O USO DE FERRAMENTAS PARA A CRIAÇÃO DE PAINÉIS ADMINISTRATIVOS EM APLICAÇÕES LARAVEL

Isak Gabriel Chedid Girardello[1], Rogério Antônio Casagrande[2]

Resumo: Este trabalho apresenta um estudo aplicado sobre o uso de ferramentas do ecossistema Laravel para a criação de painéis administrativos, com ênfase na biblioteca Filament, a partir do desenvolvimento de um protótipo de sistema de contas a pagar. A pesquisa, de natureza descritiva e base tecnológica, combina revisão exploratória de trabalhos correlatos com implementação prática em Laravel 12.x e Filament 4.x. O sistema foi estruturado com *migrations*, *seeders* e Eloquent ORM, contemplando autenticação nativa, formulários com validações, relacionamentos entre entidades e *widgets* gráficos para análise de gastos por unidade, tipo e período. A aplicação foi implantada em ambiente local (MySQL) e em nuvem (Render + Docker + SQLite), apresentando funcionamento estável, responsivo e consistente entre interface e banco de dados. A observação técnica realizada durante o desenvolvimento evidenciou agilidade na geração automática de interfaces e baixo esforço de codificação, apontando o potencial do Filament como ferramenta para construção de sistemas administrativos em projetos com prazos curtos e equipes reduzidas.

Palavras-chave: laravel; filament; painéis administrativos; contas a pagar; desenvolvimento web; prototipagem; produtividade no desenvolvimento.

[1]Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (Unesc), isakgabrielcg@gmail.com

[2]Orientador. Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (Unesc), roc@unesc.net

ABSTRACT: This paper presents an applied study on the use of Laravel ecosystem tools for building administrative dashboards, with emphasis on the Filament library, through the development of an accounts payable system prototype. The research, descriptive in nature and with a technological basis, combines an exploratory review of related works with practical implementation using Laravel 12.x and Filament 4.x. The system was structured with migrations, seeders, and the Eloquent ORM, including native authentication, validated forms, entity relationships, and graphical widgets for analyzing expenses by unit, type, and period. The application was deployed in both local (MySQL) and cloud environments (Render + Docker + SQLite), demonstrating stable, responsive, and consistent performance between the interface and the database. The technical observation carried out during development revealed agility in the automatic generation of interfaces and low coding effort, highlighting Filament's potential as a tool for building administrative systems in projects with short deadlines and small development teams.

Keywords: laravel; filament; administrative panels; accounts payable; web development; prototyping; development productivity.

1 INTRODUÇÃO

A transformação digital tem remodelado profundamente os processos internos das empresas, impulsionando a necessidade por sistemas de informação capazes de oferecer agilidade, controle e precisão na gestão (Ferreira, 2023). Em meio a esse processo, os painéis administrativos, também chamados de *admin panels*, se destacam como uma das ferramentas mais utilizadas por times de desenvolvimento para facilitar operações rotineiras, como o cadastro e manutenção de dados, controle de usuários, aplicação de regras de negócio e auditoria de ações. A construção desses painéis, no entanto, exige tempo e conhecimento técnico, o que representa um desafio adicional para equipes pequenas ou com prazos curtos.

De acordo com Chapetta (2018), fatores como complexidade do projeto, domínio da tecnologia e ferramentas de apoio influenciam diretamente na produtividade do desenvolvimento de software, o que reforça a importância de soluções que agilizem as etapas de codificação e manutenção. Assim, torna-se relevante investigar ferramentas e práticas que otimizem o desenvolvimento de sistemas administrativos, especialmente em contextos com restrições de tempo e equipe.

Nesse contexto, a busca por soluções que aumentem a produtividade sem comprometer a qualidade tem levado desenvolvedores a adotar ferramentas que automatizam a criação desses módulos administrativos. De acordo com Bach et al. (2022), a padronização visual de interfaces por meio de *dashboards* e indicadores facilita a tomada de decisão. Almasi et al. (2023) reforçam que, além da eficiência, fatores como usabilidade, operabilidade e curva de aprendizado devem ser considerados, especialmente quando o sistema será utilizado por usuários com diferentes níveis de familiaridade com tecnologia.

Dentre os frameworks mais populares para desenvolvimento web, o Laravel se destaca por sua estrutura robusta e flexível. Criado por Taylor Otwell, o Laravel adota o padrão MVC (*Model-View-Controller*) e oferece funcionalidades modernas como Eloquent ORM, autenticação, controle de sessões, roteamento e integração com banco de dados (Subecz, 2021). Essas características tornam o Laravel uma solução altamente produtiva para a criação de sistemas administrativos. Um dos grandes diferenciais do Laravel é sua sintaxe expressiva, que favorece a legibilidade do código e a manutenção do sistema, além de um ecossistema de ferramentas e pacotes que enriquecem o desenvolvimento.

Além de sua estrutura nativa, o ecossistema Laravel inclui ferramentas auxiliares específicas para a construção de interfaces administrativas. Entre elas, destacam-se o Laravel Nova, Laravel Voyager e Filament, que oferecem soluções para geração de CRUDs, personalização visual, gerenciamento de usuários e visualização de dados. O Laravel Nova, por exemplo, é uma ferramenta premium, mantida pela equipe oficial do framework e integrada nativamente ao Eloquent ORM (Laravel, 2025b). Já o Voyager é uma alternativa gratuita e de código aberto, voltada a projetos menores e de fácil configuração (Voyager, 2025). O Filament, por sua vez, combina facilidade de uso, flexibilidade e velocidade de desenvolvimento, utilizando componentes prontos como tabelas, formulários e filtros, além de basear-se na arquitetura TALL Stack, que integra *Tailwind CSS*, *Alpine.js*, *Laravel* e *Livewire*, o que o torna uma escolha popular entre desenvolvedores (Filament, 2024; Curie et al., 2019; Hussain, 2025).

A construção de sistemas administrativos demanda também uma estrutura sólida de banco de dados. O Laravel oferece, nesse aspecto, ferramentas como as *migrations*, que permitem versionar e controlar a estrutura das tabelas, e as *seeders*, que populam automaticamente o banco com dados de exemplo. Esse tipo de versionamento favorece a consistência en-

tre ambientes e facilita a evolução controlada do sistema (Athota, 2024). No contexto do Laravel, as migrations funcionam como um controle de versão do banco de dados, possibilitando criar, modificar e reverter estruturas de forma segura (Aurora, 2025). Integrados ao ORM Eloquent, esses recursos tornam o desenvolvimento mais rápido e confiável, reduzindo falhas e simplificando a manutenção do sistema.

Entretanto, apesar da oferta crescente dessas ferramentas, a escolha da mais adequada ainda representa um dilema técnico. Questões como custo de licenciamento, compatibilidade com bibliotecas externas, facilidade de personalização e suporte da comunidade influenciam diretamente a decisão. Além disso, a maioria dos relatos disponíveis sobre essas ferramentas é baseada em experiências pessoais ou documentação técnica, o que dificulta uma comparação objetiva entre elas (Silva, 2019). Assim, a presente pesquisa busca contribuir com uma análise aplicada e documentada, explorando empiricamente as vantagens e limitações dessas ferramentas no contexto do desenvolvimento real.

Esse desafio torna-se ainda mais relevante quando se observa o aumento da demanda por sistemas administrativos no Brasil. Segundo o Ministério da Economia (2023), foram registradas mais de 3,8 milhões de novas empresas em um único ano, muitas delas com necessidades semelhantes de gestão e controle interno. Em paralelo, estudos como o da Prophix Software (2024) apontam que o uso contínuo de planilhas em processos de gestão, ainda comum entre pequenas empresas, compromete a confiabilidade dos dados, dificulta auditorias e aumenta os riscos operacionais. Dessa forma, torna-se evidente a necessidade de soluções acessíveis, escaláveis e rápidas de implementar.

Com o objetivo de suprir essa demanda, ferramentas como o Filament permitem que desenvolvedores criem sistemas administrativos com rapidez, mantendo padrões de segurança, organização e usabilidade. A integração com recursos como migrations, seeders e Eloquent ORM facilita a consistência entre banco de dados e interface administrativa. A abordagem visual do Filament também promove maior clareza e simplicidade na gestão de sistemas, o que é essencial em ambientes com usuários não técnicos (Filament, 2024). Assim, este trabalho propõe investigar como o uso dessas ferramentas pode impactar o equilíbrio entre produtividade, flexibilidade e curva de aprendizado no desenvolvimento de sistemas internos.

Diante desse cenário, este trabalho é orientado pela seguinte questão exploratória: como ferramentas administrativas do ecossistema La-

ravel, como o Filament, podem contribuir para o equilíbrio entre produtividade, flexibilidade e curva de aprendizado no desenvolvimento de sistemas internos por equipes pequenas?

A pesquisa tem como foco o Filament, mas também contempla um levantamento exploratório sobre outras ferramentas populares do ecossistema Laravel. Com base na implementação prática de um sistema de contas a pagar, pretende-se observar aspectos como tempo de desenvolvimento, facilidade de personalização, curva de aprendizado e adequação à realidade de pequenas empresas, com o intuito de compreender as possibilidades e limitações envolvidas no uso dessas ferramentas. A abordagem metodológica adota características de uma pesquisa aplicada e descritiva, conduzida sob a forma de estudo de caso tecnológico, com observação empírica dos resultados.

O que justifica este estudo é a crescente necessidade de soluções administrativas eficientes em um contexto de expansão do empreendedorismo e da digitalização no Brasil. Ao relatar uma experiência prática de implementação, busca-se oferecer subsídios aplicados que auxiliem desenvolvedores e equipes técnicas na escolha de ferramentas apropriadas ao seu contexto. Além disso, pretende-se incentivar o uso de soluções acessíveis e sustentáveis, promovendo boas práticas de desenvolvimento de sistemas internos em ambientes corporativos.

Este artigo está dividido em cinco seções. A Seção 1 apresenta a introdução ao tema, o problema, os objetivos e a justificativa da pesquisa. A Seção 2 aborda os trabalhos correlatos, discutindo soluções existentes e estudos prévios relacionados à criação de sistemas administrativos com Laravel. A Seção 3 detalha os materiais e métodos utilizados para o desenvolvimento do protótipo. A Seção 4 apresenta os resultados obtidos e a discussão crítica com base nos dados coletados. Por fim, a Seção 5 expõe as conclusões do estudo, bem como sugestões para trabalhos futuros.

2 TRABALHOS CORRELATOS

Como parte da fundamentação técnica e conceitual desta pesquisa, foram analisados estudos que exploram o uso do framework Laravel e ferramentas relacionadas na construção de sistemas administrativos. Esses trabalhos apresentam abordagens semelhantes às adotadas neste projeto, permitindo uma análise crítica que contribui para fundamentar as decisões técnicas tomadas ao longo do desenvolvimento.

2.1 DESENVOLVIMENTO DE UM SISTEMA WEB PARA CONTROLE DE FALTAS DA UNIVERSIDADE FEDERAL DE OURO PRETO

O trabalho de Almeida (2018) apresenta o desenvolvimento de um sistema web voltado para o controle de faltas em uma instituição pública de ensino superior. A autora parte do contexto da Universidade Federal de Ouro Preto (UFOP) para justificar a necessidade de informatização do processo de registro e acompanhamento de faltas dos alunos, que era até então realizado de forma manual ou com ferramentas pouco integradas. O estudo enfatiza a relevância da digitalização de processos educacionais, com foco na padronização e na segurança dos dados.

Do ponto de vista técnico, o sistema foi desenvolvido utilizando tecnologias web amplamente conhecidas, como HTML, CSS, JavaScript e PHP. Embora o framework Laravel não tenha sido utilizado, o projeto destaca aspectos importantes como o controle de acesso, persistência de dados em banco relacional e a construção de uma interface intuitiva para os usuários administrativos. O trabalho também explora preocupações com a experiência do usuário e a organização lógica do sistema, mesmo com limitações tecnológicas da época.

Ainda que o escopo do sistema seja distinto do foco deste trabalho, voltado à área administrativa empresarial, a proposta de Almeida contribui por apresentar uma metodologia aplicada na organização e registro eficiente de dados operacionais. A principal diferença está na tecnologia de desenvolvimento e na ausência de ferramentas modernas como Filament, mas ambos os trabalhos compartilham o objetivo de digitalizar processos com foco na clareza, acessibilidade e confiabilidade das informações manipuladas.

2.2 DESENVOLVIMENTO DE UMA FERRAMENTA DE GESTÃO PARA CONSTRUÇÃO CIVIL UTILIZANDO O FRAMEWORK LARAVEL MVC

O trabalho desenvolvido por Silva (2022) descreve a construção do sistema EngPack Budget 2.0, voltado ao ensino da orçamentação na engenharia civil. A proposta surgiu diante da dificuldade de acesso a softwares que atendessem às necessidades acadêmicas, já que muitas soluções do mercado eram pagas, limitadas a plataformas fixas ou baseadas em tecnologias ultrapassadas.

Para atender a essa demanda, o sistema foi desenvolvido com o framework Laravel, seguindo o padrão MVC. O autor utilizou recursos como Eloquent ORM, migrations, autenticação e validação de dados. A

interface foi feita com HTML, CSS, Blade, Bootstrap e o template AdminLTE, formando um painel com áreas de controle financeiro, cadastro de obras e geração de relatórios.

Mesmo sendo um sistema voltado ao contexto educacional, ele compartilha aspectos técnicos relevantes com o presente trabalho. Ambos utilizam o Laravel como base de estrutura. Um dos pontos em que eles se diferenciam é a camada visual, enquanto o sistema de Silva (2022) utiliza um template pronto, o sistema proposto neste trabalho faz uso do Filament, que facilita a criação de interfaces com aparência mais moderna e organizada. Essa diferença permite observar como o uso de ferramentas recentes pode influenciar positivamente no desenvolvimento e na experiência do usuário.

2.3 SISTEMA DE INFORMAÇÃO PARA CLÍNICAS ODONTOLÓGICAS USANDO LARAVEL E FILAMENT

O estudo conduzido por Pramudya et al. (2025) mostra o desenvolvimento de um sistema de informação voltado ao atendimento de clínicas odontológicas de pequeno e médio porte. O objetivo central foi digitalizar o gerenciamento de prontuários médicos, estoques de medicamentos e faturamento, como forma de superar limitações causadas por processos manuais e desorganizados, ainda comuns em muitas unidades de saúde desse porte.

A aplicação foi construída utilizando o framework Laravel, adotando a arquitetura MVC, e teve o Filament como base para o painel administrativo. Essa escolha visou otimizar o gerenciamento das funcionalidades e proporcionar uma experiência mais fluida aos usuários. O processo de desenvolvimento seguiu a metodologia Extreme Programming (XP), que favorece entregas rápidas e ajustes contínuos a partir do retorno dos usuários envolvidos.

Nos testes realizados com o sistema, por meio da técnica de *Black Box Testing*, verificou-se a eficácia da solução em termos de confiabilidade, usabilidade e desempenho. Embora o sistema tenha sido construído para o setor da saúde, seu desenvolvimento com Laravel e Filament mostra a versatilidade dessas tecnologias, o que reforça sua aplicabilidade em projetos de outras áreas, como o proposto neste trabalho.

3 MATERIAIS E MÉTODOS

O presente trabalho configura-se como uma pesquisa aplicada, de base tecnológica e natureza descritiva, com ênfase no desenvolvimento de um protótipo funcional de sistema de contas a pagar. A abordagem metodológica adotada foi o estudo de caso tecnológico, conduzido a partir da implementação prática do sistema e da observação empírica de seu funcionamento. Para a implementação, utilizou-se a linguagem PHP 8.4.11 em conjunto com o framework Laravel 12.x, que adota o padrão arquitetural Model-View-Controller (MVC) e disponibiliza recursos modernos, como migrations, seeders e o Eloquent ORM (Laravel, 2025a). A camada administrativa foi construída com o Filament 4.x, ferramenta que se integra ao Laravel e fornece componentes prontos para criação de interfaces administrativas (Filament, 2024).

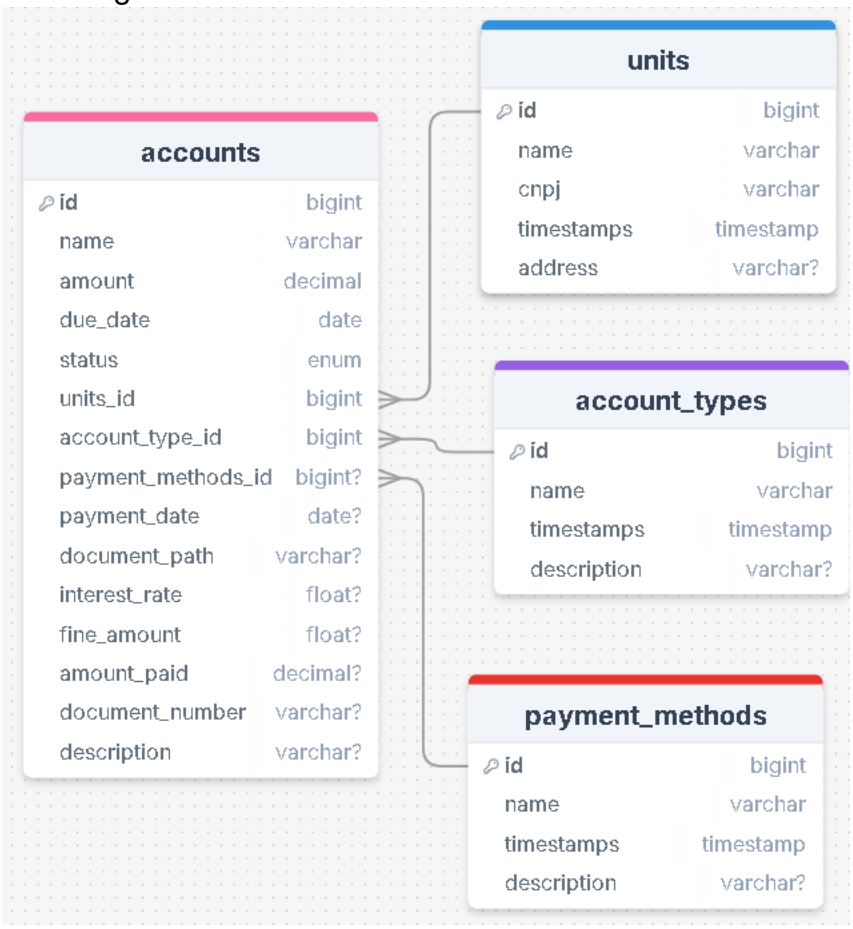
O ambiente de desenvolvimento foi configurado em um computador pessoal com sistema operacional Windows 10. Foram utilizadas as seguintes ferramentas: Composer 2.8.10, para gerenciamento de dependências PHP; Node.js 20.13.1 e NPM 10.8.2, para compilação de recursos front-end; MySQL 8.4.6, como banco de dados local; Visual Studio Code, como editor de código; e Git, integrado ao GitHub, para controle de versão.

O projeto foi iniciado a partir da execução dos comandos disponibilizados pelo instalador oficial do Laravel, responsável por configurar automaticamente a estrutura inicial da aplicação. Em seguida, realizaram-se as configurações das dependências front-end e os testes de execução local para verificar o funcionamento correto do ambiente. Após essa etapa, procedeu-se à instalação do Filament 4.x, adicionando o pacote ao projeto por meio do Composer e executando a configuração da ferramenta, que criou a camada administrativa do sistema. O processo foi conduzido de forma incremental, com registros e testes sucessivos para assegurar a consistência metodológica das etapas.

A modelagem inicial do banco de dados foi elaborada em ferramenta online de diagramas relacionais e posteriormente exportada para migrations do Laravel. Durante esse processo, foram necessários ajustes manuais nas chaves estrangeiras, de forma a garantir a integridade referencial entre as tabelas. Também foi realizada a alteração da migration de usuários para incluir a coluna *role*, permitindo diferenciar perfis de administrador e usuário padrão. Após essas configurações, foi criado o primeiro usuário administrativo para validação da autenticação e acesso à interface

do sistema, etapa que também serviu como verificação funcional da integração entre backend e banco de dados. A Figura 1 apresenta o modelo relacional do banco de dados utilizado na aplicação, evidenciando as principais entidades e seus relacionamentos.

Figura 1 - Modelo relacional do banco de dados.



Fonte: Autor.

Um aspecto que merece destaque é a autenticação disponibilizada pelo próprio Filament. Após sua instalação, a ferramenta gera automaticamente uma tela de login integrada ao Laravel, com toda a lógica de autenticação já configurada. Essa funcionalidade permite validar as credenciais do usuário diretamente no banco de dados, sem necessidade de programação manual. Além disso, o sistema protege todas as rotas administrativas, mesmo que alguém tente acessar diretamente uma URL sem estar autenticado, redirecionando para a página de login. Esse recurso reduz significativamente a complexidade do desenvolvimento, uma vez que elimina a necessidade de construir manualmente fluxos de autenticação e controle de sessões, comuns em frameworks front-end como React, e foi considerado essencial para validar o comportamento de segurança do pro-

tótipo.

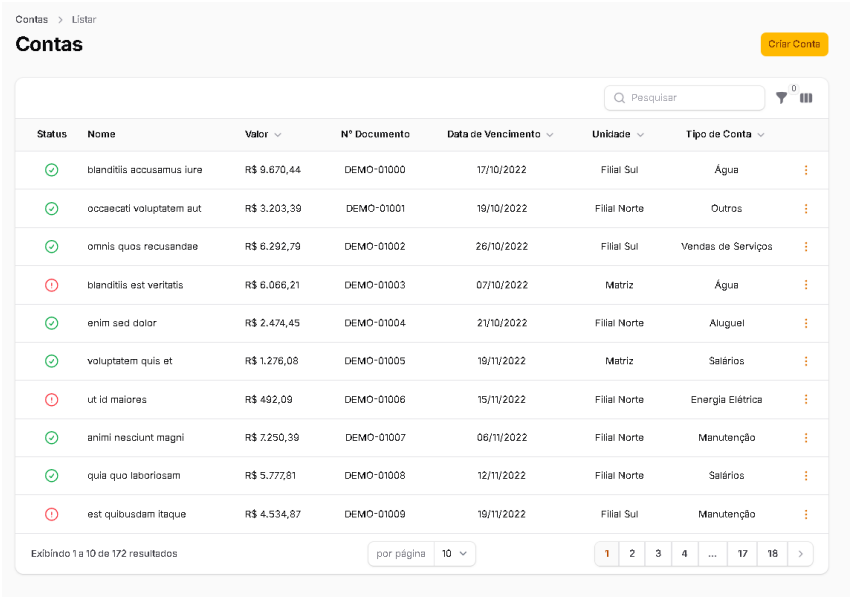
Além da autenticação, foram realizadas pequenas customizações na interface gerada pelo Filament, como a definição de permissões por perfil e ajustes visuais nos formulários administrativos. Essa etapa teve como objetivo garantir uma experiência consistente entre os diferentes tipos de usuários, mantendo o padrão estético do painel e a segurança nas operações internas.

Com a estrutura inicial consolidada, iniciou-se a criação das *seeders* responsáveis por popular o banco de dados com informações de teste. Cada *seeder* foi desenvolvida individualmente e, posteriormente, centralizada em uma *DemoSeeder*, que executa todas as demais de forma automatizada. Essa estratégia facilitou a inserção de dados padronizados e garantiu maior agilidade na verificação do protótipo.

Na etapa seguinte, foram criados os *models* do Laravel correspondentes às entidades do banco de dados, como Account, AccountType, PaymentMethod e Unit. Em cada *model*, definiram-se os atributos *fillable*, que determinam quais colunas podem ser preenchidas em operações de cadastro, além da declaração explícita da tabela associada por meio da propriedade *$table*. Para representar os relacionamentos entre as entidades, foram implementados os métodos Eloquent *belongsTo* e *hasMany*. Durante o desenvolvimento, cada módulo foi testado individualmente, buscando avaliar a estabilidade e o comportamento esperado em cada operação.

A interface administrativa foi estruturada a partir da criação de *resources* do Filament, vinculados aos respectivos *models* da aplicação. Esse recurso gera automaticamente as telas de listagem, criação, edição e exclusão de registros, além de configurar formulários básicos de entrada de dados. Dessa forma, grande parte do sistema pôde ser implementada sem necessidade de codificação manual de formulários simples, o que acelerou significativamente o processo de desenvolvimento. A Figura 2 apresenta um exemplo da listagem de contas exibida no painel administrativo, com recursos de filtragem, ordenação e paginação.

Figura 2 - Exemplo de listagem de contas no painel administrativo



Fonte: Autor.

Além das operações básicas, foram implementados recursos de apoio à usabilidade, como ordenação de colunas e pesquisa textual. O Filament fornece suporte nativo para tais funcionalidades, bastando declarar atributos como `->sortable()` e `->searchable()` nos campos de listagem. Dessa forma, o usuário pode reorganizar os registros conforme a coluna desejada ou realizar buscas rápidas, sem necessidade de programação adicional.

Outro ponto relevante foi a criação de indicadores visuais para facilitar a interpretação das informações. O campo de status, por exemplo, foi configurado para exibir ícones e cores distintas conforme a situação da conta: verde para pagas, vermelho para vencidas, azul para contas que vencem no dia e amarelo para pendências em aberto.

Outro recurso de usabilidade oferecido nativamente pelo Filament é a alternância de tema da interface. O sistema disponibiliza três opções: tema claro, tema escuro e adaptação automática conforme a configuração do navegador do usuário. Essa funcionalidade melhora a experiência de uso, permitindo que cada pessoa escolha a visualização que considerar mais confortável.

Para o cálculo do valor final pago, considerando multas e juros por atraso, foi criada uma classe auxiliar denominada *Financeiro*, localizada em uma pasta específica de *helpers*. Essa classe recebe os dados da conta (valor base, percentual de multa, taxa de juros e datas de vencimento e pagamento) e, com base nessas informações, aplica as regras de cálculo.

O resultado é formatado no padrão monetário brasileiro, exibindo ao usuário o valor total efetivamente pago.

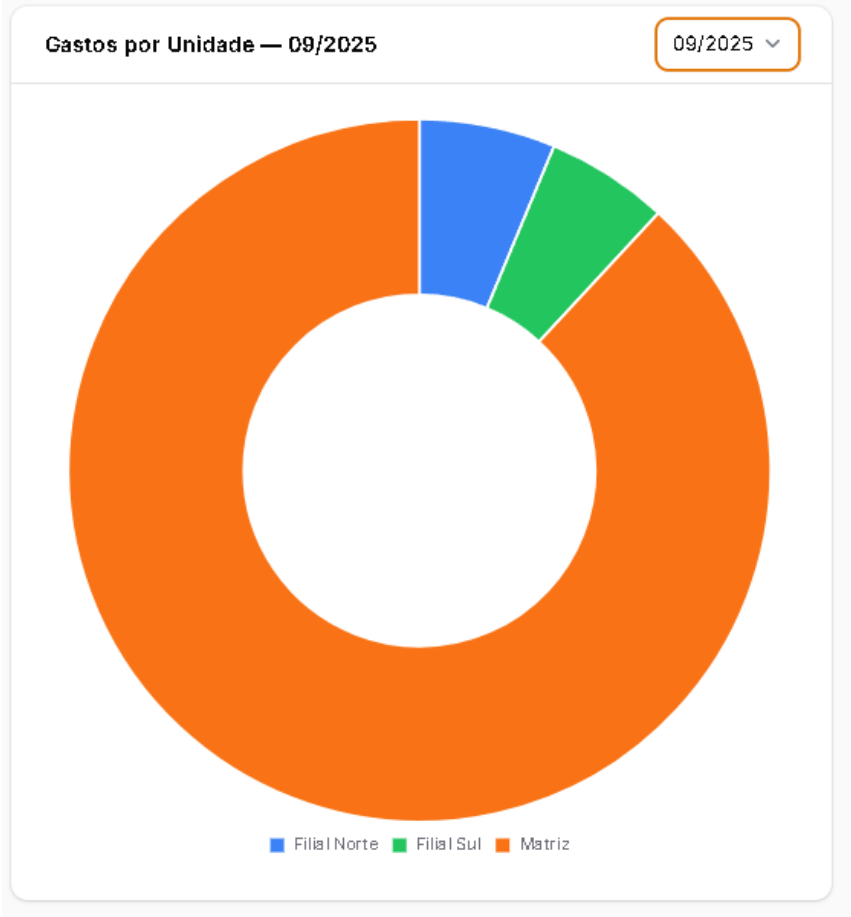
Outro elemento importante da aplicação foram os gráficos implementados para fornecer uma visão consolidada dos dados. Para isso, utilizaram-se os *widgets* do Filament, que permitem criar componentes dinâmicos de visualização integrados à interface administrativa. Durante a criação, foram escolhidos os modelos de gráfico do tipo linha (*line*), barra (*bar*) e rosca (*doughnut*), disponibilizados pela biblioteca Chart.js, já integrada ao framework.

A lógica de cada gráfico foi implementada diretamente nos arquivos gerados pelo Filament, sendo necessário apenas inserir as consultas adequadas ao banco de dados. Para tornar a visualização dinâmica, foram adicionados filtros que permitem ao usuário selecionar períodos específicos. Como o sistema foi executado em ambientes distintos, foi necessário adaptar as consultas para garantir compatibilidade. No ambiente local, com MySQL, empregaram-se funções como `DATE_FORMAT` e `whereYear`, enquanto no ambiente em nuvem, baseado em SQLite, utilizaram-se funções como `strftime` para extrair ano e mês. Essa estratégia assegurou que os relatórios funcionassem corretamente em ambos os bancos de dados, sem comprometer a consistência dos resultados, conforme ilustrado na Figura 3, que apresenta o gráfico do tipo *doughnut* utilizado para demonstrar a distribuição dos gastos por unidade.

No total, foram implementados três gráficos principais:

- **AmountByTypeChart** (doughnut), representando os gastos totais por unidade;
- **AccountsByTypeChart** (bar), exibindo os gastos por tipo de conta no ano corrente;
- **AccountsByMonthChart** (line), apresentando a evolução dos gastos por mês.

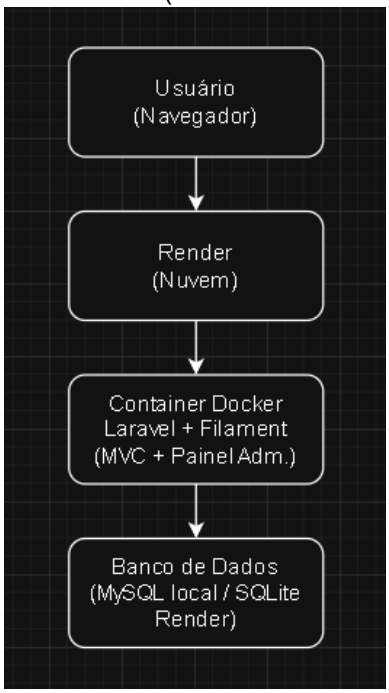
Figura 3 - Exemplo de gráfico de gastos por unidade (tipo doughnut)



Fonte: Autor.

Para a disponibilização online do sistema, utilizou-se a plataforma Render, um serviço de hospedagem em nuvem que permite implantar aplicações diretamente a partir de repositórios GitHub. A plataforma oferece suporte a containers Docker e possibilita integração contínua, de modo que cada atualização realizada no repositório é refletida automaticamente no ambiente de produção. O Render dispõe de planos pagos e gratuitos, sendo que a versão gratuita apresenta algumas limitações, como tempo de inicialização maior e ausência de persistência de dados entre os deploys. Ainda assim, tais características foram suficientes para a montagem do protótipo e para a demonstração prática de suas funcionalidades, cuja estrutura geral de execução é representada na Figura 4, que ilustra o fluxo entre o usuário, o painel Filament, o framework Laravel e o banco de dados.

Figura 4 - Arquitetura do sistema (Usuário → Filament → Laravel → BD)



Fonte: Autor.

Foi criado um *Dockerfile* na raiz do projeto, baseado na imagem oficial do PHP 8.3 CLI, contendo a instalação de extensões indispensáveis (PDO, SQLite, ZIP e intl), configuração do Composer e otimização do autoload. Além disso, o Dockerfile foi configurado para executar automaticamente, no momento da inicialização do container, comandos do Laravel que garantem o funcionamento imediato do sistema.

Para simplificar a configuração e evitar a necessidade de provisionamento de um banco MySQL externo, optou-se pelo uso do SQLite. Assim, foi adicionado ao projeto um arquivo vazio denominado `database.sqlite`, localizado na pasta `database`. Esse arquivo serviu como base para a aplicação, e a cada atualização no Render o banco era reconstruído a partir dos *seeders* definidos.

A configuração final foi realizada diretamente na seção *Environment* do Render, onde foram definidas variáveis como a chave de segurança da aplicação, idioma padrão, conexão ao banco SQLite, parâmetros de cache e sessão, além da URL pública de acesso. Dessa forma, foi possível manter o sistema acessível em ambiente online, permitindo sua demonstração prática em diferentes dispositivos e facilitando a verificação do funcionamento do protótipo.

Por fim, o protótipo passou por verificação funcional em ambiente local e em nuvem, verificando-se o funcionamento das operações de cadas-

tro, edição e exclusão de contas, bem como a correta exibição de relatórios e gráficos nos painéis administrativos. Em ambos os ambientes, a aplicação demonstrou consistência entre banco de dados e interface, atendendo aos requisitos funcionais e não funcionais previamente estabelecidos. A análise dos resultados foi conduzida de forma qualitativa e descritiva, considerando estabilidade, responsividade e usabilidade observadas durante a execução do protótipo.

4 RESULTADOS E DISCUSSÃO

O protótipo resultou em um sistema administrativo estável, responsivo e alinhado aos requisitos definidos, construído com Laravel 12.x e Filament 4.x. Durante o desenvolvimento, foi criado um módulo auxiliar denominado *Category*, utilizado apenas para observação técnica de produtividade. A geração automática do recurso foi concluída em aproximadamente três minutos e vinte segundos, com criação estruturada das pastas `Pages`, `Schemas` e `Tables`. As edições manuais se restringiram a ajustes mínimos, como a adição de rótulos, um filtro booleano e a definição da opção de ordenação de colunas, totalizando cerca de doze linhas de código distribuídas em três arquivos. Essa atividade evidenciou a capacidade do Filament de gerar, em poucos minutos, uma interface administrativa totalmente funcional, demandando baixa intervenção do desenvolvedor.

O sistema completo foi implantado e testado em dois ambientes: local (MySQL) e nuvem (Render, com SQLite). O comportamento foi semelhante nos dois cenários, apresentando tempo médio de carregamento de 4,87 s no ambiente local e 2,01 s na nuvem, além de valores médios de *DOMContentLoaded* de 4,75 s e 1,76 s, respectivamente. Esses resultados indicam estabilidade e boa portabilidade entre ambientes, possivelmente favorecida pela compressão e pelo cache oferecidos pelo serviço de hospedagem. Para a publicação em nuvem, utilizou-se o Docker como ferramenta de implantação, garantindo compatibilidade entre os ambientes de desenvolvimento e produção e simplificando o processo de disponibilização.

Nos testes funcionais, o sistema respondeu rapidamente às operações de cadastro, edição e exclusão de contas, com sincronização correta entre a interface e o banco de dados. A renderização da listagem principal executou oito consultas SQL, número considerado baixo para aplicações administrativas com relacionamentos, demonstrando eficiência no uso do ORM Eloquent. O fluxo de marcação de contas como pagas exigiu apro-

ximadamente cinco interações do usuário, abrir o registro, alterar o status, selecionar o método de pagamento, confirmar e salvar, mantendo um nível adequado de usabilidade. O campo de data é preenchido automaticamente com o dia atual, o que reduz a necessidade de ações adicionais.

A camada de *widgets* financeiros, composta por gráficos e indicadores, operou de forma estável e manteve a consistência das informações exibidas. A arquitetura MVC do Laravel, combinada aos componentes administrativos do Filament, garantiu separação clara entre lógica de negócio, interface e persistência de dados, facilitando a manutenção e a evolução futura do sistema. Os recursos visuais integrados permitiram a visualização de indicadores em tempo real sem dependência de ferramentas externas, concentrando funcionalidades administrativas em um único ambiente.

Em comparação com Pramudya et al. (2025), que desenvolveu um sistema para clínicas odontológicas com Laravel e Filament adotando práticas do Extreme Programming (XP), observa-se convergência no uso de ferramentas que aceleram o desenvolvimento de painéis administrativos. A principal diferença está no domínio do problema: enquanto o trabalho de Pramudya et al. (2025) aborda operações mais estáveis (cadastros e relatórios clínicos), este estudo contempla um contexto financeiro dinâmico, com regras automatizadas (juros e multas), filtragem temporal e gráficos interativos. Além disso, a validação em dois bancos de dados e a implantação em nuvem ampliam o alcance prático do protótipo.

O trabalho de Almeida (2018), *UFOP Faltas*, é uma referência no cenário de informatização acadêmica. Embora utilize uma base tecnológica distinta e anterior, o estudo enfatiza organização de dados, controle de acesso e usabilidade, aspectos que dialogam com este projeto. Aqui, o foco é mais técnico e corporativo, priorizando consistência entre ambientes, portabilidade e automação de rotinas financeiras. Ainda assim, ambos convergem na busca por eficiência e padronização de processos administrativos.

Por fim, o estudo de Silva (2019) aborda a migração de um sistema legado em PHP para uma arquitetura MVC com Laravel, destacando desempenho e manutenibilidade. O presente trabalho complementa essa abordagem ao integrar a camada administrativa do Filament, eliminando a necessidade de construção manual de painéis, formulários e filtros, além de oferecer autenticação e personalização nativas. A implantação em nuvem via Docker também contribuiu para manter o funcionamento consistente da

aplicação. Em conjunto, o protótipo de contas a pagar pode ser compreendido como uma evolução natural de aplicações MVC tradicionais, adequada a contextos que demandam rapidez de desenvolvimento e compatibilidade multiplataforma.

De modo geral, os resultados obtidos demonstram que o Filament se mostrou produtivo e consistente para a construção de sistemas administrativos em Laravel. O tempo reduzido de desenvolvimento, a execução estável em diferentes ambientes e o número limitado de consultas ao banco reforçam a adequação da ferramenta ao propósito do estudo. As observações também indicam potencial de evolução do protótipo, especialmente com a implementação futura de ações em lote e métricas complementares de usabilidade.

5 CONCLUSÃO

O presente trabalho apresentou o desenvolvimento de um protótipo de sistema administrativo para controle de contas a pagar, utilizando o framework Laravel 12.x e a biblioteca Filament 4.x. O estudo teve como propósito analisar, de forma aplicada, o uso de ferramentas do ecossistema Laravel na criação de painéis administrativos modernos e portáteis. A aplicação foi estruturada com base no padrão MVC e implantada em ambiente local e em nuvem, demonstrando estabilidade e consistência entre as execuções.

A pesquisa contemplou o levantamento exploratório das principais ferramentas administrativas do ecossistema Laravel e a implementação prática de um protótipo funcional, evidenciando a integração entre os componentes nativos do framework e o Filament. A observação técnica realizada durante o desenvolvimento permitiu identificar ganhos de produtividade, padronização de código e facilidade de manutenção, reforçando o potencial da ferramenta para aplicações corporativas de pequeno e médio porte.

Como contribuição, o trabalho reúne práticas de desenvolvimento que podem servir de referência para projetos que demandam rapidez, portabilidade e coerência visual em sistemas internos. A combinação entre *migrations*, *seeders*, Eloquent ORM e Filament demonstrou eficiência na construção de aplicações administrativas completas com baixo esforço de codificação. Entre as limitações observadas, destacam-se a utilização de dados de demonstração e a ausência de métricas quantitativas mais amplas de desempenho e usabilidade.

Para trabalhos futuros, recomenda-se a ampliação das regras de negócio financeiras, a inclusão de relatórios e auditorias detalhadas e a comparação controlada entre diferentes ferramentas administrativas, como Filament, Nova e Voyager. Também se sugere a integração com serviços externos de pagamento, a implementação de testes automatizados e a análise de métricas de usabilidade, de modo a aprofundar a compreensão sobre a produtividade e o desempenho de frameworks administrativos no ecossistema Laravel.

REFERÊNCIAS

ALMASI, S. et al. **Usability evaluation of dashboards: a systematic literature review of tools**. BioMed Research International, 2023, Wiley Online Library, v. 2023, n. 1,

ALMEIDA, B. C. **Desenvolvimento de um sistema web para controle de faltas da universidade federal de ouro preto.**

ATHOTA, K. **Database Migration and Version Control: The Ultimate Guide for Beginners**. 2024. <https://www.xcubelabs.com/blog/database-migration-and-version-control-the-ultimate-guide-for-beginners/>. Acesso em: 18 out. 2025.

AURORA, C. **The Art of Database Migrations in Laravel — Common Mistakes**. 2025. <https://dev.to/cyber_aurora_/the-art-of-database-migrations-in-laravel-common-mistakes-3kkc>. Acesso em: 18 out. 2025.

BACH, B. et al. **Dashboard design patterns**. arXiv preprint arXiv:2205.00757, 2022.

CHAPETTA, W. A. **A influência de fatores na produtividade do desenvolvimento de software de acordo com um modelo de estruturas teóricas**. 2018,

CURIE, D. H. et al. Analysis on web frameworks. In: IOP PUBLISHING. **Journal of Physics: Conference Series**. [S.l.], 2019. v. 1362, n. 1, p. 012114.

FERREIRA, C. F. **Melhorias na gestão empresarial de micro e pequenos negócios com a implementação de ferramentas de processos**. 2023,

Filament. **Documentação do Filament**. 2024. Acesso em: 22 set. 2024. Disponível em: <https://filamentphp.com/docs>.

HUSSAIN, D. A. A. **Design and implementation of library management system**.

Laravel. **Documentação oficial do Laravel**. 2025. Acesso em: 21 out. 2025. Disponível em: <https://laravel.com/docs/12.x>.

Laravel. **Laravel Nova Documentation**. 2025. Acesso em: 8 maio 2025. Disponível em: <https://nova.laravel.com>.

Ministério da Economia. **Mapa de Empresas: Boletim 3º Quadrimestre 2023**. 2023. Acesso em: 22 set. 2024. Disponível em: <https://www.gov.br/empresas-e-negocios/pt-br/mapa-de-empresas/boletins/mapa-de-empresas-boletim-3o-quadrimestre-2023.pdf>.

PRAMUDYA, B. et al. **Implementation of extreme programming (xp) in the development of dental clinic information systems**. Journal of Enhanced Studies in Informatics and Computer Applications, 2025, v. 2, n. 1,

Prophix Software. **Excel vs. Specialized Planning Software: Making the Right Choice**. 2024. Acesso em: 5 out. 2024. Disponível em: <https://br.prophix.com/blog/excel-vs-specialized-planning-software-making-the-right-choice/>.

SILVA, C. F. S. **Automação residencial via smartphone**. [S.l.: s.n.], 2022.

SILVA, E. D. de M. **Desenvolvimento de uma ferramenta de gestão para construção civil utilizando o framework laravel mvc.** 2019,

SUBECZ, Z. **Web-development with laravel framework**. Gradus, 2021, John von Neumann University, v. 8, n. 1,

Voyager. **Documentação do Laravel Voyager**. 2025. Acesso em: 21 out. 2025. Disponível em: <https://voyager.devdojo.com>.