# Finance Vision: Sistema de Controle de Finanças Pessoais para Jovens

**Anthoni Liederson da Luz**[1], **Elias Vinicius Raitz de Oliveira**[1], **Edinilson da Silva Vida**[1]

[1]Instituto Federal de Santa Catarina – Câmpus Lages
R. Heitor Villa Lobos, 225 – São Francisco, Lages – SC, 88506-400

{anthoni.l, elias.vr27}@aluno.ifsc.edu.br

{edinilson.vida@ifsc.edu.br}

***Abstract.*** *This paper describes the development of Finance Vision, a web system designed to assist young people in managing their personal finances. The platform integrates essential features such as recording income and expense transactions, importing bank statements, viewing graphical reports, and setting personalized goals. As a key feature, the system incorporates gamification elements and financial tips to promote user engagement and financial literacy. The solution was developed using the Laravel framework and MySQL DBMS, prioritizing an intuitive and responsive interface. The adopted methodology included an incremental development cycle and usability validation through the System Usability Scale (SUS) questionnaire. The results indicated high acceptance and usability (95.00 in SUS score), confirming the tool's potential to contribute to the financial awareness and organization of the target audience.*

***Resumo.*** *Este artigo descreve o desenvolvimento do Finance Vision, um sistema web projetado para auxiliar jovens no controle de suas finanças pessoais. A plataforma integra funcionalidades essenciais como registro de transações de receitas e despesas, importação de extratos bancários, visualização de relatórios, gráficos e definição de metas personalizadas. Como diferencial, o sistema incorpora elementos de gamificação e dicas financeiras para promover o engajamento e a educação financeira dos usuários. A solução foi desenvolvida com o framework Laravel e SGBD MySQL, priorizando uma interface intuitiva e responsiva. A metodologia adotada incluiu um ciclo de desenvolvimento incremental e a validação da usabilidade por meio do questionário SUS (System Usability Scale). Os resultados indicaram alta aceitação e usabilidade (95.00 no score do SUS), confirmando o potencial da ferramenta para contribuir com a conscientização e organização financeira do público-alvo.*

## 1. Introdução

O controle e a organização financeira são fundamentais para o desenvolvimento pessoal e profissional dos jovens. Entretanto, a falta de educação financeira nas escolas e universidades dificulta a criação de hábitos financeiros saudáveis entre os jovens brasileiros (Reizes, 2023). Embora não exija conhecimentos matemáticos avançados, a gestão financeira requer disciplina e constância para que os objetivos pessoais e profissionais possam ser alcançados. Contudo, segundo Hill et al. (2014), a rotina corrida e a falta de tempo

podem contribuir para a ausência de interesse em manter uma organização financeira adequada. Os autores também ressaltam que fatores do cotidiano exercem influência direta nos hábitos financeiros dos jovens.

Diante desse cenário, é importante compreender como a falta de educação financeira se conecta diretamente ao crescente endividamento juvenil no Brasil. O endividamento precoce entre jovens brasileiros com idades entre 15 e 30 anos é um problema crescente e preocupante. Segundo Batista (2023), 33% desses jovens estão simultaneamente desempregados e endividados, 71% dos endividados permanecem inadimplentes há mais de um ano, e 50% não acompanham suas dívidas e nem conhecem as taxas de juros aplicadas. Esses dados refletem uma combinação de fatores, como a má organização financeira, o uso excessivo do cartão de crédito, a carência de educação financeira e o fácil acesso a bancos digitais. Além dos impactos financeiros e psicológicos, o endividamento pode acarretar restrições no Cadastro de Pessoa Física (CPF). Essas limitações podem dificultar etapas importantes da vida adulta, como a obtenção de financiamentos para compra de imóveis, veículos ou, até mesmo, crédito pessoal no mercado. Diante desse cenário, torna-se evidente a necessidade de soluções tecnológicas que incentivem hábitos financeiros mais conscientes e saudáveis entre os jovens.

Considerando o cenário apresentado, de que maneira a tecnologia pode contribuir de forma eficaz para o desenvolvimento de hábitos financeiros mais organizados entre os jovens? Tendo em vista esse panorama, percebe-se a necessidade de soluções digitais que auxiliem os jovens a gerenciar melhor seu dinheiro de forma prática, acessível e integrada às tecnologias que fazem parte de sua rotina, buscando um hábito mais saudável e responsável no planejamento financeiro em seu cotidiano. Assim, este projeto tem como objetivo geral propor o desenvolvimento de uma plataforma *web* chamada Finance Vision, voltada ao controle financeiro pessoal de jovens. Para atingir o objetivo geral, os seguintes objetivos específicos foram determinados:

- Integrar elementos de gamificação à plataforma com o intuito de aumentar o engajamento dos usuários;
- Avaliar a usabilidade e a eficácia do sistema por meio de testes com o usuário;
- Analisar e comparar as principais tecnologias de Inteligência Artificial (IA), avaliando sua eficácia na extração e estruturação de dados de diferentes *layouts* de arquivos CSV;
- Realizar *benchmarking* de soluções de educação financeira com uso de IA para identificar limitações e lacunas, justificando a criação de uma nova solução.

A metodologia foi composta por cinco etapas, visando alcançar os objetivos propostos com foco no desenvolvimento de funcionalidades robustas, visualizações intuitivas e estratégias de engajamento. Na etapa 1, foi realizada a coleta e análise dos requisitos funcionais e não funcionais da plataforma. Essa fase envolveu a definição das funcionalidades de cadastro de usuários, movimentações financeiras, geração de relatórios e integração de gamificação. Na etapa 2, foi elaborado o projeto da arquitetura da aplicação, contemplando uma estrutura escalável e modular. O *front-end* foi desenvolvido utilizando HTML, CSS, JavaScript juntamente com o *template* Blade, enquanto o *back-end* foi implementado com PHP e o *framework* Laravel. O Sistema Gerenciador de Banco de Dados (SGBD) utilizado foi o MySQL.

Dando continuidade ao processo, na etapa 3, foram desenvolvidas as funcionali-

dades básicas da plataforma, incluindo o cadastro e autenticação de usuários, registro de transações financeiras (entradas e saídas), e controle de categorias de gastos e receitas. Na etapa 4, ocorreu a criação de funcionalidades interativas que permitiram o usuário analisar seu comportamento financeiro ao longo do tempo, com visualizações de relatórios e painéis gráficos. Também foram implementados elementos de gamificação, como sistema de pontos, *badges* e metas financeiras. Na etapa 5, após o desenvolvimento das funcionalidades, a plataforma foi submetida a testes com usuários reais. Foram aplicados testes de usabilidade, no caso o *System Usability Scale* (SUS) para validar a experiência de uso. Com base no *feedback* dos testes, foram realizadas melhorias contínuas na interface e nas funcionalidades.

O estudo adotou uma abordagem qualitativa, utilizando o teste *System Usability Scale* (SUS) com o objetivo de coletar dados consistentes sobre a usabilidade do sistema. Em relação à sua natureza, trata-se de uma pesquisa aplicada, pois busca desenvolver uma solução prática voltada ao controle financeiro dos jovens com o apoio de tecnologias. O objetivo da pesquisa é exploratório, já que procura entender o problema de forma mais profunda e reunir informações que ajudem a aprimorar a solução proposta. Os métodos utilizados envolvem pesquisa bibliográfica, que fornece a base teórica e prática tanto para o desenvolvimento do sistema quanto para a compreensão do cenário financeiro estudado.

Este trabalho está organizado em seis seções. A seção 1 faz uma introdução ao tema central, destacando sua relevância e explicando por que o projeto é necessário e importante. A seção 2 apresenta o referencial teórico que dá base e sustenta a pesquisa. Já na seção 3, são descritos os requisitos do sistema e a modelagem da plataforma. A seção 4 detalha as funcionalidades implementadas durante o desenvolvimento. Em seguida, a seção 5 traz uma análise e a discussão dos resultados obtidos. Por fim, na seção 6, são apresentadas as considerações finais, com um resumo das principais descobertas e definindo a conclusão do sistema.

## 2. Referencial Teórico

Esta seção é dividida em quatro partes. A seção 2.1 apresenta o contexto da educação financeira no Brasil, com ênfase na realidade enfrentada pelos jovens, incluindo dados estatísticos, comportamento financeiro e falta de conhecimento sobre finanças pessoais. A seção 2.2 aborda conceitos técnicos da ferramenta como gamificação, explicando suas características, benefícios e como esses recursos podem ser aplicados para incentivar hábitos saudáveis no contexto da educação financeira. Na sequência, na seção 2.3 é apresentado o teste SUS de usabilidade, utilizado para avaliar a experiência dos usuários com a plataforma, por meio de uma escala padronizada que mede a facilidade de uso, a eficiência e a satisfação percebida durante a interação com o sistema. Por fim, na seção 2.4, são analisados trabalhos semelhantes, destacando funcionalidades, pontos positivos, limitações e contribuições de cada sistema identificado para fundamentar o sistema proposto.

### 2.1. Educação Financeira e Controle Financeiro

A educação financeira é considerada um dos pilares fundamentais para a formação de indivíduos mais conscientes, capazes de administrar seus recursos com responsabilidade e autonomia. Quando inserida desde cedo no processo educativo, ela favorece o desenvolvimento de competências essenciais para o controle financeiro pessoal, como o planeja-

mento, a definição de metas, a avaliação de prioridades e a tomada de decisões informadas sobre consumo, poupança e investimento.

No Brasil, o tema ganhou maior relevância com a instituição da Estratégia Nacional de Educação Financeira (ENEF) e do Fórum Brasileiro de Educação Financeira (FBEF), por meio do Decreto nº 10.393/2020. O principal objetivo da ENEF é promover ações de orientação voltadas à construção de hábitos financeiros saudáveis, abrangendo o consumo consciente, o planejamento orçamentário, a educação previdenciária e a cidadania fiscal. A iniciativa busca integrar a educação financeira aos currículos escolares, às ações comunitárias e aos meios de comunicação, fortalecendo o papel das políticas públicas na redução do endividamento, na promoção da inclusão social e no estímulo à cidadania econômica (Brasil, 2020).

Para os jovens, especialmente, a educação financeira contribui diretamente para o desenvolvimento de hábitos de controle e organização desde a fase inicial da vida econômica. Segundo Silva et al. (2018), ao compreenderem a lógica do dinheiro, seus direitos e deveres como consumidores, e o impacto de suas escolhas de curto prazo, os jovens se tornam mais aptos a adotar práticas responsáveis no uso dos recursos tornando o indivíduo com mais autonomia na construção de um futuro sustentável.

Em complemento, de acordo com Silva e Monteiro (2023), a educação financeira busca promover o conhecimento e as habilidades necessárias para uma gestão eficiente do dinheiro, tomada de decisões financeiras embasadas e conquista da estabilidade financeira.

### 2.1.1. Conceitos de Controle Financeiro Pessoal

O controle financeiro pessoal pode ser compreendido como um conjunto de práticas e estratégias utilizadas por indivíduos para administrar seus recursos, com o propósito de equilibrar receitas e despesas, evitar o endividamento e alcançar metas econômicas definidas. Trata-se de um processo contínuo que envolve planejamento, organização, monitoramento e decisões conscientes sobre o uso do dinheiro no cotidiano.

Nesse contexto, Andrade e Carraro (2018) ressaltam que o domínio sobre as finanças pessoais contribui significativamente para a previsibilidade econômica, além de reduzir o estresse financeiro e promover a qualidade de vida. Os autores reforçam o papel central da educação financeira nesse processo, ao capacitar o indivíduo para reconhecer seus próprios padrões de consumo e realizar escolhas mais coerentes com seus objetivos pessoais e profissionais.

Entre os principais conceitos que viabilizam o controle financeiro estão o registro sistemático das movimentações, o acompanhamento do fluxo de caixa, a definição de metas e orçamentos, e a análise crítica dos hábitos de consumo. Quando aplicadas de forma consistente, essas práticas auxiliam especialmente os jovens a identificar desequilíbrios, repensar seus gastos e tomar decisões alinhadas à sua realidade. Assim, o controle financeiro pessoal torna-se não apenas uma ferramenta de gestão, mas também um pilar formativo da educação financeira, promovendo autonomia e responsabilidade no uso consciente dos recursos.

### 2.1.2. Métodos Tradicionais e Digitais de Controle de Despesas e Receitas

Ao longo de décadas, o controle financeiro pessoal foi conduzido por métodos tradicionais, como anotações manuais em cadernos, agendas ou planilhas físicas. Essas ferramentas, apesar de simples e acessíveis, exigem alto grau de constância e motivação por parte do usuário, uma vez que cada transação precisa ser registrada manualmente. Segundo Cerbasi (2015), esse processo favorece a consciência dos gastos, pois torna o indivíduo mais atento às suas movimentações financeiras ao envolver um esforço ativo de monitoramento.

Entretanto, os métodos manuais apresentam limitações importantes, especialmente no que diz respeito à categorização automática de despesas, à geração de relatórios analíticos e à visualização de *dashboards* e gráficos. Essa ausência de recursos automatizados pode dificultar a identificação de padrões de consumo e comprometer a agilidade na tomada de decisões financeiras mais estratégicas, principalmente em contextos de alta complexidade ou fluxo intenso de transações.

Com o avanço da tecnologia e a crescente digitalização da vida cotidiana, principalmente dos jovens, surgiram soluções digitais voltadas à gestão financeira pessoal, como aplicativos móveis, planilhas eletrônicas automatizadas e plataformas *online*. De acordo com Tavares et al. (2024), o uso de ferramentas digitais pode contribuir significativamente para o desenvolvimento da autonomia financeira, especialmente entre jovens familiarizados com tecnologias digitais. Nesse contexto, estratégias como a gamificação vêm sendo usadas para tornar o processo de aprendizagem e controle financeiro mais atrativo e engajador.

## 2.2. Gamificação

A gamificação é definida como o uso de elementos e mecânicas de jogos em contextos que não são, originalmente, voltados para jogos, como ambientes corporativos, educacionais e de saúde (Deterding et al., 2011). Trata-se de uma abordagem que busca incorporar características típicas dos jogos, como desafios, recompensas e *feedbacks* constantes para promover o engajamento, a motivação e a participação ativa dos usuários em tarefas do cotidiano (Zichermann e Cunningham, 2011). Ao contrário dos jogos tradicionais, a gamificação não exige que a atividade gamificada seja, de fato, um jogo completo, mas sim que utilize aspectos de *design* de jogos para melhorar a experiência do usuário.

A distinção entre gamificação estrutural e de conteúdo também é fundamental. A gamificação estrutural se preocupa com a forma como o usuário interage com um conteúdo sem necessariamente alterá-lo, enquanto a gamificação de conteúdo modifica o próprio conteúdo para torná-lo mais parecido com um jogo (Alves, 2015). Ambas as abordagens têm valor didático e podem ser utilizadas em diferentes contextos, desde o ambiente educacional até aplicativos de finanças pessoais, dependendo dos objetivos da aplicação.

Werbach e Hunter (2012) destacam que a gamificação é uma forma de transformar comportamentos através da ludificação de processos. Sua eficácia está diretamente relacionada à capacidade de provocar experiências positivas, como o sentimento de progresso, superação e pertencimento. A utilização da gamificação em sistemas e plataformas digitais, como *sites* de finanças pessoais, tem como objetivo central facilitar a adoção de

comportamentos desejáveis, como a organização financeira e o cumprimento de metas pessoais, por meio de experiências mais envolventes e agradáveis.

### 2.2.1. Elementos de Gamificação: Pontos, *Badges*, Níveis e Recompensas

A Figura 1 ilustra como a mecânica de jogos pode ser aplicada de três formas distintas: para motivar por meio da gamificação, engajando usuários em atividades não lúdicas; para compensar, através de programas de recompensa que incentivam comportamentos desejados; e para entreter, no uso tradicional em *videogames*, oferecendo diversão e desafios.

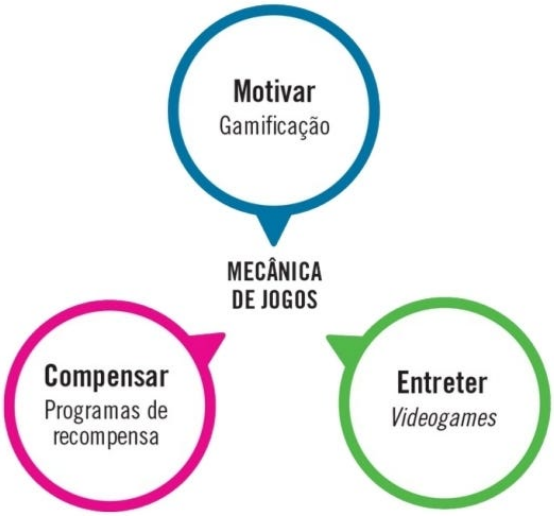



**Figura 1. Gamificação, programas de recompensa e videogames. Fonte: Ferraz (2023).**

A Tríade PBL — *Points* (pontos), *Badges* (insígnias) e *Leaderboards* (quadros de liderança) — constitui a base dos sistemas de gamificação mais utilizados. Esses elementos funcionam como mecanismos de recompensa e *feedback* que orientam o progresso dos usuários, reforçam comportamentos positivos e promovem a competitividade saudável (Zichermann e Cunningham, 2011). Pontos são atribuídos por ações realizadas; *badges* servem como reconhecimento por conquistas; níveis representam o avanço contínuo; e recompensas podem ser simbólicas ou tangíveis.

O uso consciente dos elementos clássicos de gamificação, como pontos, *badges* e níveis, não garantem o sucesso de uma iniciativa. Esses elementos devem fazer parte de um sistema maior de engajamento, no qual o *design* considera as metas do negócio e os desejos dos participantes. Um sistema gamificado eficaz é aquele que combina esses elementos com desafios significativos, *feedback* contínuo e recompensas que representem conquistas reais para o usuário (Burke e Gartner, 2015).

### 2.2.2. Gamificação no Contexto da Educação e Mudança de Comportamento

A Figura 27,descrita no Apêndice 1, destaca os impactos positivos da gamificação no ambiente corporativo, com dados da Tamboro (2020). Segundo a pesquisa, 83% dos

participantes de treinamentos gamificados se sentem mais motivados, enquanto 61% dos que fazem treinamentos não gamificados relatam tédio. Além disso, 89% afirmam que a gamificação aumenta sua produtividade e proporciona sentimento de pertencimento e propósito. No recrutamento, 79% consideram empresas com gamificação mais atrativas, reforçando que essa abordagem vai além do engajamento, influenciando positivamente tanto no desempenho quanto na percepção sobre a organização.

A gamificação tem se mostrado eficaz no contexto educacional por sua capacidade de tornar o processo de aprendizagem mais atrativo e interativo. Como afirmam Reguze e Silva (2016), o engajamento do aluno é um dos principais fatores para o sucesso educacional, e a gamificação atua como um mecanismo para sustentar esse engajamento por meio da motivação contínua. Elementos como desafios progressivos, *feedback* imediato e reconhecimento promovem uma aprendizagem ativa, incentivando a participação e o comprometimento dos estudantes.

Essa abordagem também influencia mudanças comportamentais significativas. Através da gamificação, é possível desenvolver competências do século XXI, como pensamento crítico, criatividade, colaboração e comunicação. Além disso, ela contribui para a autorregulação da aprendizagem, onde o aluno se torna responsável por seu próprio progresso e resultados.

Para Kapp (2012), a gamificação pode ser compreendida como o uso do pensamento e das mecânicas dos jogos para engajar pessoas, motivar ações, promover o aprendizado e resolver problemas. Esta prática está enraizada em fundamentos da psicologia comportamental, como o reforço positivo, e da teoria da motivação, que distinguem entre motivação intrínseca (vontade interna de realizar uma atividade) e extrínseca (motivada por recompensas externas). A aplicação desses conceitos permite criar experiências que direcionam comportamentos de forma estratégica e planejada.

### 2.2.3. Aplicação da Gamificação para Promoção da Educação Financeira

Aplicando a gamificação de forma prática, Burke e Gartner (2015) argumentam que há um grande potencial em áreas que envolvem educação e aprendizagem, influenciando o comportamento e criando expectativas sólidas e engajadoras para se obter conhecimento. Ao conciliar a gamificação com educação financeira, sistemas bem projetados ajudam as pessoas a manterem o foco, superarem dificuldades e se sentirem recompensadas pelo esforço. Mais do que isso, a gamificação pode transformar o processo de aprendizagem em algo mais pessoal e motivador, facilitando o desenvolvimento de competências importantes como resiliência, disciplina e autorregulação.

Hamari et al. (2014) afirma que a gamificação tem o potencial de melhorar a motivação de maneira intrínseca nos indivíduos ao tornar o processo de aprendizagem mais envolvente e divertido, o que é particularmente relevante em temas complexos como finanças pessoais. Isso permite que as pessoas desenvolvam habilidades financeiras importantes de forma mais natural e contínua, reduzindo a resistência a conteúdos tradicionalmente considerados complexos.

Outro aspecto importante é o papel da personalização em sistemas gamificados voltados à educação financeira. Deterding et al. (2011) reforça que adaptar os desa-

fios e recompensas ao perfil do usuário pode aumentar significativamente a eficácia da gamificação, pois promove uma experiência mais significativa e alinhada aos objetivos individuais. Por exemplo, um usuário com dificuldades em poupar dinheiro pode ser motivado por conquistas relacionadas à criação de uma meta financeira. Assim, ao alinhar objetivos financeiros pessoais com mecânicas de pequenas conquistas, a gamificação promove um aprendizado mais eficaz e orientado a resultados.

### 2.2.4. Aderência da Gamificação à Lei Geral de Proteção de Dados (LGPD)

A Lei Geral de Proteção de Dados (LGPD), instituída pela Lei nº 13.709/2018, descreve o tratamento de dados pessoais no âmbito digital, de pessoa física, jurídica, de direito público e privado, visando proteger os direitos à integridade, privacidade e liberdade das pessoas Brasil (2018). O que motivou o surgimento de leis relacionadas à proteção de dados pessoais de maneira consolidada e regulamentada foi o desenvolvimento de modelos de negócios da economia digital, que passou a depender de um fluxo internacional de base de dados relacionados às pessoas (Pinheiro, 2021).

No contexto da aplicação da gamificação para a educação financeira, é essencial considerar a aderência à Lei Geral de Proteção de Dados Pessoais (LGPD), uma vez que sistemas gamificados frequentemente coletam e processam dados sensíveis dos usuários, como informações de consumo, hábitos financeiros e metas pessoais. A LGPD estabelece diretrizes claras para o tratamento de dados, exigindo transparência, consentimento explícito e finalidade específica na coleta dessas informações (Pinheiro, 2021). Assim, garantir a conformidade com a legislação não apenas protege os direitos dos usuários, mas, também, fortalece a confiança no sistema gamificado.

Outrossim, de acordo com o artigo 52 da Lei nº 13.709/2018, as penalidades para o descumprimento das disposições legais incluem advertências, multas simples de até 2% do faturamento da empresa no Brasil, limitadas a R$ 50 milhões por infração, multas diárias, publicização da infração, bloqueio ou eliminação dos dados pessoais envolvidos, suspensão parcial ou total do funcionamento do banco de dados ou das atividades de tratamento de dados pessoais por até seis meses, prorrogável por igual período, e até mesmo a proibição parcial ou total do exercício de atividades relacionadas ao tratamento de dados. Essas sanções são aplicadas pela Agência Nacional de Proteção de Dados (ANPD) após procedimento administrativo que assegure o direito à ampla defesa e ao contraditório (Brasil, 2018).

Como é possível, identificar, é essencial que as plataformas gamificadas, especialmente aquelas voltadas à educação financeira, adotem medidas rigorosas de conformidade com a LGPD para evitar tais penalidades e garantir a confiança dos usuários.

## 2.3. Teste SUS de Usabilidade

Antes de abordar o teste SUS de usabilidade, é fundamental compreender o conceito de usabilidade, que pode ser definido como um atributo fundamental na avaliação de sistemas interativos, sendo definida como o grau em que um produto pode ser utilizado por usuários específicos para alcançar objetivos determinados com eficácia, eficiência e satisfação em um determinado contexto de uso. De acordo com ISO (1998), esse conceito está diretamente relacionado à experiência do usuário ao interagir com interfaces,

abrangendo aspectos como facilidade de aprendizado, facilidade de uso, memorização de comandos, prevenção de erros e satisfação geral.

Em complemento, o SUS é um instrumento criado por Brooke (1986) com o objetivo de medir, de forma rápida e eficaz, a percepção dos usuários relacionado à usabilidade de um sistema. Em uma análise posterior, Brooke (2013) ressaltou a eficiência e a ampla aceitação do SUS em pesquisas de usabilidade, evidenciando sua versatilidade na avaliação de diferentes tipos de sistemas. No contexto deste trabalho, a aplicação do SUS busca identificar oportunidades de aprimoramento na experiência do usuário, de forma prática e com base em dados claros.

Para analisar a usabilidade da plataforma, foi aplicado o questionário padrão do SUS, composto por 10 afirmações que abordam aspectos como facilidade de aprendizado, simplicidade da interface e integração das funcionalidades. Cada item é avaliado em uma escala do tipo *Likert*, variando de 1 (discordo totalmente) a 5 (concordo totalmente). A pontuação do SUS é calculada a partir das respostas individuais.

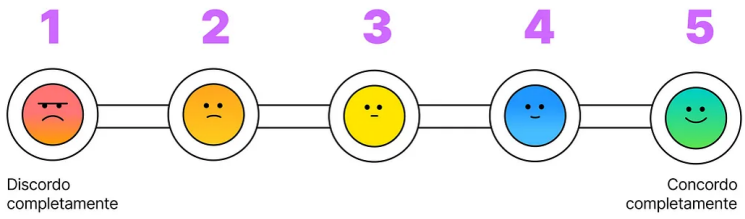



**Figura 2. Escala *Likert* do SUS. Fonte: UX Design Brasil (2022).**

De acordo com a Figura 2, é possível visualizar a representação gráfica da escala *Likert* utilizada no questionário SUS para avaliação da usabilidade do sistema. Cada ponto da escala está associado a uma expressão facial que facilita a interpretação visual por parte dos participantes, indo de uma face descontente (vermelha) até uma face satisfeita (verde). Essa abordagem tem como objetivo tornar o processo de resposta mais intuitivo, principalmente para usuários com menor familiaridade com avaliações formais, contribuindo para respostas mais espontâneas e confiáveis.

As dez afirmações utilizadas no questionário SUS, aplicadas aos participantes desta pesquisa, foram as seguintes:

1. Eu acredito que gostaria de utilizar este sistema com frequência.
2. Considero o sistema desnecessariamente complexo.
3. Achei o sistema fácil de usar.
4. Acredito que precisaria da ajuda de uma pessoa com conhecimentos técnicos para utilizar o sistema.
5. As funcionalidades do sistema me pareceram bem integradas.
6. Percebi inconsistências no funcionamento do sistema.
7. Suponho que a maioria das pessoas aprenderia a utilizar este sistema com rapidez.
8. Achei o sistema confuso ou desorganizado em sua utilização.
9. Senti-me confiante ao usar o sistema.
10. Precisei aprender muitas coisas novas antes de conseguir utilizar o sistema de forma eficaz.

A Figura 3 ilustra a pontuação de aceitabilidade do SUS, que varia de 0 a 100. Essa pontuação reflete a percepção dos usuários em relação a facilidade de uso e a eficácia sistema avaliado.

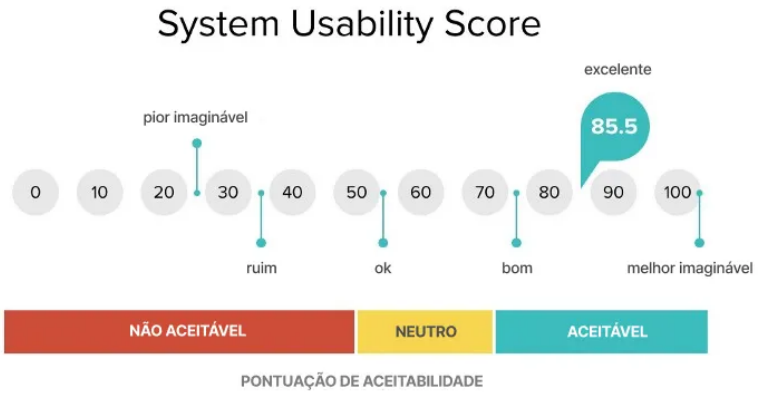



**Figura 3. *System Usability Score* Fonte: UX Design Brasil (2022).**

O cálculo do SUS segue um processo padronizado que envolve três etapas principais. Primeiramente, é necessário ajustar as respostas conforme a posição da pergunta no questionário. Para os itens de número ímpar (1, 3, 5, 7 e 9), subtrai-se 1 do valor da resposta fornecida pelo participante. Já para os itens pares (2, 4, 6, 8 e 10), o cálculo é feito subtraindo o valor da resposta de 5. Esse procedimento tem como objetivo padronizar a escala para que todos os itens contribuam de forma coerente para a pontuação total.

Na segunda etapa, todos os valores ajustados são somados, gerando uma pontuação intermediária. Por fim, na terceira etapa, esse total é multiplicado por 2,5, resultando na pontuação final do SUS, que varia de 0 a 100. Essa pontuação representa a percepção geral de usabilidade do sistema avaliado, sendo que pontuações acima de 68 geralmente indicam uma boa experiência de uso.

## 2.4. Trabalhos Similares

Nesta subseção, são analisados trabalhos voltados para o controle e educação financeira. A análise considera as principais funcionalidades, vantagens, limitações e contribuições de cada projeto. O objetivo é identificar tecnologias, boas práticas e lacunas existentes, que possam fundamentar e justificar as escolhas feitas no desenvolvimento do Finance Vision.

Durante o processo de levantamento e seleção de referências, foi realizada uma pesquisa no Google Acadêmico, abrangendo o período de 2017 a 2023. A pesquisa teve como objetivo identificar estudos relacionados ao controle de finanças pessoais entre jovens, com ênfase em práticas de educação financeira, definição de metas e uso de ferramentas digitais de monitoramento financeiro.

Para avaliar os trabalhos selecionados, elaborou-se uma matriz de pertinência com base em critérios previamente definidos, inspirados na abordagem metodológica de Garrard (2022). Os critérios adotados foram: (i) portabilidade do sistema, (ii) se possui *dashboards*, (iii) *framework* utilizado, (iv) se possui geração de relatórios, (v) Sistema

Gerenciador de Banco de Dados utilizado, (vi) se apresenta recursos de gamificação e (vii) se disponibiliza guia de educação financeira. Cada estudo foi analisado quanto à presença ou ausência desses critérios, o que permitiu avaliar sua relevância para os objetivos deste trabalho. Com base nessa análise, foram destacados os cinco estudos mais alinhados ao escopo da pesquisa, os quais foram organizados em uma tabela comparativa que sintetizam suas principais características conceituais, metodológicas e tecnológicas.

Algumas das expressões utilizadas durante a etapa de busca bibliográfica incluíram: controle de finanças pessoais, educação financeira para jovens, uso de aplicativos para controle de gastos, gestão financeira com tecnologia digital, hábitos de consumo e educação financeira, ferramentas digitais de planejamento financeiro e autonomia financeira entre estudantes.

Ao término da análise, os trabalhos que atenderam a um maior número de critérios definidos na matriz de pertinência foram considerados os mais relevantes para embasar a fundamentação teórica do Finance Vision. Esses estudos apresentaram alinhamento conceitual e metodológico com os objetivos da proposta, especialmente no que se refere ao incentivo à autonomia financeira, ao uso de tecnologias acessíveis para o controle orçamentário e à promoção de práticas educativas voltadas à organização das finanças pessoais entre jovens. Na sequência, são apresentados os principais estudos identificados na literatura, destacando suas contribuições e relação com o sistema proposto.

O sistema de Souza (2023) apresenta uma aplicação *web* para gestão de finanças pessoais, desenvolvido com o *framework* Laravel e utilizando tecnologias como Bootstrap. A aplicação oferece funcionalidades como cadastro de receitas e despesas, visualização por categorias e gráficos interativos. A proposta é funcional, com interface responsiva e de fácil usabilidade, atendendo a requisitos bem definidos. No entanto, o sistema se limita a recursos convencionais, sem a utilização de gamificação ou notificações automatizadas, elementos considerados diferenciais na proposta deste projeto para aumentar o engajamento do público jovem. Além disso, o trabalho não contempla testes reais de usabilidade com usuários finais, nem explora os mecanismos de segurança de dados, o que representa uma limitação relevante para um sistema que lida com informações financeiras.

Em Oliveira (2023) é proposto um sistema *web* para controle financeiro pessoal que integra dados de contas bancárias em uma única plataforma. O sistema se destaca pela personalização de categorias, visualização de fluxo de caixa baseado em transações recorrentes e notificações via WhatsApp, recursos que ampliam o controle e a previsibilidade dos usuários. No entanto, o projeto não aborda diretamente aspectos de educação financeira ou estratégias de gamificação, limitando o potencial do projeto, especialmente entre o público jovem.

Nos estudos de Reizes (2023) é apresentado um aplicativo *web* denominado Finanças Pessoais, que futuramente pode ser portado para dispositivos móveis. Faz o uso de tecnologias como a linguagem PHP, o *framework* Laravel, e o Sistema Gerenciador de Banco de Dados MySQL. Utiliza a arquitetura *Model-View-Controller* (MVC) para a organização do projeto e se destaca pelo uso do *template* AdminLte, trazendo uma interface bem intuitiva relacionada a aspectos financeiros. Contudo, há uma carência de funcionalidades relacionadas à educação financeira e, em relação ao trabalho proposto,

não possui características de gamificação que visam uma melhor interação e engajamento dos usuários.

Ramos (2021) criou uma aplicação *mobile* nativa para Android, voltada ao auxílio no controle financeiro de pessoas com pouca experiência ou leigas no assunto. Utiliza Kotlin e Java como linguagens de programação, além do Firebase Realtime Database para realizar a persistência dos dados financeiros por meio da nuvem. Tem como diferencial o uso de um *web service* para fazer conversões monetárias com moedas, como o Dólar, Euro, Iene e a Libra Esterlina em tempo real. Também possui a funcionalidade de simulação de investimentos, trazendo perspectivas futuras em relação a como o usuário pode realizar seus investimentos. Apresenta funcionalidades inovadoras, porém, não possui a implementação de outras básicas e fundamentais, como a geração de relatórios seguida de exportações para arquivos de diferentes formatos, o uso de recursos de gamificação e educação financeira para melhorar a usabilidade e o engajamento do sistema.

No trabalho de Alves (2017) é desenvolvido um sistema *web* móvel para controle de finanças pessoais com foco na geração de dados para declaração do imposto de renda. O sistema se destaca pela leitura automatizada de QR Code de notas fiscais eletrônicas, facilitando o registro de despesas. No entanto, a solução não contempla abordagens educativas ou gamificadas, o que limita seu impacto no desenvolvimento da educação financeira entre jovens usuários.

| **Autores** | **Portabilidade** | ***Dashboards*** | ***Framework*** | **Relatórios** | ***Database*** | **Gamificação** | **Educação** |
|---|---|---|---|---|---|---|---|
| Souza (2023) | *Web* | Sim | Laravel | Não | MariaDB | Não | Não |
| Oliveira (2023) | *Web* | Sim | Spring/Angular | Sim | PostgreSQL | Sim | Não |
| Reizes (2023) | *Web* | Sim | Laravel | Sim | MySQL | Não | Não |
| Ramos (2021) | *App* | Sim | Kotlin/Java | Não | Realtime DB | Não | Não |
| Alves (2017) | *Web/App* | Sim | MCV VRaptor | Sim | MySQL | Não | Não |
| (Sistema Proposto) | *Web* | Sim | Laravel | Sim | MySQL | Sim | Sim |

**Quadro 1. Comparação entre trabalhos similares.**

Conforme apresentado no Quadro 1, é possível observar uma análise comparativa entre o sistema proposto e soluções similares desenvolvidas em trabalhos anteriores. Embora todos os sistemas apresentem portabilidade *web* ou *mobile* e incluam *dashboards* como recurso visual de acompanhamento, nota-se que apenas o sistema proposto contempla, simultaneamente, funcionalidades de gamificação e recursos voltados à educação financeira. Essa combinação confere à solução um caráter diferencial, ao buscar não apenas o controle das finanças pessoais, mas também o estímulo à mudança de comportamento do usuário por meio do aprendizado e do engajamento contínuo. O sistema proposto adota o *framework* Laravel e o SGBD MySQL, tecnologias consolidadas e amplamente utilizadas no desenvolvimento *web*, garantindo uma aplicação robusta e escalável.

## 3. Especificações de Requisitos e Modelagem

Esta seção detalha a concepção e a estrutura do sistema proposto, sendo organizado em quatro seções. A seção 3.1 estabelece o que o sistema deve fazer e sob quais condições de operação. Em seguida, a seção 3.2 define as lógicas e restrições que nortearam os processos da aplicação. A seção 3.3 justifica as escolhas realizadas relacionadas as ferramentas e tecnologias. Por fim, a seção 3.4 apresenta os diagramas de arquitetura e as interfaces que serviram como base para a implementação.

## 3.1. Requisitos Funcionais e Não Funcionais

Nesta subseção, são apresentados os requisitos funcionais e não funcionais do Finance Vision. Os requisitos essenciais para o pleno funcionamento da plataforma estão organizados em duas categorias principais: requisitos funcionais, que descrevem as funcionalidades e comportamentos esperados do sistema; e requisitos não funcionais, que definem atributos de qualidade como segurança, desempenho, usabilidade e conformidade.

### 3.1.1. Requisitos Funcionais

Segundo Sommerville (2011), os Requisitos Funcionais (RF) especificam as operações fundamentais que o sistema deve executar para atender de forma eficaz e segura às necessidades dos usuários, promovendo autonomia financeira e organização pessoal.

- **RF01** – O sistema deve permitir o cadastro, *login* e alteração da senha do usuário, caso necessário.
- **RF02** – O sistema deve permitir o registro de receitas e despesas, com categorização por tipo, data e valor.
- **RF03** – O sistema deve disponibilizar relatórios para consulta do usuário.
- **RF04** – O sistema deve permitir a criação de metas financeiras personalizadas, com acompanhamento de progresso.
- **RF05** – O sistema deve permitir a visualização de gráficos no formato de pizza para acompanhar o desempenho das receitas do usuário.
- **RF06** – O sistema deve fornecer dicas e conteúdos educativos sobre controle financeiro.

### 3.1.2. Requisitos Não Funcionais

Segundo Sommerville (2011), os Requisitos Não Funcionais (RNF) estão relacionados a aspectos e características de qualidade e comportamento do sistema, como eficácia, confiabilidade, usabilidade e também a atender às demandas do usuário e do *software*.

- **RNF01** – A usabilidade do sistema deve ser intuitiva.
- **RNF02** – O sistema deve ser desenvolvido utilizando a linguagem PHP juntamente com o *framework* Laravel e o *template* Blade.
- **RNF03** – A persistência de dados deve ser implementada utilizando o SGBD MySQL.
- **RNF04** – O sistema deve estar em conformidade com a Lei Geral de Proteção de Dados (LGPD).
- **RNF05** – A aplicação deve conter estratégias de segurança, como uma autenticação via *login*, e consequentemente a criptografia de senhas.
- **RNF06** – O sistema deve ser compatível com os principais navegadores. Sendo eles, o Google Chrome, Mozilla Firefox e Microsoft Edge.

## 3.2. Regras de Negócio

As Regras de Negócio (RN), segundo Sommerville (2011), definem a estrutura e como o sistema deve se portar em cada situação, através de restrições, critérios e lógicas que reagem ao comportamento da plataforma. Isso garante segurança, coerência e eficácia na utilização do sistema.

- **RN01** – Os usuários devem realizar os cadastros de suas contas para terem acesso e utilização da plataforma, garantindo, assim, a autenticidade dos mesmos.
- **RN02** – Cada lançamento financeiro, seja ele uma receita ou uma despesa, deve conter dados mínimos. Sendo eles: data, categoria, descrição e o valor.
- **RN03** – O sistema deve aplicar regras de gamificação baseadas em comportamento financeiro. Os usuários recebem pontos por metas alcançadas, bem como pela frequência de uso da plataforma e o registro de despesas e receitas.
- **RN04** – A geração de relatórios deve abranger opções personalizadas, isto é, o usuário poderá escolher uma geração de relatórios de fluxo de caixa ou uma geração de metas financeiras.
- **RN05** – A exclusão de dados devem requerer confirmação. Todo dado cadastrado e registrado, ao ser excluído pelo usuário, pede uma confirmação explícita ao mesmo, garantindo segurança no tratamento de ações realizadas de forma acidental.
- **RN06** – A coleta e tratamento de dados deve estar em conformidade com a Lei Geral de Proteção de Dados, incluindo o consentimento do usuário e a segurança da informação.

## 3.3. Ferramentas, Processo de Desenvolvimento e Tecnologias Utilizadas

Para o desenvolvimento do Finance Vision, planejou-se adotar uma abordagem incremental baseada na metodologia ágil *Scrum*, visando maior organização e flexibilidade durante as etapas do projeto. O gerenciamento de tarefas foi realizado por meio do *Trello*, enquanto para o controle de versão foi utilizado o *Git*, com repositório hospedado no *GitHub*. A implementação da aplicação foi desenvolvida em PHP, utilizando o *framework* Laravel para o *back-end*, aliado a tecnologias como HTML, CSS e JavaScript no *front-end*. O SGBD relacional MySQL foi utilizado para armazenar e estruturar as informações. A escolha desse conjunto de ferramentas e tecnologias visou garantir escalabilidade, facilidade de manutenção e eficiência no desenvolvimento do sistema.

### 3.3.1. PHP

O servidor *web* funciona como um bibliotecário da *Internet*: ele recebe os pedidos dos usuários, como acessar uma página e entrega os recursos solicitados. Para recursos simples, como imagens ou páginas HTML, ele apenas fornece o arquivo. Já para conteúdo dinâmico, o servidor precisa de uma ferramenta extra, como o PHP, para processar o código e gerar a resposta certa (Silva, 2025). Rápida, flexível e dinâmica, a linguagem de programação PHP foi desenvolvida especialmente para aplicações *web* e integrações de aplicações para agilizar no desenvolvimento de um sistema (PHP, 2025).

A escolha do PHP se deve por ser uma linguagem de programação robusta, muito utilizada no desenvolvimento de *sites*, com uma vasta quantidade de usuários. É uma linguagem de fácil aprendizagem e altamente compatível com servidores *web*, possibilitando flexibilidade para a criação de *Application Programming Interface* (Interface de Programação de Aplicações) e sistemas dinâmicos. Em complemento, é uma linguagem confiável, com mais de 30 anos no mercado, oferecendo segurança, integridade e estabilidade.

### 3.3.2. Laravel

O Laravel é um *framework* PHP baseado no padrão MVC (*Model-View-Controller*) e segue o paradigma de orientação a objetos. O padrão MVC surgiu na década de 1980 e se popularizou na criação de aplicações *web*. Nesse padrão, a estrutura da aplicação é dividida em três camadas (Ferreira, 2021):

- ***Model* (modelo):** responsável pelo gerenciamento dos dados e regras de negócio.
- ***View* (visão):** cuida da apresentação, ou seja, da interface com o usuário.
- ***Controller* (controlador):** serve como intermediário entre o modelo e a visão, recebendo as requisições do usuário, processando-as e decidindo qual modelo utilizar para retornar a resposta adequada à visão.

A pretensão pelo Laravel foi pelo fato de oferecer uma arquitetura nativa baseada no padrão MVC, capaz de promover a separação de responsabilidades no desenvolvimento do código, facilitando a manutenção e escalabilidade do sistema. Além disso, possui uma grande comunidade de colaboradores que incentivam no desenvolvimento, trocando experiências e dicas em relação a resoluções de problemas e possíveis dificuldades encontradas no processo de desenvolvimento.

### 3.3.3. MySQL

O MySQL é um Sistema Gerenciador de Banco de Dados Relacional (SGBDR) multitarefa e multiusuário. Além disso, ele é *open source* (código aberto), significa que uma pessoa pode alterar seu código-fonte, adaptando-o às suas necessidades, seguindo regras descritas em sua GNU (Licença Pública Geral). O MySQL faz uso da arquitetura cliente/servidor, foi desenvolvido com o uso da linguagem de programação C e sofre atualizações regularmente (Tonsig, 2006).

O MySQL utiliza a linguagem *Structured Query Language* (SQL) para gerenciar e consultar dados em bancos de dados relacionais. Ela é altamente compatível com diversas plataformas, incluindo sistemas operacionais como Linux, Windows e macOS, além de suportar várias linguagens de programação, como PHP, Python, Java, C e C++ (MySQL, 2005).

O MySQL foi escolhido por ser um sistema simples de utilizar, tendo como vantagem a sua performance, confiabilidade e fácil usabilidade, é *open source* (código aberto), gratuito e se integra nativamente com o PHP e o Laravel, simplificando o desenvolvimento do *back-end* e garantindo uma persistência dos dados otimizada e eficiente.

### 3.3.4. *Scrum*

O *Scrum* é um *framework* de gerenciamento de projetos baseado na metodologia ágil que auxilia pessoas e organizações a elaborarem e gerenciarem o trabalho por meio de um conjunto de práticas, princípios e valores, incentivando as equipes de trabalho a aprenderem de forma empírica, isto é, através das experiências adquiridas. De forma prática, o *Scrum* define um conjunto de ferramentas, reuniões e papéis que ocorrem juntos para que as equipes envolvidas se organizem nas resoluções de problemas, proporcionando reflexões sobre êxitos e fracassos, promovendo uma melhoria contínua (Atlassian, 2025).

No contexto de desenvolvimento de *software*, o *Scrum* define três principais papéis fundamentais para o sucesso do projeto (Machado e Medina, 2009):

- ***Product Owner*:** responsável por maximizar o valor do produto, possui conhecimento das regras de negócio e atua como porta-voz dos clientes.
- ***Scrum Master*:** responsável por assegurar o funcionamento adequado da metodologia *Scrum*, garantindo que a equipe trabalhe com eficácia e produtividade. Deve possuir amplo conhecimento sobre os processos e práticas do *Scrum*.
- ***Scrum Team* (equipe de trabalho):** responsável por transformar as tarefas priorizadas em incrementos funcionais do *software*, prontos para entrega ao final de cada iteração.

O uso do *Scrum* foi aderido por incorporar um *framework* ágil de gerenciamento de projetos, sendo eficiente na organização das equipes e tarefas atribuídas. Ele permite que os projetos sejam desenvolvidos de maneira iterativa e incremental, facilitando a adaptação às mudanças de requisitos e situações no decorrer das implementações de um sistema.

### 3.3.5. *Trello*

O *Trello* é uma ferramenta de gerenciamento de projetos que possibilita a organização através de um fluxo de quadros, utilizando uma lista em um sistema de cartas (Kaur, 2018).

Ele é baseado na metodologia *Kanban*, que possibilita a exibição de todas as tarefas pertencentes ao projeto em um único painel visual, acessível a toda a equipe envolvida. Com o auxílio do *Trello*, os usuários podem organizar seus projetos em quadros, listas e subdividir essas listas em cartões de tarefas. Possui uma interface intuitiva e de fácil usabilidade, se adaptando a projetos de várias áreas do conhecimento e de diferentes situações (Johnson, 2017).

A escolha do *Trello* se deve por conta de sua interface intuitiva e de fácil organização, permitindo um melhor acompanhamento das equipes, tarefas submetidas e o progresso dos projetos. Sua acessibilidade via *web* e dispositivos móveis, além das diversas integrações com outras ferramentas, como *Google Drive* e *GitHub*, tornam o *Trello* uma solução prática e eficiente para o desenvolvimento de *software*.

### 3.3.6. *Git* e *GitHub*

O *Git* é um sistema de controle de versionamento distribuído que permite uma averiguação do processo de um projeto ao decorrer do tempo. Ele salva todas as versões de mudanças que ocorreram, possibilitando a verificação de quais mudanças houve, quem as fez e até mesmo regressar a uma versão anterior, se necessário (Astigarraga e Cruz-Alonso, 2022).

Já o *GitHub* é uma plataforma *online* baseada em nuvem, no qual é possível compartilhar, armazenar e trabalhar com outras pessoas de maneira simultânea em diferentes projetos. Sua utilização é ideal para a construção de um *software*, funciona como uma

rede social de código aberto, os usuários têm acesso a projetos do mundo todo e podem contribuir com a execução dos mesmos (GitHub, 2025).

A utilização do *Git* e *GitHub* como ferramenta de versionamento de código decorre de sua flexibilidade para alterar qualquer versão, caso seja necessário, além de permitir o desenvolvimento de aplicações com mais de uma pessoa de maneira simultânea e colaborativa.

### 3.3.7. *Figma*

O *Figma* é uma plataforma *web* voltada ao *design* de interfaces e à prototipação de sistemas interativos. Por ser uma ferramenta baseada em nuvem, permite que múltiplos usuários colaborem em tempo real na criação de telas, fluxos de navegação e elementos visuais, otimizando a comunicação entre *designers* e desenvolvedores. O *Figma* permite validar a organização visual, a experiência do usuário e os caminhos funcionais antes da implementação definitiva do sistema (Figma, 2016).

O *Figma* foi escolhido para realizar a prototipação do projeto por possuir uma interface moderna e de fácil usabilidade. O *Figma* possui muitas vantagens em relação à imensa quantidade de *plug-ins* que podem se integrar a ele, bem como recursos avançados de prototipagem, criação de componentes reutilizáveis e entrega de especificações para desenvolvedores, agilizando o processo de *design* e desenvolvimento.

## 3.4. Modelagem da Solução

A modelagem do Finance Vision foi elaborada com o objetivo de representar, de forma visual e estruturada, os principais aspectos funcionais, comportamentais e de navegação da aplicação. A construção desses modelos permitiu antecipar o entendimento da lógica do sistema e facilitar a comunicação entre as etapas de projeto e desenvolvimento.

Esta seção apresenta os principais elementos de modelagem e concepção do sistema, abrangendo desde o planejamento visual e definição das interações dos usuários até a estruturação técnica e lógica da aplicação. São descritos de forma integrada os aspectos relacionados à interface, arquitetura, comunicação entre componentes e organização dos dados, fornecendo uma visão geral do processo de desenvolvimento e das bases que sustentam a solução proposta.

### 3.4.1. Protótipo da Interface da Ferramenta

A Figura 4 apresenta o fluxo de navegação, ilustrando a estrutura geral das principais telas e seus caminhos de acesso. Após a autenticação do usuário por meio da tela de *login* ou cadastro, o sistema direciona para o *dashboard*, que atua como o ponto central de acesso às demais funcionalidades. A partir do *dashboard*, o usuário pode navegar para os módulos de lançamentos, metas financeiras, relatórios, dicas financeiras e configurações. Também há a possibilidade de recuperar a senha a partir da tela inicial. Esse modelo de navegação prioriza a simplicidade, a fluidez entre as seções e a usabilidade, oferecendo um ambiente organizado e acessível ao público jovem.

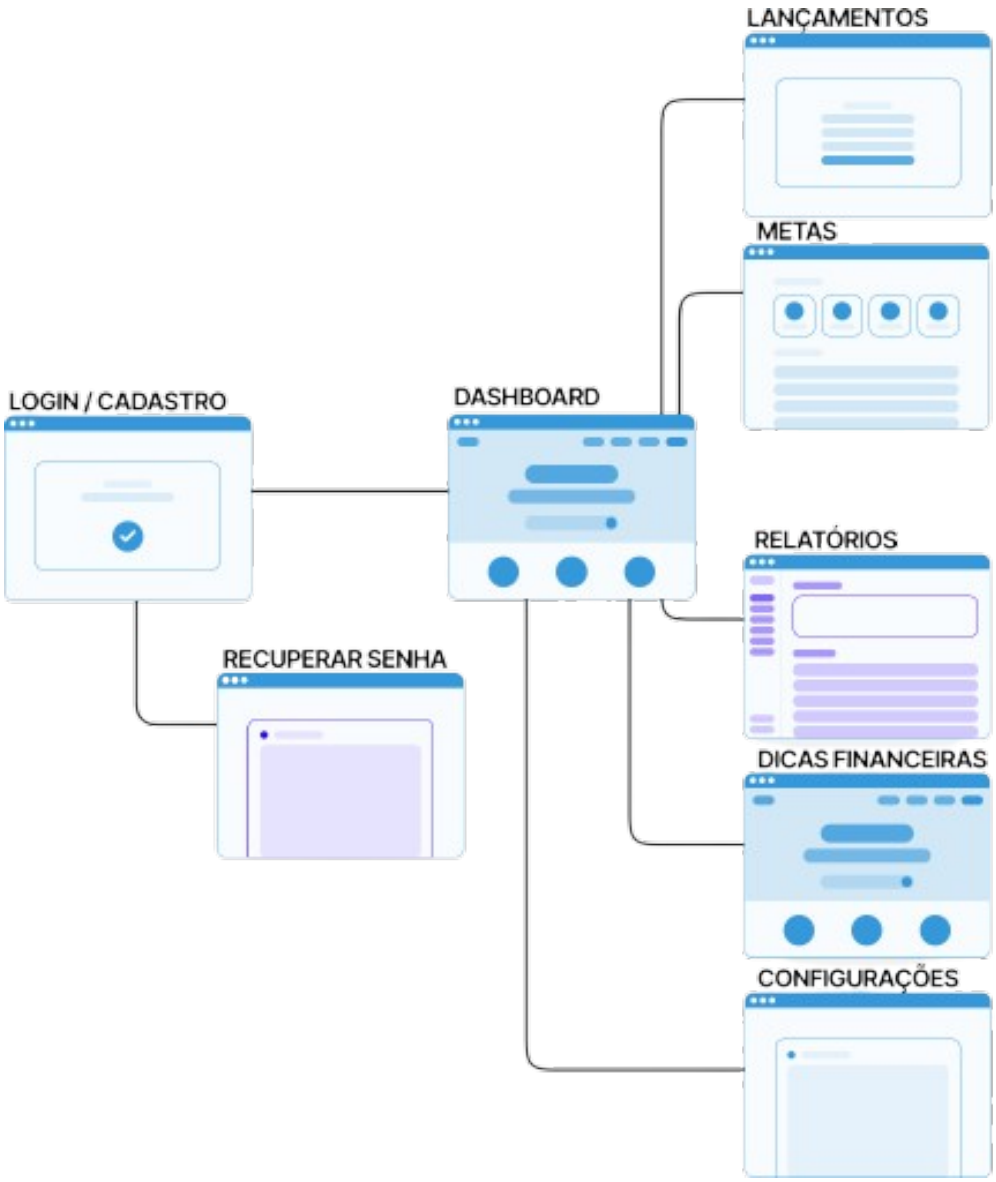



**Figura 4. Fluxo de navegação. Disponível em: shre.ink/xJyE.**

O protótipo desenvolvido permitiu simular cenários de uso real, validar a usabilidade da aplicação e realizar ajustes na interface antes da implementação final. A Figura 5 apresenta a tela inicial do sistema proposto, acessada após o *login*. Nessa interface, o usuário tem acesso rápido a *cards* interativos e gráficos que representam informações essenciais, como o saldo atual, evolução de receitas e despesas. Usuários têm acesso rápido às principais funcionalidades, como lançamentos financeiros, metas, relatórios, dicas financeiras e configurações do sistema. A interface foi projetada para ser clara, objetiva e funcional, deixando de fácil acesso a navegação e oferecendo uma visão geral do progresso financeiro do usuário. O *layout* foi projetado para ser claro, funcional e intuitivo, centralizando os caminhos de navegação e oferecendo uma visão geral do comportamento financeiro do usuário de forma visual e acessível.

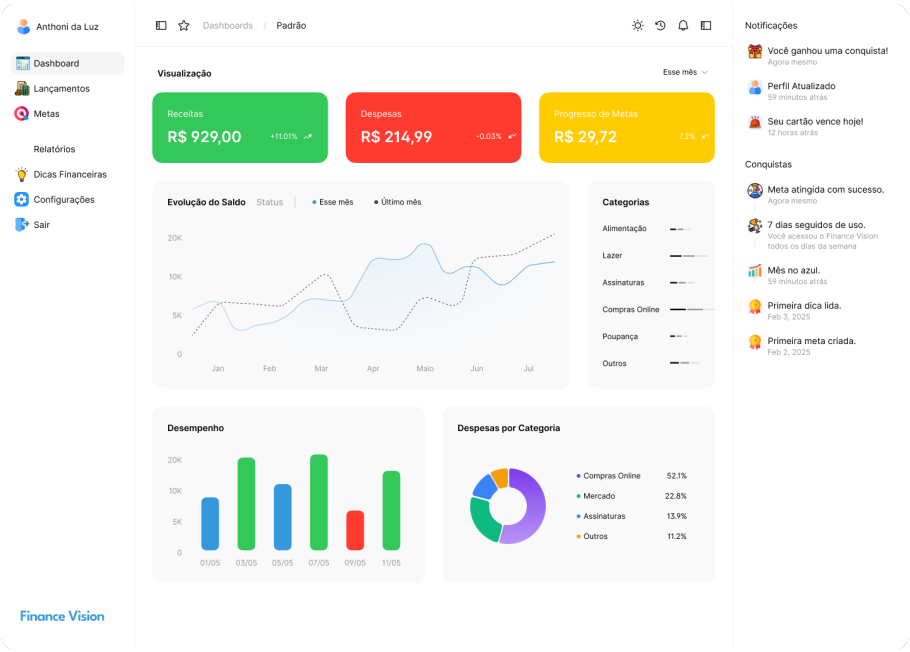



**Figura 5. Tela de *Dashboard*. Disponível em: shre.ink/xJZ9.**

A Figura 29, descrita no Apêndice 1, apresenta a tela de Lançamentos, onde os usuários podem registrar manualmente ou de forma automática suas receitas e despesas de forma prática e organizada. O formulário permite cadastrar informações como nome, valor, data, categoria da transação, vinculação a metas previamente cadastradas e descrição opcional. A interface também disponibiliza um quadro por tipo de movimentação e exibe os lançamentos mais recentes em ordem cronológica, facilitando o acompanhamento contínuo do histórico financeiro. Além disso, a tela oferece a funcionalidade de criação de relatórios, otimizando o processo de inserção de dados para usuários que desejam integrar suas finanças automaticamente ao sistema.

A Figura 30, descrita no Apêndice 1, apresenta a tela de Relatórios, responsável por consolidar e visualizar, de forma simples e eficaz, os dados financeiros. O usuário pode aplicar filtros por intervalo de data, adaptando a visualização de acordo com suas preferências ou necessidades de análise. Também é possível selecionar qual relatório será emitido, fluxo de caixa ou acompanhamento de metas financeiras. Essa tela tem como objetivo transformar os dados inseridos em informações acessíveis e interpretáveis, contribuindo para a análise dos hábitos de consumo, a identificação de tendências ao longo do tempo e a tomada de decisões mais conscientes e estratégicas.

A Figura 31, descrita no Apêndice 1, apresenta a tela de Metas Financeiras, onde o usuário pode visualizar, editar e acompanhar seus objetivos econômicos. Cada meta é exibida em um cartão com título, *status*, valores (objetivo e poupado), prazo e categoria, além de um gráfico circular que indica o progresso da meta de forma visual. A interface também oferece filtros por prazo e *status*, e um botão para adicionar novas metas. O *layout* busca promover o planejamento financeiro pessoal, tornando os objetivos mais claros, mensuráveis e acessíveis.

### 3.4.2. Diagrama de Casos de Uso

O Diagrama de Casos de Uso representa, de forma abstrata, as principais interações entre o usuário e as funcionalidades do Finance Vision, descrevendo os atores envolvidos e os casos de uso correspondentes, como realizar *login*, gerenciar lançamentos financeiros, definir metas, visualizar relatórios e acessar dicas educativas. Esse diagrama, descrito na Figura 6, contribui para a compreensão dos requisitos funcionais do sistema, servindo como base para o planejamento e a validação das funcionalidades esperadas na fase de desenvolvimento.

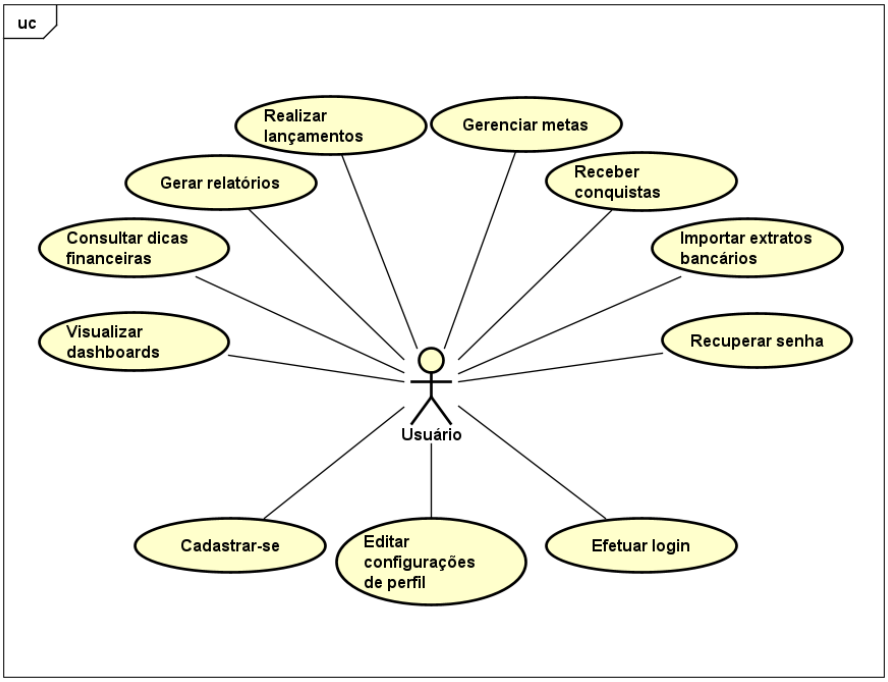



**Figura 6. Diagrama de Casos de Uso.**

### 3.4.3. Arquitetura do Sistema

A arquitetura adotada no desenvolvimento do Finance Vision segue o padrão MVC (*Model-View-Controller*), comum em aplicações *web* modernas por sua organização clara de responsabilidades. Como é possível observar na Figura 7, as interações do usuário iniciam-se pela camada de visualização (*view*), responsável por exibir as interfaces da aplicação. As requisições são tratadas inicialmente pelo *middleware*, que atua como filtro e é responsável por gerenciar autenticação e outras verificações antes de direcionar o fluxo. Em seguida, as *routes* definem para qual *controller* a requisição será encaminhada. O *controller*, por sua vez, processa a lógica da aplicação, interagindo com a camada de dados (*model*) para realizar operações como inserção, consulta e atualização no banco de dados. Após o tratamento, os dados retornam à camada de visualização, concluindo o ciclo de resposta ao usuário.

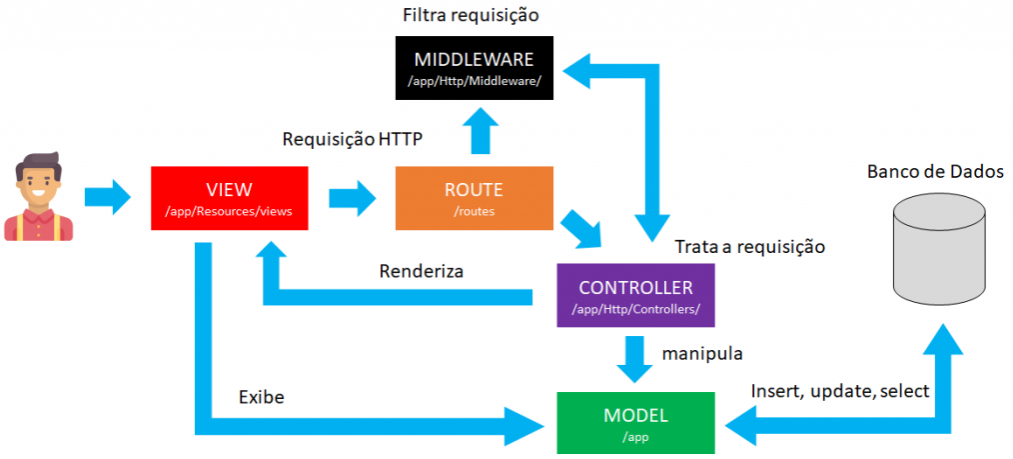



**Figura 7. Arquitetura do sistema. Documentação Laravel (2024).**

### 3.4.4. API REST

A API REST (*Representational State Transfer*) é um estilo arquitetural definido por Fielding (2000), que baseia-se em princípios como comunicação *stateless*, interface uniforme, identificação de recursos via URIs e uso de métodos HTTP padrão (GET, POST, PUT, DELETE) para manipular dados representados em formatos como JSON ou XML. Estudos com especialistas indicam que a adoção de boas práticas REST, melhora atributos como manutenibilidade, usabilidade e compatibilidade da API. A arquitetura REST é amplamente adotada por ser leve, escalável, independente de plataforma e altamente compatível com microserviços e aplicações móveis, permitindo integrações eficientes e comunicação clara entre componentes (Kotstein e Bogner, 2021).

Nesse projeto, a API REST foi empregada principalmente para organizar de forma clara e modular todas as funcionalidades do sistema, permitindo que cada recurso, como lançamentos, metas, categorias e conquistas, possua rotas próprias e bem definidas. Essa estrutura facilita a comunicação interna da aplicação, garantindo que o *front-end* acesse os dados de maneira consistente e previsível.

### 3.4.5. Diagrama de Entidade-Relacionamento

O Diagrama de Entidade-Relacionamento (DER) apresenta a estrutura lógica do banco de dados, evidenciando as principais entidades, seus atributos e os relacionamentos entre elas. Entre as entidades modeladas estão *users*, *lancamentos*, *metas*, *categorias*, *sessions* e *password_reset_token*, organizadas de forma a refletir a estrutura real de armazenamento e manipulação de dados no sistema. Esse modelo permite visualizar as conexões lógicas entre os dados, além de apoiar o desenvolvimento da camada de persistência e garantir a integridade referencial das informações. O DER foi elaborado com o auxílio da ferramenta *MySQL Workbench*, amplamente utilizada para modelagem de banco de dados relacionais.

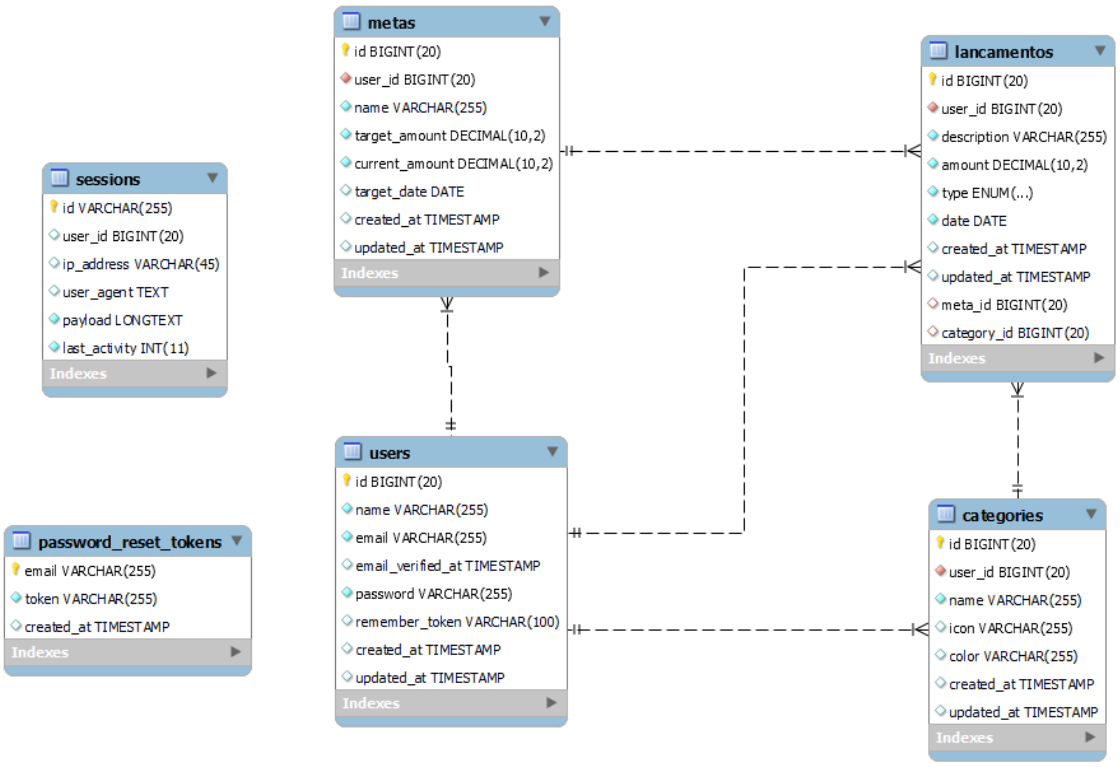



**Figura 8. Modelo lógico do banco de dados no *MySQL Workbench*.**

## 4. Desenvolvimento

A plataforma do Finance Vision foi desenvolvida com a linguagem PHP juntamente do *framework* Laravel, a estruturação e estilização das páginas com HTML e CSS. Para converter o protótipo criado no *Figma*, renderizar e criar *views* de maneira dinâmica, também foi utilizado a ferramenta Blade, nativa do Laravel, que serve para compilar *templates* com códigos PHP. O banco de dados foi elaborado com o Sistema Gerenciador de Banco de Dados MySQL conforme a Figura 8.

A seção de desenvolvimento foi estruturada em módulos: o Módulo de Autenticação faz o cadastro e o *login* dos usuários, verificando a identidade de cada um que deseja efetuar a entrada e validando a navegabilidade com o auxílio do *middleware Auth* do Laravel. O Módulo de *Dashboards* e Visualização exibe o balanço financeiro geral de forma intuitiva, em formato de gráficos. O Módulo de Lançamentos realiza o cadastro das despesas e receitas do usuário. O Módulo de Metas Financeiras possibilita o registro de metas a serem atingidas, além de atingir a meta, o usuário receberá uma conquista em forma de elo. O Módulo de Relatórios permite a criação de relatórios contendo os gastos e as receitas. O Módulo de Dicas Financeiras traz conteúdos sobre investimento, consumo, orçamento e metas de forma teórica. O Módulo de Configurações permite que o usuário gerencie algumas notificações do sistema e sua senha.

Foi utilizada uma metodologia adaptativa de desenvolvimento do *Scrum*, com ambos os orientandos se envolvendo em todos os papéis descritos na metodologia. A organização e o andamento do projeto foram organizados em *Sprints*, com entregas re-

alizadas semanalmente, bem como o registro e acompanhamento das fases do projeto através de quadros no Trello conforme a Figura 28, disponível no Apêndice 1. Para assegurar o desenvolvimento síncrono e o versionamento da plataforma, foi utilizado o *Git* e o *GitHub*.

A Figura 9, apresenta a listagem de rotas utilizadas no sistema, que evidenciam a arquitetura API REST implementada. As principais rotas seguem o padrão CRUD (*Create, Read, Update e Delete*) e estão organizadas por recursos como categorias, lançamentos, metas, relatórios e perfil do usuário. Cada módulo possui rotas específicas para exibir, criar, editar, atualizar e excluir registros, sendo gerenciadas por controladores dedicados. Além disso, destacam-se as rotas relacionadas à importação automatizada de extratos que integram o recurso de Inteligência Artificial para leitura de arquivos bancários. Também estão presentes rotas voltadas à autenticação e segurança, incluindo *login*, registro, redefinição de senha e verificação de *e-mail*, assegurando o controle de acesso dos usuários.

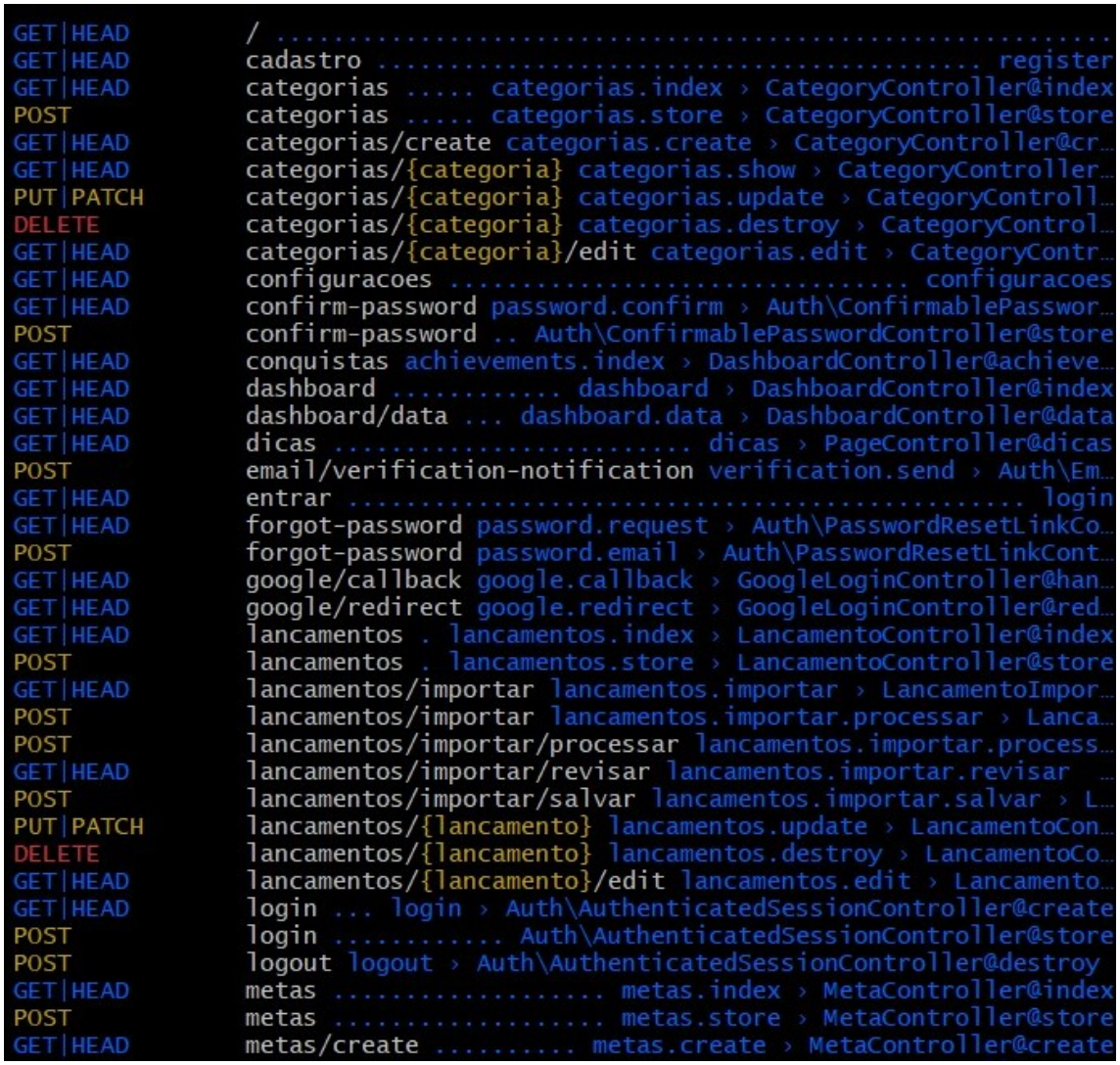



**Figura 9. Lista de rotas seguindo a arquitetura API REST.**

## 4.1. Módulo de Autenticação

Ao acessar a plataforma *web* do Finance Vision, os usuários são direcionados para a Tela de *Login*, apresentada na Figura 32, disponível no Apêndice 1, onde podem efetuar a entrada no sistema inserindo suas credenciais (*e-mail* e senha), assegurando a identidade única de cada usuário. Em complemento, permite a recuperação da senha, caso o usuário a esqueça e deseje recuperá-la, e também é possível realizar o *login* de forma integrada com uma conta Google.

Caso o usuário esteja entrando no sistema pela primeira vez, ao clicar em "crie uma conta gratuitamente", ele é direcionado à Tela de Cadastro de Contas, Figura 33, disponível no Apêndice 1, podendo, assim, realizar a criação de sua conta digitando seu nome completo, *e-mail* e senha com confirmação.

Na Figura 34, apresentada no Apêndice 1, observa-se um trecho do *controller* de usuário, responsável pelo processo de cadastro e autenticação de novos usuários no sistema Finance Vision. O código define o *RegisteredUserController*, que possui o método *create*() para retornar a *view* de registro e o método *store*() para tratar o envio do formulário, validando os dados de entrada (nome, *e-mail* e senha) conforme regras estabelecidas pelo *framework*. Após a validação, um novo usuário é criado no banco de dados com a senha devidamente criptografada utilizando o *Hash*, sendo em seguida disparado o evento *Registered* e realizada a autenticação automática do usuário com o método *Auth::login*(). Por fim, ocorre o redirecionamento para a rota do painel principal (*dashboard*), garantindo uma experiência fluida no fluxo de registro.

No *model* de usuários, Figura 35, conforme apresentado no Apêndice 1, a classe *User* centraliza as relações do sistema, conectando usuários a lançamentos, metas e categorias por meio dos métodos *hasMany*. Além disso, define os atributos *fillable*, *hidden* e *casts*, garantindo segurança na manipulação de dados sensíveis como senha e *token* de autenticação. Essa estrutura oferece robustez ao modelo principal da aplicação, integrando todas as funcionalidades do sistema.

Na *migration* de usuários, Figura 36, conforme apresentado no Apêndice 1, além da criação da tabela principal *users* com campos como *name*, *email*, *password* e *email_verified_at*, foram criadas tabelas auxiliares para redefinição de senha *password_reset_tokens* e para controle de sessões (*sessions*). Essa estrutura segue o padrão do Laravel para autenticação e gerenciamento seguro de acessos, garantindo integridade e escalabilidade no sistema de *login* e persistência de sessão.

## 4.2. Módulo de *Dashboards* e Visualização

Após a realização do *login*, o usuário acessa a Tela de *Dashboards*, Figura 10. Esta é a principal tela do sistema, contendo um menu com todas as telas à esquerda para possibilitar a navegabilidade dos usuários em todas as outras telas da plataforma. Além disso, possui uma visualização objetiva e intuitiva do controle financeiro dos usuários, destacando as receitas, despesas e saldo mensal, além de possibilitar análises visuais em forma de gráfico da evolução financeira e das despesas por categoria. Também contempla um acompanhamento das metas pré-cadastradas pelo usuário e as conquistas alcançadas pelo mesmo, trazendo uma estratégia de gamificação para engajar no uso da plataforma.

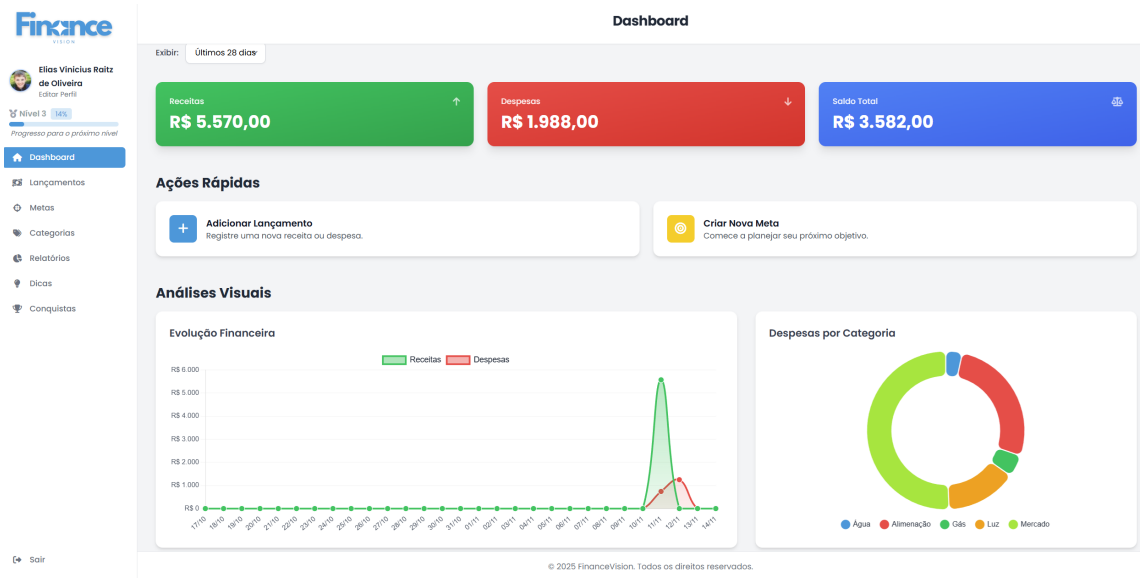



**Figura 10. Tela de *Dashboards*.**

## 4.3. Módulo de Lançamentos

Ao clicar em *Lançamentos* no menu ou no acesso rápido de adicionar lançamentos da Tela de *Dashboards*, Figura 11, é possível realizar o cadastro dos mesmos, selecionando o tipo (se é uma receita ou uma despesa), e preenchendo os campos de descrição, valor, data e categoria. Também há como vincular a uma meta cadastrada, como guardar dinheiro para uma viagem. Na parte inferior da tela é realizada a exibição de um histórico de lançamentos feitos anteriormente.

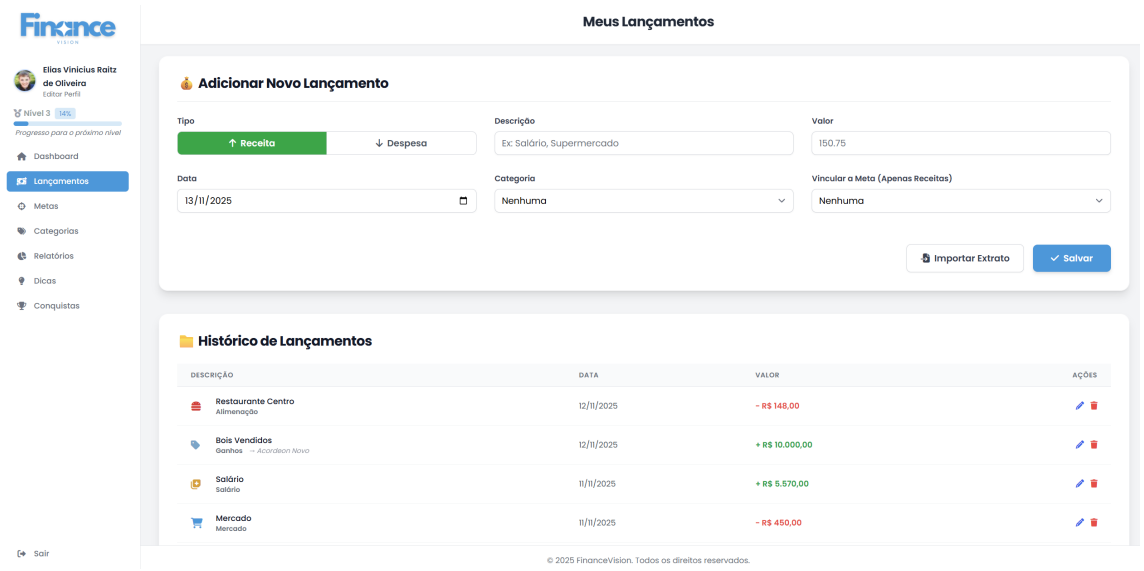



**Figura 11. Tela de Lançamentos.**

O *controller* de lançamentos financeiros, Figura 12, gerencia a criação e listagem

de lançamentos (receitas e despesas). A função *index()* retorna todos os lançamentos do usuário autenticado, relacionados às categorias e metas, além de retornar também as listas de metas e categorias para uso na *view*. No método *store()*, ocorre a validação dos dados do formulário, seguida da criação do lançamento. Caso o lançamento esteja vinculado a uma meta, o sistema busca a meta correspondente e atualiza seu progresso por meio de uma função específica. Essa implementação garante integridade relacional e encapsula regras de negócio dentro do próprio *controller*.

```php
class LancamentoController extends Controller
{

    public function index()
    {
        $lancamentos = Auth::user()->lancamentos()->with(['category', 'meta'])->latest('date')->get();
        $metas = Auth::user()->metas()->get();
        $categories = Auth::user()->categories()->get();

        return view('lancamentos.index', [
            'lancamentos' => $lancamentos,
            'metas' => $metas,
            'categories' => $categories,
        ]);
    }

    public function store(Request $request)
    {
        $validated = $request->validate([
            'description' => 'required|string|max:255',
            'amount' => 'required|numeric|min:0.01',
            'type' => 'required|in:receita,despesa',
            'meta_id' => 'nullable|exists:metas,id',
            'category_id' => 'nullable|exists:categories,id',
            'date' => 'required|date',
        ]);

        DB::transaction(function () use ($validated) {
            $lancamento = Auth::user()->lancamentos()->create($validated);
            if (isset($validated['meta_id'])) {
                $meta = Auth::user()->metas()->findOrFail($validated['meta_id']);
                $this->updateMetaProgress($meta, $lancamento->amount, $lancamento->type);
            }
        });

        return redirect()->route('lancamentos.index')->with('success', 'Lançamento adicionado com sucesso!');
    }
```

**Figura 12. *Controller* dos lançamentos.**

No *model* de lançamentos, Figura 13, a classe *Lancamento* define os atributos *description*, *amount*, *type*, *date*, *meta_id* e *category_id*. Ela também estabelece relacionamentos com *Meta* e *Category*, permitindo vincular cada lançamento tanto a uma meta financeira quanto a uma categoria específica. Essa modelagem garante flexibilidade na análise de dados, possibilitando agrupar os registros em diferentes perspectivas.

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Lancamento extends Model
{
    use HasFactory;

    protected $fillable = [
        'description',
        'amount',
        'type',
        'date',
        'meta_id',
        'category_id',
    ];

    public function meta()
    {
        return $this->belongsTo(Meta::class);
    }

    public function category()
    {
        return $this->belongsTo(Category::class);
    }
}
```

**Figura 13. *Model* dos lançamentos.**

A *migration* de lançamentos, Figura 14, é responsável por armazenar os registros financeiros de um usuário no sistema. Nela, é definido um campo *id* como chave primária e um campo *user_id* como chave estrangeira, estabelecendo uma relação com a tabela de usuários, onde a exclusão em cascata remove automaticamente os lançamentos associados a um usuário excluído. Além disso, a tabela contém os campos *description* (descrição do lançamento), *amount* (valor decimal com duas casas decimais), *type* (definido como um *enum* que pode ser receita ou despesa) e *date* (data do lançamento), além dos campos automáticos *created_at* e *updated_at* adicionados pelo método *timestamps()*. O método *down()* garante a reversão da *migration*, removendo a tabela caso seja necessário.

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration {

    public function up(): void
    {
        Schema::create('lancamentos', function (Blueprint $table) {
            $table->id();
            $table->foreignId('user_id')->constrained()->cascadeOnDelete();
            $table->string('description');
            $table->decimal('amount', 10, 2);
            $table->enum('type', ['receita', 'despesa']);
            $table->date('date');
            $table->timestamps();
        });
    }

    public function down(): void
    {
        Schema::dropIfExists('lancamentos');
    }
};
```

**Figura 14. *Migrations* de lançamentos.**

Ainda na Tela de Lançamentos, Figura 15, ao clicar em Importar Extrato, o usuário tem acesso a tela Importar Lançamentos, nela foi implementada a funcionalidade automatizada de importação de extratos bancários em formato CSV com recursos de IA, ao analisar o arquivo selecionado, o sistema lê o extrato e preenche automaticamente os campos de lançamentos, reduzindo o processo manual do usuário ao cadastrar as receitas e despesas.


**Figura 15. Tela de Importação de Lançamentos.**

O controlador *LancamentoImportController* (Figura 16) é o componente central da funcionalidade de importação automatizada de extratos bancários, denominada "Importador Mágico". Essa funcionalidade foi desenvolvida para eliminar a entrada manual de dados, utilizando Inteligência Artificial (Google Gemini) para processar os arquivos enviados pelos usuários.

As *Large Language Models* (LLMs) são modelos de Inteligência Artificial treinados com grandes volumes de dados textuais, capazes de compreender e interpretar informações em linguagem natural. Esses modelos permitem a análise automatizada de documentos e a extração de dados relevantes de forma eficiente. Conforme destacado por Bommasani (2021), as LLMs ampliam as possibilidades de automação em diferentes contextos. Nesse cenário, o Google Gemini é uma LLM avançada que possibilita o processamento inteligente de arquivos, sendo utilizada no sistema para analisar extratos bancários, reduzir a necessidade de entrada manual de dados e aumentar a eficiência do processo de importação.

O fluxo de execução, representado na Figura 17, é dividido em quatro fases principais:

**Fase 1 (upload):** o usuário inicia o processo na Tela de Lançamentos e navega para a rota *lancamentos.importar*. O *LancamentoImportController* renderiza a *view importar.blade.php*, que permite o *upload* de um arquivo CSV. O controlador valida esse arquivo quanto ao tamanho máximo de 2048 KB, ou seja, 2 MB e formato.

**Fase 2 (processamento e IA):** o método *processar()* lê as primeiras 20 linhas do arquivo e obtém a lista de categorias do utilizador. Esses dados são formatados em um *prompt* personalizado, enviado via HTTP para a API do Google Gemini. Esse *prompt* instrui a IA a analisar os dados brutos e devolver uma resposta estruturada em JSON, identificando data, descrição, valor, tipo (receita/despesa) e sugerindo uma categoria.

**Fase 3 (armazenamento temporário):** após receber a resposta JSON da IA, o controlador trata e converte os dados. Para lidar com arquivos grandes e evitar erros de *timeout* ou limites de tamanho de sessão, os dados analisados não são armazenados na sessão. Em vez disso, são colocados no *cache* do Laravel, associados a uma chave única do utilizador.

**Fase 4 (revisão e persistência):** o utilizador é redirecionado para a rota *lancamentos.importar.revisar*, onde o método *revisar()* lê os dados do *cache* e os exibe na visão *revisar.blade.php*. Nessa tela, o utilizador confirma (ou corrige) as sugestões de lançamento quanto a data, descrição, categoria e valor, preenchidas automaticamente pela IA. Ao submeter, o método *salvar()* recebe o *array* de lançamentos validados, executa uma *DB::transaction()* para garantir a integridade dos dados e cria cada *Lancamento*. A cada criação, o evento *LancamentoCreated* é disparado, integrando a importação ao sistema de gamificação e atribuindo conquista e XP (*Experience Points*, pontos de experiência) ao usuário.

```php
class LancamentoImportController extends Controller
{

    public function index()
    {
        return view('lancamentos.importar');
    }

    public function processar(Request $request)
    {
        $request->validate([
            'extrato' => 'required|file|mimes:csv,txt,plain|max:2048',
        ]);

        $file = $request->file('extrato');
        $content = file_get_contents($file->getRealPath());
        $lines = explode("\n", $content);
        $csvSample = implode("\n", array_slice($lines, 0, 20));
        $userCategories = Auth::user()->categories()->pluck('name')->implode(', ');
        $prompt = $this->criarPrompt($csvSample, $userCategories);

        try {
            $apiKey = config('gemini.api_key');
            $endpoint = config('gemini.endpoint');

            $response = Http::timeout(60)->withHeaders([
                'Content-Type' => 'application/json',
            ])->post("{$endpoint}?key={$apiKey}", [
                        'contents' => [['parts' => [['text' => $prompt]]]]
                    ]);

            if (!$response->successful()) {
                $errorDetails = $response->json('error.message',
'Erro desconhecido da API.');
                return back()->with('error', "A API de IA falhou: " . $errorDetails);
            }

            $jsonResponse = $response->json();
            $iaRawText = $jsonResponse['candidates'][0]['content']['parts'][0]['text'
] ?? null;

            if (is_null($iaRawText)) {
                return back()->with('error',
'A IA devolveu uma resposta vazia. Verifique o seu ficheiro CSV.');
            }

            $jsonString = $this->limparRespostaJson($iaRawText);
            $parsedData = json_decode($jsonString, true);

            session()->flash('import_data', $parsedData['lancamentos'] ?? []);
            return redirect()->route('lancamentos.importar.revisar');

        } catch (\Exception $e) {
            return back()->with('error', 'Ocorreu um erro na ligação com a API: ' . $e
->getMessage());
        }
    }


    public function revisar()
    {
        $importData = session('import_data');

        if (empty($importData)) {
            return redirect()->route('lancamentos.importar')->with('error',
'Nenhuns dados para importar. Por favor, envie o ficheiro novamente.');
        }

        $userCategories = Auth::user()->categories()->orderBy('name')->get();

        return view('lancamentos.revisar', [
            'lancamentos' => $importData,
            'categories' => $userCategories
        ]);
    }

    public function salvar(Request $request)
    {
        $validated = $request->validate([
            'lancamentos' => 'required|array',
```

**Figura 16. *Controller* de importação de lançamentos.**

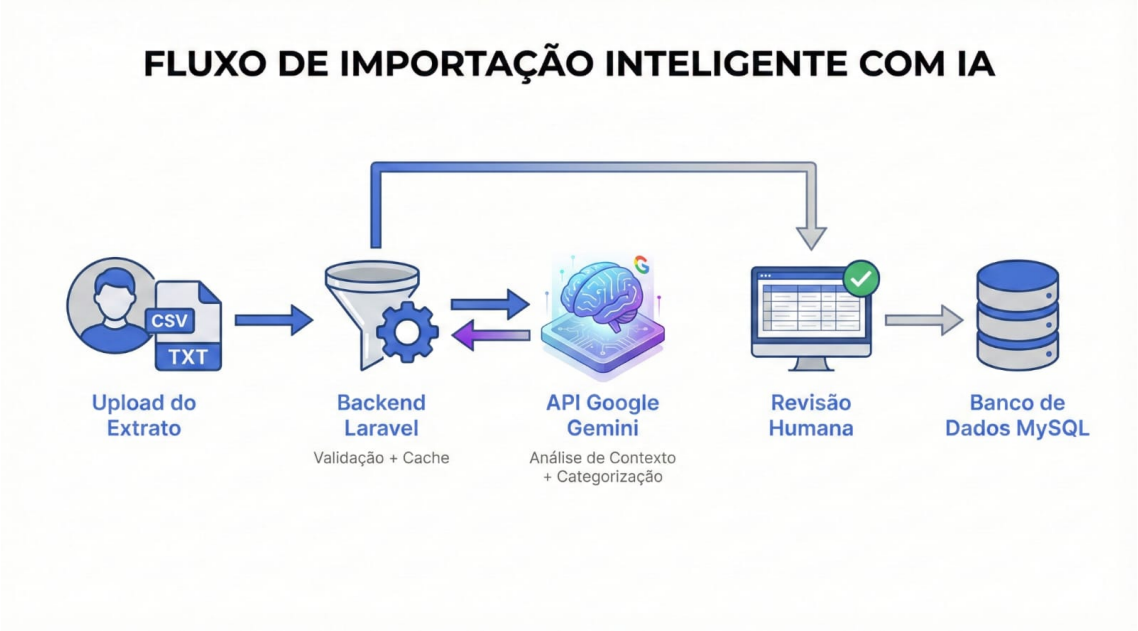



**Figura 17. Fluxo de importação de extrato utilizando Inteligência Artificial.**

Atrelada ao Módulo de Lançamentos, através do menu à esquerda, o usuário pode acessar a Tela de Categorias, Figura 18, nela é possível realizar a criação de categorias personalizáveis para posteriormente serem associadas a um lançamento. Ao criar uma categoria, é necessário dar um nome a categoria e também há a opção de selecionar uma cor e escolher um ícone entre os disponíveis para ela, visando uma melhor organização e visualização das receitas e despesas.

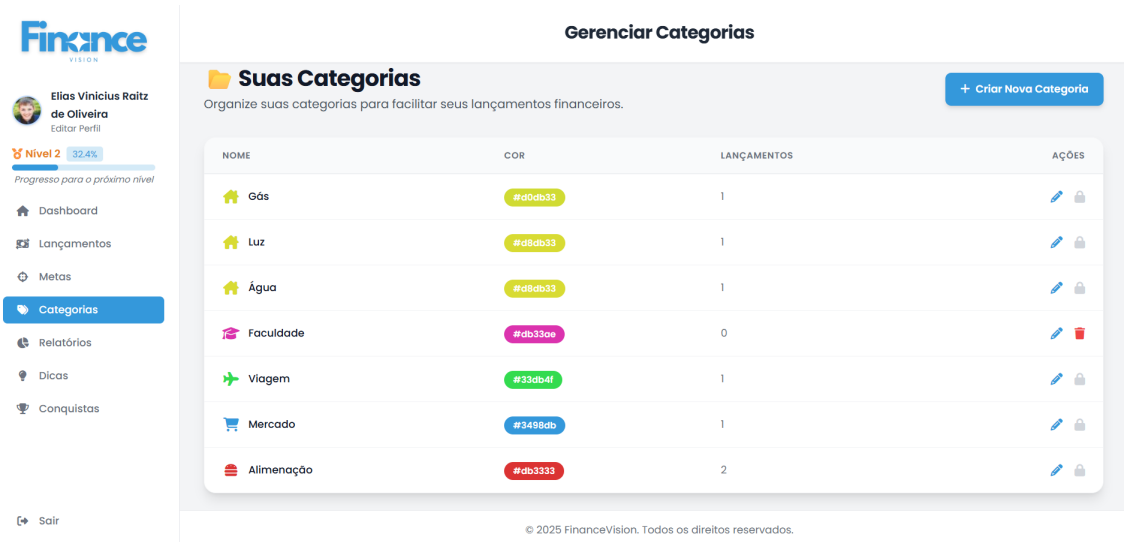



**Figura 18. Tela de Categorias.**

O *controller* de categorias, Figura 38, detalhado no Apêndice 1, é responsável por

permitir que cada usuário gerencie suas próprias categorias personalizadas para organizar seus lançamentos. O método *index()* recupera todas as categorias do usuário autenticado e as envia para a *view*. A função *store()* valida os campos de entrada como nome, ícone e cor antes de persistir os dados no banco. Já os métodos *edit()* e *update()* garantem que apenas o dono da categoria possa editá-la, promovendo segurança no acesso e manipulação dos dados. Toda a estrutura aproveita os recursos nativos do Laravel, como validação, autenticação e rotas nomeadas.

No *model* de categorias, Figura 39, detalhado no Apêndice 1, a classe *Category* define os atributos que podem ser preenchidos (*name*, *icon*, *color*, *user_id*) e estabelece os relacionamentos com usuário (*belongsTo*) e lançamentos (*hasMany*). Assim, cada categoria pertence a um usuário e pode agrupar múltiplos lançamentos financeiros, favorecendo a organização e classificação dos dados.

Na *migration* de categorias, Figura 40, detalhada no Apêndice 1, foi criada a tabela *categories* que armazena as informações de cada categoria vinculada a um usuário. A estrutura inclui os campos *id*, *user_id* (chave estrangeira relacionada à tabela de usuários), *name*, *icon* e *color*, além dos registros de data de criação e atualização. A relação de dependência entre usuário e categoria foi implementada com a cláusula *cascadeOnDelete*, garantindo que categorias associadas sejam removidas automaticamente caso o usuário seja excluído.

## 4.4. Módulo de Metas Financeiras

Acessando a Tela de Metas Financeiras, Figura 19, os usuários poderão ter um plano de finanças para atingir um objetivo futuro, como, por exemplo, comprar um veículo ou um imóvel. Ao clicar em “Adicionar nova meta”, abre uma nova tela para realizar o cadastro, o qual exige um nome para a meta, um valor alvo e a opção de selecionar uma data alvo. Na sequência, é possível visualizar o progresso da meta através de um gráfico de evolução. Além da possibilidade de editar ou excluir a meta, também há como filtrar a visualização por *status* ou tempo (recente ou antigo).

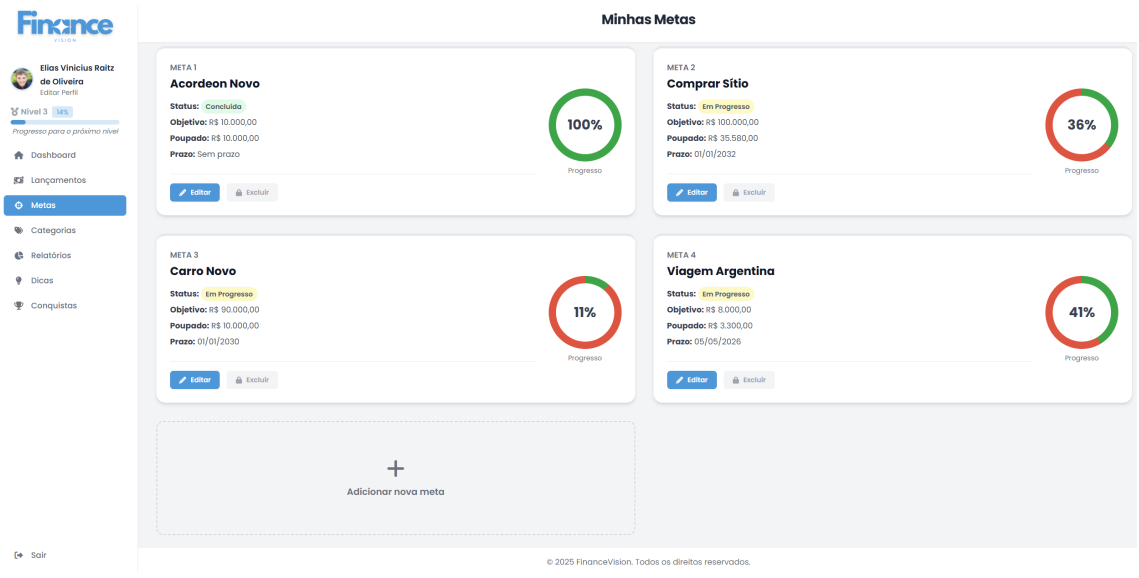



**Figura 19. Tela de Metas Financeiras.**

No trecho de código apresentado do *controller* de metas financeiras, Figura 20, foi implementado o controle das metas associadas ao usuário autenticado utilizando o *framework* Laravel. A função *index()* realiza a busca das metas filtrando por *status* e ordenação, além de carregar a contagem dos lançamentos relacionados. O método *create()* apenas exibe a *view* de criação, enquanto o método *store()* trata a validação dos dados e salva uma nova meta no banco de dados, associando-a ao usuário logado. Essa estrutura segue as boas práticas do Laravel, utilizando Eloquent ORM, autenticação via *Auth* e redirecionamento com mensagens de sucesso após as operações.

```php
<?php

namespace App\Http\Controllers;

use App\Models\Meta;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class MetaController extends Controller
{

    public function index(Request $request)
    {
        $statusFilter = $request->query('status');
        $sortOrder = $request->query('sort');

        $metas = Auth::user()->metas()
            ->withCount('lancamentos')
            ->ofStatus($statusFilter)
            ->sortBy($sortOrder)
            ->get();

        return view('metas.index', [
            'metas' => $metas,
            'selectedStatus' => $statusFilter,
            'selectedSort' => $sortOrder,
        ]);
    }

    public function create()
    {
        return view('metas.create');
    }

    public function store(Request $request)
    {
        $validated = $request->validate([
            'name' => 'required|string|max:255',
            'target_amount' => 'required|numeric|min:0.01',
            'target_date' => 'nullable|date',
        ]);

        Auth::user()->metas()->create($validated);

        return redirect()->route('metas.index')->with('success',
'Nova meta criada com sucesso!');
    }
```

**Figura 20. *Controller* das metas financeiras.**

No *model* de metas, Figura 21, a classe *Meta* define os atributos *name*, *target_amount* e *target_date*, além de relacionar metas a lançamentos (*hasMany*) e usuários (*belongsTo*). Foi implementado ainda o método *getProgressAttribute()*, que calcula dinamicamente a porcentagem de progresso de uma meta com base nos valores acumulados, e um escopo *scopeOfStatus()* para consultas filtradas, demonstrando um uso avançado de recursos do Eloquent ORM.

```php
class Meta extends Model
{
    use HasFactory;

    protected $fillable = [
        'name',
        'target_amount',
        'target_date',
    ];

    public function lancamentos()
    {
        return $this->hasMany(Lancamento::class);
    }

    protected $casts = [
        'target_amount' => 'float',
        'current_amount' => 'float',
        'target_date' => 'date',
    ];

    public function user()
    {
        return $this->belongsTo(User::class);
    }

    public function getProgressAttribute(): float
    {
        if ($this->target_amount == 0) {
            return 100.0;
        }
        $progress = ($this->current_amount / $this->target_amount) * 100;
        return round(min($progress, 100), 2);
    }

    public function scopeOfStatus(Builder $query, ?string $status): void
    {
        if ($status === 'completed') {
            $query->whereRaw('current_amount >= target_amount');
        } elseif ($status === 'progress') {
            $query->whereRaw('current_amount < target_amount')
                ->where(fn($q) => $q->where('target_date', '>=', now())->
orWhereNull('target_date'));
        } elseif ($status === 'overdue') {
            $query->whereRaw('current_amount < target_amount')
                ->where('target_date', '<', now());
        }
    }

    public function scopeSortBy(Builder $query, ?string $sort): void
    {
        if ($sort === 'newest') {
            $query->orderBy('created_at', 'desc');
        } elseif ($sort === 'closest_due_date') {
            $query->orderBy('target_date', 'asc');
        } else {
            $query->latest();
        }
    }
}
```

**Figura 21. *Model* das metas financeiras.**

Na *migration* de metas financeiras, Figura 37, detalhada no Apêndice 1, foi criada a tabela *metas* que possibilita ao usuário definir e acompanhar seus objetivos financeiros. Cada meta possui *id*, *user_id* associado, *name*, *target_amount* (valor alvo), *current_amount* (valor atual, inicializado em zero), e *target_date* (prazo estabelecido). Essa estrutura permite medir o progresso de forma gradual até o cumprimento da meta, possibilitando também a remoção em cascata caso o usuário seja deletado.

## 4.5. Módulo de Relatórios

O Módulo de Relatórios, Figura 22, contém uma das principais funcionalidades do sistema, que consiste na geração de relatórios, permitindo uma consulta mais rápida do controle financeiro pelo usuário. Ao gerar o relatório, escolhe-se o tipo e o período em que irá englobar (dia, mês, ano); nesta tela também será possível visualizar um gráfico de despesas por categoria, ajudando a identificar quais categorias detêm as maiores despesas para um melhor entendimento de seus gastos.

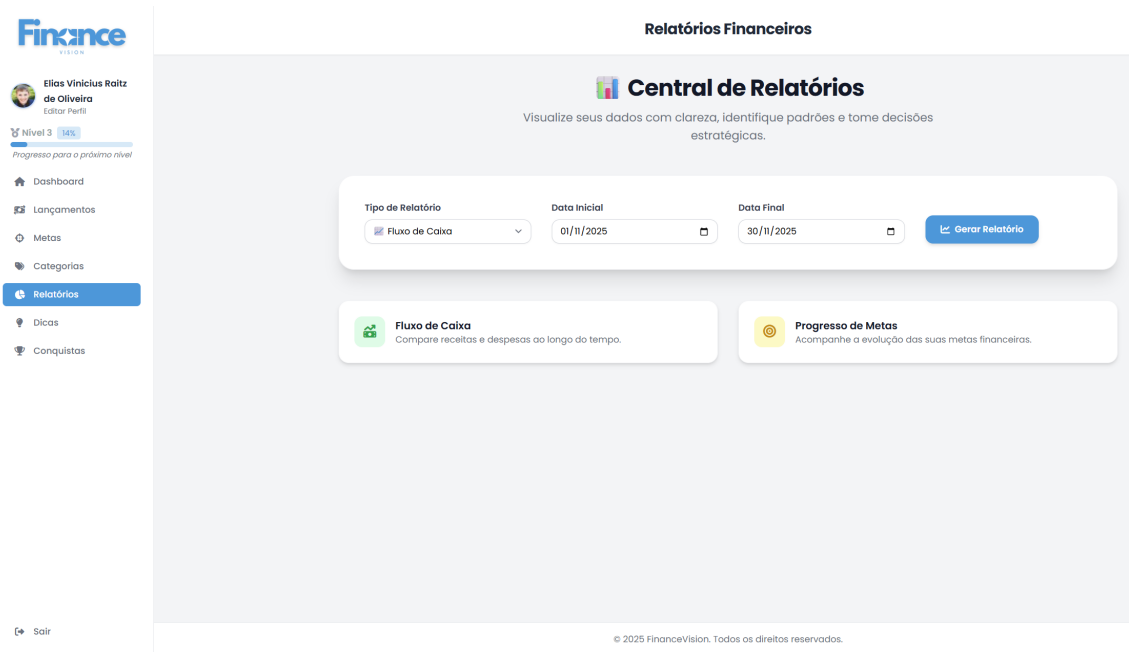



**Figura 22. Tela de Relatórios.**

## 4.6. Módulo de Educação Financeira

No módulo de Educação Financeira, foi desenvolvida a tela de dicas financeiras, Figura 23, que contém conteúdos sobre orçamento, metas, consumo e investimento. Ela serve como um guia, buscando enriquecer o conhecimento do usuário sobre o controle financeiro e, dessa forma, permitindo uma melhor decisão monetária a se fazer.

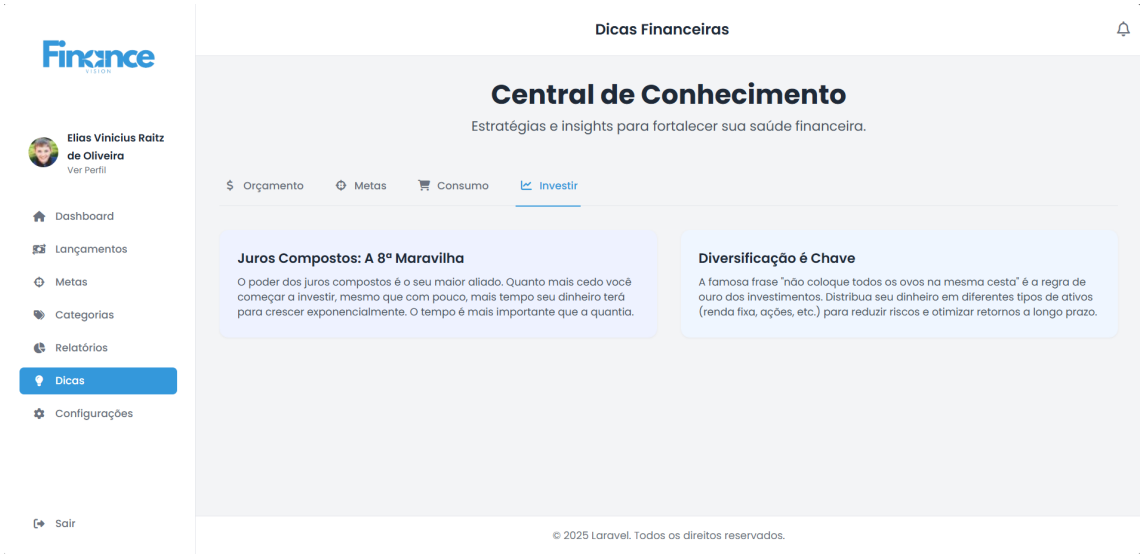



**Figura 23. Tela de Dicas Financeiras.**

## 4.7. Módulo de Gamificação

Na Figura 24, é apresentada a tela de conquistas do sistema. Essa interface permite que o usuário visualize seu progresso e desempenho financeiro por meio de um sistema de níveis e recompensas. As conquistas são exibidas em formato de cartões, contendo título, descrição, raridade e pontuação em XP. O objetivo dessa funcionalidade é estimular o engajamento e a continuidade no uso da plataforma, recompensando ações como o registro de lançamentos, a criação de metas financeiras e a geração de relatórios. O sistema utiliza a arquitetura *Event-Driven* (orientado a eventos), representada na Figura 25, promovendo assim, a motivação e o aprendizado contínuo do usuário, tornando a gestão financeira uma experiência mais interativa e atrativa.

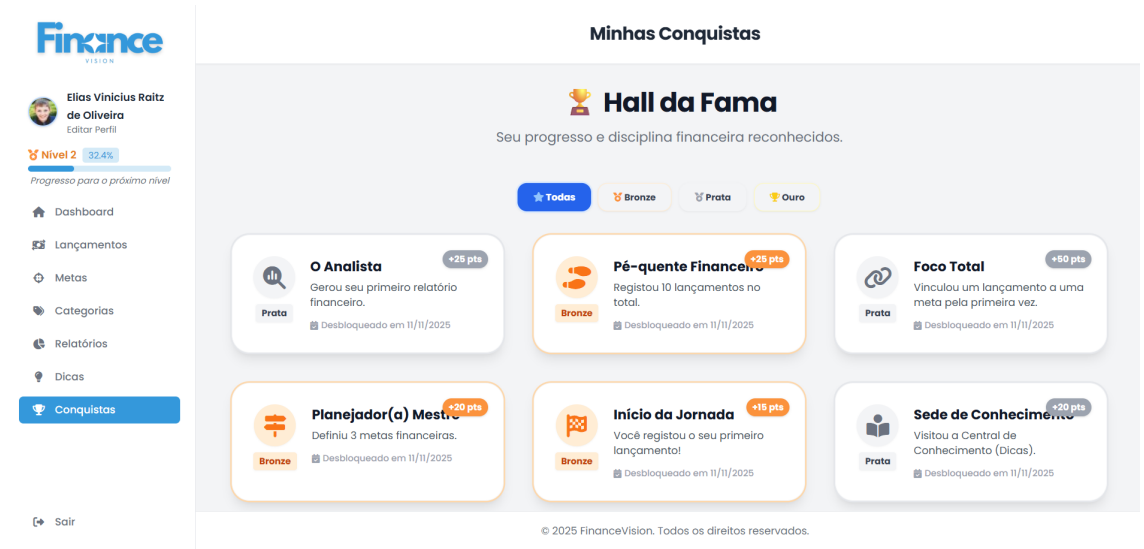



**Figura 24. Tela de Conquistas.**


**Figura 25. Arquitetura de gamificação do sistema.**

No Quadro 2, são apresentadas as relações de conquistas utilizadas como base lógica para o funcionamento da gamificação. Cada conquista possui uma raridade associada, um nome, uma descrição resumida e uma pontuação específica em XP. Esses parâmetros definem as condições de desbloqueio e o valor de recompensa atribuído a cada ação realizada dentro do sistema. Assim, ao atingir determinadas metas ou concluir tarefas específicas, o usuário acumula pontos e medalhas que refletem sua evolução financeira.

| Raridade | Conquista | Descrição resumida | Pontos (XP) |
|---|---|---|---|
| Bronze | Início da jornada | Primeiro lançamento registrado | 15 XP |
| Bronze | Organizador(a) | Criar 5 categorias | 40 XP |
| Bronze | Sonhador(a) | Criar 3 metas | 20 XP |
| Bronze | Pé-quente financeiro | 10 Lançamentos registrados | 25 XP |
| Prata | Foco total | Vinculou um lançamento a uma meta | 50 XP |
| Prata | Atividade constante | 20 lançamentos registrado | 75 XP |
| Prata | O analista | Gerou o primeiro relatório | 25 XP |
| Prata | Sede de conhecimento | Visitou a página de dicas | 20 XP |
| Ouro | Objetivo alcançado | Completou uma meta (100) | 150 XP |
| Ouro | Mês no verde | Terminou o mês com saldo positivo | 100 XP |

**Quadro 2. Quadro de conquistas.**

No Quadro 3, é exibida a tabela de níveis, responsável por determinar a progressão do usuário conforme o total de pontos acumulados. O sistema foi dividido em quatro níveis: Padrão, Bronze, Prata e Ouro. Cada um com uma pontuação mínima e uma cor representativa. Essa hierarquia permite que o usuário acompanhe sua evolução de

maneira clara e objetiva, servindo como incentivo para a continuidade das boas práticas financeiras.

| Nível | Pontos mínimos | Cor |
|---|---|---|
| 1 | 0 | Padrão |
| 2 | 100 | Bronze |
| 3 | 270 | Prata |
| 4 | 520 | Ouro |

**Quadro 3. Quadro de níveis.**

### 4.8. Módulo de Edição de Perfil

O Módulo de Edição de Perfil, Figura 41, apresentado no Apêndice 1, permite aos usuários realizarem o gerenciamento de suas contas, podendo alterar seu nome, *e-mail* e também a senha. Além disso, há a possibilidade de exclusão de conta, caso o usuário não queira continuar usando o sistema.

### 4.9. Política de Privacidade

A política de privacidade apresentada pelo sistema, Figura 42, apresentado no Apêndice 1, estabelece de forma clara como os dados pessoais dos usuários são coletados, utilizados, compartilhados e protegidos dentro da plataforma, em conformidade com os princípios da Lei Geral de Proteção de Dados (LGPD). São coletados apenas dados essenciais para o funcionamento do sistema, como informações de conta, lançamentos financeiros e dados técnicos.

Além disso, destaca-se medidas de segurança, como senhas criptografadas (*hashed*), conexões seguras via TLS, controle de acesso e monitoramento constante, garantindo a integridade e confidencialidade das informações. Em caso de incidentes relevantes, os usuários afetados serão notificados, conforme determina a LGPD. A política também assegura direitos previstos na lei, como o de acessar, corrigir ou excluir dados pessoais, reforçando o compromisso da plataforma com a transparência, segurança e responsabilidade no tratamento de dados pessoais.

## 5. Análise e Discussão

Nesta seção, são analisados e discutidos os resultados obtidos na fase de validação do Finance Vision. O objetivo é fazer uma análise crítica do sistema através do teste de usabilidade *System Usability Scale* (SUS), interpretar os dados coletados e transformá-los em métricas quantitativas para, posteriormente, haver um diagnóstico conciso e consistente da plataforma. O código-fonte do Finance Vision pode ser visualizado no repositório do projeto[1].

Na aplicação do teste SUS, um total de 13 pessoas na faixa etária de 18 a 25 anos, tiveram acesso e interação com o Finance Vision, podendo testar todas as telas de interface e suas funcionalidades contidas. Após o uso da plataforma, os usuários foram submetidos a um formulário de avaliação com 10 perguntas referente a usabilidade, facilidade de aprendizado, integração das funcionalidades e simplicidade da interface. Cada

[1]Repositório do projeto no *GitHub*: `https://github.com/anthonidaluz/FinanceVision`.

uma das perguntas do questionário é pontuada pelo usuário em uma escala *Likert* de 1 a 5, conforme Figura 2.

O cálculo do SUS envolve as seguintes etapas:

1. Identificar as perguntas ímpares e pares:
    - Perguntas ímpares (1, 3, 5, 7, 9): subtraia 1 da resposta.
    - Perguntas pares (2, 4, 6, 8, 10): subtraia o valor da resposta de 5.
2. Somar os valores ajustados:
    - Após o ajuste das respostas (ímpares e pares), some os valores.
3. Multiplicar o total por 2,5:
    - O valor total ajustado deve ser multiplicado por 2,5 para obter a pontuação final do SUS, que varia de 0 a 100.

O sistema alcançou uma pontuação média de 95.00 no *score* do SUS. Este resultado é classificado como Excelente (grau A de aceitação), de acordo com a escala de adjetivos de Bangor et al. (2009), posicionando o Finance Vision entre os sistemas com maior avaliação de usabilidade.

A análise detalhada e cálculo das respostas (Figura 26, disponível no Apêndice 1), demonstra um alto nível de consistência nas respostas. As pontuações individuais variaram de 85.0 a 100, com todos os 13 usuários classificando o sistema na faixa Excelente. Quatro usuários avaliaram o sistema com a pontuação individual máxima (100). Este resultado valida o desenvolvimento adotado, que priorizou a prototipação de alta fidelidade e um *design* de interface intuitivo, atingindo e superando os objetivos de usabilidade propostos por este trabalho.

## 6. Considerações Finais

O presente trabalho teve como propósito desenvolver e analisar o sistema Finance Vision, uma plataforma *web* voltada ao controle de finanças pessoais de jovens, com o objetivo de promover a organização financeira e incentivar hábitos econômicos mais conscientes. O estudo partiu do reconhecimento da falta de educação financeira entre os jovens brasileiros e buscou, por meio da tecnologia, oferecer uma solução acessível, interativa e educativa. Para alcançar esse objetivo, foram estabelecidas etapas de levantamento de requisitos, modelagem e desenvolvimento, resultando na criação de um sistema funcional que alia gestão financeira e gamificação como estratégia de engajamento e aprendizagem.

Os resultados obtidos evidenciam que a plataforma atendeu aos objetivos propostos, oferecendo funcionalidades completas para o registro de receitas e despesas, acompanhamento por meio de gráficos e relatórios, definição de metas e integração de elementos de gamificação. A pontuação de 95.00 atingida na aplicação do teste SUS, reforça também a aceitação dos usuários ao interagirem com a plataforma. Dessa forma, o Finance Vision representa uma contribuição significativa de soluções tecnológicas voltadas ao público jovem, reforçando o potencial de ferramentas digitais para auxiliar as pessoas no desenvolvimento da educação financeira.

Entretanto, reconhece-se que o trabalho apresenta limitações, especialmente em relação à ausência de uma ampla integração com instituições financeiras reais. Além disso, por se tratar de um protótipo inicial, futuras melhorias poderão incluir o desenvolvimento de um aplicativo móvel, o uso de Inteligência Artificial para personalizar

recomendações financeiras e a ampliação das funcionalidades gamificadas. Dessa forma, sugere-se que pesquisas futuras explorem novas abordagens tecnológicas e metodológicas que aprimorem ainda mais a experiência do usuário e ampliem o impacto do sistema no contexto da educação financeira juvenil.

## Referências


Alves, F. (2015). Gamification: como criar experiências de aprendizagem engajadoras. Disponível em: https://shre.ink/xz4U. Acesso em: 11 jun. 2025.

Alves, M. B. M. (2017). Bolso virtual: aplicação web móvel para controle de finanças pessoais. Disponível em: https://shre.ink/eZPX. Acesso em: 09 abr. 2025.

Andrade, L. M. e Carraro, W. H. (2018). Mudanças nos hábitos do controle financeiro pessoal com educação financeira sustentável. *Saber Humano*. Acesso em: 30 abr. 2025.

Astigarraga, J. e Cruz-Alonso, V. (2022). ¡se puede entender cómo funcionan git y github! Disponível em: https://shre.ink/xzYE. Acesso em: 10 jun. 2025.

Atlassian (2025). O que é Scrum e como começar. Disponível em: https://shre.ink/xzY6. Acesso em: 17 jun. 2025.

Bangor, A., Kortum, P., e Miller, J. (2009). Determining what individual SUS scores mean: adding an adjective rating scale. *Journal of Usability Studies*, 4(3):114–123.

Batista, M. A. (2023). Endividamento precoce: breve descrição da situação financeira dos jovens brasileiros nos anos de 2019 a 2023. Disponível em: https://shre.ink/eZPS. Acesso em: 09 abr. 2025.

Bommasani, R. (2021). On the opportunities and risks of foundation models. Acesso em: 16 dez. 2025.

Brasil (2018). Lei Geral de Proteção de Dados Pessoais (LGPD). Disponível em: https://shre.ink/xz41. Acesso em: 11 jun. 2025.

Brasil (2020). Decreto nº 10.393. Disponível em: https://shre.ink/eZYv. Institui a nova Estratégia Nacional de Educação Financeira (ENEF) e o Fórum Brasileiro de Educação Financeira (FBEF). Acesso em: 02 maio 2025.

Brooke, J. (1986). SUS: a quick and dirty usability scale. Disponível em: https://shre.ink/e7tp. Acesso em: 16 maio 2025.

Brooke, J. (2013). SUS: a retrospective. Disponível em: https://shre.ink/e7Sg. Published in Journal of Usability Studies, 8(2), pp. 29–40. Acesso em: 16 maio 2025.

Burke, B. e Gartner, I. (2015). *Gamificar: como a gamificação motiva as pessoas a fazerem coisas extraordinárias*. DVS Editora, São Paulo.

Cerbasi, G. (2015). *Como organizar sua vida financeira*. Sextante, Rio de Janeiro.

Deterding, S., Dixon, D., Khaled, R., e Nacke, L. (2011). From game design elements to gamefulness: defining gamification. Disponível em: https://shre.ink/e7So. Acesso em: 30 abr. 2025.

Documentação Laravel (2024). Laravel - the PHP framework for web artisans. Disponível em:https://shre.ink/xR1n. Acesso em: 12 jul. 2025.

Ferraz, F. (2023). Gamificação, programas de recompensa e videogames. Disponível em: https://shre.ink/xz4l. Acesso em: 27 maio 2025.

Ferreira, B. S. (2021). Framework Laravel: um estudo de caso full stack development. Disponível em: https://shre.ink/xzYY. Acesso em: 11 jun. 2025.

Fielding, R. T. (2000). Architectural styles and the design of network-based software architectures. Disponível em:https://shre.ink/o4tV. Acesso em: 17 set. 2025.

Figma (2016). Figma: collaborative interface design tool. Acesso em: 18 jun. 2025.

GitHub (2025). Sobre o GitHub e o Git. Disponível em: https://shre.ink/xzYF. Acesso em: 18 jun. 2025.

Hamari, J., Koivisto, J., e Sarsa, H. (2014). Does gamification work? — A literature review of empirical studies on gamification. In *Hawaii International Conference on System Science*, Hawaii, HA.

Hill, R. P., Ortega, M., e Williams, J. D. (2014). *Financial literacy and the young adult consumer: a study of attitudes and behavior*. Journal of Consumer Affairs.

ISO (1998). Iso 9241-11: Ergonomic requirements for office work with visual display. Disponível em: https://www.iso.org/standard/16883.html. Acesso em: 11 jun. 2025.

Johnson, H. A. (2017). Trello. Acesso em: 8 jun. 2025.

Kapp, K. M. (2012). *The gamification of learning and instruction: game-based methods and strategies for training and education*. Wiley, San Francisco, EUA. Acesso em: 02 jun. 2025.

Kaur, A. (2018). App review: Trello. Disponível em: https://shre.ink/xzYO. Acesso em: 18 jun. 2025.

Kotstein, S. e Bogner, J. (2021). Which restful api design rules are important and how do they improve software quality? A delphi study with industry experts. Disponível em:https://shre.ink/o4tw. Acesso em: 25 set. 2025.

Machado, M. e Medina, S. G. (2009). Scrum – método ágil: uma mudança cultural na gestão de projetos de desenvolvimento de software. Acesso em: 17 jun. 2025.

MySQL (2005). MySQL reference manual. Disponível em: https://www.mysql.com/. Acesso em: 13 jun. 2025.

Oliveira, C. P. (2023). Sistema web para controle financeiro pessoal. Disponível em: https://shre.ink/eZPF. Acesso em: 18 abr. 2025.

PHP (2025). PHP documentation. Disponível em: https://www.php.net/docs.php. Acesso em: 16 jun. 2025.

Pinheiro, P. P. (2021). *Proteção de dados pessoais: comentários à Lei n. 13.709/2018 (LGPD)*. Saraiva, São Paulo.

Ramos, B. S. (2021). Desenvolvimento de um aplicativo para controle financeiro pessoal. Disponível em: https://shre.ink/eZPY. Acesso em: 30 abr. 2025.

Reguze, T. e Silva, R. P. (2016). Gamificação aplicada a ambientes de aprendizagem. Disponível em: https://shre.ink/xz4m. Acesso em: 11 jun. 2025.

Reizes, M. B. d. (2023). Sistema de gerenciamento para finanças pessoais. Disponível em: https://shre.ink/eZPJ. Acesso em: 18 abr. 2025.

Silva, A. L. P., Benevides, F. T., Duarte, F. V., da Nobrega Oliveira, J., e Cordeiro, R. (2018). Finanças pessoais: análise do nível de educação financeira de jovens estudantes do IFPB. *Principia: Caminhos da Iniciação Científica*. Acesso em: 30 abr. 2025.

Silva, B. A. B. d. e Monteiro, J. M. (2023). Educação financeira: um estudo sobre a sua importância na gestão pessoal. Disponível em: https://shre.ink/e7tx. Acesso em: 02 maio 2025.

Silva, L. F. d. (2025). Apostila de PHP. Disponível em: https://shre.ink/xzYR. Acesso em: 10 jun. 2025.

Sommerville, I. (2011). *Engenharia de software*. Pearson Prentice Hall, São Paulo.

Souza, J. F. d. (2023). Sistema web para gestão de finanças pessoais. Disponível em: https://shre.ink/eZP7. Acesso em: 09 abr. 2025.

Tamboro (2020). Gamificação. Disponível em: https://shre.ink/xz4n. Acesso em: 27 maio 2025.

Tavares, A. H. B., Nascimento, D. E. C., e Souza, D. N. (2024). FINANC+ - software de gestão financeira. Disponível em: `https://shre.ink/euPe`. Acesso em: 30 abr. 2025.

Tonsig, S. L. (2006). *MySQL: aprendendo na prática*. Ciência Moderna, Rio de Janeiro.

UX Design Brasil (2022). Guia atualizado de como utilizar a escala SUS (System Usability Scale) no seu produto. Disponível em: `https://shre.ink/eG41`. Acesso em: 15 maio 2025.

Werbach, K. e Hunter, D. (2012). *For the win: how game thinking can revolutionize your business*. Wharton Digital Press, Philadelphia, EUA. Acesso em: 30 abr. 2025.

Zichermann, G. e Cunningham, C. (2011). *Gamification by design: implementing game mechanics in web and mobile apps*. O'Reilly Media, Sebastopol, Canadá. Acesso em: 30 abr. 2025.

## A. Apêndice 1 - Interfaces do Sistema

| Respondente | 1. Eu acho que gostaria de usar esse sistema com frequência. | 2. Eu acho o sistema desnecessariamente complexo. | 3. Eu achei o sistema fácil de usar. | 4. Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema. | 5. Eu acho que as várias funções do sistema estão muito bem integradas. | 6. Eu acho que o sistema apresenta muita inconsistência. | 7. Eu imagino que as pessoas aprenderão como usar esse sistema rapidamente. | 8. Eu achei o sistema atrapalhado de usar. | 9. Eu me senti confiante ao usar o sistema. | 10. Eu precisei aprender várias coisas novas antes de conseguir usar o sistema. | Pontuação Bruta do SUS | Pontuação Final do SUS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 4 | 1 | 5 | 1 | 4 | 1 | 5 | 2 | 5 | 1 | 37 | 92,5 |
| B | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 40 | 100 |
| C | 4 | 3 | 5 | 1 | 4 | 2 | 5 | 1 | 4 | 1 | 34 | 85 |
| D | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 40 | 100 |
| E | 4 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 39 | 97,5 |
| F | 5 | 1 | 5 | 1 | 1 | 1 | 5 | 1 | 5 | 1 | 36 | 90 |
| G | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 40 | 100 |
| H | 4 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 39 | 97,5 |
| I | 5 | 3 | 5 | 2 | 5 | 2 | 5 | 1 | 5 | 1 | 36 | 90 |
| J | 4 | 3 | 5 | 2 | 5 | 1 | 5 | 1 | 5 | 1 | 36 | 90 |
| K | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 40 | 100 |
| L | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 40 | 100 |
| M | 5 | 1 | 5 | 2 | 5 | 1 | 4 | 1 | 5 | 2 | 37 | 92,5 |
| | | | | | | | | | | | Média | 95,0 |

**Figura 26. Cálculo do teste SUS.**


**Figura 27. Estatística sobre a gamificação. Fonte: Tamboro (2020).**

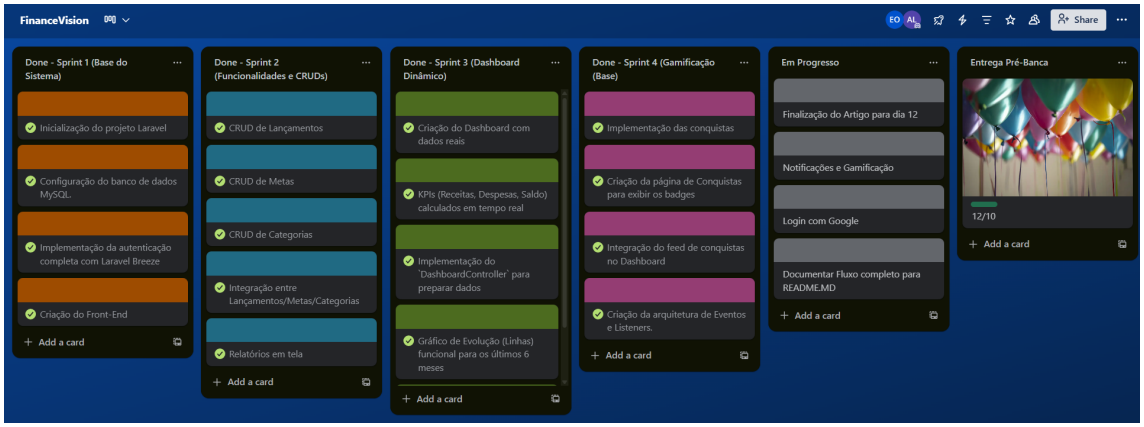



**Figura 28. Quadro *Kanban* criado no *Trello*.**

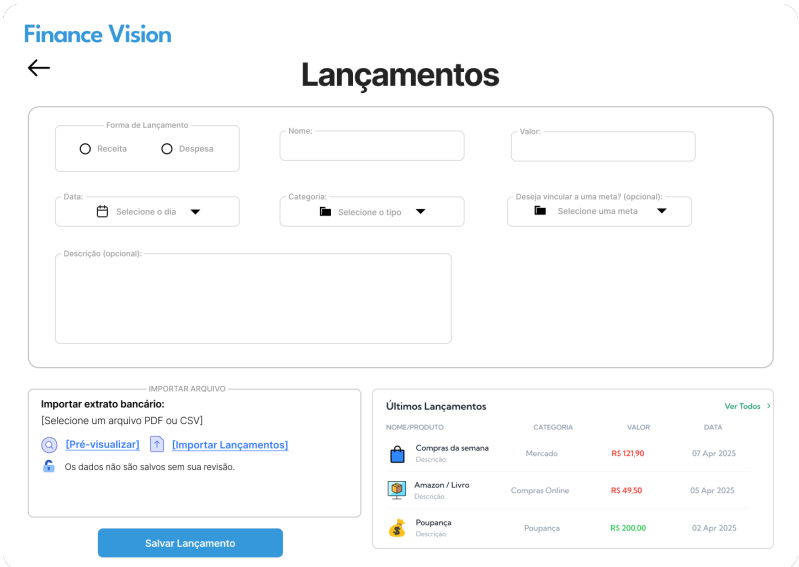



**Figura 29. Tela de Lançamentos. Disponível em: shre.ink/xJZ9.**


**Figura 30. Tela de Relatórios. Disponível em: shre.ink/xJZ9.**

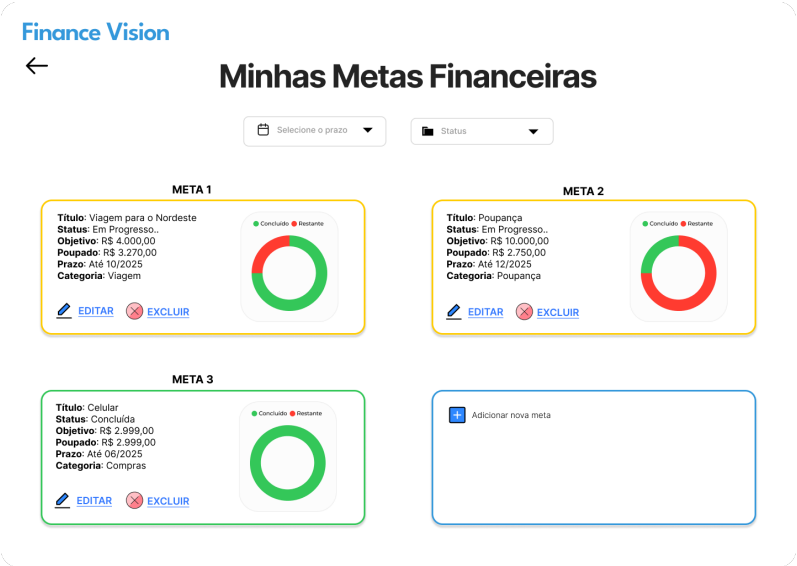



**Figura 31. Tela de Metas. Disponível em: shre.ink/xJZ9.**

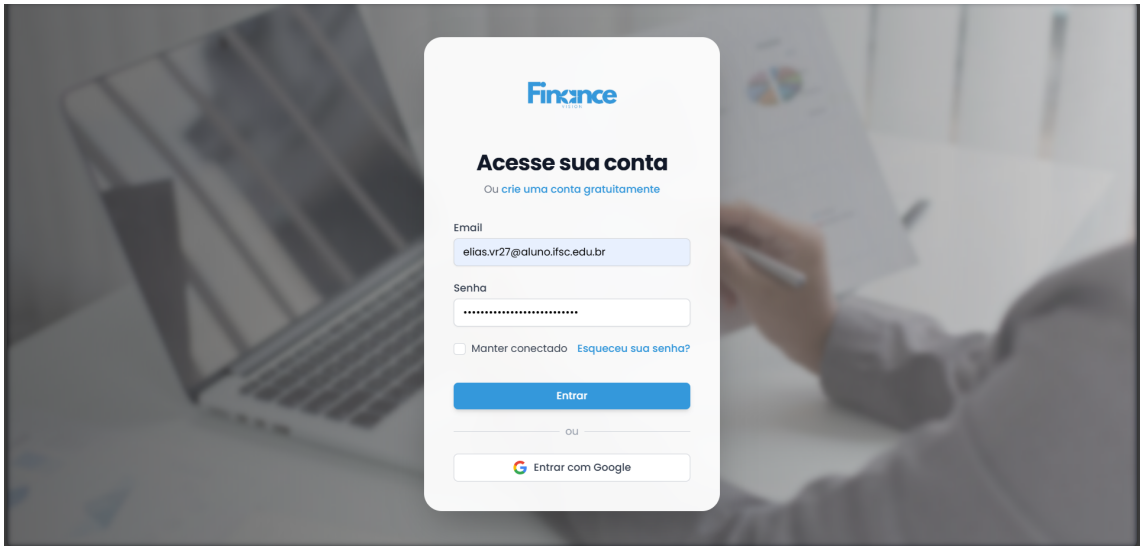



**Figura 32. Tela de *Login*.**

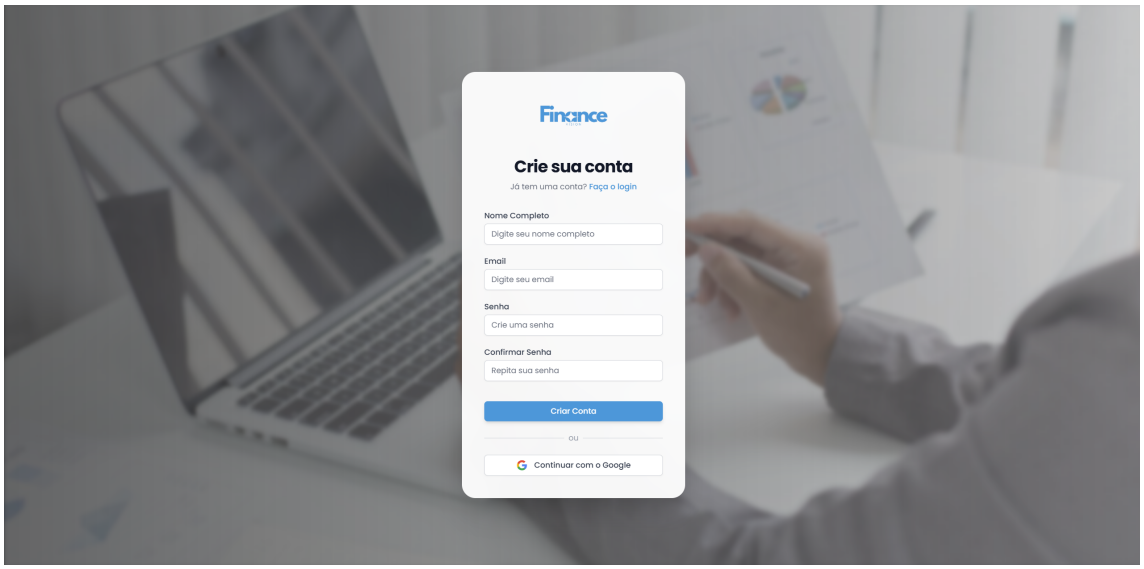



**Figura 33. Tela de Cadastro de Contas.**

```php
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Models\User;
use Illuminate\Auth\Events\Registered;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\Rules;
use Illuminate\View\View;

class RegisteredUserController extends Controller
{

    public function create(): View

    {
        return view('auth.register');
    }

    public function store(Request $request): RedirectResponse
    {
        $request->validate([
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'lowercase', 'email', 'max:255'
, 'unique:'.User::class],
            'password' => ['required', 'confirmed', Rules\Password::defaults
()],
        ]);

        $user = User::create([
            'name' => $request->name,
            'email' => $request->email,
            'password' => Hash::make($request->password),
        ]);

        event(new Registered($user));

        Auth::login($user);

        return redirect(route('dashboard', absolute: false));
    }
}
```

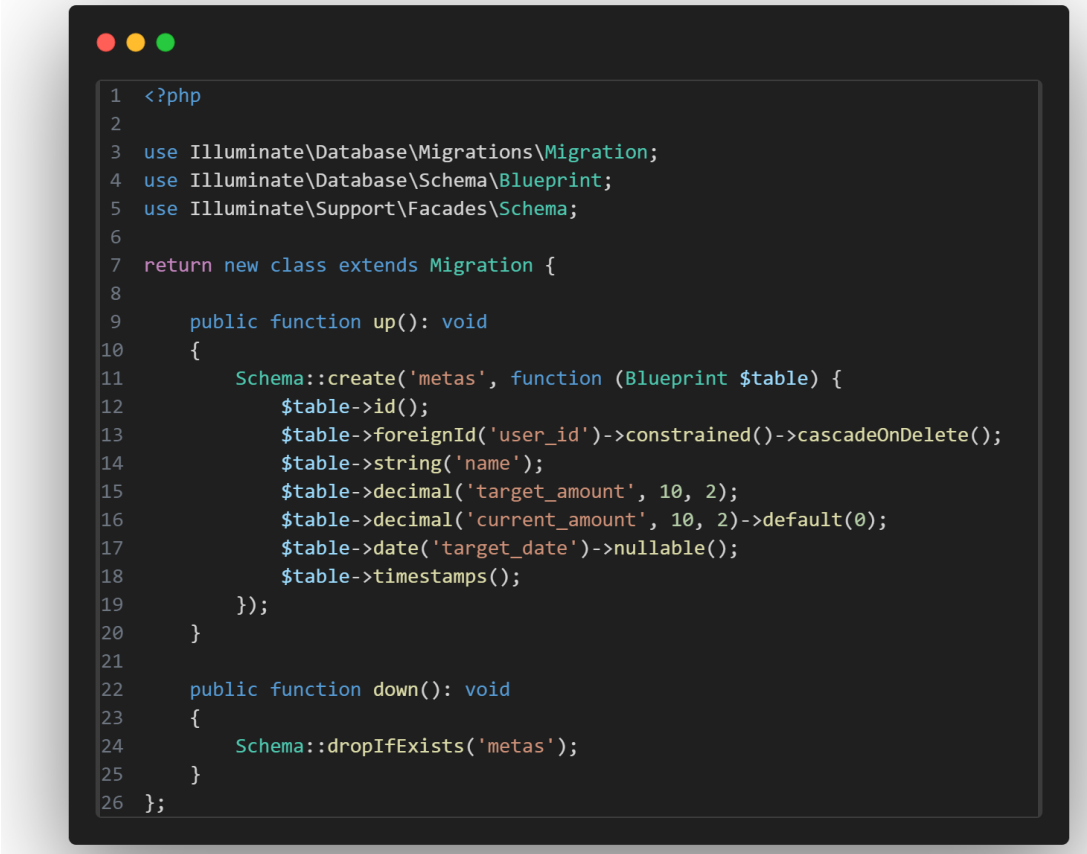
**Figura 34. *Controller* dos usuários.**

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

class User extends Authenticatable
{
    use HasFactory, Notifiable;

    protected $fillable = [
        'name',
        'email',
        'password',
    ];

    public function lancamentos()
    {
        return $this->hasMany(Lancamento::class);
    }

    public function metas()
    {
        return $this->hasMany(Meta::class);
    }

    public function categories()
    {
        return $this->hasMany(Category::class);
    }

    protected $hidden = [
        'password',
        'remember_token',
    ];

    protected function casts(): array
    {
        return [
            'email_verified_at' => 'datetime',
            'password' => 'hashed',
        ];
    }
}
```

**Figura 35. *Model* do usuário.**

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{

    public function up(): void
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });

        Schema::create('password_reset_tokens', function (Blueprint $table) {
            $table->string('email')->primary();
            $table->string('token');
            $table->timestamp('created_at')->nullable();
        });

        Schema::create('sessions', function (Blueprint $table) {
            $table->string('id')->primary();
            $table->foreignId('user_id')->nullable()->index();
            $table->string('ip_address', 45)->nullable();
            $table->text('user_agent')->nullable();
            $table->longText('payload');
            $table->integer('last_activity')->index();
        });
    }

    public function down(): void
    {
        Schema::dropIfExists('users');
        Schema::dropIfExists('password_reset_tokens');
        Schema::dropIfExists('sessions');
    }
};

```

**Figura 36. *Migrations* de criação de usuários.**



```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration {

    public function up(): void
    {
        Schema::create('metas', function (Blueprint $table) {
            $table->id();
            $table->foreignId('user_id')->constrained()->cascadeOnDelete();
            $table->string('name');
            $table->decimal('target_amount', 10, 2);
            $table->decimal('current_amount', 10, 2)->default(0);
            $table->date('target_date')->nullable();
            $table->timestamps();
        });
    }

    public function down(): void
    {
        Schema::dropIfExists('metas');
    }
};
```

**Figura 37. *Migrations* das metas financeiras.**

```php
class CategoryController extends Controller
{
    public function index()
    {
        $categories = Auth::user()->categories()->latest()->get();
        return view('categorias.index', compact('categories'));
    }

    public function create()
    {
        return view('categorias.create');
    }

    public function store(Request $request)
    {
        $validated = $request->validate([
            'name' => 'required|string|max:255',
            'icon' => 'nullable|string|max:255',
            'color' => 'nullable|string|max:7',
        ]);

        Auth::user()->categories()->create($validated);

        return redirect()->route('categorias.index')->with('success',
'Categoria criada com sucesso!');
    }

    public function edit(Category $category)
    {
        if ($category->user_id !== Auth::id())
            abort(403);
        return view('categorias.edit', compact('category'));
    }

    public function update(Request $request, Category $category)
    {
        if ($category->user_id !== Auth::id())
            abort(403);
        $validated = $request->validate([
            'name' => 'required|string|max:255',
            'icon' => 'nullable|string|max:255',
            'color' => 'nullable|string|max:7',
        ]);
        $category->update($validated);
        return redirect()->route('categorias.index')->with('success',
'Categoria atualizada com sucesso!');
    }

    public function destroy(Category $category)
    {
        if ($category->user_id !== Auth::id())
            abort(403);
        $category->delete();
        return redirect()->route('categorias.index')->with('success',
'Categoria excluída com sucesso!');
    }
}
```

**Figura 38. *Controller* das categorias.**

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Database\Eloquent\Relations\HasMany;

class Category extends Model
{
    use HasFactory;

    protected $fillable = [
        'name',
        'icon',
        'color',
        'user_id',
    ];


    public function user(): BelongsTo
    {
        return $this->belongsTo(User::class);
    }

    public function lancamentos(): HasMany
    {
        return $this->hasMany(Lancamento::class);
    }
}
```

**Figura 39. *Model* das categorias.**

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration {

    public function up(): void
    {
        Schema::create('categories', function (Blueprint $table) {
            $table->id();
            $table->foreignId('user_id')->constrained()->cascadeOnDelete();
            $table->string('name');
            $table->string('icon')->nullable();
            $table->string('color')->nullable();
            $table->timestamps();
        });
    }

    public function down(): void
    {
        Schema::dropIfExists('categories');
    }
};
```

**Figura 40. *Migrations* das categorias.**

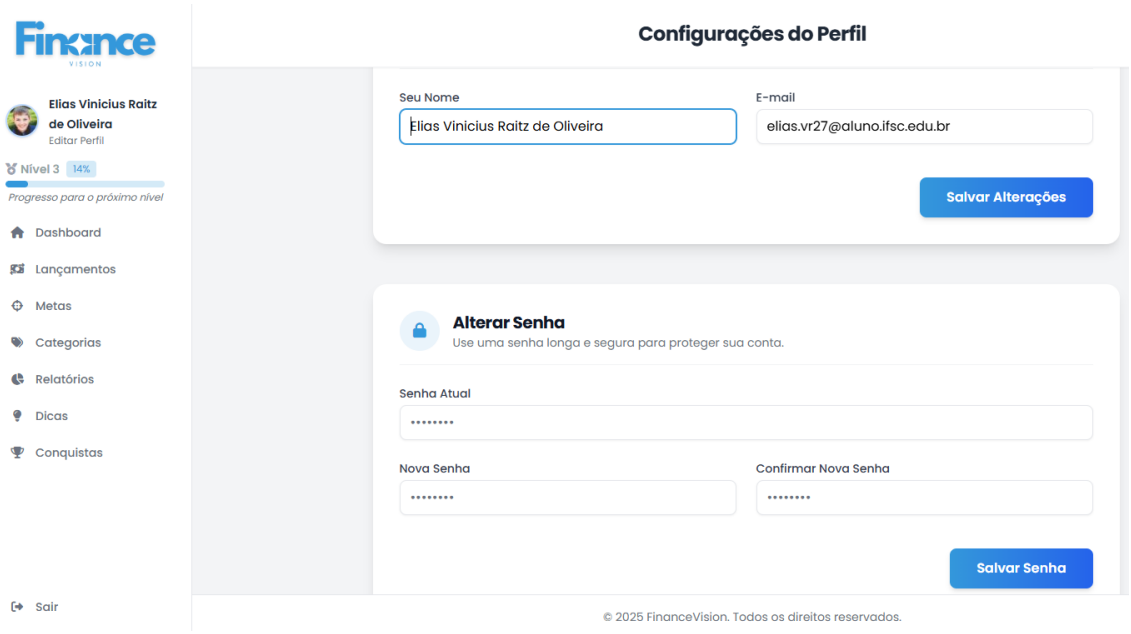



**Figura 41. Tela de Edição de Perfil.**

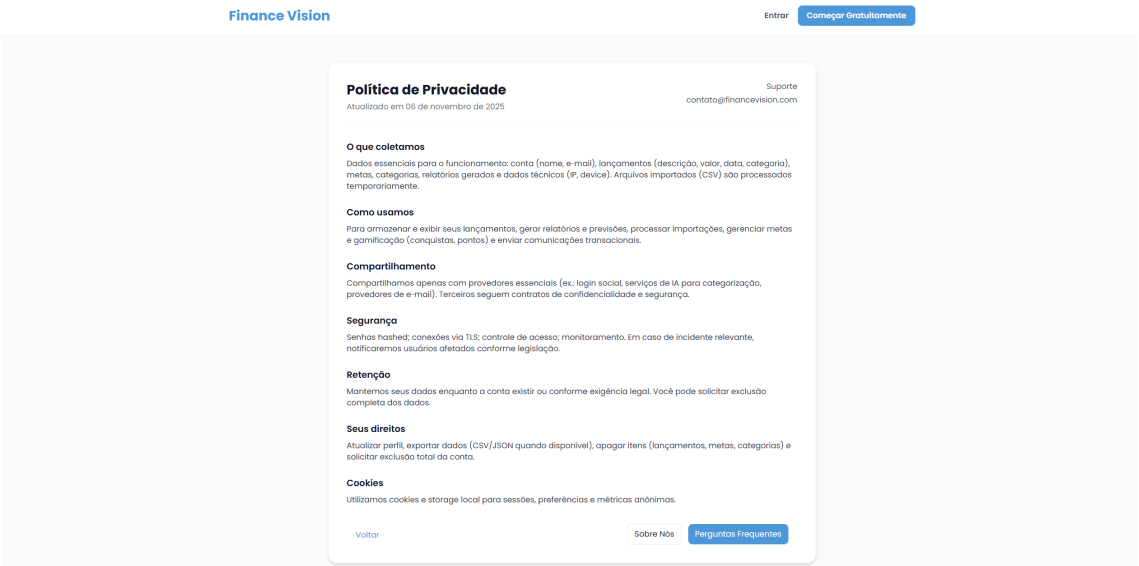



**Figura 42. Tela de Políticas de Privacidade.**