRATOS no MySQL agora",
use_container_width=True):
        res = salvar_mysql(df_final, modo=modo_interno)
        st.success(
```

```
            f"EXTRATOS → Inseridos: {res.get('inseridos',0)} | "
            f"Atualizados: {res.get('atualizados',0)} | "
            f"Ignorados: {res.get('ignorados',0)} | "
            f"Erros: {res.get('erros',0)}"
        )
        st.caption("Base atualizada com sucesso. Dados prontos para
relatórios. ✅")

# ======================== PÁGINA: RELATÓRIOS – RESUMO LIVRO CAIXA RURAL
(LCDPR) ========================
def pagina_relatorios():
    from reportlab.platypus import SimpleDocTemplate, Table, TableStyle,
Paragraph
    from reportlab.lib.pagesizes import A4
    from reportlab.lib import colors
    from reportlab.lib.styles import getSampleStyleSheet

    st.title("📊 Relatório Resumo Livro Caixa Rural (LCDPR)")

    with st.form("filtro_relatorio"):
        col1, col2 = st.columns(2)
        data_ini = col1.date_input("Data inicial", datetime(2024, 1, 1))
        data_fim = col2.date_input("Data final", datetime.today())
        gerar = st.form_submit_button("Gerar Relatório Rural (LCDPR)")

    if not gerar:
        return

    try:
        # ------------------------------
        # BUSCA DO BANCO
        # ------------------------------
        conn = pymysql.connect(**DB_CONFIG)
        cur = conn.cursor()

        sql = """
            SELECT
                MONTH(data) AS MES_NUM,
```

```
            natureza,
            SUM(valor) AS TOTAL
        FROM extrato
        WHERE data BETWEEN %s AND %s
        GROUP BY MES_NUM, natureza
        ORDER BY MES_NUM
    """

    cur.execute(sql, (data_ini, data_fim))
    linhas = cur.fetchall()
    conn.close()

    # ----------------------------
    # BASE DOS MESES
    # ----------------------------
    meses_label =
["JAN","FEV","MAR","ABR","MAI","JUN","JUL","AGO","SET","OUT","NOV","DEZ"]

    tabela = pd.DataFrame({
        "MÊS": meses_label,
        "RECEITA BRUTA": [0]*12,
        "DESPESAS RURAIS / CUSTO RURAL / INVESTIMENTO RURAL": [0]*12
    })

    # ----------------------------
    # DEFINIÇÕES RURAIS
    # ----------------------------
    RECEITAS_RURAIS = ["RECEITA RURAL"]

    DESPESAS_RURAIS = [
        "DESPESA RURAL",
        "CUSTO RURAL",
        "INVESTIMENTO RURAL"
    ]

    # ----------------------------
    # PREENCHER VALORES MÊS A MÊS
    # ----------------------------
```

```
        for mes_num, natureza, total in linhas:
            idx = mes_num - 1
            natureza = (natureza or "").upper().strip()

            if natureza in RECEITAS_RURAIS:
                tabela.loc[idx, "RECEITA BRUTA"] += float(total)

            if natureza in DESPESAS_RURAIS:
                tabela.loc[idx, "DESPESAS RURAIS / CUSTO RURAL / INVESTIMENTO
RURAL"] += float(total)

        # ------------------------------
        # LINHA TOTAL
        # ------------------------------
        linha_total = pd.DataFrame({
            "MÊS": ["TOTAL"],
            "RECEITA BRUTA": [tabela["RECEITA BRUTA"].sum()],
            "DESPESAS RURAIS / CUSTO RURAL / INVESTIMENTO RURAL": [
                tabela["DESPESAS RURAIS / CUSTO RURAL / INVESTIMENTO
RURAL"].sum()
            ]
        })

        tabela_final = pd.concat([tabela, linha_total], ignore_index=True)

        # ------------------------------
        # EXIBIR NA TELA
        # ------------------------------
        st.subheader("📃 Resumo Livro Caixa Rural – LCDPR")
        st.dataframe(
            tabela_final.style.format({
                "RECEITA BRUTA": "R$ {:,.2f}",
                "DESPESAS RURAIS / CUSTO RURAL / INVESTIMENTO RURAL": "R$
{:,.2f}"
            }),
            use_container_width=True
        )
```

```
        # ------------------------------
        # EXPORTAR CSV / EXCEL
        # ------------------------------
        csv = tabela_final.to_csv(index=False).encode("utf-8")
        st.download_button(
            label="⬇️ Exportar Excel/CSV – Livro Caixa Rural",
            data=csv,
            file_name="resumo_livro_caixa_rural_lcdpr.csv",
            mime="text/csv"
        )

        # ------------------------------
        # EXPORTAR PDF
        # ------------------------------
        if st.button("📄 Exportar PDF – Resumo Livro Caixa Rural (LCDPR)"):
            nome_pdf = "resumo_livro_caixa_rural_lcdpr.pdf"

            styles = getSampleStyleSheet()
            conteudo = []

            conteudo.append(Paragraph("<b>Resumo Livro Caixa Rural – 
LCDPR</b>", styles["Title"]))
            conteudo.append(Paragraph(f"Período: {data_ini} até {data_fim}", 
styles["Normal"]))
            conteudo.append(Paragraph(" ", styles["Normal"]))

            dados_pdf = [tabela_final.columns.tolist()] + 
tabela_final.astype(str).values.tolist()

            tabela_pdf = Table(dados_pdf)
            tabela_pdf.setStyle(TableStyle([
                ("BACKGROUND", (0,0), (-1,0), colors.lightgrey),
                ("GRID", (0,0), (-1,-1), 0.5, colors.black),
                ("ALIGN", (0,0), (-1,-1), "CENTER"),
                ("FONTNAME", (0,0), (-1,0), "Helvetica-Bold"),
                ("FONTSIZE", (0,0), (-1,-1), 9),
            ]))
```

```
            conteudo.append(tabela_pdf)

            caminho_pdf = f"/mnt/data/{nome_pdf}"
            pdf = SimpleDocTemplate(caminho_pdf, pagesize=A4)
            pdf.build(conteudo)

            with open(caminho_pdf, "rb") as f:
                st.download_button(
                    "📄 Baixar PDF – Resumo Livro Caixa Rural (LCDPR)",
                    data=f.read(),
                    file_name=nome_pdf,
                    mime="application/pdf"
                )

    except Exception as e:
        st.error(f"Erro ao gerar relatório: {e}")


# ======================== RELATÓRIO DETALHADO (ESCRITURAÇÃO RURAL)
========================
def pagina_relatorio_detalhado():
    from reportlab.platypus import SimpleDocTemplate, Table, TableStyle,
Paragraph
    from reportlab.lib.pagesizes import A4
    from reportlab.lib import colors
    from reportlab.lib.styles import getSampleStyleSheet

    st.title("📃 Relatório Detalhado – Escrituração Rural (LCDPR)")

    with st.form("filtro_detalhado"):
        col1, col2, col3 = st.columns(3)
        data_ini = col1.date_input("Data inicial", datetime(2024, 1, 1))
        data_fim = col2.date_input("Data final", datetime.today())
        cultura = col3.text_input("Cultura (opcional)").strip()

        col4, col5 = st.columns(2)
        natureza = col4.text_input("Natureza (opcional)").strip()
        safra = col5.text_input("Safra (opcional)").strip()
```

```
    gerar = st.form_submit_button("Gerar Relatório Detalhado")

if not gerar:
    return

try:
    conn = pymysql.connect(**DB_CONFIG)
    cur = conn.cursor()

    sql = """
        SELECT
            data,
            documento,
            cultura,
            safra,
            natureza,
            valor,
            historico
        FROM extrato
        WHERE data BETWEEN %s AND %s
    """

    params = [data_ini, data_fim]

    if cultura:
        sql += " AND cultura = %s"
        params.append(limpar_ui(cultura))

    if natureza:
        sql += " AND natureza = %s"
        params.append(natureza.upper())

    if safra:
        sql += " AND safra = %s"
        params.append(safra.upper())

    sql += " ORDER BY data ASC"
```

```
        cur.execute(sql, params)
        linhas = cur.fetchall()
        conn.close()

        if not linhas:
            st.warning("Nenhuma movimentação encontrada para os filtros
selecionados.")
            return

        colunas = ["DATA", "DOCUMENTO",
                   "CULTURA", "SAFRA", "NATUREZA", "VALOR", "HISTÓRICO"]

        df = pd.DataFrame(linhas, columns=colunas)
        df["VALOR"] = pd.to_numeric(df["VALOR"], errors="coerce").fillna(0)

        st.dataframe(df, use_container_width=True, height=400)

        # Exportar CSV
        csv = df.to_csv(index=False).encode("utf-8")
        st.download_button(
            label="⬇️ Exportar Excel/CSV",
            data=csv,
            file_name="relatorio_detalhado_lcdpr.csv",
            mime="text/csv"
        )

        # Exportar PDF
        if st.button("📄 Exportar PDF – Relatório Detalhado"):
            nome_pdf = "relatorio_detalhado_lcdpr.pdf"

            styles = getSampleStyleSheet()
            conteudo = []

            conteudo.append(Paragraph("<b>Relatório Detalhado – Escrituração
Rural (LCDPR)</b>", styles["Title"]))
            conteudo.append(Paragraph(f"Período: {data_ini} até {data_fim}",
styles["Normal"]))
```

```
            conteudo.append(Paragraph(" ", styles["Normal"]))

            dados_pdf = [df.columns.tolist()] + df.astype(str).values.tolist()

            tabela_pdf = Table(dados_pdf)
            tabela_pdf.setStyle(TableStyle([
                ("BACKGROUND", (0,0), (-1,0), colors.lightgrey),
                ("GRID", (0,0), (-1,-1), 0.5, colors.black),
                ("ALIGN", (0,0), (-1,-1), "CENTER"),
                ("FONTNAME", (0,0), (-1,0), "Helvetica-Bold"),
                ("FONTSIZE", (0,0), (-1,-1), 7),
            ]))

            conteudo.append(tabela_pdf)

            caminho_pdf = f"/mnt/data/{nome_pdf}"
            pdf = SimpleDocTemplate(caminho_pdf, pagesize=A4)
            pdf.build(conteudo)

            with open(caminho_pdf, "rb") as f:
                st.download_button(
                    label="📄 Baixar PDF – Relatório Detalhado",
                    data=f.read(),
                    file_name=nome_pdf,
                    mime="application/pdf"
                )

    except Exception as e:
        st.error(f"Erro ao gerar relatório detalhado: {e}")



# ======================== RELATÓRIO POR CULTURA (GERENCIAL)
========================
def pagina_relatorio_por_cultura():
    from reportlab.platypus import SimpleDocTemplate, Table, TableStyle,
Paragraph
    from reportlab.lib.pagesizes import A4
```

```
from reportlab.lib import colors
from reportlab.lib.styles import getSampleStyleSheet

st.title("▨ Relatório por Cultura – Gerencial")

with st.form("filtro_cultura"):
    col1, col2 = st.columns(2)
    data_ini = col1.date_input("Data inicial", datetime(2024, 1, 1))
    data_fim = col2.date_input("Data final", datetime.today())
    gerar = st.form_submit_button("Gerar Relatório por Cultura")

if not gerar:
    return

try:
    conn = pymysql.connect(**DB_CONFIG)
    cur = conn.cursor()

    sql = """
        SELECT
            cultura,
            natureza,
            SUM(valor) AS TOTAL
        FROM extrato
        WHERE data BETWEEN %s AND %s
        GROUP BY cultura, natureza
        ORDER BY cultura ASC
    """

    cur.execute(sql, (data_ini, data_fim))
    linhas = cur.fetchall()
    conn.close()

    if not linhas:
        st.warning("Nenhuma movimentação encontrada.")
        return

    RECEITAS = ["RECEITA RURAL"]
```

```
        DESPESAS = ["DESPESA RURAL", "CUSTO RURAL", "INVESTIMENTO RURAL"]

        culturas = {}
        for cultura, natureza, total in linhas:
            cultura = (cultura or "SEM CULTURA")

            if cultura not in culturas:
                culturas[cultura] = {"receita": 0, "despesa": 0}

            if natureza.upper() in RECEITAS:
                culturas[cultura]["receita"] += float(total)

            if natureza.upper() in DESPESAS:
                culturas[cultura]["despesa"] += float(total)

        dados = []
        total_geral = 0

        for cult, valores in culturas.items():
            receita = valores["receita"]
            despesa = valores["despesa"]
            resultado = receita - despesa
            dados.append([cult, receita, despesa, resultado])
            total_geral += receita

        colunas = ["CULTURA", "RECEITA", "DESPESA", "RESULTADO"]
        df = pd.DataFrame(dados, columns=colunas)

        df["% PARTICIPAÇÃO"] = (df["RECEITA"] / df["RECEITA"].sum() *
100).round(2)

        st.dataframe(df, use_container_width=True)

        csv = df.to_csv(index=False).encode("utf-8")
        st.download_button(
            "⬇ Exportar Excel/CSV – Por Cultura",
            data=csv,
            file_name="relatorio_por_cultura.csv",
```

```
            mime="text/csv"
        )

        if st.button("📄 Exportar PDF – Relatório por Cultura"):
            nome_pdf = "relatorio_por_cultura.pdf"

            styles = getSampleStyleSheet()
            conteudo = []

            conteudo.append(Paragraph("<b>Relatório por Cultura – 
Gerencial</b>", styles["Title"]))
            conteudo.append(Paragraph(f"Período: {data_ini} até {data_fim}", 
styles["Normal"]))
            conteudo.append(Paragraph(" ", styles["Normal"]))

            dados_pdf = [df.columns.tolist()] + df.astype(str).values.tolist()

            tabela_pdf = Table(dados_pdf)
            tabela_pdf.setStyle(TableStyle([
                ("BACKGROUND", (0,0), (-1,0), colors.lightgrey),
                ("GRID", (0,0), (-1,-1), 0.5, colors.black),
                ("ALIGN", (0,0), (-1,-1), "CENTER"),
                ("FONTNAME", (0,0), (-1,0), "Helvetica-Bold"),
                ("FONTSIZE", (0,0), (-1,-1), 9),
            ]))

            conteudo.append(tabela_pdf)

            caminho_pdf = f"/mnt/data/{nome_pdf}"
            pdf = SimpleDocTemplate(caminho_pdf, pagesize=A4)
            pdf.build(conteudo)

            with open(caminho_pdf, "rb") as f:
                st.download_button(
                    "📄 Baixar PDF – Por Cultura",
                    data=f.read(),
                    file_name=nome_pdf,
                    mime="application/pdf"
```

```
                )

    except Exception as e:
        st.error(f"Erro ao gerar relatório por cultura: {e}")


# ======================== RELATÓRIO GERAL – TODAS AS NATUREZAS
========================
def pagina_relatorio_geral():
    from reportlab.platypus import SimpleDocTemplate, Table, TableStyle,
Paragraph
    from reportlab.lib.pagesizes import A4
    from reportlab.lib import colors
    from reportlab.lib.styles import getSampleStyleSheet

    st.title("📊 Relatório Geral – Totais por Natureza")

    with st.form("filtro_geral"):
        col1, col2 = st.columns(2)
        data_ini = col1.date_input("Data inicial", datetime(2024, 1, 1))
        data_fim = col2.date_input("Data final", datetime.today())

        gerar = st.form_submit_button("Gerar Relatório Geral")

    if not gerar:
        return

    try:
        conn = pymysql.connect(**DB_CONFIG)
        cur = conn.cursor()

        sql = """
            SELECT natureza, SUM(valor) AS TOTAL
            FROM extrato
            WHERE data BETWEEN %s AND %s
            GROUP BY natureza
            ORDER BY natureza ASC
        """
```

```
        cur.execute(sql, (data_ini, data_fim))
        linhas = cur.fetchall()
        conn.close()

        if not linhas:
            st.warning("Nenhuma movimentação encontrada no período
selecionado.")
            return

        dados = []
        total_geral = sum([float(l[1]) for l in linhas])

        for natureza, total in linhas:
            percentual = (float(total) / total_geral * 100) if total_geral
else 0
            dados.append([natureza, float(total), round(percentual, 2)])

        colunas = ["NATUREZA", "TOTAL", "% PARTICIPAÇÃO"]
        df = pd.DataFrame(dados, columns=colunas)

        # Exibir na tela
        st.dataframe(
            df.style.format({
                "TOTAL": "R$ {:,.2f}",
                "% PARTICIPAÇÃO": "{:.2f}%"
            }),
            use_container_width=True
        )

        # Exportar CSV
        csv = df.to_csv(index=False).encode("utf-8")
        st.download_button(
            "⬇️ Exportar Excel/CSV – Relatório Geral",
            data=csv,
            file_name="relatorio_geral_naturezas.csv",
            mime="text/csv"
        )
```

```
        # Exportar PDF
        if st.button("📄 Exportar PDF – Relatório Geral"):
            nome_pdf = "relatorio_geral_naturezas.pdf"

            styles = getSampleStyleSheet()
            conteudo = []

            conteudo.append(Paragraph("<b>Relatório Geral – Totais por
Natureza</b>", styles["Title"]))
            conteudo.append(Paragraph(f"Período: {data_ini} até {data_fim}",
styles["Normal"]))
            conteudo.append(Paragraph(" ", styles["Normal"]))

            dados_pdf = [df.columns.tolist()] + df.astype(str).values.tolist()

            tabela_pdf = Table(dados_pdf)
            tabela_pdf.setStyle(TableStyle([
                ("BACKGROUND", (0,0), (-1,0), colors.lightgrey),
                ("GRID", (0,0), (-1,-1), 0.5, colors.black),
                ("ALIGN", (0,0), (-1,-1), "CENTER"),
                ("FONTNAME", (0,0), (-1,0), "Helvetica-Bold"),
                ("FONTSIZE", (0,0), (-1,-1), 9),
            ]))

            conteudo.append(tabela_pdf)

            caminho_pdf = f"/mnt/data/{nome_pdf}"
            pdf = SimpleDocTemplate(caminho_pdf, pagesize=A4)
            pdf.build(conteudo)

            with open(caminho_pdf, "rb") as f:
                st.download_button(
                    "📄 Baixar PDF – Relatório Geral",
                    data=f.read(),
                    file_name=nome_pdf,
                    mime="application/pdf"
                )
```

```
    except Exception as e:
        st.error(f"Erro ao gerar relatório geral: {e}")

# ======================== ROTEAMENTO PRINCIPAL ========================
if "auth" not in st.session_state:
    st.session_state["auth"] = False

if not st.session_state["auth"]:
    tela_login()
else:
    escolha = menu_lateral()
    if escolha == "🏠 Início":
        pagina_inicio()
    elif escolha == "📥 Extrato Bancário (PDF)":
        pagina_extrato()
    elif escolha == "📝 Lançamentos Manuais":
        pagina_lancamentos_manuais()
    elif escolha == "⚙️📝 Classificação (Extratos)":
        pagina_classificacao_extratos()
    elif escolha == "📂 Lançamentos no Banco (Consulta/Edição)":
        pagina_consultar_editar_banco()
    elif escolha == "💾 Salvar Extratos no MySQL":
        pagina_salvar_extratos()
    elif escolha == "📈 Relatório Resumo Rural (LCDPR)":
        pagina_relatorios()
    elif escolha == "📈 Relatório Detalhado":
        pagina_relatorio_detalhado()
    elif escolha == "📈 Relatório por Cultura":
        pagina_relatorio_por_cultura()
    elif escolha == "📈 Relatório Geral por Natureza":
        pagina_relatorio_geral()
```

APÊNDICE B – Código fonte do arquivo funcoes.py

```
# -*- coding: utf-8 -*-
import re
import pdfplumber
import pandas as pd
import pymysql
from datetime import datetime
import unicodedata

# ============================================================
# UTILITÁRIOS GERAIS
# ============================================================
def remover_acentos(txt: str) -> str:
    if txt is None:
        return ""
    return "".join(
        c for c in unicodedata.normalize("NFD", str(txt))
        if unicodedata.category(c) != "Mn"
    )

def limpar_texto_padrao(txt: str) -> str:
    return remover_acentos(txt).upper().strip()

def ler_pdf(caminho_pdf: str) -> str:
    texto = ""
    with pdfplumber.open(caminho_pdf) as pdf:
        for pagina in pdf.pages:
            texto += pagina.extract_text() + "\n"
    return texto

# ============================================================
# IDENTIFICAÇÃO DO BANCO
# ============================================================
def identificar_banco(texto: str) -> str:
    texto_upper = texto.upper()
```

```
            if (
                re.search(r"BANCO\s+DO\s+BRASIL", texto_upper)
                or re.search(r"EXTRATO\s+DE\s+CONTA\s+CORRENTE", texto_upper)
            ):
                return "BRASIL"
            elif "SICOOB" in texto_upper or "CREDICITRUS" in texto_upper or
"SISBR" in texto_upper:
                return "SICOOB"
            elif "BRADESCO" in texto_upper:
                return "BRADESCO"
            elif "SANTANDER" in texto_upper:
                return "SANTANDER"
            elif "ITAU" in texto_upper:
                return "ITAU"
            else:
                return "BANCO"

        # ============================================================
        # EXTRAÇÃO DE AGÊNCIA E CONTA
        # ============================================================
        def extrair_agencia_conta(texto: str) -> str:
            texto_upper = texto.upper()

            # --- SICOOB / CREDICITRUS ---
            m_coop = re.search(r"COOP\.\s*[:\-]?\s*(\d{3,5}-?\d?)",
texto_upper)
            m_conta = re.search(r"CONTA\s*[:\-]?\s*([\d\.]{3,}-?\d?)",
texto_upper)
            if m_coop or m_conta:
                return f"{m_coop.group(1) if m_coop else ''} /
{m_conta.group(1) if m_conta else ''}".strip(" /")

            # --- BANCO DO BRASIL / BRADESCO ---
            m_ag_bb = re.search(r"AG[ÊE]NCIA[:\s\-]*(\d{3,5}-?[A-Z0-
9]*)\s+CONTA[:\s\-]*(\d{3,10}-\d)", texto_upper)
            if m_ag_bb:
                return f"{m_ag_bb.group(1)} / {m_ag_bb.group(2)}"
```

```
        return ""

    # ============================================================
    # TRATAMENTO DAS LINHAS DO EXTRATO
    # ============================================================
    def tratar_linhas(texto: str) -> pd.DataFrame:
        """
        Lê o texto bruto do PDF do extrato e devolve um DataFrame com:
        DATA, BANCO, AGENCIA_CONTA, DOCUMENTO, HISTORICO, VALOR.

        Regras:
        - DATA: usa a última data válida (repete até aparecer outra).
        - DOCUMENTO: para o Bradesco, captura o número da coluna "Docto.";
          para o Sicoob, identifica "DOC.: xxxx" no histórico.
        """
        banco = identificar_banco(texto)
        ag_conta = extrair_agencia_conta(texto)
        linhas = texto.splitlines()

        registros = []
        data_atual = None
        desc_buffer = ""

        padrao_data  = re.compile(r"\b(\d{1,2}/\d{1,2}/\d{2,4})\b")
        padrao_valor = re.compile(r"(-?\d{1,3}(?:\.\d{3})*,\d{2})")
        padrao_doc   = re.compile(r"DOC\.\s*:\s*(.+)", re.IGNORECASE)
        padrao_doc_bradesco = re.compile(r"^\s*(\d{3,8})\s")  # número tipo
1109, 4734671 etc.

        def parse_valor(txt: str) -> float:
            txt = txt.replace(".", "").replace(",", ".")
            try:
                return float(txt)
            except Exception:
                return 0.0

        for linha in linhas:
            linha_original = linha
```

```
                linha = " ".join(linha.strip().split())
                if not linha:
                    continue

                # Ignora cabeçalhos e blocos não relevantes
                if "INFORMACOES ADICIONAIS" in linha.upper() or "INFORMACOES
COMPLEMENTARES" in linha.upper():
                    break

                # --- DOC.: padrão Sicoob ---
                m_doc = padrao_doc.search(linha_original)
                if m_doc:
                    doc_txt = limpar_texto_padrao(m_doc.group(1))
                    if registros:
                        registros[-1]["DOCUMENTO"] = doc_txt
                    continue

                # --- Bradesco: identifica número no início da linha ---
                doc_num = ""
                if banco == "BRADESCO" and not m_doc:
                    m_doc_br = padrao_doc_bradesco.match(linha)
                    if m_doc_br:
                        doc_num = m_doc_br.group(1)
                        linha = linha.replace(doc_num, "", 1).strip()

                # --- Data ---
                m_data = padrao_data.search(linha)
                if m_data:
                    data_atual = m_data.group(1).strip()
                    linha = padrao_data.sub("", linha).strip()

                # --- Valor ---
                valores = padrao_valor.findall(linha)
                if not valores:
                    desc_buffer = (desc_buffer + " " + linha).strip() if
desc_buffer else linha
                    continue
```

```
                val_txt = valores[0]
                valor = parse_valor(val_txt)
                linha_sem_valores = padrao_valor.sub("", linha).strip()
                historico = limpar_texto_padrao(" ".join([desc_buffer,
linha_sem_valores]).strip())
                desc_buffer = ""

                if not data_atual:
                    continue

                registros.append({
                    "DATA": data_atual,
                    "BANCO": banco,
                    "AGENCIA_CONTA": ag_conta,
                    "DOCUMENTO": doc_num,
                    "HISTORICO": historico,
                    "VALOR": valor
                })

            # --- Monta DataFrame final ---
            df = pd.DataFrame(registros,
columns=["DATA","BANCO","AGENCIA_CONTA","DOCUMENTO","HISTORICO","VALOR"])

            if not df.empty:
                df["DATA"] = pd.to_datetime(df["DATA"], errors="coerce",
dayfirst=True)
                df["DATA"] = df["DATA"].ffill().bfill().dt.strftime("%d/%m/%Y")
                df["VALOR"] = pd.to_numeric(df["VALOR"],
errors="coerce").fillna(0.0)

            return df

        # ============================================================
        # BANCO DE DADOS
        # ============================================================
        DB_CONFIG = dict(host="localhost", user="root",
password="Rogerio2025@", database="Financeiro")
```

```
        def salvar_mysql(df: pd.DataFrame, modo="sobrepor"):
            res = dict(inseridos=0, atualizados=0, erros=0)
            try:
                conn = pymysql.connect(**DB_CONFIG)
                cur = conn.cursor()

                for _, linha in df.iterrows():
                    try:
                        data_val = linha["DATA"]
                        if isinstance(data_val, datetime):
                            data_fmt = data_val.date()
                        else:
                            txt = str(data_val).strip().split(" ")[0]
                            data_fmt = None
                            for fmt in ("%d/%m/%Y", "%Y-%m-%d"):
                                try:
                                    data_fmt = datetime.strptime(txt,
fmt).date()
                                    break
                                except ValueError:
                                    continue
                        if data_fmt is None:
                            raise ValueError(f"Formato de data inválido:
{txt}")

                        sql = """
                            INSERT INTO extrato
                                (data, banco, agencia_conta, documento,
historico, valor,
                                 natureza, cultura, safra, atividades, insumos)
                            VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)
                            ON DUPLICATE KEY UPDATE
                                valor=VALUES(valor)
                        """
                        cur.execute(sql, (
                            data_fmt,
                            linha["BANCO"],
                            linha["AGENCIA_CONTA"],
```

```
                    linha["DOCUMENTO"],
                    linha["HISTORICO"],
                    linha["VALOR"],
                    linha.get("NATUREZA", ""),
                    linha.get("CULTURA", ""),
                    linha.get("SAFRA", ""),
                    linha.get("ATIVIDADES", ""),
                    linha.get("INSUMOS", "")
                ))
                res["inseridos"] += 1

            except Exception:
                res["erros"] += 1

        conn.commit()
        cur.close()
        conn.close()

    except Exception:
        res["erros"] += 1

    return res
```

APÊNDICE C – Código fonte do arquivo main.py

```
        # -*- coding: utf-8 -*-
        import pymysql

        DB_CONFIG = dict(host="localhost", user="root",
password="Rogerio2025@", database="Financeiro")

        def garantir_esquema_mysql():
            conn = pymysql.connect(**DB_CONFIG); cur = conn.cursor()
            cur.execute("""
                CREATE TABLE IF NOT EXISTS extrato (
                    id INT AUTO_INCREMENT PRIMARY KEY,
                    data DATE NOT NULL,
                    banco VARCHAR(50) NOT NULL,
                    agencia_conta VARCHAR(50) NOT NULL,
                    documento VARCHAR(50) NULL,
                    historico TEXT NOT NULL,
                    valor DECIMAL(15,2) NOT NULL,
                    natureza  VARCHAR(20)  NOT NULL DEFAULT '',
                    cultura   VARCHAR(30)  NOT NULL DEFAULT '',
                    safra     VARCHAR(20)  NOT NULL DEFAULT '',
                    atividades VARCHAR(80) NOT NULL DEFAULT '',
                    insumos    VARCHAR(120) NOT NULL DEFAULT ''
                ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
            """)
            conn.commit(); cur.close(); conn.close()

        if __name__ == "__main__":
            garantir_esquema_mysql()
            print("Schema checado/atualizado com NATUREZA, CULTURA, SAFRA,
ATIVIDADES, INSUMOS.")
```