

FACULDADE SENAC PALHOÇA
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE
SISTEMAS

Adriene Batista De Paula
Anderson Raulino Dutra
Guilherme dos Santos Thiel
Gustavo Bonella dos Santos
Laura Prass Senna

MONETAR.IA:
APLICATIVO MÓVEL DE GESTÃO FINANCEIRA INTEGRADO COM
INTELIGÊNCIA ARTIFICIAL

Palhoça
2024

Adriene Batista De Paula
Anderson Raulino Dutra
Guilherme dos Santos Thiel
Gustavo Bonella dos Santos
Laura Prass Senna

MONETAR.IA:
APLICATIVO MÓVEL DE GESTÃO FINANCEIRA INTEGRADO COM
INTELIGÊNCIA ARTIFICIAL

Trabalho apresentado ao Faculdade Senac Palhoça
como requisito parcial para obtenção do título de
Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientadores: Prof. Jaime Elias Vieira, Me.
Prof. Telma Alessandra Correa da Silva,
Me.
Prof. Wildson Caio Felipe, Esp.

Coorientador: Prof. Luciano Figueiredo Coelho, Me.

Palhoça
2024

Ficha de identificação da obra

M742p Monetar.IA: aplicativo móvel de gestão financeira integrado com inteligência artificial / Adriene Batista de Paula... [et al.] – Palhoça (SC): Faculdade Senac Palhoça, 2024.
88 p.: il. color.

Orientadores: Jaime Elias Vieira, Telma Alessandra Correa da Silva, Wildson Caio Felipe. Coorientador: Luciano Figueiredo Coelho.

Trabalho de Conclusão de Curso (Graduação em Análise e Desenvolvimento de Sistemas) – Faculdade Senac Palhoça, 2024.

1. Inteligência Artificial Generativa. 2. Finanças pessoais. 3. Arquitetura em Microsserviços I. Paula, Adriene Batista de. II. Dutra, Anderson Raulino. III. Thiel, Guilherme dos Santos. IV. Santos, Gustavo Bonella dos. V. Senna, Laura Prass. VI. Vieira, Jaime Elias de. VII. Silva, Telma Alessandra Correa da. VIII. Felipe, Wildson Caio IX. Figueiredo, Luciano Coelho. X. Título.

CDD 22 ed.: 371.3078

Ficha catalográfica elaborada pela bibliotecária Jamile de Sousa – CRB 14/1563



Creative Commons - Atribuição - Não Comercial CC BY-NC

**ADRIENE BATISTA DE PAULA
ANDERSON RAULINO DUTRA
GUILHERME DOS SANTOS THIEL
GUSTAVO BONELLA DOS SANTOS
LAURA PRASS SENNA**

MONETAR.IA:


**APLICATIVO MÓVEL DE GESTÃO FINANCEIRA INTEGRADO COM
INTELIGÊNCIA ARTIFICIAL**

Trabalho apresentado ao Faculdade Senac Palhoça como requisito parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

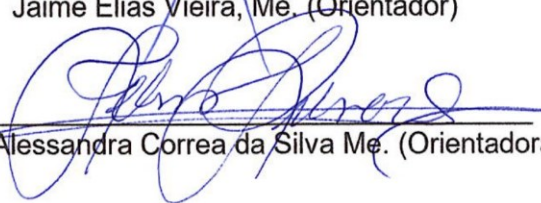
Orientador: Prof. Jaime Elias Vieira, Me.
Prof. Telma Alessandra Correa da Silva, Me.
Prof. Wildson Caio Felipe, Esp.

Coorientador: Prof. Luciano Figueiredo Coelho, Me.

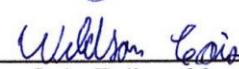
Banca:



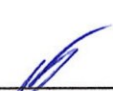
Jaime Elias Vieira, Me. (Orientador)



Telma Alessandra Correa da Silva Me. (Orientadora)



Wildson Caio Felipe, Me. (Orientador)



Luciano Figueiredo Coelho, Me. (Orientador)

Palhoça, 03 de dezembro de 2024.

AGRADECIMENTOS

Gostaríamos de expressar nossa mais profunda gratidão aos orientadores Jaime Elias Vieira, Telma Alessandra Correa da Silva e Wildson Caio Felipe, pelo conhecimento e dedicação para a realização deste trabalho acadêmico. Agradecemos também ao coordenador do curso de Análise e Desenvolvimento de Sistemas, Luciano Figueiredo Coelho e toda a equipe da Faculdade Senac Palhoça, que nos proporcionou um ambiente acadêmico impecável para a concretização desta etapa em nossa jornada.

Agradecemos aos nossos familiares, que nos ofereceram um apoio imensurável durante a realização deste projeto, em especial: Rodolfo Batista de Paula, Zeloira de Fatima Rodrigues de Souza, Elza Dutra, Letícia Guimarães Dutra, Thalyta de Oliveira Guimarães Dutra, Roseli Maria dos Santos, Sirlei Araujo Thiel, Jaíne Rodrigues da Luz, João Ademir dos Santos, Renê Maria Bonella dos Santos, Ana Paula Prass, Artur Prass e Fernando Prass. Sua paciência, compreensão e incentivo constante foram fundamentais para superarmos os desafios e alcançarmos este marco tão significativo.

Agradecemos também aos professores Marcelo da Silva e Vlademir Jean Machado que nos ajudaram, respectivamente, no diagrama de banco de dados e nas tabelas de requisitos. Nossa gratidão se estende à bibliotecária Jamile de Sousa, pela valiosa assistência na formatação deste trabalho. Sem o apoio de vocês, nossos esforços certamente teriam sido muito mais desafiadores.

E, por fim, agradecemos aos nossos colegas de turma e professores, pelo companheirismo e ajuda durante esses cinco semestres da graduação.

“O que todos devemos fazer é nos certificar que estamos usando a inteligência artificial de uma maneira que beneficie a humanidade, e não que a deteriore” (Tim Cook).

RESUMO

O seguinte trabalho acadêmico propõe uma aplicação móvel com foco em gestão financeira pessoal com integração com inteligência artificial generativa para auxílio educacional, automação de tarefas e análise preditiva das informações cadastradas pelo usuário. O objetivo principal da proposta é o exame de hábitos econômicos e comportamentais através das informações de renda, despesas e metas disponibilizadas pelo usuário e analisadas por inteligência artificial, com seus resultados computados por relatórios e *dashboards*. Para tanto, foi desenvolvido uma robusta API Rest em uma arquitetura de microsserviços onde integra-se o modelo de inteligência artificial generativo com o banco de dados em PostgreSQL e o *front-end* da aplicação, criando nas linguagens Flutter e Dart. Utilizou-se uma metodologia de pesquisa quantitativa em relação a abordagem e exploratório quanto aos objetivos. A escolha do modal se deu em razão da alta aceitabilidade dos aparelhos *mobile* para a gestão financeira e bancária no Brasil nos últimos anos. O desenvolvimento do aplicativo iniciou-se com o levantamento de requisitos, a criação de um protótipo em Figma e a modelagem do sistema, utilizando-se a metodologia ágil *scrumban* para a condução dos trabalhos de desenvolvimento de *software*. Nos resultados obtidos o LLM do ChatGPT mostrou-se superior ao Gemini com respostas de *prompt* mais objetivos e precisos, a API desenvolvida ostentou forte desempenho e ampla capacidade de manutenção e escalabilidade. Conclui-se que a aplicação desenvolvida pode ser um fator essencial para a melhora da gestão financeira pessoal e apresenta uma capacidade didática para a tomada de decisões econômicas realizada de maneira esclarecida.

Palavras-chave: Inteligência Artificial Generativa. Finanças Pessoais. Análise Preditiva. Arquitetura em Microsserviços.

ABSTRACT

This academic work proposes a mobile application focused on personal finance management, integrated with generative artificial intelligence to assist in education, task automation, and predictive analysis of the information provided by the user. The main objective of the proposal is to examine economic and behavioral habits through income, expenses, and goals provided by the user, which are analyzed by artificial intelligence, with the results computed into reports and dashboards. To achieve this, a robust REST API was developed using a microservices architecture, integrating the generative artificial intelligence model with PostgreSQL database and the application's front end, built using Flutter and Dart. A quantitative research methodology was used in terms of approach, and exploratory in relation to the objectives. The choice of the mobile platform was based on the high acceptance of mobile devices for financial and banking management in Brazil in recent years. The development of the application began with requirements gathering, the creation of a prototype in Figma, and system modeling, using the agile Scrumban methodology to guide the software development process. In the results obtained, the ChatGPT LLM proved to be superior to Gemini, with more objective and precise prompt responses. The developed API demonstrated strong performance and a high capacity for maintenance and scalability. It is concluded that the developed application can be an essential factor in improving personal financial management and offers an educational capacity for making economic decisions in a clear and informed manner.

Keywords: Generative Artificial Intelligence. Personal Finance. Predictive Analytics. Microservices Architecture.

LISTAS DE ILUSTRAÇÕES

Figura 1 - Estrutura de uma <i>Sprint Planning</i>	30
Figura 2 - Estrutura da lógica de versionamento Git	37
Figura 3 - Diagrama de Caso de Uso	56
Figura 4 - Diagrama de Caso de Atividades	57
Figura 5 - Diagrama de Arquitetura de Software	59
Figura 6 - Diagrama de Banco de Dados	60
Figura 7- Visão Geral do Notion	61
Figura 8 - Visão Geral do <i>Board</i> de Tarefas	61
Figura 9 - Repositório do Projeto no GitHub	62
Figura 10 - Conceito de Design do Protótipo	63
Figura 11 - Visão Geral da Landing Page	64
Figura 12 - Visão Geral do Digital Ocean	65
Figura 13 - Visão Geral do CapRover	65
Figura 14 - Informações Inseridas no <i>Prompt</i>	66
Figura 15 - Resposta da Gemini	67
Figura 16 - Resposta do ChatGPT	67
Figura 17 - Documentação de Funções da API	68
Figura 18 - Visão Geral da Tela Inicial da Monetar.ia	69
Figura 19 - Visão Geral da Tela Cadastro da Monetar.ia	70
Figura 20 - Visão Geral da Tela Principal da Monetar.ia	71
Figura 21 - Visão Geral da Tela Cadastro de Metas da Monetar.ia	72
Figura 22 - Visão Geral da Tela Cadastro de Receitas da Monetar.ia	73
Figura 23 - Visão Geral da Tela Cadastro de Despesas da Monetar.ia	74
Figura 24 - Visão Geral da Tela Assistente Virtual da Monetar.ia	75
Gráfico 1 - Com que frequência você monitora suas finanças pessoais?	84
Gráfico 2 - Você acha útil ter uma visão geral dos seus gastos e recebimentos em um único lugar?	84
Gráfico 3 - Você utiliza alguma ferramenta ou aplicativo para gestão financeira atualmente?	84

Gráfico 4 - Se sim, qual(is) ferramenta(s) ou aplicativo(s) você utiliza?	85
Gráfico 5 - Quais são os principais desafios que você enfrenta na gestão das suas finanças pessoais?	85
Gráfico 6 - Você gostaria de criar e acompanhar metas financeiras, como poupança, pagamento de dívidas e aumento de receitas, dentro do aplicativo?	85
Gráfico 7 - Você consideraria útil um aplicativo que analise seus hábitos de consumo de forma inteligente (utilizando inteligência artificial) e forneça sugestões personalizadas de economia e otimização financeira?	86
Gráfico 8 - Você acharia útil receber notificações e alertas personalizados sobre limites de gastos, vencimentos de contas e recebimentos esperados?	86
Gráfico 9 - Você gostaria de poder registrar suas despesas e receitas utilizando comandos de texto e voz em um aplicativo?	86
Gráfico 10 - Você acharia útil poder gerar e exportar relatórios detalhados de suas despesas e receitas em formatos como PDF ou Excel?	87
Gráfico 11 - Quão importante é para você que o aplicativo seja intuitivo e fácil de usar?	87
Gráfico 12 - Há alguma funcionalidade ou característica que você gostaria de ver no aplicativo?	87
Gráfico 13 - Quanto você estaria disposto a pagar por um aplicativo com essas funcionalidades?	88

LISTAS DE TABELAS

Tabela 1 - Definições e Siglas.....	49
Tabela 2 - Tabela de Prioridades	50
Tabela 3 - Tabela de Requisitos Funcionais	50
Tabela 4 - Tabela de Requisitos Não Funcionais.....	54

LISTA DE SIGLAS

ACID – Atomicidade, Consistência, Isolamento e Durabilidade

API – *Application Programming Interface*

BI – *Business Intelligence*

CPU – *Central Processing Unit*

DDD – *Domain-Driven Design*

HTML – *Hyper Text Markup Language*

HTTP – *Hyper Text Transfer Protocol*

HTTPS – *Hyper Text Transfer Protocol Secure*

JSON – *JavaScript Object Notation*

JWT – *JSON Web Tokens*

REST – *Representational State of Transfer*

SFN – Sistema Financeiro Nacional

SQL – *Structured Query Language*

SSL – *Secure Sockets Layer*

UI – *User Interface*

UML – *Unified Modelling Language*

VPS – *Virtual Private Server*

VUI – *Voice User Interface*

XML – *Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO	15
1.1 OBJETIVOS	17
1.1.1 Objetivo geral	18
1.1.2 Objetivos específicos	18
2 FUNDAMENTAÇÃO TEÓRICA	19
2.1 GESTÃO FINANCEIRA	19
2.2 APLICATIVOS MÓVEIS	20
2.2.1 Interface de Voz	22
2.3 INTELIGÊNCIA ARTIFICIAL.....	23
3 METODOLOGIA DE PESQUISA	26
3.1 METODOLOGIA DA CRIAÇÃO DO PRODUTO	29
3.1.1 Metodologia ágil	29
3.2 ARQUITETURA.....	31
3.2.1 API Rest	34
3.3 FERRAMENTAS	36
3.3.1 Notion	36
3.3.2 Git	36
3.3.3 GitHub	38
3.3.4 Figma	38
3.3.5 Visual Studio Code	39
3.4 DESENVOLVIMENTO MOBILE	39
3.4.1 Flutter	39
3.4.2 Dart	40
3.4.3 PostgreSQL	40
3.5 API.....	41
3.5.1 Python	41
3.5.2 Digital Ocean	42
3.5.3 CapRover	42
3.5.4 Docker	43
3.5.5 Nginx	43
3.6 DESENVOLVIMENTO DA LANDING PAGE.....	44
3.6.1 HTML	44

3.6.2 CSS	45
3.6.3 JavaScript	45
3.7 INTELIGÊNCIA ARTIFICIAL GENERATIVA	46
4 RESULTADOS E DISCUSSÃO	49
4.1 ANÁLISE E DEFINIÇÃO DE REQUISITOS	49
4.1.1 Requisitos Funcionais	50
4.1.2 Requisitos Não Funcionais	54
4.1.3 Diagrama de Caso de Uso	55
4.1.3 Diagrama de Caso de Atividades	57
4.1.4 Diagrama de Arquitetura	58
4.1.5 Diagrama de Banco de Dados	59
4.2 GERENCIAMENTO DO PROJETO	60
4.2.1 Landing Page	63
4.3 DESENVOLVIMENTO DA API	64
4.3.1 Documentação da API	67
4.4 FUNCIONAMENTO DO SISTEMA	68
5 CONCLUSÃO	77
REFERÊNCIAS	78
APÊNDICE A – Formulário de pesquisa semiestruturada com respostas	84

1 INTRODUÇÃO

A gestão financeira e econômica é um fator importante para a organização individual e familiar brasileira, todavia, mesmo de extrema importância, ainda é uma área pouco explorada e de difícil compreensão para a maioria da população. Tal fator se potencializa no Brasil em decorrência da ausência do ensino formal em educação financeira no currículo básico aplicado para todo o país, obscurecendo ainda mais o conhecimento do tema para os brasileiros.

Ademais, nota-se fatores históricos e culturais para a precarização econômica dos brasileiros. Para Savoia, Saito e Santana (2008), o extenso período de inflação comprometeu a capacidade de planejamento econômico-financeiro de longo prazo. Com a abertura econômica, no início dos anos 1990 e o processo de estabilização do Plano Real, o mercado financeiro nacional se transformou e criou instrumentos, aumentando a complexibilidade dos produtos oferecidos.

Tal panorama de instabilidade da economia brasileira resultou em gerações de indivíduos carentes de alfabetização financeira, incapazes de tomar decisões eficientes sobre seus gastos e ganhos, comprometendo sua estabilidade social e qualidade de vida.

Na visão de Sebben (2020, *apud* Reis, 2020), a inadimplência intensificou no Brasil após o pacote de medidas implementadas pelo Presidente Collor de Melo durante a década de 1990, diminuiu-se a circulação de moedas e movimentações em conta correntes além de aplicações e poupança serem congeladas, com o intuito de diminuir a taxa de inflação e obrigar a baixa dos preços por parte dos empresários. Tal medida foi determinante para a alteração do comportamento financeiro das pessoas, pois o novo cenário era instável e os consumidores mantinham hábitos obrigatórios de consumo — alimentação por exemplo —, gerando uma piora significativa do padrão de consumo e estabilidade financeira das décadas seguintes.

Com o sucesso do Plano Real e, posteriormente, o “boom” das *comodities* durante o início dos anos 2000, surge uma “nova classe média” no Brasil, impulsionada principalmente pelo aumento do valor real do salário e a facilidade de crédito. Porém, mesmo com um ambiente favorecido, esse desenvolvimento econômico de forma rápida e fácil não foi determinante para uma conclusão do endividamento e inadimplência da população brasileira, tal fenômeno se explica, em

sua maioria, pelo pouco conhecimento em educação financeira e a falta de planejamento econômico pessoal (Pochmann, 2012 *apud* Reis, 2020).

Segundo levantamento do Banco Central do Brasil (2023), percebe-se um aumento do número de tomadores de crédito e o percentual de endividados de risco, principalmente em decorrência do maior comprometimento de renda com dívidas e da inadimplência das famílias brasileiras durante o período da pandemia de covid-19. Para tanto, afirma-se:

Em março de 2020 e junho de 2021, período da pandemia de covid-19, houve uma queda percentual do endividamento de risco de 11,8% para 9,9% dos tomadores de crédito. A partir de março de 2021, a tendência mudou, e o aumento percentual do endividamento de risco no SFN tornou-se contínuo, com crescimento observado em todos os trimestres analisados desde então, atingido 14,2% em março de 2023 [...]. Aproximadamente vinte milhões de brasileiros tornaram-se tomadores de crédito, provenientes da expansão da bancarização no período da pandemia, atingindo o total de 106,2 milhões de clientes com ao menos uma operação de crédito ativa considerados na análise. Ainda assim, o número de endividados de risco cresceu proporcionalmente mais que o crescimento do número de tomadores de crédito: passou de 8,5 milhões em março de 2021 para 15,1 milhões em março de 2023, resultando em um forte crescimento do endividamento de risco em termos relativos (Banco Central do Brasil, 2023).

Como antídoto para a situação apresentada, a educação financeira poderia proporcionar uma compreensão abrangente do valor real do dinheiro, da gerência de despesas pessoais e da tomada de decisão prevista e ponderada. Dotando o indivíduo de capacidade e habilidade de administrar suas finanças (Levino; Santos, 2019). A qualidade das decisões financeiras pode ser fundamental para o planejamento saudável das finanças pessoais a longo prazo, resultando em uma situação de maior disponibilidade de renda, maior direcionamento à poupança, planejamento da aposentadoria e menor endividamento.

Observa-se nas últimas décadas um aumento substancial ao acesso à *internet* e tecnologias variadas que democratizam o acesso à informação. O controle das finanças pode ser facilitado pelo uso de computadores, *smathphones* e *tablets*, por meio das plataformas que auxiliam o usuário, além disso, a quantidade de aplicativos de extratos de cartões de crédito, *internet banking*, planilhas de controle financeiro e outras plataformas de auxílio de finanças pessoais crescem exponencialmente (Venâncio; Malaquias, 2017).

O Banco Central do Brasil (2021) observa um cenário de maior digitalização dos serviços financeiros, principalmente proporcionados pela pandemia de Covid-19,

onde o crescimento de quase 10% da quantidade de pessoas com relacionamento com o Sistema Financeiro Nacional, saindo de 149 milhões em 2020 para 163 milhões de pessoas. Tal crescimento resultado em partes pela distribuição do Auxílio Emergencial e o início da operação do Pix, no qual se mostrou de rápida adoção, possibilitando a adoção de novos paradigmas financeiros e um impacto na cidadania financeira.

Esse acontecimento se deu em parte pela popularização de dispositivos de comunicação conectados na internet, com destaque para os *smartphones*, segundo a Febraban (2024) registra-se um número de 186 bilhões de transações bancárias no país, um acréscimo de 19% comparada ao ano de 2023; desde total, 130 bilhões foram através de modal *mobile banking*, que registrou um aumento de 22% comparado ao ano anterior, enquanto as transações via *internet banking* se mantêm estáveis desde 2019.

Assim, por mais que o campo das finanças seja amplo e dinâmico, o uso da tecnologia pode ser fundamental para proporcionar mudanças de hábitos de economia e finanças pessoais, afetando diretamente a vida de todas as pessoas e organizações. Desta forma, questiona-se, portanto, se uma ferramenta de gestão financeira pessoal seria capaz de auxiliar no processo de diminuição de dívidas, o aprendizado de educação financeira e na melhoria da gestão econômica individual.

Neste contexto, este trabalho acadêmico propõe o desenvolvimento de uma aplicação móvel com um assistente virtual de finanças personalizado, integrado com inteligência artificial generativa, visando tornar a gestão econômica mais ágil, informada e acessível. A metodologia adotada será quantitativa em sua abordagem e exploratória em relação aos objetivos, com intuito de analisar de maneira aprofundada o impacto da solução na vida financeira dos usuários. O objetivo final é proporcionar uma ferramenta capaz de auxiliar o usuário a sair de uma situação financeira precária, oferecendo suporte e clareza para que ele alcance suas metas e objetivos econômicos de forma sustentável e assertiva.

1.1 OBJETIVOS

Criar uma ferramenta de gestão econômica que proponha cultivar hábitos de controle e fiscalização das receitas e despesas pessoais com propósito de instruir sobre praxes de educação financeira e melhorar o bem-estar financeiro do usuário.

1.1.1 Objetivo geral

Desenvolver um aplicativo móvel de gestão financeira que permita o usuário a disponibilizar informações de receitas, despesas e metas que poderão ser computadas em *dashboards* e relatórios de administração de finanças pessoais, integradas com o uso de inteligência artificial para análise preditiva e categorização de padrões de comportamento e hábitos econômicos.

1.1.2 Objetivos específicos

Os objetivos específicos deste trabalho serão:

- Identificar os requisitos e fundamentos básicos para o desenvolvimento da aplicação móvel;
- Projetar e desenvolver a aplicação utilizando ferramentas e tecnologias pertinentes;
- Desenvolver funcionalidade de reconhecimento de voz com o objetivo de tornar a aplicação intuitiva;
- Integrar a aplicação com inteligência artificial com o intuito de trazer avaliações sobre padrões de comportamentos e gastos do usuário.

2 FUNDAMENTAÇÃO TEÓRICA

Com razão de completar os objetivos propostos, far-se-á necessário aprofundar em temas chave para o desenvolvimento da ferramenta, para tanto, apresenta-se os seguintes temas como principais para a realização da aplicação móvel.

2.1 GESTÃO FINANCEIRA

Entende-se por finanças uma série de princípios econômicos e financeiros para maximizar um determinado resultado de valor num intervalo de tempo. Esse resultado, é a riqueza que pode ser medida através do lucro ou o aumento do patrimônio de uma pessoa ou empresa (Fonseca, 2013). Em outras palavras, a administração financeira nada mais é do que a melhor utilização dos princípios microeconômicos e a utilização das ferramentas financeiras para obter o melhor resultado de um negócio.

Quando se aborda sobre finanças, fala-se principalmente sobre a gestão de capitais ou de produtos, onde necessita de um cálculo para apurar custos e receitas onde o produto é uma margem de lucro ou uma despesa (Schenini; Bonavita, 2012).

Para obter um bom planejamento, é necessário internalizar uma visão de futuro obtido pelo desejo de realização de um projeto ou o estabelecimento de metas claras e objetivas. Dessa forma, torna-se elementar a existência de um orçamento, onde seja registrada e organizada todas as receitas (rendas) e todas as despesas (gastos), além de investimentos e metas (Banco Central do Brasil, 2013).

Para a criação do orçamento é fundamental começar pelos itens mais importantes, de maior peso para o orçamento, como, por exemplo, pelos rendimentos, seguido por despesas fixas, como aluguel ou condomínio, seguidos pelos gastos menos importantes (Schenini; Bonavita, 2012). Tal comportamento deve ser documentado todos os meses para atingir uma certa lógica de controle a longo prazo.

A origem da renda, geralmente, advém do resultado do trabalho, obtido pelo salário, porém, pode ter outras origens, sendo as formas mais comuns: comissões, diárias, honorários, pró-labore, prêmios, planos de previdência social ou privada, faturamento do rendimento de aplicações financeiras ou em bolsa de valores ou de

aplicações não financeiras como aluguel de imóveis, heranças, *royalties* ou através de programas de assistência social. Por outro lado, as principais despesas podem ser feitas para o mantimento de funções básicas e inerentes da vida humana, como, por exemplo, alimentação, moradia, educação, saúde, lazer e viagens, ou por meio de dívidas e da cobrança de juros em casos excepcionais (Banco Central do Brasil, 2013).

O nivelar entre a renda e a despesa é chamada de ponto de equilíbrio, que permite realizar uma avaliação da saúde financeira de uma entidade, seus diferentes níveis de faturamento ou nível de operações (Schenini; Bonavita, 2012). O ponto de equilíbrio pode ser representado através de cálculos ou de forma gráfica, sendo a maneira gráfica a mais acessível para a compreensão, no qual, um gráfico em linha, representa receitas e despesas. Caso a receita seja a linha dominante, mostra-se uma boa saúde financeira, em contraponto, a situação inversa mostra-se uma má administração financeira.

Dessa maneira, obtendo determinado sucesso de lucros, chega-se ao estágio de poupar dinheiro, tornando o ambiente financeiro pessoal mais seguro para eventuais situações necessárias. Para Halfeld (2007), “a poupança está a maior dor que imediatamente é sentida quando se renuncia ao consumo imediato na esperança de ser recompensado em um futuro ainda muito distante”, esse problema se torna ainda maior em grupos etários mais jovens da população.

No que tange ao planejamento financeiro familiar, é importante levar em consideração sobre os diferentes padrões de comportamento e consumo dos membros familiares. Nesse sentido, é interessante abordar diferentes estratégias, podendo impor limites ou buscar limites (Banco Central do Brasil, 2013).

É comum as pessoas desanimarem ao longo da realização de uma meta, para impossibilitar isso, é importante a criação de objetivos intermediários entre o local onde está até o objetivo que se espera alcançar (Schenini; Bonavita, 2012). Definindo esses objetivos intermediários a propensão de desanimar é menor e as metas de longo prazo se mantêm mais esclarecidas.

2.2 APLICATIVOS MÓVEIS

As aplicações móveis estão nas mãos de praticamente todas as pessoas. Os celulares estão cada vez mais modernos e inteligentes, eles constam com recursos

cada avançados além das funções antigas como efetuar ligação, registrar contatos e mandar mensagens. Uma das principais ferramentas que os *smartphones* tem em seu dispor é a possibilidade de instalar aplicações variadas, popularmente conhecidos como *apps*, algumas vezes capazes de substituir funções nativas do próprio celular. Esses *apps* podem contar para usos diversos como práticas corporativas, assistencial, governamental, entretenimento ou lazer (Cardoso, 2024).

Um aplicativo móvel pode ser desenvolvido em diversas linguagens de programação, cada linguagem pode ter seu próprio *framework*. A utilização de *framework* enfatiza a reutilização de projetos em oposição a reutilização de código, ditando o fluxo da aplicação através de configurações pré-estabelecidas, permitindo o reuso de arquiteturas e funções facilitando a criação de um *software*. É importante citar que a contribuição das comunidades criadas em torno destes projetos, que contribuem compartilhando e disseminando informações, fomentando e expandindo a comunidade e conseqüentemente os produtos e serviços produzidos. Como consequência o desenvolvedor possui um vasto número de ferramentas para auxiliar na sua produção de aplicativos (Rosário, 2015).

O *Flutter* é um SDK de desenvolvimento de aplicativos móveis de código aberto desenvolvido e patrocinado principalmente pelo Google, usado para desenvolver aplicativos para Android e iOS. É escrito nas linguagens C, C++ e Dart e usa Skia Graphics Engine. Ele oferece um rico conjunto de *widgets* personalizados para a criação de interfaces nativas, incluindo bibliotecas *Material Design*, APIs de movimento, rolagem natural suave que ajudam a criar interfaces de usuário rapidamente sem perder o estado em emuladores, simuladores e qualquer *hardware* para iOS e Android (Freitas, 2019).

Para Cardoso (2024) alguns *designs* padrões para a aplicações móveis são: menu de lista ou de abas: onde pode-se organizar opções da ferramenta; *dashboards*: para resumo de parâmetros de desempenho; metáfora: que ajuda o usuário a catalogar e categorizar itens e megamenu: que consiste em um grande painel com personalização de opções.

Decorrente de uma gama de dispositivos móveis de tamanhos de telas diferentes e sistemas operacionais variados, bem como versões, surge a necessidade do desenvolvimento de aplicações multiplataforma, necessitando a cautela sobre questões como responsividade, usabilidade e acessibilidade (Rosário, 2015). Dessa maneira, a experiência do usuário pode ser melhorada através de

componentes pré-definidos que envolvem funcionalidades comuns facilitando a familiaridade da aplicação.

A usabilidade, desta maneira, pode ser definido como “a capacidade de usar um produto com efetividade, eficiência e satisfação em um contexto específico de uso” (Eskelsen, 2023). Para tanto, usa-se alguns critérios para definir o valor de usabilidade de uma aplicação, por exemplo, Dourado e Canedo (2018) trazem os seguintes pontos a serem avaliados: a facilidade de aprender as funcionalidades principais da aplicação; o desempenho e eficiência que o usuário obtém ao executar tarefas na ferramenta; a capacidade de memorização de componentes e funcionalidades; a baixa taxa de erros que o usuário poderá realizar ao usar a aplicação e, enfim, a satisfação do uso do *design* da aplicação.

Para a criação de um aplicativo móvel deve-se racionalizar sobre sua mecânica funcional, escolha de tipografia considerando boa legibilidade das informações, imagens, ícones e cores que versem sobre a identidade visual do produto. Tais elementos devem ser dispostos e organizados de maneira coerente e atraente na interface, visando chamar e prender a atenção pelo maior tempo possível (Cardoso, 2024).

As interfaces, por sua vez, analisam a comunicação por meio da interação entre o ser humano e o computador, com o avanço da tecnologia, essa interação pode ser integrada com aspectos cognitivos e emocionais do usuário. Nos dispositivos móveis existem determinadas peculiaridades nessa integração, cuja composição abrange um conjunto que tornam possível ao usuário ver, ouvir e interagir com as informações, sendo composta por quatro tipos de elementos: não textuais; interativos; *layout* e interpretáveis pelos navegadores (Batista, 2008 *apud* Freitas, 2019).

Visto a boa usabilidade da aplicação, destaca-se, em especial, o uso da interface de voz, em decorrência do seu alto grau de aceitação entre usuários de aplicativos móveis e a facilidade de seu uso.

2.2.1 Interface de Voz

O comando de voz, também chamado de interface por voz — ou VUI — trata-se da tecnologia que permite o usuário interagir com computadores e dispositivos através do uso da voz. Para Bento e Abreu (2023) a *VUI* permite uma usabilidade

mais rápida, fácil e agradável de executar uma determinada tarefa. As interfaces de voz podem ser usadas como assistentes virtuais, ou a opção de falar no lugar da digitação em um teclado físico ou *touchscreen*, tornando a facilidade de uso mais intuitiva e a facilidade de registrar ou executar uma ação mais dinâmica.

A VUI torna as aplicações onde é integrada em ambientes mais intuitivos para o uso humano, pois, ela é projetada para reencenar a sensação de conversas entre o usuário, ajudando a pessoa a completar tarefas ou buscar informações sem usar suas mãos ou mesmo seus olhos. Além disso, a conveniência e as facilidades de uma conversa por voz aprimoram a experiência do usuário, permitindo uma retenção a longo prazo entre o cliente final e a ferramenta (Bento; Abreu, 2023).

Muitas empresas de tecnologia já implementaram interfaces de voz em seus assistentes virtuais, como é o caso da Siri (*Apple*), Alexa (*Amazon*), Bixby (*Samsung*), entre outras. Os usuários podem simplesmente fazer perguntas, como: "Qual é a temperatura agora?". Ou instruir ações específicas, como: "Definir um cronômetro para 30 minutos" ou "Criar um evento no calendário para uma consulta médica às 14h de sexta-feira" (Bento; Abreu, 2023).

2.3 INTELIGÊNCIA ARTIFICIAL

A definição de inteligência artificial é ampla e imprecisa, nota-se um debate acadêmico e científico sobre uma identificação concludente e resoluto sobre sua delimitação.

Historicamente a definição de inteligência artificial é variada, alguns pesquisadores definiram inteligência artificial em termos de fidelidade ao desempenho humano, enquanto outros preferem uma definição abstrata de inteligência, ao qual é caracterizada como "inteligência racional", em que a máquina faz ou toma uma decisão "correta". Outros pesquisadores caracterizam como uma propriedade de processos internos de pensamento e raciocínio ou um comportamento inteligente como característica externa variado pela percepção humana (Russell; Norvig, 2021).

Na definição de Oliveira (2018), "a inteligência artificial é um conjunto de ações que, se fossem realizadas por um ser humano, seriam consideradas inteligentes", trata-se de uma definição que nos obriga a comparar as ações de uma

máquina com as ações de um ser vivo e racional para definir se algum comportamento é inteligente ou não.

Complementa-se, ainda, que a inteligência artificial é a capacidade de os computadores de realizar tarefas que os seres humanos inerentemente realizam melhor até o momento, ou melhor, quando um computador consegue fazer uma tarefa que até então apenas seres humanos são capazes. Este computador, assim, exibe traços de inteligência, e por ser uma entidade fabricada — não existindo espontaneamente na natureza —, mostra-se como artificial (Oliveira, 2018).

Para Russell e Norvig (2021), essa reprodução da cognição humana pode ser definida em três pontos-chaves: A resolução de problemas, o aprendizado e a percepção.

No que tange a resolução de problemas, pode ser entendido como a capacidade de realizar tarefas julgadas corretas por algum critério de avaliação, usando algoritmos e técnicas de programação. No caso de um problema matemático qualquer, a resolução pode ser compreendida como a resposta correta do problema definido através de regras de matemática, caso essa resolução trazida pela máquina se adeque ao consenso científico, a sua definição poderá ser caracterizada como “inteligente” (Oliveira, 2018).

O aprendizado, por outro lado, como definido por Russell e Norvig (2021) é a característica dos algoritmos de melhorar o seu desempenho por meio de experiência. Esse atributo também é chamado de *machine learning*, ou “aprendizado de máquina”, no qual o algoritmo pode ajustar seus parâmetros internos com dados oriundos de execuções anteriores, atingindo resultados melhores em cada execução futura.

Por último, a percepção pode ser referida a dois tipos de fenômenos: físico, no qual, é a capacidade de coletar dados acerca do ambiente a nossa volta; ou mental, que se define como a aptidão de deduzir situações que ocorrem a nossa volta. No caso dos sistemas de um computador, vai desde a leitura de dados por um software até sensores físicos que interagem com o ambiente melhorando a capacidade da máquina de trazer uma informação que se adeque a realidade tangível (Oliveira, 2018).

Dentro de um contexto de finanças pessoais, a inteligência artificial pode ser aplicada sob a ótica da teoria da decisão, para Russell e Norvig (2021) essa teoria se baseia em a combinação da “teoria da probabilidade com a teoria da utilidade,

fornecendo uma estrutura formal e completa para decisões individuais (econômicas ou não) tomadas sob incerteza”, isto é, em casos em que a probabilidade captura o ambiente do tomador de decisão.

Ainda sobre o assunto os Russell e Norvig (2021) trazem a seguinte definição para o uso de inteligência artificial no campo da economia e finanças:

A economia e pesquisa operacional contribuiu muito para a nossa noção de agentes de racionalidade, mesmo assim, durante muitos anos a pesquisa de inteligência artificial se desenvolveu ao longo de caminhos totalmente separados, um dos motivos, foi a aparente complexidade de tomar decisões racionais. O pioneiro pesquisador de inteligência artificial e ganhador do Prêmio Nobel de Economia em 1978 Herbert Simon (1916-2001), trouxe em um dos seus primeiros trabalhos um modelo baseado em satisfação, capaz de tomar decisões que são “boas o suficiente” em vez de calcular laboriosamente uma decisão ideal, assim, dando uma descrição mais precisa sobre o real comportamento humano.

Dessa maneira, com o intuito de se aproximar de uma decisão satisfatória e emular o comportamento humano para a tomada de decisão assertiva, o uso da análise preditiva torna-se essencial. A análise preditiva pode ser definida como a prática de determinados padrões de dados para prever resultados e as tendências futuras, que permite a reação conforme os acontecimentos ocorrem, e que as ações tomadas em reação sejam proativas com relação ao seu próprio futuro (Turban, 2017 *apud* Soares, 2018).

Ressalta-se que o uso de inteligência artificial não se dá com o objetivo de substituir o ser humano, mas para desenvolver de maneira mais ágil processos e transformar o ambiente de trabalho, trazendo inovação, produtividade e resultados, ao mesmo tempo que facilita atividades.

Para a Pesquisa de Inteligência Bancária realizada pela Febraban (2024), o uso da inteligência artificial tem melhorado a eficiência dos processos bancários, aumentando índices de produtividade e performance dos clientes, trazendo soluções inovativas, impacto na cultura organizacional, bem-estar, crescimento pessoal e desenvolvimento de habilidades, função essa que se mostra essencial para a aplicação que será desenvolvida.

3 METODOLOGIA DE PESQUISA

A presente pesquisa é caracterizada como quantitativa em relação a abordagem e exploratório quanto a seus objetivos, com amostragem composta por potenciais usuários no intuito de obter dados que reforcem os objetivos propostos. Trata-se da etapa da pesquisa científica que busca compreender e resolver problemas plurais fazendo uso de procedimento técnicos e metodológicos com propósito de obter respostas para questões hipoteticamente levantadas no aparecimento de uma suspeita ou de um problema (Rodrigues; Oliveira; Santos, 2021).

Na visão de Knechtel (2024), a pesquisa quantitativa:

É uma modalidade de pesquisa sobre um problema, humano ou social, baseada na testagem de uma teoria composta de variáveis, quantificadas em números e analisadas por meio de procedimentos estatísticos, de forma a determinar se as generalizações previstas na teoria se sustentam ou não.

O uso da abordagem quantitativa se justifica no interesse por analisar, dimensionar e avaliar a aplicabilidade de recursos ou técnicas alicerçada na quantidade de dados coletados durante uma investigação, com objetivo de medir opiniões e informações, fazendo uso de recursos da estatística e seus elementos de demonstração de porcentagem, a média e o desvio-padrão (Rodrigues; Oliveira; Santos, 2021), somando-se a isso o uso de elementos visuais como tabelas, gráficos ou textos.

As informações foram obtidas utilizando-se um questionário estruturado, realizado durante o período de dez de agosto de 2024 até dezoito de agosto de 2024, com 183 pessoas distintas, na plataforma *Google Forms* (2024).

O questionário propôs identificar os seguintes tópicos: idade; frequência do monitoramento das finanças pessoais; a centralização do monitoramento dessas finanças; o uso de alguma ferramenta de monitoramento, — se positivo — a descrição de qual ferramenta usada; os principais desafios para a gestão das finanças pessoais; a necessidade de criar metas financeiras; a utilidade do uso de inteligência artificial para o apoio da conquista de metas financeiras; a utilidade de notificações sobre gestão das finanças pessoais; a necessidade do uso de comando de texto e voz para o contexto de interface de usuário; a utilidade de relatórios em

PDF ou *Excel*; a importância da intuitividade do uso da ferramenta; o desejo de alguma funcionalidade ou característica específica; e, por fim, a potencial disposição para o pagamento de uso da ferramenta.

Tais perguntas realizadas no questionário foram elaboradas com o objetivo de validar a estrutura do *software* proposto no objetivo geral desta obra, os resultados da pesquisa podem ser verificados no Apêndice A e se mostram da seguinte forma:

A primeira questão (gráfico 1) refere-se sobre o padrão de comportamento referente à frequência de monitoramento das finanças pessoais, obtendo-se os seguintes resultados: monitora as finanças pessoais semanalmente: 40,1%; monitora as finanças pessoais diariamente: 30,2%; monitora as finanças pessoais mensalmente: 24,7%; monitora as finanças pessoais raramente: 3,8%; e, nunca monitora as finanças pessoais: 1,1%.

A segunda questão (gráfico 2) refere-se à utilidade de uma visão geral de gastos e recebimentos em um único lugar, obtendo-se os seguintes resultados: 98,4% afirmaram que acham útil tal funcionalidade e apenas 1,6% não acham útil.

A terceira questão (gráfico 3) refere-se ao uso de alguma ferramenta ou aplicativo de gestão financeira, obtendo-se os seguintes resultados: 67,6% afirmam que não usam nenhuma ferramenta e 32,4% afirmam que usam algum tipo de ferramenta de controle. A quarta questão (gráfico 4) é um complemento a quarta questão, se positivo, quais ferramentas usam, nos quais, as respostas mais frequentes foram: planilhas ou ferramentas de BI obtiveram a maior parte de respostas, seguidos por aplicativos bancários e aplicativos de finanças.

A quinta questão (gráfico 5) refere-se aos principais desafios para a gestão de finanças pessoais, nos quais, as respostas mais frequentes foram: falta de controle e organização para o controle financeiro, além de desafio para equilibrar gastos, seguido de outras respostas como salários baixos, dificuldade de lembrar datas para pagamentos e inflação.

A sexta questão (gráfico 6) refere-se à criação e acompanhamento de metas financeiras, nos quais, 92,9% das respostas foram positivas, em contraponto com os 7,1% de respostas negativas em relação a essa funcionalidade.

A sétima questão (gráfico 7) refere-se à utilização de inteligência artificial para a análise de hábitos de consumo e sugestões personalizadas, nos quais, 94,4% das respostas foram positivas, em contraponto com os 5,6% de respostas negativas em relação a essa funcionalidade.

A oitava questão (gráfico 8) refere-se aos alertas e notificações personalizadas sobre limite de gastos, vencimento de contas e recebimentos, nos quais, 94% das respostas foram positivas, em contraponto com os 6% de respostas negativas em relação a essa funcionalidade.

Ressalta-se que, na sexta questão, alguns potenciais usuários trouxeram a falta de controle e a dificuldade de lembrar o pagamento de despesas como um desafio para o controle de gastos. Desta forma, mostra-se validado o recebimento de alertas e notificações como uma função essencial para o desenvolvimento da aplicação.

A nona questão (gráfico 9) refere-se à possibilidade do uso de comando de voz para o registro de despesas e receitas, nos quais, 82,5% das respostas foram positivas, em contraponto com os 17,5% de respostas negativas em relação a essa funcionalidade.

A décima primeira (gráfico 10) questão refere-se à possibilidade de exportar relatórios detalhados de despesas e receitas em formato *PDF* ou *Excel*, nos quais, 86,8% das respostas foram positivas, em contraponto com os 13,2% de respostas negativas em relação a essa funcionalidade.

A pergunta acima exposta se assemelha com a resposta de potenciais usuários obtidos na quinta questão, na qual, o maior número de pessoas afirmou que usam ativamente planilhas como uma forma de controle de gastos e despesas financeiras, desta maneira, mostra-se interessante trazer a exportação de planilha composta com gastos e despesas do usuário como uma funcionalidade da aplicação que será desenvolvida.

A décima primeira questão (gráfico 11) refere-se à intuitividade e facilidade de uso da aplicação, obtendo-se os seguintes resultados: 73,6% acham muito importante, 24,2% acham importante, 1,6% acham pouco importante, e 0,5% não acha importante.

A décima segunda questão (gráfico 12) refere-se a funcionalidades e características interessantes de serem adicionadas ao aplicativo, nos quais, as respostas mais frequentes foram: gerenciar investimentos, além de, mais uma vez, a importância da usabilidade da plataforma. Outras questões como gráficos, relatórios e o uso da inteligência artificial também foram registradas nessa questão, complementando e reforçando algumas questões anteriores aplicadas.

Por último, a décima terceira questão (gráfico 13) refere-se à possibilidade do pagamento para o uso de uma aplicação com as funcionalidades apresentadas, obtendo-se os seguintes resultados: preferem usar de maneira gratuita com anúncios: 43,9%; pagaria até R\$ 10,00 (dez reais) por mês: 28,3%; pagaria entre R\$ 10,00 (dez reais) até R\$ 30,00 (trinta reais) por mês: 21,1%; pagaria entre R\$ 30,00 (trinta reais) até R\$ 50,00 (cinquenta reais) por mês: 6,1%; e, pagaria mais de R\$ 50,00 (cinquenta reais) por mês: 0,6%.

3.1 METODOLOGIA DA CRIAÇÃO DO PRODUTO

Baseado nas respostas obtidas durante a fase do questionário de pesquisa, obteve-se uma estrutura para o desenvolvimento do aplicativo *mobile* que atenda os objetivos gerais e específicos propostos.

3.1.1 Metodologia ágil

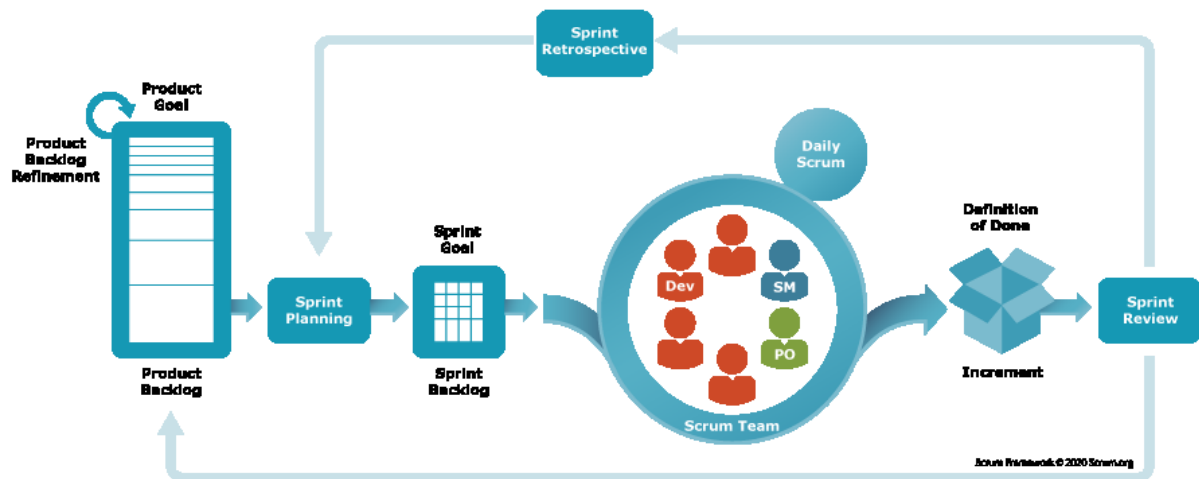
A metodologia ágil é um conjunto de ideais e princípios que servem para automatizar e integrar processos entre equipes de desenvolvimento de *software* e de operações, usa uma abordagem estruturada e iterativa para gestão de projetos e desenvolvimento de produtos, no qual, reconhece a volatilidade do desenvolvimento de produtos e proporciona uma metodologia para equipes auto-organizadas responderem a mudanças sem perder o fluxo ou lógica interna (Rehkopf, 2024). Dentre os *frameworks* mais populares das metodologias ágeis destaca-se o *Scrum* e o *Kanban*.

O *Scrum* emprega uma abordagem iterativa e incremental para otimizar a previsibilidade e controlar os riscos, envolvendo um grupo de pessoas que coletivamente tem todas as aptidões e conhecimentos para realizar as tarefas necessárias. Esse grupo geralmente é dividido em três responsabilidades sem subequipes ou hierarquia: *Scrum Master* — trata-se de um facilitador das práticas do *Scrum* que atua diretamente com o *Product Owner* para garantir qualidade e eficiência nos processos e remover qualquer tipo de impedimentos que surjam ao longo do projeto —, *Product Owner* — também conhecido pela sigla “P.O”, é a pessoa responsável pelos requisitos da empresa, transformando ideias em um

produto tangível — e desenvolvedores, responsáveis pela construção técnica da ferramenta (Mendes, 2022).

O *Scrum* tem como pilares os conceitos de transparência, inspeção e adaptação, trata-se de um processo empírico para a tomada de decisão baseada na observação, experiência e experimentação. Esse processo ocorre durante a *sprint*, que consiste em quatro eventos: *Sprint Planning*, *Daily Scrum*, *Sprint Review* e *Sprint Retrospective*, que funcionam de maneira organizada e continua conforme figura abaixo:

Figura 1 - Estrutura de uma *Sprint Planning*



Fonte: Scrum (2024).

Por outro lado, o *Kanban* é um método para a gestão de trabalho e conhecimento, aplicado através de princípios e práticas, utilizando *boards* para visualizar o fluxo de trabalho, limitações do trabalho em processo e gerenciamento de fluxo para tornar políticas explícitas, com *feedbacks* constante para a melhora colaborativa e evolução (Mendes, 2022).

Embora ambas as metodologias tenham diferenças práticas, os princípios em sua maior parte se mantêm uníssonos, podendo ser utilizadas em conjunto, essa abordagem mista das duas práticas é popularmente conhecida pelo termo "*Scrumban*". O uso híbrido das duas ferramentas já é adotado por diversas equipes de gerenciamento de projeto, permitindo maior liberdade para a implementação de medidas personalizadas que possam funcionar em determinados projetos de

desenvolvimento de produtos (Rehkopf, 2024). No caso do desenvolvimento do projeto do aplicativo Monetar.ia, o *scrumban* será a metodologia de gerenciamento do trabalho, com aplicação de *Sprints Plannings* semanais, com reuniões de grupo aos domingos para realização de *Sprint Review* e *Sprint Retrospective*.

3.2 ARQUITETURA

A arquitetura de *software* pode ser definida como o conjunto de elementos arquiteturais — sejam elas de dados, de processamento ou de conexão — que possuem alguma organização ou lógica interna, que foram tomadas para satisfazer objetivos e restrições que se agregam respeitando a regra de negócio que refere o desenvolvimento de um sistema (Perry; Wolf, 1992 *apud* Silva, 2016).

Dentre os diversos padrões de arquitetura de *software* pode-se destacar o sistema monolítico pois é um dos exemplos mais comuns usados para o desenvolvimento de aplicações. Esse tipo de arquitetura, que também pode ganhar a nomenclatura de sistema distribuídos simples, se baseia na implantação do código em um único processo. Eventualmente, o sistema monolítico também pode ser aplicado em dois processos acoplados ou através de sua variação, baseado no desenvolvimento de subconjuntos de processos que trabalham de maneira independente, essa variação ganha o nome de sistema monolítico modular (Newman, 2022, p. 37).

Por mais que a arquitetura monolítica possa fazer sentido em algumas aplicações ou regras de negócio, ou pela sua facilidade de implementação, que se resume a apenas um *deploy*, e da sua manutenção facilitada, esse padrão arquitetural não é recomendado para aplicações grandes e complexas visto que para atualizar o sistema para novas funcionalidades que possam surgir no futuro, seria preciso atualizar o sistema por completo, consumindo tempo demais e oferecendo risco para o funcionamento perfeito do sistema (Conceição; Pinto, 2021).

Outra vantagem é a capacidade de reutilização e simplificação de código dentro da própria aplicação. Em um sistema distribuído, algumas decisões devem ser tomadas sobre a cópia do código, a separação em bibliotecas ou a passagem de funcionalidades compartilhadas para um serviço. Porém, à medida que mais pessoas trabalham em um único código monolítico, contradições e *bugs* podem ser geradas, além da confusão relacionada a quais partes do código são

responsabilidade de cada desenvolvedor, resultando em responsabilidades confusas para os administradores do projeto. Para Newman (2022, p. 40), esse problema pode ser chamado de “desavença nas entregas” (*delivery contention*).

A presença da arquitetura monolítica é maior em sistemas legados, sendo que em vários casos uma migração é vista como uma tarefa extremamente complexa. Seu uso é envolto em problemas de acoplamento, manutenção e complexidade, ainda assim, é uma arquitetura indicada para sistemas pequenos e início de projetos de software, onde o escopo não é totalmente previsto pela equipe de desenvolvimento (Pereira; Diniz, 2022).

Cada vez mais o sistema monolítico entra em desuso por equipes de desenvolvimento, e no seu lugar a arquitetura de microsserviços ganha voga. Esse tipo de arquitetura se baseia na orientação a serviços onde, um serviço encapsula uma funcionalidade e a torna acessível a outros serviços através das redes, possibilitando a criação de um sistema mais complexo a partir de blocos de construção (Newman, 2022, p. 24).

Os microsserviços são propulsores de inovação tecnológica, permitindo que sejam desenvolvidos serviços pequenos, leves e de rápida construção e aplicação. Além disso, garante segurança para, caso um serviço individual falhar, o resto da aplicação continuará funcionando normalmente (Conceição; Pinto, 2021).

De forma técnica, os microsserviços adotam o conceito de ocultação de informações, no qual, oculta-se o máximo possível de informações dentro de um componente e expõe o mínimo possível por meio de interfaces externas, permitindo uma clara separação entre o que pode ser alterado com facilidade e o que é mais difícil — ou até mesmo impossível — de ser alterado. Esse aspecto é fundamental para que se possa trabalhar em microsserviços de forma isolada, resultando em componentes coesos (Newman, 2022, p. 26).

Dessa maneira, outro conceito possível para a arquitetura de microsserviços é as funcionalidades de uma aplicação decomposta em funções menores e independentes residindo em recursos próprios de infraestrutura. Uma característica básica de um ambiente de microsserviços é a replicabilidade, ou seja, um determinado microsserviço pode ser facilmente replicado quantas vezes forem necessárias para alcançar escalabilidade ou simplesmente reutilização, o que implica ao microsserviço a incorporação da lógica da função e o ambiente de provisionamento (Pereira; Diniz, 2022).

Outro conceito essencial para a arquitetura de microsserviços é a possibilidade de implantações independentes, trata-se da ideia de que alterações podem ser feitas afetar a eficiência da aplicação para o usuário final. Para garantir implantações independentes necessita de uma característica de baixo acoplamento, no qual, convenções explícitas, bem definidas e estáveis entre os serviços (Newman, 2022, p. 28).

Essa divisão granular dos componentes lógicos de uma aplicação pode gerar responsabilidades coesas, baixo acoplamento e o uso de comunicações padronizadas entre os componentes. Visando assim, uma maior taxa de atualizações dos componentes lógicos da aplicação, em razão de os componentes não terem efeitos ou dependências em outros componentes fora do escopo da aplicação (Pereira; Diniz, 2022).

A modelagem com base em um domínio de negócio é o próximo conceito dos microsserviços, técnicas como o DDD (*Domain-Driven Design* ou *Design Orientado a Domínios*), permite estruturar o código de modo a facilitar o lançamento de novas funcionalidades e recombinar serviços de maneira distinta, desta forma permitindo uma maior escalabilidade do projeto no espaço de tempo da maneira mais eficiente possível (Newman, 2022, p. 29).

Assim, o *software* é construído dividindo cada funcionalidade em módulos com especialidades, permitindo uma maior preocupação com as estruturas do negócio em si, para dividir cada parcela de serviço em um módulo desacoplado e isolado da aplicação, e então integrar os vários serviços para compor a aplicação em sua totalidade (Pereira; Diniz, 2022).

Os microsserviços devem evitar o uso de banco de dados compartilhados, se um microsserviço solicitar dados para outro, deve-se solicitar esse dado, dotando a capacidade de decidir o que será compartilhado e o que se manterá oculto, permitindo fazer distinção clara de quais funcionalidades podem ser alteradas livremente. Permitindo uma lógica geral de que o microsserviço é uma fatia completa de funcionalidades do negócio, as quais, quando apropriado, encapsulam a interface de usuário (UI), a lógica de negócios e os dados armazenados (Newman, 2022, p. 30).

Dessa forma, a arquitetura de microsserviços se destaca em sistemas grandes e complexos. A divisão dos vários serviços em menores e granulares permite uma manutenção mais simples e tangível, muito difícil de ser aplicada em

blocos monolíticos de grande tamanho. Além disso, permite uma maior capacidade de inovação nas aplicações em que é utilizada, visto que a independência dos serviços permite vários ciclos de desenvolvimento simultâneos de novas funcionalidades (Pereira; Diniz, 2022).

Referente ao seu tamanho e flexibilidade, os microsserviços devem se preocupar em ter a menor interface possível, alinhando-se ao conceito de ocultar-se informações, não podendo ser extremamente complexos e se preocupando com a flexibilidade resultante no geral no sistema, devendo ser organizado, técnico, propenso a escalabilidade e a robustez (Newman, 2022, p. 32).

O ciclo de desenvolvimento de projeto que se propõe a ser em microsserviços deverá ser realizado em etapas, para Asher (2019 *apud* Conceição; Pinto, 2021), esse ciclo ocorrerá em oito etapas: o conceito, no qual, deverá ser alinhado requisitos de mercado, documentando as informações coletadas em alguma ferramenta — ou algum *framework* de metodologia como o *Scrumban* — para gerenciamento do projeto; as especificações de API, no qual se propõe um *design* que se adequem as especificações; seguido pela própria modelagem da API desenvolvida pelos programadores do projeto; o protótipo do serviço que possa ser validado com possíveis usuário; o lançamento de versões *Alpha* e *Beta*, com intuito de encontrar problemas e *bugs* no sistema; a aplicação de testes para prevenir erros, latências e vulnerabilidades de segurança; a *release* de atualizações ou melhorias com garantias de versionamento do sistema; e por fim, a finalização do projeto, no qual o sistema se encontra em perfeito funcionamento.

3.2.1 API Rest

O *Representational State of Transfer* (Transferência de Representação de Estado), representado pela sigla REST, pode ser definida com um estilo arquitetural para o desenvolvimento de serviços dentro de um sistema, ou, pelas palavras do próprio criador do REST, Roy T. Fielding (2000 *apud* Campos, 2013): “Esse tipo de arquitetura não é um padrão e sim um estilo híbrido de outras arquiteturas de rede que são combinadas com algumas restrições adicionais definindo uma interface de conexão única”.

Alguns princípios básicos do REST: funcionalidade de comunicação em interface uniforme, no qual, adota-se o protocolo HTTP que garante o tráfego íntegro

pela rede, nos quais são usados os métodos: OPTIONS, GET, HEAD, POST, PUT, DELETE e TRACE, onde cada método opera com apenas um recurso, não mudando seu comportamento nem sua sintaxe independentemente da aplicação usada; múltiplas representações: onde a representação é o encapsulamento da informação em uma codificação específica, seja ela XML, JSON ou HTML; recursos *stateless*: que está associado ao estado em que alguns recursos se encontram no servidor e a interação da aplicação com esse recurso, geralmente, o servidor considera cada requisição de maneira isolada, com objetivo de interpretar e entregar da maneira mais íntegra possível; sistema em camadas: onde, dentro da arquitetura do REST, os serviços devem ser organizados de maneira hierárquica, sendo divididos em componentes que possuem funções únicas e bem definidas facilitando a comunicação entre as partes; e, por último, o mecanismo do REST em identificar se as informações de resposta de um servidor devem ou não serem armazenadas para posteriores utilizações, eliminando a necessidade de carregamento de um mesmo conteúdo (Campos, 2013).

Para a implementação de uma API, devem-se levar em consideração alguns princípios de *design* e boas práticas de origem da arquitetura REST, como por exemplo: o estilo de arquitetura deve ter foco no tipo *Pragmatic*, proporcionando um conjunto de funcionalidades de fácil aprendizado, auxílio no consumo de serviços e suporte para expansão de outras tecnologias, aumentando a produtividade e sucesso da API; a comunicação padronizada entre elementos será realizada através do formato JSON, o uso OAuth, que fornece um mecanismo de autenticação e autorização componentes de segurança, além do uso do Swagger para a documentação automática das funções implementadas no *back-end* da aplicação (Campos, 2013).

Seus benefícios mais perceptíveis são a separação de cliente e servidor aprimorando a experiência do usuário que consome a API, escalabilidade da aplicação resultando em uma maior agilidade de implementação de soluções inovadoras destacando a competitividade do recurso, a acessibilidade facilitada aos contratos da aplicação, a segurança e confiabilidade dos componentes (Feltrin, 2023), tornando-se a escolha para o desenvolvimento da API que funcionará baseado em uma arquitetura de microsserviços.

3.3 FERRAMENTAS

Referente ao escopo do projeto e a arquitetura escolhida para estruturar a aplicação, as seguintes ferramentas foram selecionadas para o desenvolvimento do produto.

3.3.1 Notion

O Notion é uma ferramenta de criação de documentos em conjunto voltado para facilitar a colaboração e organização de equipes. Trata-se de uma ferramenta que possibilita a criação de documentos, permitindo edições e comentários em tempo real. Além disso, sua plataforma armazena diversos tipos de documentos como trechos de código, imagens e vídeos, permite a criação de *charts* e gráficos para acompanhamento de métricas de projeto, criação de *boards Kanban* que contribui para o acompanhamento visual das tarefas, organização do progresso e priorização de atividades. Além disso, o Notion permite de maneira nativa a conexão com outras ferramentas como o Figma e o Github tornando-o ideal para a aplicação em projetos de tecnologia (Notion, 2024). Desta maneira, para o gerenciamento do *board kanban* e das documentações criadas durante a fase de desenvolvimento do produto, o Notion aparenta ser uma ferramenta que atende as expectativas da equipe, oferecendo uma plataforma integrada, intuitiva e segura, facilitando a colaboração e o acompanhamento das tarefas em tempo real.

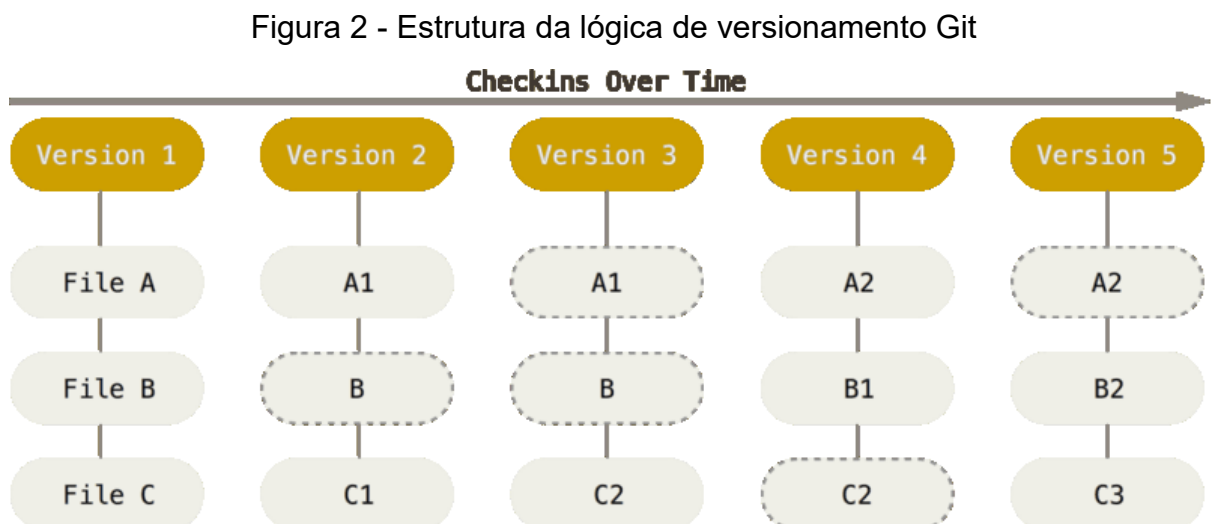
3.3.2 Git

O Git é um *software* de controle de versionamento de documentos, sua principal característica é a maneira como armazena dados, a maior parte dos softwares que se propõem ao controle de versões armazena dados em uma lista de alterações baseadas em um conjunto de arquivos ao longo do tempo, comumente descrito como controle de versão baseado em *delta*. O Git, por outro lado, não salva um novo arquivo em lista, mas sim as modificações realizadas no arquivo base e a referência do arquivo idêntico armazenado na versão anterior (Chacon; Straub, 2024).

O controle de versionamento, durante o processo de desenvolvimento do produto, será realizado com o uso dessa ferramenta, que torna possível o trabalho conjunto de diferentes desenvolvedores em um único diretório — chamado de *main* —, permitindo realizar alterações no código, registro de alterações aprovadas ou reprovadas e o armazenamento seguro dessas informações (Palestino, 2015).

O Git possibilita a criação de espaços separados para testes ou desenvolvimento de projetos, além do controle de alterações do código, que podem ser aprovadas ou negadas por meio do *code review*. Um diferencial em relação a maioria dos sistemas de controle de versão é a possibilidade de criação de *snapshots*, tratando as informações e mantendo-as como um conjunto de arquivos organizado de maneira temporal. Para tanto, existe a possibilidade de criar várias ramificações de um mesmo projeto, chamadas de *branches*, que consistem em cópias do estado do projeto em determinado momento, facilitando a funcionalidade e coerência da aplicação (Freitas, 2017).

Na imagem abaixo, podemos entender através de um esquema visual como funciona a alteração de diferentes arquivos por meio do versionamento organizado pelo Git:



Fonte: Chacon e Straub (2024).

O desenvolvedor poderá realizar um *pull* do repositório remoto para o local onde esteja realizando as modificações. Após o *code review* positivo das alterações realizadas é possível realizar um *commit* da *Branch* para o *main*, realizando o

processo de *merge* que deverá ocorrer automaticamente quando não houver conflitos entre as diferentes versões do documento (Freitas, 2017).

Em razão do consenso do uso do Git entre os desenvolvedores e gerentes de projeto como uma ferramenta essencial para o controle de versão do código, a equipe de projeto adotou seu uso durante a fase de criação do produto, facilitando a resolução de conflitos durante o processo de criação, além de melhorar o trabalho colaborativo entre os membros da equipe.

3.3.3 GitHub

O GitHub é uma ferramenta de versionamento coletivo *opensource* com armazenamento em nuvem, onde é possível configurar grupos de contas de usuários de uma única organização ou projeto, com níveis de permissionamentos diferentes para desenvolvedores distintos e sincronização em tempo real. Com a criação de um repositório de um projeto, seus membros podem configurar métricas, detalhar problemas, criar relatórios sobre o progresso do projeto, planejar tarefas ou adicionar anotações ou informações necessárias (GitHub, 2024). Visto que o Git será usado para o versionamento de código, a escolha do GitHub é um processo natural decorrente da escolha do grupo.

3.3.4 Figma

O Figma é uma ferramenta de *design* usada para a criação de protótipos ou de interfaces de usuário. Apresenta uma funcionalidade de produção colaborativa em tempo real, onde poderá criados *frames* conectados a componentes onde será simulado interações do aplicativo ao qual se deseja criar. Além disso, apresenta funcionalidades de anotação de *feedbacks*, interação entre usuários e compartilhamento de projetos. Além disso, também existe a disponibilidade de exportar arquivos pelo modo desenvolvedor, onde o código das interfaces é gerado automaticamente (Figma, 2024).

Baseado na facilidade de uso e compartilhamento ágil de protótipos, aliados com a baixa curva de aprendizado para o desenvolvimento de interfaces intuitivas, a equipe adotou o uso do Figma como ferramenta principal para a criação do protótipo

do produto. Sua plataforma permite maior agilidade para o processo de validação do *design*, permitindo maior eficiência e alinhamento entre os membros do projeto.

3.3.5 Visual Studio Code

O Visual Studio Code, ou apenas VS Code, é um editor de código-fonte *open source*, multiplataforma e gratuito desenvolvido pela Microsoft, altamente popular entre desenvolvedores. Entre seus recursos, oferece a personalização total da interface de usuário, a conexão com o versionamento de código através do Git, terminal integrado na plataforma, a capacidade de executar, depurar ou compilar o código, além da possibilidade de criação de máquinas virtuais. Ademais, o Visual Studio Code oferece suporte para as principais linguagens de programação, além de disponibilizar um vasto número de extensões para adicionar recursos e tornar o ambiente de desenvolvimento mais funcional (Visual Studio Code, 2024).

3.4 DESENVOLVIMENTO MOBILE

Para o desenvolvimento da aplicação no ambiente móvel, foram selecionadas as seguintes tecnologias especializadas, definida por sua eficiência, compatibilidade e robustez para garantir a melhor experiência ao usuário.

3.4.1 Flutter

O Flutter é uma estrutura de código aberto para criar, testar e implantar interfaces de aplicativos móveis, *web* e *desktop* incorporados a partir de uma única base de código. Apresenta algumas vantagens como a compilação rápida, atualização de modificações dinâmica e a criação de *designs* adaptáveis para qualquer tamanho de tela (Flutter, 2024).

Um dos diferenciais notáveis do Flutter é a utilização de *widgets* para a elaboração do *layout* de telas. Os *widgets* podem ser considerados elementos que funcionam como blocos de construção básicos da interface de usuário, sendo uma declaração de tudo o que será desenhado em tela, sendo eles, a estrutura, a aparência, o conteúdo, posição e comportamento. Apresentam a característica de serem *stateless* e *stateful*, que correspondem ao estado do objeto. O *stateful* é

geralmente usado para o cumprimento de atividades assíncronas ou para situações que o estado vá mudar em algum momento a lógica do *widget*, e o *stateless* não muda seu estado em nenhuma situação (Gradin, 2019).

3.4.2 Dart

O Dart é uma linguagem desenvolvida e mantida pela Google com sintaxe familiar ao Java. O Flutter e todas as suas aplicações elaboradas são escritas e funcionam em Dart (Gradin, 2019).

Trata-se de uma linguagem acessível e extremamente produtiva para aplicativos de alta qualidade. Apresenta características que facilitam o desenvolvimento, como, por exemplo, o *hot reload* de alterações do código-fonte, que permite a visualização instantânea da funcionalidade aplicada; tipos flexíveis de análise estática; compilador nativo para inicialização instantânea da aplicação e ferramentas poderosas para auxiliar o desenvolvedor (Dart, 2024).

Considerando que a equipe escolheu o uso do Flutter para a estruturação de *layout* de telas da aplicação móvel, o uso do Dart mostra-se uma escolha natural para ser aplicada em conjunto na fase de desenvolvimento.

3.4.3 PostgreSQL

O *PostgreSQL* é um sistema de banco de dados objeto-relacional escalável de código aberto e gratuito, que usa a linguagem de programação *SQL* combinada com alguns recursos de armazenamento, dimensionamento e segurança de dados. O *PostgreSQL* ajuda desenvolvedores a criar aplicações, proteger a integridade dos dados e criar ambientes tolerantes a falhas, além do gerenciamento dos dados armazenados no banco de dados (Postgresql, 2024).

Um dos pontos de destaque para o uso do *PostgreSQL* é o conjunto de procedimentos para preservação de integridade de transações em um banco de dados relacional, essa característica é chamada de "ACID" definida pelas propriedades de atomicidade, consistência, isolamento e durabilidade. Em resumo, a propriedade "ACID" garante que as transações devem ser concluídas com sucesso para serem consideradas válidas, assegurando que todas as ações de uma transação sejam executadas ou que nenhuma seja, em caso de falha, um *rollback*

retorna para o estado de pré execução da operação; regras e restrições estabelecidas devem ser cumpridas garantindo a passagem de um estado consistente para outro; uma transação não deve ser interferida por outra; e por fim, os resultados de uma transação são permanentes abonando que um dado não poderá ser perdido (Silva, 2023).

Relacionado ao consumo de API e a aplicação de microsserviços, a atomicidade tem protagonismo, visto que, a falta de atomicidade pode resultar em problemas significativos, por exemplo, nas operações de serviços que envolvem banco de dados a garantia da execução entre múltiplas funcionalidades precisa ser assegurada, pois, a falha nessa execução acarretará o cancelamento total da operação, podendo gerar um resultado não esperado pelos desenvolvedores (Newman, 2022, p. 216).

O PostgreSQL está em conformidade com o padrão exigido pelo SQL, mesmo que em alguns casos a sintaxe ou função seja ligeiramente diferente. O PostgreSQL tem pelo menos 170 dos 177 dos recursos obrigatórios para estar em conformidade com o SQL:2023 Core, sendo o sistema de gerenciamento de banco de dados relacional disponível em mercado que está em maior conformidade com este padrão (Postgresql, 2024).

3.5 API

Com o proposito de tornar a aplicação móvel mais robusta e dotada de escalabilidade, torna-se necessário o desenvolvimento de uma API que usará as seguintes ferramentas para seu desenvolvimento.

3.5.1 Python

O Python é uma linguagem de programação de baixa curva de aprendizado, com comunidade de desenvolvedores ativa, com uma série de bibliotecas, módulos criados por terceiros e *frameworks* exclusivos para a linguagem, além de sua característica de código aberto (Python, 2024).

Um dos *frameworks* mais modernos da linguagem Python é o FastAPI, exclusivo para a construção de APIs. Alguns dos pontos fortes que reforçam seu uso são seu alto desempenho, sua facilidade de uso, a baixa probabilidade de *bugs*

resultado seus recursos para declaração de parâmetros e sua robustez (Fastapi, 2024).

O FastAPI utiliza recursos como tipagem estática para fornecer um ambiente de desenvolvimento mais eficiente e seguro, ademais, oferece suporte nativo para chamadas assíncronas, possibilitando a construção de APIs rápidas e escaláveis. Além disso, tem a capacidade de geara automaticamente documentação interativa para a API, incluindo exemplos de uso, tipo de dados esperados e até mesmo a possibilidade de testar rotas diretamente na documentação (Habbema, 2023).

3.5.2 Digital Ocean

A *Digital Ocean* é uma plataforma de nuvem em IaaS (infraestrutura como serviço) e PaaS (plataforma como serviço) que disponibiliza a integração de variadas ferramentas necessárias para a implementação de uma API, como, por exemplo: criação de máquinas virtuais simples e escaláveis para hospedagem web e VPS, gerenciamento de banco de dados e armazenamento em nuvem por meio de *block storage*. Mostra-se uma ferramenta de simples e de fácil aprendizado e com custos previsíveis para o desenvolvimento da API (Digital Ocean, 2024).

3.5.3 CapRover

O *CapRover* é um gerenciador de implantação de aplicativos em PaaS (plataforma como serviço). É uma ferramenta que facilita a configuração e manutenção de um servidor, simplifica o processo de *deploy*, integrando diretamente ao repositório de projeto no *Github*, realiza espontaneamente a obtenção de certificado SSL para oferecer criptografia automática. Além disso, o CapRover apresenta integração com containers Docker para hospedar aplicações ou banco de dados e realiza o gerenciamento do Nginx, atualizando repetidamente de maneira automática sem a necessidade de lidar com *scripts*. Seu uso desponta pelo desembaraço da aplicação de múltiplas ferramentas, seu caráter *open source*, pela escalabilidade com base na demanda, permitindo adição de instâncias conforme necessário e pelo poder de gerenciamento de múltiplos servidores com recuperação automática em caso de falhas (CapRover, 2024).

3.5.4 Docker

O Docker é uma plataforma *open source* responsável por automatizar a implantação de aplicações dentro de um ambiente isolado chamado de containers. Desta forma, desenvolvedores e administradores podem embarcar, integrar e executar aplicações rapidamente, proporcionando múltiplos ambientes isolados dentro de um mesmo servidor, mas acessíveis externamente através da tradução de portas (Docker, 2024).

Uma das características principais para o uso do Docker é a sua escalabilidade, podendo ajustar recursos de forma inteligente para atender quantidades variáveis de volume de dados ou usuários, isso é possível através do conceito de orquestração, feito especificamente para automatizar tarefas de aplicações containerizadas, como a instalação, escalonamento, balanceamento de carga, monitoramento e provisionamento de recursos, em resumo, serve primariamente para gerenciar o ciclo de vida dos containers (Pelegriani; Corino; Silva, 2022).

Os containers, por sua vez, é um agrupamento de uma aplicação junto com suas dependências, que compartilham o *kernel* do sistema operacional do *host*, proporcionando melhor desempenho por conta do gerenciamento único dos recursos. Apesar de manifestar algumas semelhanças com a estrutura de um ambiente virtualizado, o Docker não necessita de uma camada intermediária de sistema operacional entre o hospedeiro e as aplicações hospedadas, utilizando o mesmo *kernel* criando ambientes isolados a nível de disco, memória e processamento (Gomes; Souza, 2015).

Ademais, o balanceamento de carga distribui o tráfego de forma inteligente para servidores e responder de volta para o usuário, garantindo maior estabilidade a uma aplicação, já que haverá múltiplos servidores processando requisições, isso ocorre buscando equalizar a utilização de *CPU* de cada servidor, não sobrecarregando máquinas específicas (Pelegriani; Corino; Silva, 2022).

3.5.5 Nginx

O Nginx é um servidor web HTTP de *proxy reverso* com cache de conteúdo e balanceador de carga, que trabalha em conjunto com o Docker. Faz uso de

servidores virtuais baseados em nome e IP que possibilitam o redirecionamento de códigos de erro, controle de acesso balanceado, controle e limitador de acesso, entre outros recursos. Sua arquitetura é flexível para configurações e atualizações sem a interrupção do atendimento ao usuário (Nginx, 2024).

3.6 DESENVOLVIMENTO DA LANDING PAGE

Com objetivo de divulgar as funcionalidades da aplicação que se propõe ser desenvolvida, será criado uma *Landing Page* que funcionará em conjunto com a API da Monetar.ia. Para tanto será necessário criar uma *webpage* usando as seguintes ferramentas de desenvolvimento web.

3.6.1 HTML

O HTML é a sigla para *Hypertext Markup Language*, ou, na tradução direta, Linguagem de Marcação de Hipertexto. Trata-se de uma linguagem baseada em marcação, onde os elementos são marcados, através de *tags*, para mostrar quais informações a página exibe (Ozelin, 2017).

A linguagem HTML pertence a uma classe de linguagens de programação conhecida como "*Tags Languages*", ou simplesmente Linguagem de Marcação. Neste tipo de linguagem, os comandos são escritos em forma de marcação denominadas *tags*. Essas *tags* aparecem em pares, delimitando o texto a ser formatado. A *tag* que abre o campo de atuação de um comando pode ser identificada pelo sinal "<" e a que encerra pelo sinal ">" (Pedroso, 2007).

Geralmente, a composição padrão de uma página HTML possui três partes básicas, estruturadas por um cabeçalho, a estrutura principal e o corpo da página, porém outras formas de estruturar uma página em HTML são viáveis. O uso difundido dessa linguagem de marcação se dá em razão da alta gama de personalização de títulos e subtítulos, estilos de texto e sua estruturação, listas ordenadas e desordenadas, listas por definição, o uso de links, a capacidade de inserção de imagens no corpo da página, entre outros (Ozelin, 2017).

No caso da *Landing Page* da Monetar.ia, o uso de HTML se dará na estruturação da página, com um cabeçalho, um corpo principal onde as informações

sobre a ferramenta serão dispostas, um *link* de acesso ao *download* da aplicação na Play Store e um *footer* com outras informações técnicas do aplicativo.

3.6.2 CSS

O CSS é a sigla para *Cascading Style Sheets*, ou, na tradução direta, Folha de Estilo em Cascata. Trata-se de um complemento ao HTML, visto que toda a personalização dos elementos em HTML se dá baseada no CSS, permitindo uma padronização na forma como a página Web é exibida (Ozelin, 2017).

A aplicação do CSS acontece de duas maneiras possíveis, internamente junto ao código HTML ou externamente, em uma pasta específica para seu uso. Quando seu uso é interno ela aparece no cabeço do documento HTML, entre o atributo “*style*”. Sua aplicação externa, que constitui da maneira mais adequada para seu uso, e feita pelo uso exclusivo de um arquivo com extensão “.css” com um *link* de ligação ao documento HTML, com sua respectiva extensão “.html” (Barros; Santos, 2008).

A sintaxe do CSS se dá pela regra de especificar um elemento e passar características ou valores para esse atributo, por exemplo, o tamanho da fonte ou a cor de algum elemento em tela. Outras propriedades podem ser definidas por CSS como o plano de fundo, características de texto, bordas de elementos, margens e *padding*s, entre outros (Ozelin, 2017).

O CSS será usado na *Landing Page* da Monetar.ia para a construção da identidade visual estabelecida para a aplicação, compondo um conjunto de elementos que possam comunicar os valores e o propósito da aplicação

3.6.3 JavaScript

O JavaScript é uma linguagem de programação desenvolvida especificamente para dinamizar páginas Web. Algumas páginas apresentam informações estáticas e podem ser chamadas de documentos, em contraponto, outras tem a capacidade de executar funções, dinamizar informações ou meramente compor elementos visuais não estáticos, isso se dá pela inserção de objetos definidos através de JavaScript em consonância com a paginação do HTML e a composição visual do CSS. Em elementos visuais, especialmente, o JavaScript

permite a alteração dinâmica de cores, textos e posições (Cruz; Petrucelli; Sotto, 2018).

Na *Landing Page* da Monetar.ia, o JavaScript será usado como uma ferramenta para tornar a página mais dinâmica e visualmente interessante para os potenciais usuário.

3.7 INTELIGÊNCIA ARTIFICIAL GENERATIVA

As inteligências artificiais generativas são ferramentas que detém a capacidade de entender, responder e criar diálogos similares aos de humanos, mantendo uma conversa longa baseado na interação em linguagem natural. Com essa característica, elas permitem a automação de diversas tarefas, como a escrita de e-mails, preencher formulários, criar textos, traduzir, resumir, sintetizar, organizar e estruturar conteúdos, realizar revisão de literatura, transcrever áudios, criar e corrigir *scripts* de programação. As inteligências artificiais generativas usam, em geral, o modelo grande de linguagem — ou *large language models* —, conhecido pela sigla LLM (Sampaio *et al.*, 2024).

Para entender as LLMs é necessário compreender os seguintes conceitos. O aprendizado da máquina: corresponde sobre um campo da inteligência artificial que desenvolve algoritmos capazes de tomar decisões sem programação humana direta, aprendendo padrões e atuando de maneira independente; a vetorização do texto: que permite que termos sejam relacionados entre si e avalia o mais importante em termos de conteúdo, permitindo que a máquina entenda e interprete a linguagem humana; e, por último, os dados massivos utilizados pelo modelo para a construção de uma resposta correta (Sampaio *et al.*, 2024).

Entres as variadas inteligências artificiais generativas destaca-se a Gemini desenvolvida e mantida pela Google e o ChatGPT criado pela OpenAI, trata-se dos maiores players do mercado de IA no momento. Ambos os modelos possuem habilidades linguísticas notáveis, porém, diferem em questões técnicas. Para a compreensão de qual LLM melhor qualifica, deverá levar em consideração alguns critérios como: precisão factual, integração de pesquisa, estilo de conversa, criatividade, viés, segurança, disponibilidade, contexto e compreensão de instruções complexas.

A Gemini possui vantagem no fornecimento de informações factuais devido a sua integração com o conhecimento do buscador do Google, trazendo informações mais completas, extensas e com citação da fonte, aumentando a transparência da informação. Por outro lado, o ChatGPT possui uma base de conhecimento considerável, porém, vacilando na precisão, principalmente em relação de assunto de nicho ou eventos recentes. Embora a precisão seja crucial, muitos usuários priorizam a fluência da conversação em contraponto ao factual da escrita (Rane; Choudhary; Rane, 2024).

O ChatGPT geralmente lidera em termos de fluxo de conversação e criatividade, se destacando na geração de diversos formatos de texto, incluindo poemas, trechos de código, scripts e composições musicais, além de poderem, em alguns casos, serem envolventes e divertidas devido a sua habilidade em capturar nuances sutis e infundir personalidade sendo, no geral, mais lúdico. Em contraste, o Gemini se inclina para a precisão e brevidade factual, apresentado uma variação linguística como característica (Rane; Choudhary; Rane, 2024).

Ambas se destacam quanto a mitigação de vieses prejudiciais em seus modelos de linguagem, no entanto, ainda existem falhas no que se refere aos vieses inerentes apresentados nos dados de treinamento que refletem os preconceitos humanos do mundo real. A OpenAI adota uma abordagem cautelosa ao gerar resultados que podem ser sensíveis ou potencialmente prejudiciais, o Google, em consonância, adota a política de não causar danos enfatizando a segurança, porém o risco de produzir conteúdo tendencioso ou ofensivo persiste em ambos os modelos. As semelhanças entre ambas as LLMs persistem no que tange ao recurso de compreender instruções avançadas e manter contexto durante conversas prolongadas, mesmo assim, nota-se discrepâncias sutis no tratamento de nuances linguísticas, o Gemini supera o ChatGPT na decifração de prompts, aproveitando segmentos de diálogo anteriores para fornecer respostas coerentes, o ChatGPT necessita de uma formulação mais precisa para obter os melhores resultados (Rane; Choudhary; Rane, 2024).

Considerando a enorme semelhança entre ambas as LLMs os pesquisadores Rane, Choudhary e Rane (2024) resumem suas conclusões no seguinte trecho:

Gemini e ChatGPT empregam abordagens arquitetônicas significativamente diferentes, moldando suas capacidades e trajetórias potenciais. O foco da Gemini na versatilidade multimodal e na integração com a infraestrutura do

Google a posiciona como uma ferramenta de conhecimento abrangente, enquanto a ênfase do ChatGPT em interações de linguagem natural e geração de texto imaginativo ressalta sua adequação para aplicativos de IA conversacional. Tanto o ChatGPT quanto o Gemini demonstram habilidades impressionantes na compreensão e geração de texto, cada um com pontos fortes únicos em vários domínios.

Em um contexto de finanças, o ChatGPT simula cenários de mercado e avalia os riscos associados à negociação de alta frequência, auxiliando o investidor na tomada de decisões informadas. Soma-se ainda, a otimização de orientação financeira personalizada, onde o ChatGPT realiza o planejamento orçamentário com base as preferências do usuário e padrões financeiros. Sobre sua capacidade analítica, o ChatGPT pode analisar padrões de comportamento e transação do usuário para a detecção de fraudes aumentando a segurança nas plataformas em que é integrada (Rane; Choudhary; Rane, 2024). Baseado nessas informações, o ChatGPT foi escolhido para ser a inteligência artificial generativa responsável pelas análises preditivas da lógica de gestão financeira do aplicativo.

4 RESULTADOS E DISCUSSÃO

Considerando a fundamentação teórica e as ferramentas propostas para o embasamento técnico e acadêmico do produto em desenvolvimento, segue-se para a parte da apresentação dos resultados obtidos durante a fase de execução do projeto, enfatizando o gerenciamento da regra de negócio, a delimitação dos requisitos, a aplicação das ferramentas e técnicas de programação e os resultados alcançados.

4.1 ANÁLISE E DEFINIÇÃO DE REQUISITOS

Para o processo de análise e definição de requisitos do sistema, aplicou-se a UML — *Unified Modelling Language* ou Linguagem de Modelagem Unificada — pois, é um padrão de notação para modelagem aceito e reconhecido por inúmeras empresas e desenvolvedores de *software*. A UML define um conjunto básico de diagramas e notações que permitem representar as múltiplas perspectivas do sistema sob análise e desenvolvimento, entre eles, noções estruturais, estáticas, comportamentais e dinâmicas (Costa, 2001).

Para o desenvolvimento da aplicação móvel Monetar.ia, foi proposto uma documentação de requisitos do sistema, contento requisitos funcionais (funções) e não-funcionais (de qualidade). A tabela a seguir representa essa lógica de hierarquia de requisitos:

Tabela 1 - Definições e Siglas

Número de Ordem	Siglas	Definição
1	RF	Requisitos funcionais
2	RFN	Requisitos não funcionais

Fonte: Elaborado pelos autores (2024).

Tais requisitos incluem as capacidades do produto, os recursos disponíveis, seus possíveis benefícios e os critérios de validação. Deverão ser identificados e listados em agrupamentos lógicos, onde cada requisito deverá conter um

identificador único em formato padronizado, com intuito de obter clareza e facilitar processos durante o desenvolvimento, facilitar a identificação e verificação.

4.1.1 Requisitos Funcionais

Os requisitos funcionais correspondem ao que o sistema deve de fazer, onde cada função pode ser expressa em um ou mais requisitos, aos quais a aplicação deverá atender para funcionar de maneira correta e satisfatória.

Com a razão de propor uma hierarquia entre os requisitos funcionais, definindo prioridades para o desenvolvimento das funções do sistema, foi desenvolvido uma tabela de prioridades, que pode ser observada abaixo:

Tabela 2 - Tabela de Prioridades

Essencial	É o requisito sem o qual o sistema não entra em funcionamento. Requisitos essenciais são requisitos imprescindíveis, que têm que ser implementados impreterivelmente.
Importante	É o requisito sem qual o sistema entra em funcionamento, mas de forma não satisfatória. Requisitos importantes devem ser implementados, mas se não forem, o sistema poderá ser implementado e usado mesmo assim.
Desejável	É o requisito que não compromete as funcionalidades básicas do sistema, isto é, o sistema pode funcionar de forma satisfatória sem ele. Requisitos desejáveis são requisitos que podem ser deixados para versão es posteriores do sistema, caso não haja tempo hábil para implementá-los na versão que está sendo especificada.

Fonte: Elaborado pelos autores (2024).

Baseado na hierarquia definida na tabela de prioridades, foi desenvolvida uma tabela de requisitos funcionais, que pode ser observada abaixo:

Tabela 3 - Tabela de Requisitos Funcionais

Indentificador	Requisito
RF001	Acessar a <i>home page</i>
Prioridade:	<input checked="" type="checkbox"/> Essencial <input type="checkbox"/> Importante <input type="checkbox"/> Desejável
Regras de negócio	
O sistema deve exibir a página inicial (<i>home page</i>), onde o usuário poderá realizar o <i>scroll</i> ou pressionar o botão para acessar a tela de <i>login</i> .	

Identificador	Requisito
RF002	Realizar cadastro
Prioridade:	<input checked="" type="checkbox"/> Essencial <input type="checkbox"/> Importante <input type="checkbox"/> Desejável
Regras de negócio	
<p>Permitir que o usuário realize o cadastro no sistema. Para isso, será necessário fornecer as seguintes informações nos respectivos campos: Nome, Sobrenome, E-mail, Senha e Confirmação de senha.</p> <p>Todos os campos são obrigatórios, devem ter pelo menos três letras e não podem incluir números ou caracteres especiais (exceto o e-mail e a senha, que podem conter esses elementos).</p> <p>Campo Senha e Confirmação de senha:</p> <ul style="list-style-type: none"> ▪ A senha deve ter entre 8 e 12 caracteres; ▪ À direita desses dois campos, terá um botão de revelação de senha representado por um ícone em formato de olho. Ao clicar no botão, o usuário ativará a exibição da senha, e o ícone mudará, exibindo uma barra sobre o olho; ▪ Ao confirmar a senha, o campo de Confirmação de Senha deve ser idêntico ao campo Senha. 	
Identificador	Requisito
RF003	Realizar <i>login</i>
Prioridade:	<input checked="" type="checkbox"/> Essencial <input type="checkbox"/> Importante <input type="checkbox"/> Desejável
Regras de negócio	
<p>Permitir que o usuário realize o login no sistema. Para isso, será necessário fornecer as seguintes informações nos respectivos campos: E-mail e Senha. No primeiro acesso, aparecerá um modal perguntando se o usuário deseja permanecer logado. Se ele escolher "Sim", ao fechar o aplicativo, poderá entrar novamente sem precisar informar seu e-mail e senha, conectando-se automaticamente. Essa conexão automática só será desativada se o usuário realizar o <i>logout</i>.</p>	
Identificador	Requisito
RF004	Recuperar senha
Prioridade:	<input type="checkbox"/> Essencial <input checked="" type="checkbox"/> Importante <input type="checkbox"/> Desejável
Regras de negócio	
<p>Permitir que o usuário solicite a recuperação de sua senha. Ao realizar a solicitação, será enviado um e-mail contendo um código de verificação. O usuário deverá inserir esse código para validar sua identidade e, assim, prosseguir com a redefinição da senha.</p>	
Identificador	Requisito

RF005	Exibir tela inicial
Prioridade:	<input checked="" type="checkbox"/> Essencial <input type="checkbox"/> Importante <input type="checkbox"/> Desejável
Regras de negócio	
<p>Permitir que o usuário acesse a tela inicial onde terá o <i>Dashboard</i> financeiro, exibindo dois campos: o último registro e a última meta.</p> <p>Um gráfico mostrará as últimas compras do usuário, com a divisão entre receitas e despesas.</p> <p>A tela inicial também contará com um calendário, permitindo ao usuário selecionar a data exata de mês e ano, que deseja visualizar.</p>	
Identificador	
RF006	Gerenciar receitas
Prioridade:	<input checked="" type="checkbox"/> Essencial <input type="checkbox"/> Importante <input type="checkbox"/> Desejável
Regras de negócio	
<p>Permitir que o usuário gerencie suas receitas. Durante o cadastro de uma receita, o usuário poderá selecionar a categoria, informar o valor em reais (R\$), fornece uma descrição e indicar a data da transação. O preenchimento de todos esses campos será obrigatório. Além disso, o sistema disponibilizará um ícone de lupa para que o usuário possa pesquisar suas receitas digitando termos específicos.</p>	
Identificador	
RF007	Gerenciar despesas
Prioridade:	<input checked="" type="checkbox"/> Essencial <input type="checkbox"/> Importante <input type="checkbox"/> Desejável
Regras de negócio	
<p>Permitir que o usuário gerencie suas despesas. Durante o cadastro de uma despesa, o usuário poderá selecionar a categoria, informar o valor em reais (R\$), fornece uma descrição e indicar a data da transação. O preenchimento de todos esses campos será obrigatório. Além disso, o sistema disponibilizará um ícone de lupa para que o usuário possa pesquisar suas despesas digitando termos específicos.</p>	
Identificador	
RF008	Gerenciar metas
Prioridade:	<input checked="" type="checkbox"/> Essencial <input type="checkbox"/> Importante <input type="checkbox"/> Desejável
Regras de negócio	
<p>Permitir que o usuário gerencie suas metas. Ao cadastrar uma meta, o usuário deverá informar o nome da meta, o valor alvo em reais (R\$), o valor atual em reais (R\$), adicionar uma descrição e definir uma data limite. O preenchimento de todos esses campos será obrigatório. O sistema também disponibilizará um ícone de pesquisa (lupa), permitindo que o usuário busque metas específicas através da digitação de termos relevantes.</p>	

Identificador	Requisito
RF009	Integrar com Inteligência Artificial
Prioridade:	<input checked="" type="checkbox"/> Essencial <input type="checkbox"/> Importante <input type="checkbox"/> Desejável
Regras de negócio	
O sistema deve disponibilizar um <i>chatbot</i> integrado com a tecnologia de inteligência artificial da OpenAI (ChatGPT), permitindo que o usuário esclareça dúvidas relacionadas ao contexto financeiro, registre despesas e receitas, e realize análises preditivas. O usuário poderá se comunicar com o <i>chatbot</i> de duas maneiras: digitando ou utilizando comandos de voz. Para que o comando de voz funcione, o usuário deve permitir o compartilhamento de seu microfone; caso o microfone esteja desativado, a funcionalidade de voz não estará disponível.	
Identificador	Requisito
RF010	Gerenciar perfil
Prioridade:	<input type="checkbox"/> Essencial <input checked="" type="checkbox"/> Importante <input type="checkbox"/> Desejável
Regras de negócio	
O sistema deve permitir que o usuário atualize suas informações pessoais, incluindo o campo para nome, sobrenome, senha atual e nova senha. Além disso, o sistema apresentará três dicas de segurança para auxiliar o usuário na criação de uma senha robusta.	
Identificador	Requisito
RF011	Realizar <i>logout</i>
Prioridade:	<input checked="" type="checkbox"/> Essencial <input type="checkbox"/> Importante <input type="checkbox"/> Desejável
Regras de negócio	
O sistema deve permitir que o usuário finalize sua sessão no aplicativo, garantindo que ele saia de sua conta.	
Identificador	Requisito
RF012	Realizar relatório em PDF
Prioridade:	<input type="checkbox"/> Essencial <input checked="" type="checkbox"/> Importante <input type="checkbox"/> Desejável
Regras de negócio	
O sistema deve fornecer um arquivo em PDF contendo informações detalhadas do estado financeiro do usuário, com base nas informações que foram cadastradas e atualizadas periodicamente.	
Identificador	Requisito
RF013	Receber notificações
Prioridade:	<input type="checkbox"/> Essencial <input checked="" type="checkbox"/> Importante <input type="checkbox"/> Desejável
Regras de negócio	
O sistema deve possibilitar que o usuário receba notificações relacionadas ao seu	

estado financeiro atual, com base nas informações que foram cadastradas.

Fonte: Elaborado pelos autores (2024).

4.1.2 Requisitos Não Funcionais

Os requisitos não funcionais correspondem aqueles aspectos que expressam qualidades específicas que o sistema proposto deve possuir ou às restrições que o software precisa atender para garantir sua eficácia e usabilidade. Diferentemente dos requisitos funcionais, que descrevem o que o sistema deve fazer, os requisitos não funcionais focam em como o sistema realiza essas tarefas, englobando aspectos como desempenho, escalabilidade, usabilidade e manutenção. Os requisitos não funcionais elaborados para o desenvolvimento do aplicativo Monetar.ia podem ser observados a seguir:

Tabela 4 - Tabela de Requisitos Não Funcionais

Identificador	Requisito
RNF001	Usabilidade
Regras de negócio	
O sistema deve ser de uso simples, com uma interface intuitiva e de fácil navegação para o usuário	
Identificador	Requisito
RNF002	Segurança de senha
Regras de negócio	
As senhas dos usuários devem ser protegidas utilizando criptografia com algoritmos de <i>hash</i> , garantindo a integridade e segurança dos dados armazenados no sistema.	
Identificador	Requisito
RNF003	Desempenho
Regras de negócio	
O sistema deve ter a capacidade de suportar o número necessário de usuário sem comprometer seu desempenho.	
Identificador	Requisito
RNF004	Responsividade
Regras de negócio	

O sistema deve ser escalável, de modo a suportar o aumento no volume de dados e no número de usuários.	
Identificador	Requisito
RNF005	Responsividade
Regras de negócio	
O sistema deve ser escalável, de modo a suportar o aumento no volume de dados e no número de usuários.	
Identificador	Requisito
RNF006	Manutenção
Regras de negócio	
O sistema deve ser simples de gerenciar e receber atualizações.	

Fonte: Elaborado pelos autores (2024).

4.1.3 Diagrama de Caso de Uso

O diagrama de caso de uso descreve uma sequência de ações que representam um cenário principal e cenários alternativos, com objetivo de demonstrar comportamentos de um sistema — ou apenas parte dele — através de interações entre casos e atores. Trata-se de um diagrama que obtém papel central na modelagem de um sistema, evidenciando interações externas com outras entidades e clareza sobre os componentes externos e como são influenciados (Rodrigues, 2013).

Ao invés de focar nas funções do sistema, o modelo de casos de uso captura os usos ou aplicações completas do sistema, buscando responder quais usos o sistema terá ou para que aplicações o sistema será empregado. Cada uso descreve uma aplicação ou uso completo do *software*, seu conceito não pode ser confundido com o módulo, visto que o caso de uso não é um componente, mas sim um de seus empregos possíveis. Ademais, também não deve ser confundido com função, pois possui um escopo muito mais limitado, traduzindo-se como um mero recurso ou utilidade do sistema (Rodrigues, 2013).

O relacionamento entre um ator e um caso sempre expressa uma comunicação entre eles, sendo essencial para representar como os atores interagem com o sistema. A notação UML para este tipo de relacionamento é

expresso por um segmento de reta ligando o ator com diferentes casos de uso, essa conexão visual ajuda a entender quais ações estão disponíveis e a complexidade das interações dentro do sistema (Stadzisz, 2002). Para o desenvolvimento do projeto do aplicativo Monetar.ia, o seguinte diagrama foi proposto:

Figura 3 - Diagrama de caso de uso



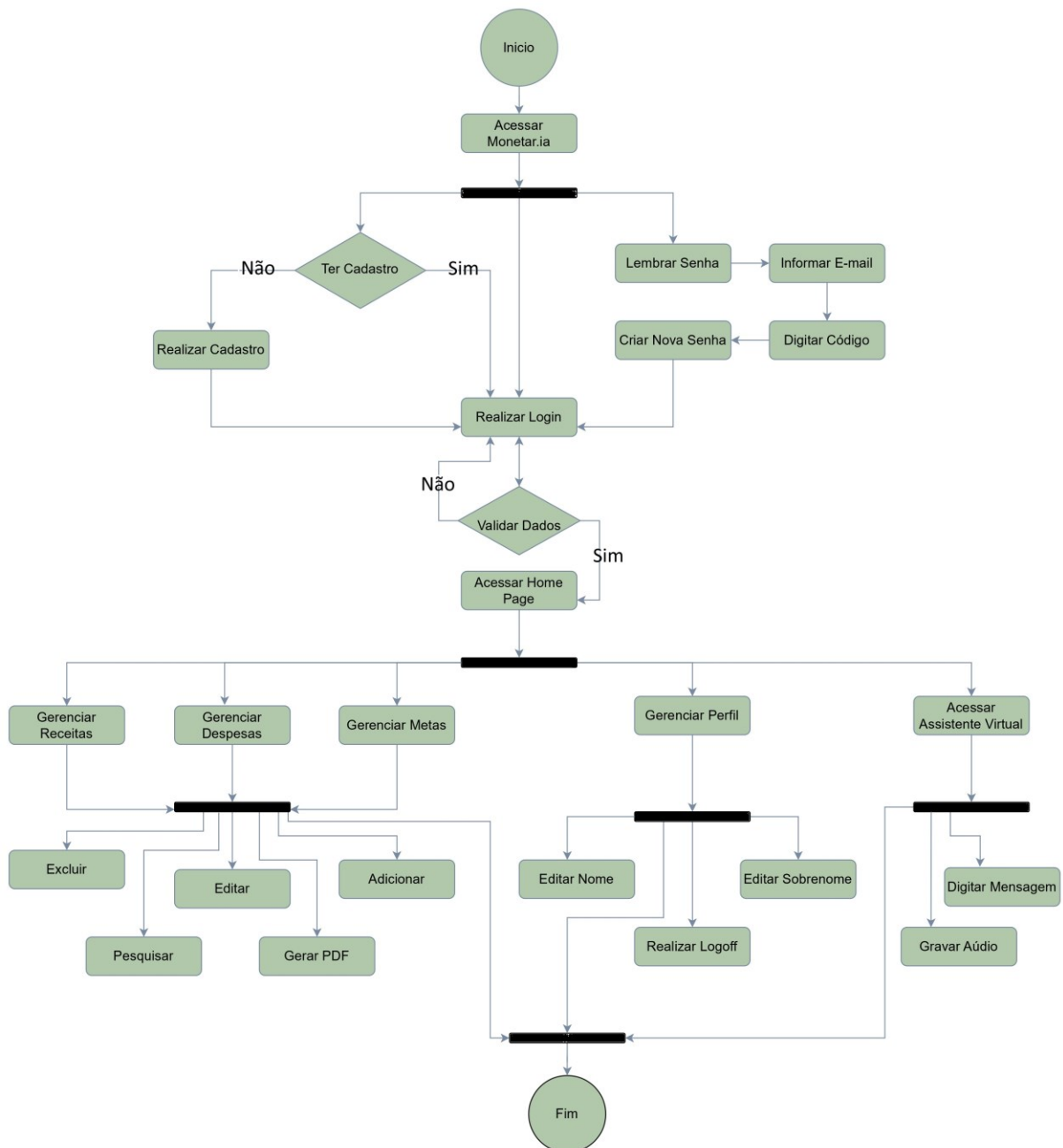
Fonte: Elaborado pelos autores (2024).

Uma vez estabelecido o caso de uso, ele se torna uma base fundamental para guiar o restante do projeto, visto que, o restante da modelagem irá se preocupar com a forma de realização dos cenários propostos no diagrama.

4.1.3 Diagrama de Caso de Atividades

O diagrama de atividades representa a sequência de ações ou atividades e os estados envolvidos na execução de um processo dentro do sistema. Para a produção da Monetar.ia foi proposto o seguinte diagrama de atividades:

Figura 4 - Diagrama de caso de atividades



Fonte: Elaborado pelos autores (2024).

Esse diagrama é amplamente utilizado para ilustrar o fluxo de controle entre atividades, destacando as transições e os caminhos possíveis que as ações podem seguir. Ademais, permite a especificação de sincronismos de concorrência, bifurcações e junções, permitindo uma visão clara sobre os processos paralelos e os pontos de decisão, facilitando a compreensão lógica do sistema, além de identificar possíveis otimizações de processo, garantindo uma execução mais eficiente e organizada das funcionalidades.

4.1.4 Diagrama de Arquitetura

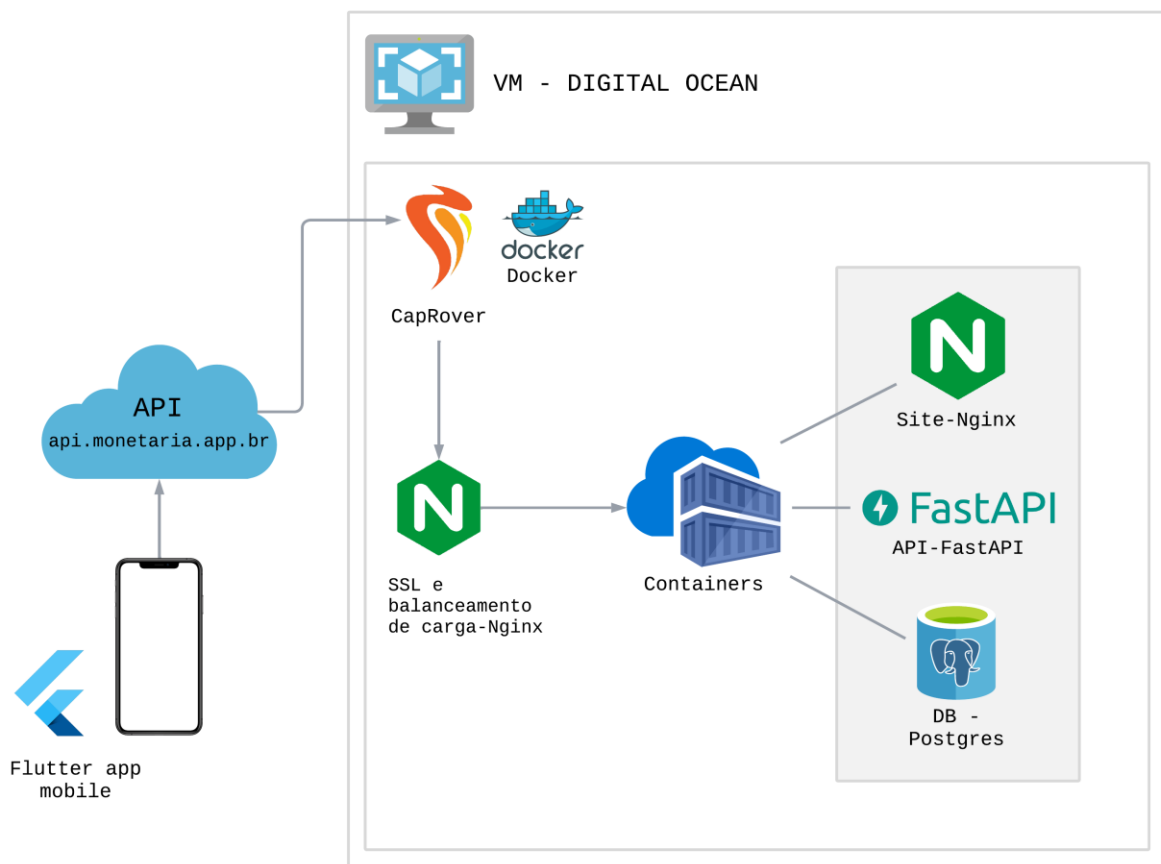
O diagrama de arquitetura é responsável por representar visualmente a dinâmica de construção da arquitetura do software, ilustrando as diversas movimentações de dados no sistema e a interação da aplicação com o ambiente ao redor.

Para garantir uma estrutura organizada e modular do sistema, foram adotadas componentes que promovem a separação de responsabilidades entre os serviços, o que permite que cada parte da aplicação funcione de forma autônoma e possa ser ajustada ou ampliada sem afetar o restante do sistema. Esse modelo facilita a manutenção e a implementação de melhorias, permitindo uma resposta mais rápida às mudanças nas demandas do projeto.

O tipo de arquitetura escolhida para o desenvolvimento da Monetar.ia é a arquitetura de microsserviços, que segue uma lógica de orientação a serviços com comunicação interna por API. Com isso, as funcionalidades da aplicação Monetar.ia podem ser dadas e implementadas de maneira independente, simplificando o desenvolvimento, garantindo resiliência contra possíveis falhas e facilitando a escalabilidade dos serviços.

Para tanto, destaca-se a organização dos serviços e componentes na nuvem, hospedados em uma máquina virtual na Digital Ocean. A aplicação é dividida em contêineres, gerenciados pela plataforma CapRover com Docker, permitindo facilmente escalável. O Nginx é utilizado para realizar o balanceamento de carga e garantir a segurança das comunicações com um certificado SSL. A API, construída com Fast API, interage com o banco de dados PostgreSQL. Todas essas operações são acessadas forma remota pela aplicação Monetar.ia. A seguir, o diagrama de arquitetura proposto para o desenvolvimento do projeto:

Figura 5 - Diagrama de arquitetura de software



Fonte: Elaborado pelos autores (2024).

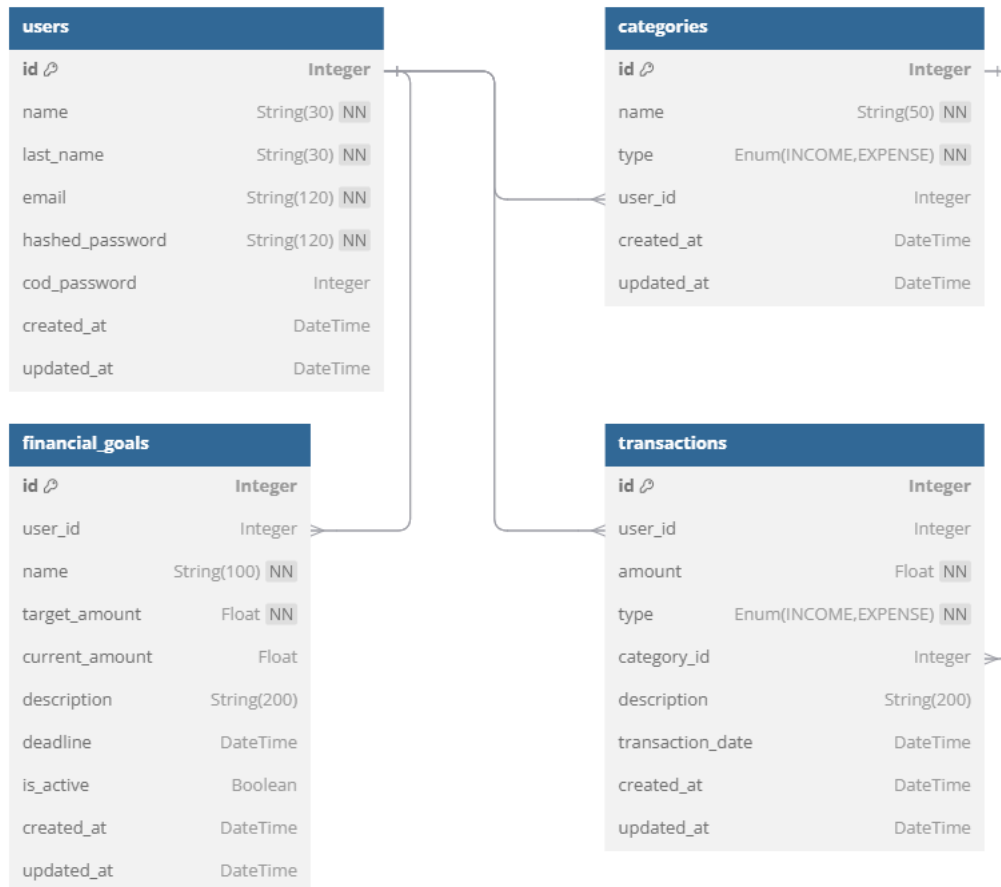
4.1.5 Diagrama de Banco de Dados

Para o desenvolvimento do banco de dados relacional em PostgreSQL, foi proposto uma modelagem de entidade e relacionamento, que representa conceitualmente de forma direta as características de relacionamento dos dados, facilitando a visualização da estrutura dos dados que se pretende armazenar.

Esse modelo é composto por entidades: que representa o objeto que necessita armazenar, como usuário, transações, objetivos financeiros e categorias; os relacionamentos: que indicam a associação entre essas entidades; e as chaves, sendo elas, primárias — a identificação exclusiva da entidade — ou estrangeiras, que compõe a chave primária identificada em outra entidade, compondo o relacionamento e combinação de informações. Além disso, o uso de chaves estrangeiras facilita a integridade referencial, assegurando que as relações entre

entidades permaneçam consistentes. Assim, esse modelo possibilita uma estrutura robusta para consultas complexas realizadas pelo sistema (Santos, 2015). Para o projeto da Monetar.ia foi proposto o seguinte diagrama:

Figura 6 - Diagrama de banco de dados



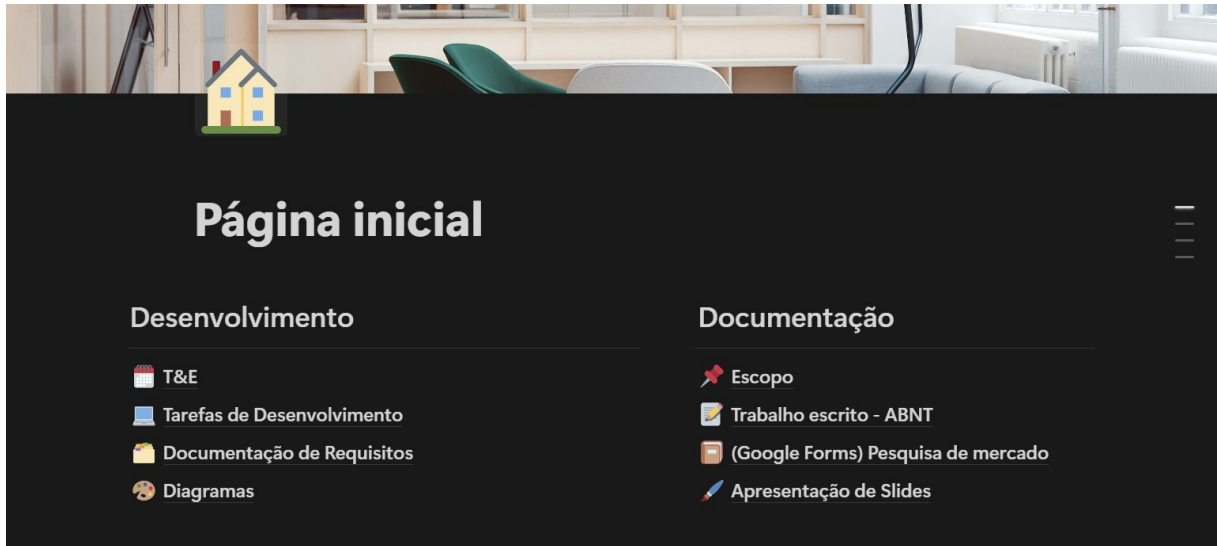
Fonte: Elaborado pelos autores (2024).

4.2 GERENCIAMENTO DO PROJETO

Para o gerenciamento do projeto de desenvolvimento da aplicação móvel da Monetar.ia, foi proposto um modelo misto de metodologia ágil, no qual, os métodos *scrum* e *kanban* foram executados em sincronia na gestão do projeto. Para tanto, foi criada uma página no Notion aberta para todos os membros da equipe de projeto, voltada para o planejamento das *sprints* e gerenciamento de *backlog*, além disso, o

Notion também foi usado como repositório da documentação de requisitos e de outras ferramentas uteis. Na imagem a seguir, a página principal do Notion:

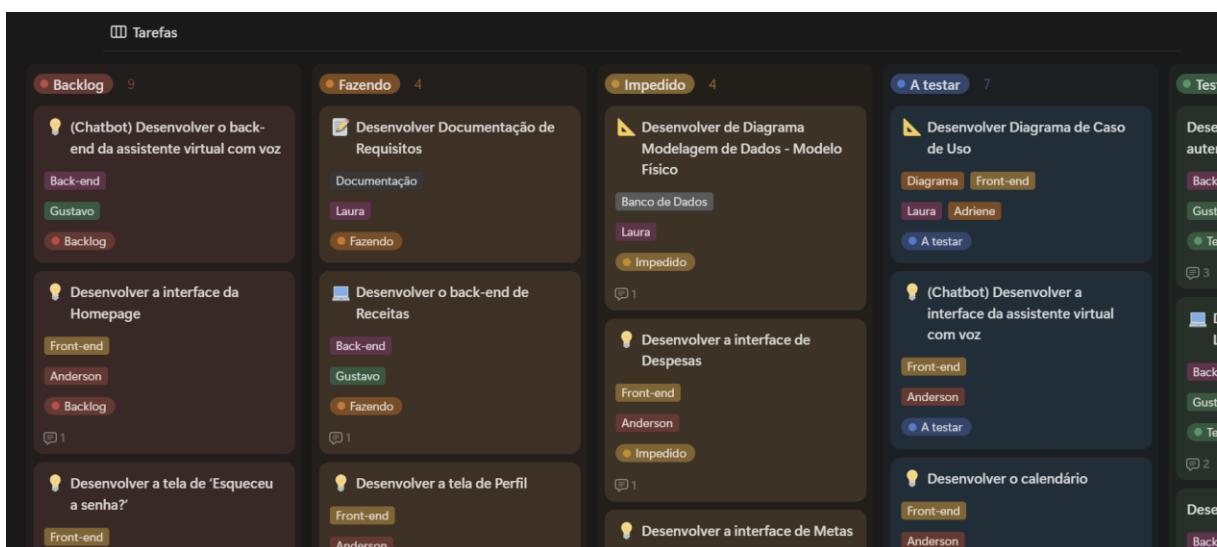
Figura 7- Visão geral do notion



Fonte: Elaborado pelos autores (2024).

Por se tratar de uma ferramenta extremamente personalizável foi possível desenvolver um *board kanban* para a administração do andamento de tarefas do time de desenvolvedores com intuito de estabelecer ordem ao fluxo de desenvolvimento. A imagem a seguir é a visão do *board*:

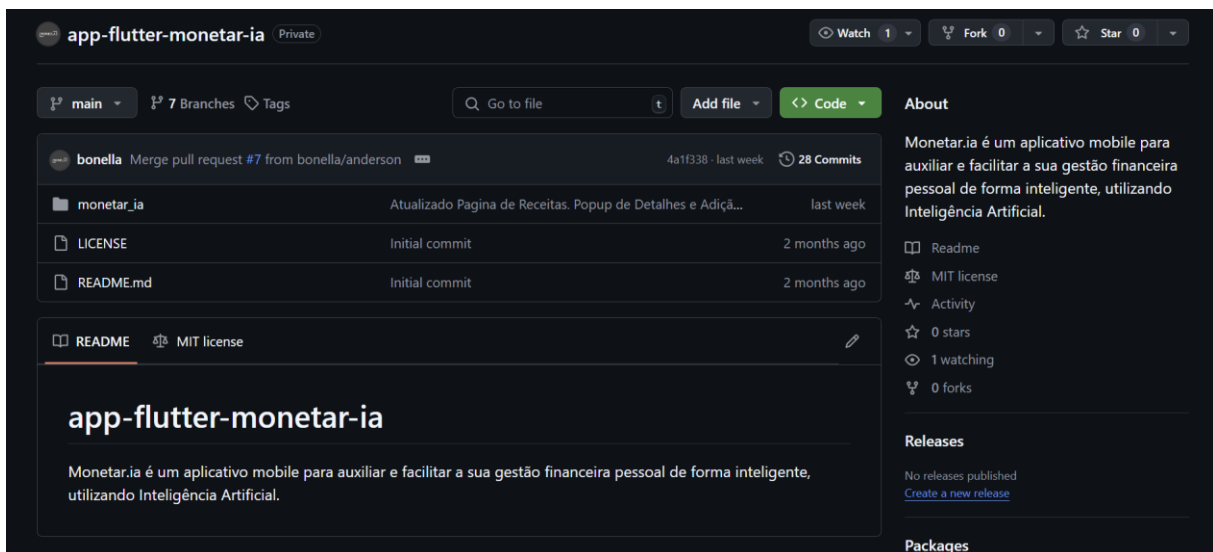
Figura 8 - Visão geral do *board* de tarefas



Fonte: Elaborado pelos autores (2024).

Todos os códigos desenvolvidos durante a fase de criação do produto foram depositados em um repositório exclusivo no GitHub (imagem abaixo), acessível apenas pela equipe de desenvolvedores para garantir segurança e controle de versões. Utilizando o terminal do Visual Studio Code, a equipe anexou a totalidade do projeto ao repositório por meio de comandos Git, permitindo manter um histórico das atualizações e alterações, documentando todo o processo. Além disso, o repositório foi configurado com *branches* específicas para desenvolvimento e produção, permitindo uma gestão mais segura e organizada das versões. Cada novo recurso ou correção foi revisado por meio de *pull requests*, garantindo qualidade com os padrões estabelecidos. Com esse fluxo, foi possível também aplicar testes, *code review* e validações, assegurando a estabilidade da aplicação antes de cada release.

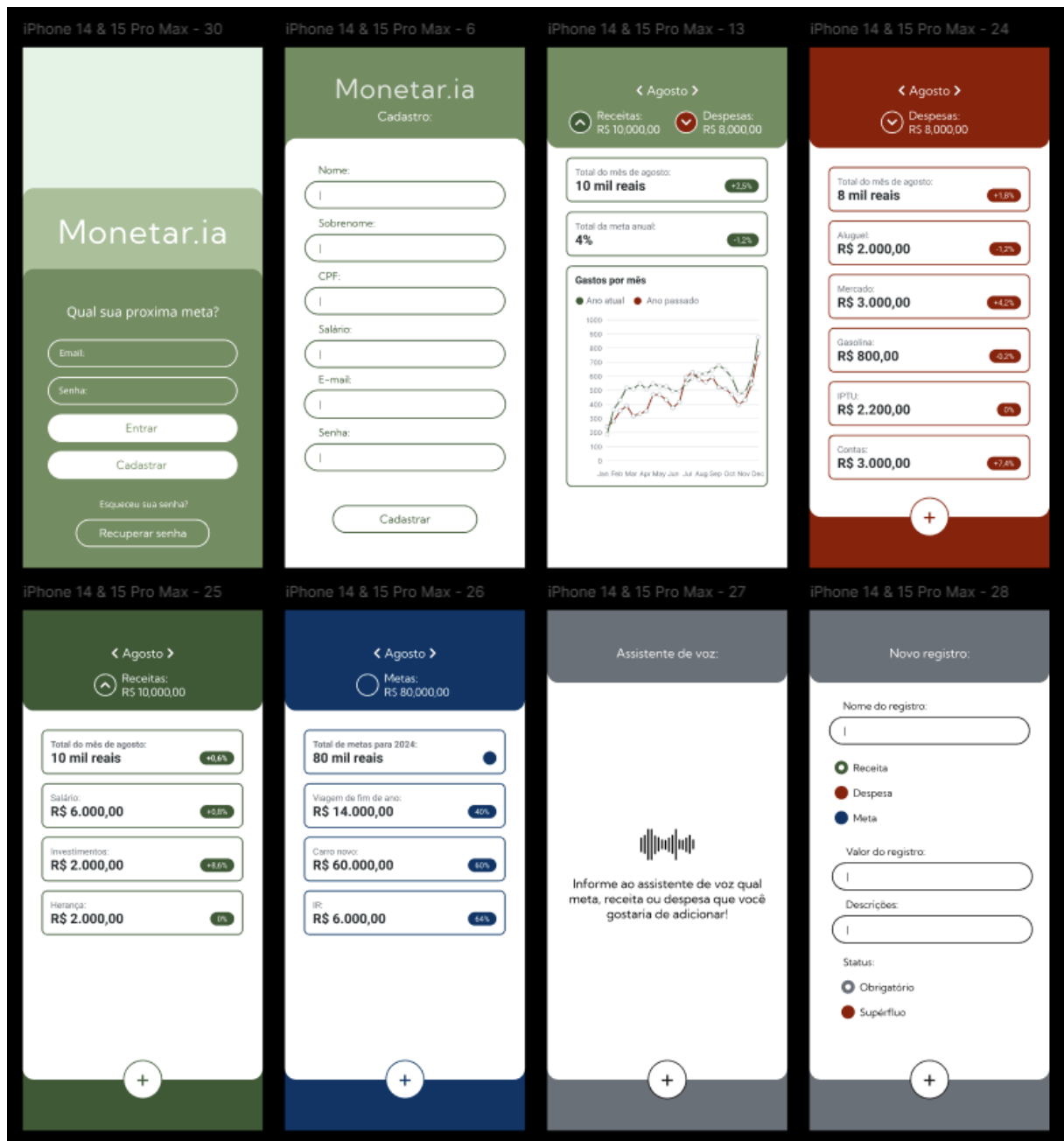
Figura 9 - Repositório do Projeto no GitHub



Fonte: Elaborado pelos autores (2024).

Para o desenvolvimento do *front-end* da aplicação móvel, foi criado um protótipo de baixa fidelidade usando a ferramenta Figma para a concepção das interfaces do aplicativo (imagem abaixo). Este protótipo inicial serve como base para aprimoramento durante a fase de desenvolvimento do produto. Além de facilitar a visualização das principais funcionalidades, o protótipo possibilita o alinhamento entre os desenvolvedores da aplicação, garantindo que todos compartilhem a mesma visão da experiência de usuário

Figura 10 - Conceito de design do protótipo



Fonte: Elaborado pelos autores (2024)

4.2.1 Landing Page

Com objetivo de divulgar as funcionalidades da aplicação Monetar.ia, foi desenvolvido uma *Landing Page* — também conhecida como página de aterrissagem — para a conversão de visitantes ou potenciais usuários em *leads*. Sua implementação foi feita na API da Monetar.ia através de um *container* exclusivo no CapRover para o website, com o seu endereço de acesso em

<https://www.monetaria.app.br>. Na imagem abaixo, podemos ver a tela única do website da Monetar.ia, que consta com informações sobre as funcionalidades da ferramenta e o link de acesso para o *download* da aplicação na Play Store.

Figura 11 - Visão geral da *landing page*



Fonte: Elaborado pelos autores (2024).

4.3 DESENVOLVIMENTO DA API

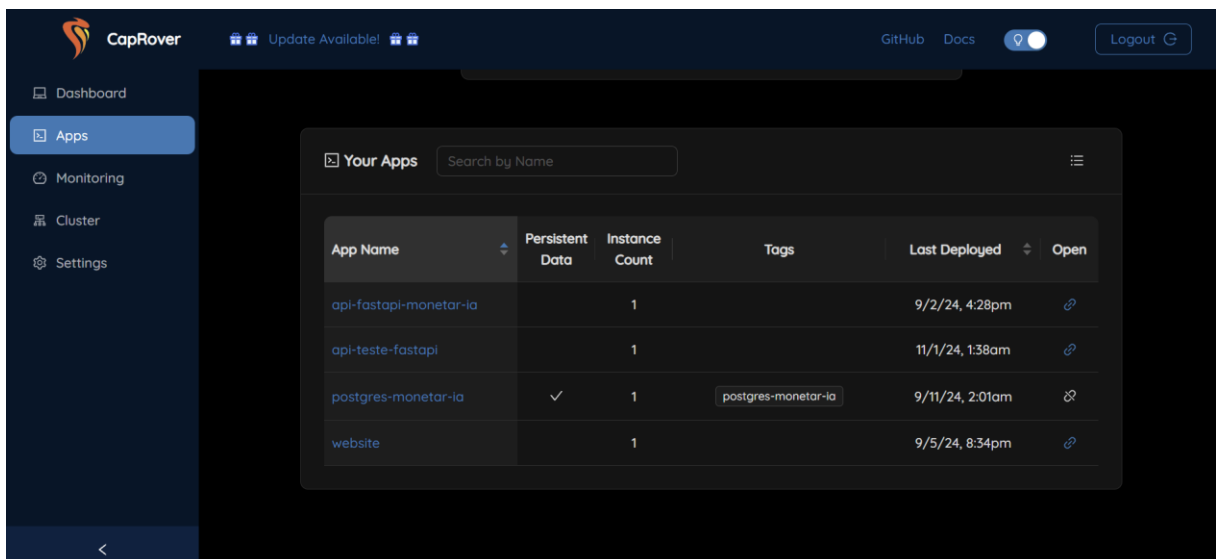
Para o desenvolvimento da API em arquitetura de microsserviços responsável pela comunicação dos componentes internos da aplicação Monetar.ia foi necessário a criação de uma máquina virtual em Ubuntu na Digital Ocean, que será responsável pelo processamento virtualizado de processamento e memória para a execução do software. Essa infraestrutura virtualizada garante flexibilidade e permite uma escalabilidade horizontal, podendo melhorar o processamento conforme o crescimento da aplicação. A escolha da Digital Ocean foi baseada em sua capacidade de fornecer uma infraestrutura confiável, além de ferramentas para monitoramento e segurança.

Na imagem abaixo mostra-se a visão geral do projeto funcionando na plataforma da *Digital Ocean*:

Figura 12 - Visão geral do *Digital Ocean*

Fonte: Elaborado pelos autores (2024).

Todo o gerenciamento de servidor, *deployment*, banco de dados e contêineres será administrado dentro da ferramenta CapRover, que está integrado ao repositório da Monetar.ia no GitHub, permitindo que qualquer alteração no código seja automaticamente disponibilizada na aplicação, graças ao *deployment* contínuo da plataforma. Na imagem abaixo pode-se observar o *dashboard* de gerenciamento do CapRover com a API funcionando em *containers*:

Figura 13 - Visão geral do *CapRover*

Fonte: Elaborado pelos autores (2024).

Como a imagem mostra, a Monetar.ia terá quatro serviços funcionando em *containers*, uma para a API oficial, uma para testes, uma onde funcionará o banco de dados em PostgreSQL e a última para a *landing page* da aplicação.

Além disso, para o gerenciamento da API foi usada o *OAuth 2.0* — um protocolo de autorização conhecido como "autorização aberta" — que permite a concessão de acesso a um conjunto de recursos de maneira segura e controlada. Esse protocolo usa tokens de acesso para representar a permissão do usuário final para acessar informações específicas, sem a necessidade de compartilhar suas credenciais diretamente. Na Monetar.ia, essa autorização é gerenciada por meio de JSON Web Tokens (JWT), permitindo que os dados de autorização sejam incluídos na própria resposta em formato JSON, o que facilita o processamento e a leitura das informações. Além disso, o OAuth estabelece um tempo de validade para os tokens, reforçando a segurança ao limitar o período em que cada token é válido para realizar requisições à API.

O processo de conexão entre o modelo de Inteligência Artificial Generativa com a aplicação da Monetar.ia será feito através do uso da API oficial. Durante a fase de desenvolvimento, a API de teste foi usada para a aplicação de avaliação de qual será a mais adequada para o uso. O processo de teste foi feito usando o mesmo input de informações em ambas os modelos — seja ela a Gemini ou o ChatGPT —, as respostas obtidas foram documentadas e comparadas individualmente. Os dados e o *prompt* usados está registrado na imagem abaixo:

Figura 14 - Informações Inseridas no *Prompt*

```
# Conjunto de dados financeiros de exemplo
dados_financeiros = {
  "transacoes": [
    {"tipo": "receita", "valor": 5000, "data": "2024-09-01"},
    {"tipo": "despesa", "valor": 1500, "data": "2024-09-05"},
    {"tipo": "receita", "valor": 2000, "data": "2024-09-10"},
    {"tipo": "despesa", "valor": 800, "data": "2024-09-15"},
    {"tipo": "despesa", "valor": 1200, "data": "2024-09-20"}
  ],
  "metas_financeiras": [
    {"meta": "economizar 1000 por mês", "prazo": "2024-12"}
  ]
}

# Formatação da mensagem para análise preditiva ou dicas financeiras em JSON
mensagem = f"""
Aqui estão meus dados financeiros:
Transações: {dados_financeiros['transacoes']}
Metas: {dados_financeiros['metas_financeiras']}

Por favor, forneça uma análise preditiva resumida das minhas finanças, 3 dicas de economia e uma notificação personalizada com emojis em formato JSON.
O formato esperado é:
{
  "analise": "string com análise preditiva",
  "dicas": ["dica1", "dica2", "dica3"],
  "notificacao": ["notificação personalizada com emojis"]
}
"""
```

Fonte: Elaborado pelos autores (2024).

Por mais que os dados informados por *prompt* sejam iguais em ambos os cenários de avaliação, o resultado obtido foi diferente, no caso da Gemini o cálculo dos dados financeiros foi calculado de maneira equivocada. Como exemplifica abaixo:

Figura 15 - Resposta da Gemini

```
{'texto_gerado_ia': '```json\n{\n"analise": "Com base em suas transações, você teve um saldo positivo de R$ 1.500 em setembro. No entanto, suas despesas estão altas, representando 60% de sua renda. Se mantiver esse ritmo, dificilmente conseguirá atingir a meta de economizar R$ 1.000 por mês até dezembro. É essencial reduzir gastos para alcançar seu objetivo.",\n"dicas": [\n"Analise seus gastos e identifique áreas onde você pode cortar custos. Por exemplo, experimente cozinhar em casa em vez de comer fora.",\n"Crie um orçamento detalhado para acompanhar suas receitas e despesas e ajustar seus hábitos de consumo.",\n"Considere utilizar aplicativos de controle financeiro para automatizar o acompanhamento e receber alertas sobre seus gastos.",\n],\n"notificacao": "\n\n⚠️ Atenção! ⚠️ Você precisa controlar seus gastos para atingir sua meta de economia. \n\n📌 Ajuste seu orçamento e comece a economizar hoje! \n\n🚀\n\n```\n'}
```

Fonte: Elaborado pelos autores (2024).

No caso do ChatGPT a resposta obtida foi concisa e o cálculo correto, a resposta está registrada na imagem abaixo:

Figura 16 - Resposta do ChatGPT

```
Análise: Com base nas suas receitas e despesas até o momento, é possível prever que seu saldo positivo no final do mês será de aproximadamente 2500. Lembre-se de controlar seus gastos para alcançar sua meta de economizar 1000 por mês até dezembro.\n\nDicas: ['1. Crie um orçamento mensal detalhado para identificar onde você pode cortar gastos desnecessários.', '2. Considere a possibilidade de investir parte das suas receitas para aumentar seu patrimônio a longo prazo.', '3. Compare preços e pesquise por promoções antes de realizar compras, para economizar ainda mais.']\n\nNotificações: ['★ Parabéns! Você está no caminho certo para atingir suas metas financeiras! ★']\n\n(venv) PS D:\> █
```

Fonte: Elaborado pelos autores (2024).

Em razão da precisão do cálculo e da característica abreviada da resposta obtida, preferiu-se implementar o ChatGPT como o modelo de Inteligência Artificial Generativa que será usada como assistente virtual da aplicação móvel da Monetar.ia.

4.3.1 Documentação da API

Com objetivo de facilitar o processo de desenvolvimento das interfaces de usuário da aplicação Monetar.ia, a equipe de desenvolvedores do projeto optou pelo uso do Swagger para a documentação automática da API. Essa ferramenta pode ser utilizada em diversas linguagens de programação, permitindo uma lista de recursos,

operações, parâmetros e informações dentro da API, sem precisar disponibilizar acesso ao código-fonte da aplicação. Essa documentação gerada ficará disponível no *link* <https://api.monetaria.app.br/docs>, como está exemplificado na imagem abaixo:

Figura 17 - Documentação de funções da API

GET Gerar Descrição

POST Criar transação

GET Listar transações

GET Listar transações - despesas

GET Listar transações - receitas

GET Listar transação

PUT Atualizar transação

DEL Deletar transação

POST Criar meta

GET Listar metas

GET Listar meta

PUT Atualizar meta

DEL Deletar meta

POST Criar categoria

GET Listar categorias

PUT Atualizar categoria

DEL Deletar categoria

Criar transação

Cria uma nova transação para o usuário autenticado.

- transaction:** Dados da transação a ser criada.
- current_user:** O usuário autenticado que está criando a transação.

Retorna:

- A transação criada com um status 201 se bem-sucedido.
- 403 se o `user_id` fornecido não corresponder ao usuário autenticado.

AUTHORIZATIONS: > OAuth2PasswordBearer

REQUEST BODY SCHEMA: application/json

user_id required	integer (User Id)
amount > required	Amount (number) or Amount (string) (Amount)
type required	string (TransactionTypeEnum) Enum: "INCOME" "EXPENSE"
category_id required	integer (Category Id)
description >	Description (string) or Description (null) (Description)
transaction_date >	Transaction Date (string) or Transaction Date (null) (Transaction Date)

Responses

- > 201 Successful Response
- > 422 Validation Error

Fonte: Elaborado pelos autores (2024).

4.4 FUNCIONAMENTO DO SISTEMA

O aplicativo móvel de gestão financeira Monetaria foi desenvolvido para ser uma ferramenta prática e intuitiva, oferecendo uma experiência simplificada aos usuários. Na razão de garantir usabilidade e acessibilidade, a interface foi projetada

seguindo uma abordagem minimalista, com pouco elementos visuais, destacando-se apenas informações essenciais, para que o usuário possa navegar com facilidade e entender rapidamente os recursos oferecidos. Por exemplo, na tela inicial da aplicação, optou-se por um pequeno texto de apresentação da ferramenta seguido pelas funcionalidades de acesso, cadastro e recuperação de senha, como pode ser observado na imagem abaixo:

Figura 18 - Visão geral da tela inicial da Monetar.ia



Fonte: Elaborado pelos autores (2024).

Para o acesso a Monetar.ia, o usuário utiliza um sistema de autenticação com e-mail e senha. Foram implementadas funcionalidades de validação como a

exigência de um endereço de e-mail válido, conforme padrões de formatação e senha com oito e doze caracteres, garantido um nível de proteção.

Caso seja a primeira vez usando a Monetar.ia, o usuário poderá realizar o cadastro diretamente na aplicação. Para isso, ele precisará fornecer seu e-mail e criar uma senha que atenda aos requisitos de segurança já especificados, além do preenchimento do nome e sobrenome, conforme pode-se observar na imagem abaixo:

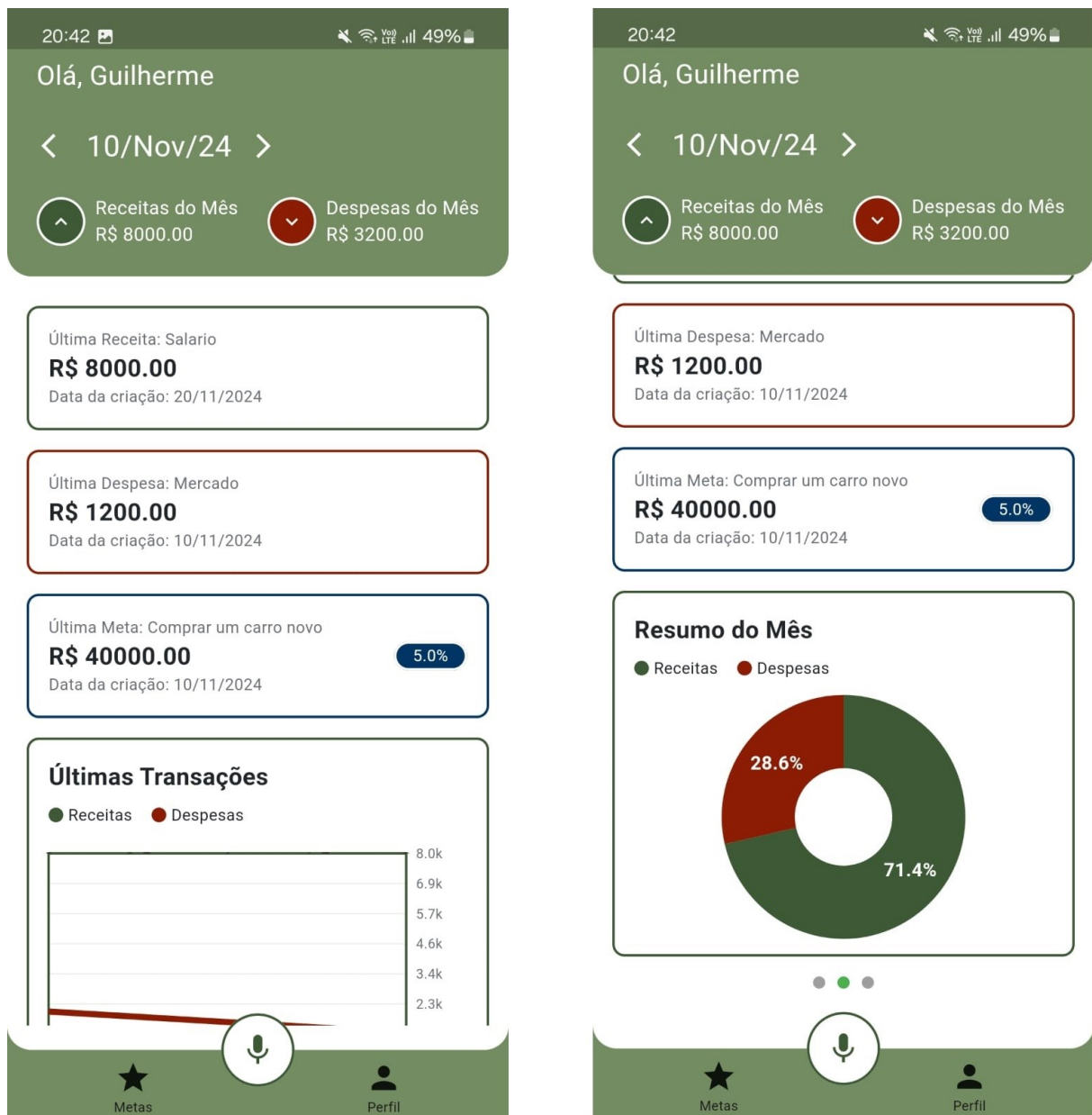
Figura 19 - Visão geral da tela cadastro da Monetar.ia

The image displays two side-by-side screenshots of the Monetar.ia registration screen. Both screens have a dark green header with the Monetar.ia logo and the word 'Cadastro'. The left screenshot shows the registration form with empty input fields for 'Nome:', 'Sobrenome:', 'E-mail:', 'Senha:', and 'Confirmar Senha:'. Each field has a small eye icon to toggle visibility. The right screenshot shows the same form with the following data entered: 'Nome:' is 'Guilherme', 'Sobrenome:' is 'Thiel', and 'Senha:' is filled with dots. A red error message is displayed below the password field: 'Senha deve ter entre 8 e 12 caracteres'. The 'Confirmar Senha:' field is also filled with dots. At the bottom of each screen is a white 'Cadastrar' button on a dark green background.

Fonte: Elaborado pelos autores (2024).

Após o *login* bem-sucedido, o usuário é direcionado à tela principal da aplicação, onde encontrará uma visão da totalidade de suas informações financeiras, destacando-se os *dashboards* da situação financeira, além da síntese de receitas, despesas e metas do mês atual, como pode-se observar na imagem a seguir:

Figura 20 - Visão geral da tela principal da Monetar.ia



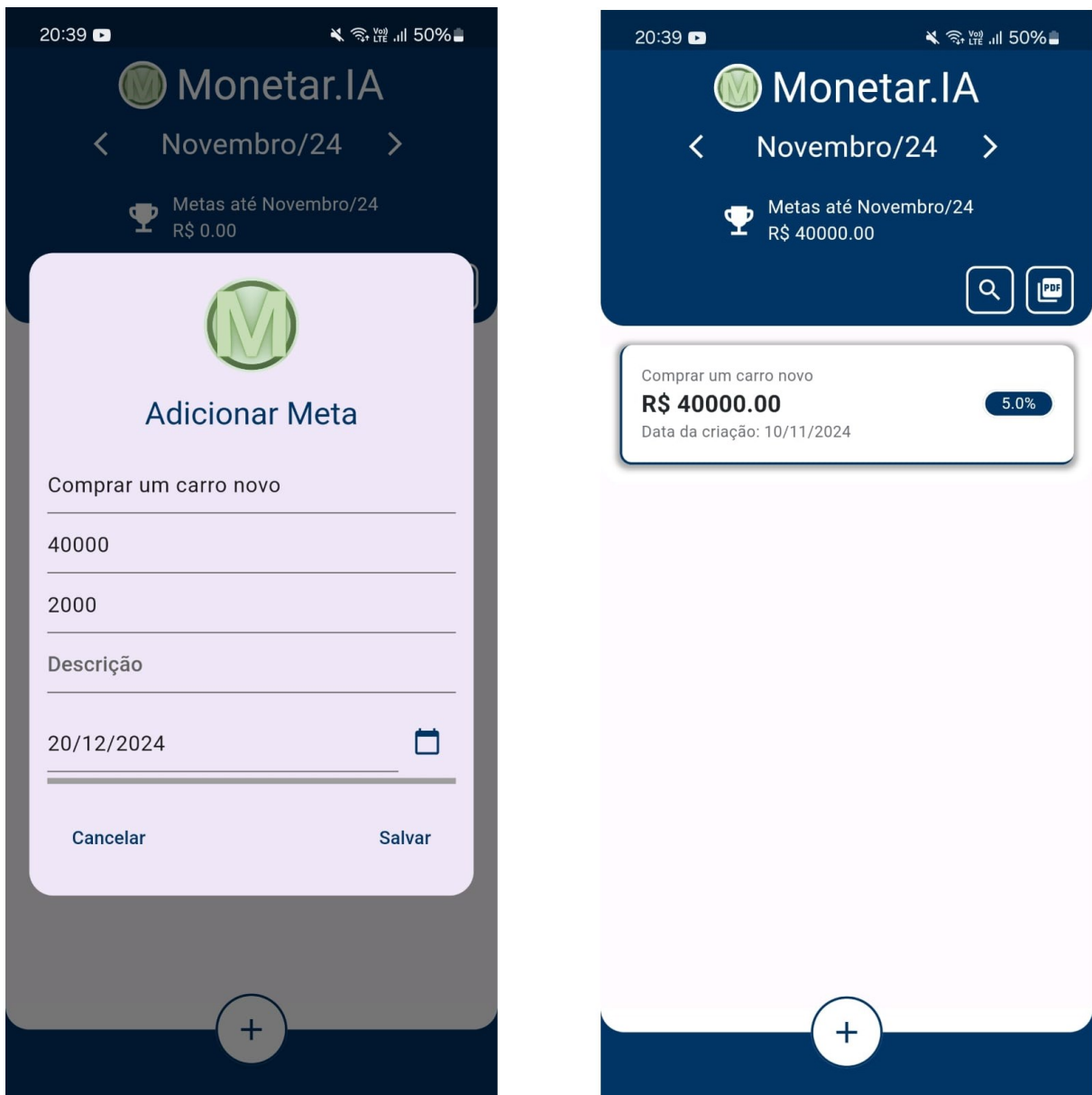
Fonte: Elaborado pelos autores (2024).

Ademais, salienta-se que em todos os recursos da Monetar.ia, foram implementados *feedbacks* imediatos para informações inseridas de maneira

equivoca. Seguindo-se, dessa maneira, a aplicação de heurísticas de usabilidade como a visibilidade do *status* do sistema e prevenção de erros, essenciais para a retenção e bem-estar do usuário.

Em seguida, o usuário poderá cadastrar suas informações financeiras, incluindo metas econômicas que deseja alcançar ou acompanhar, na imagem abaixo observa-se a tela de cadastro de metas:

Figura 21 - Visão geral da tela cadastro de metas da Monetar.ia

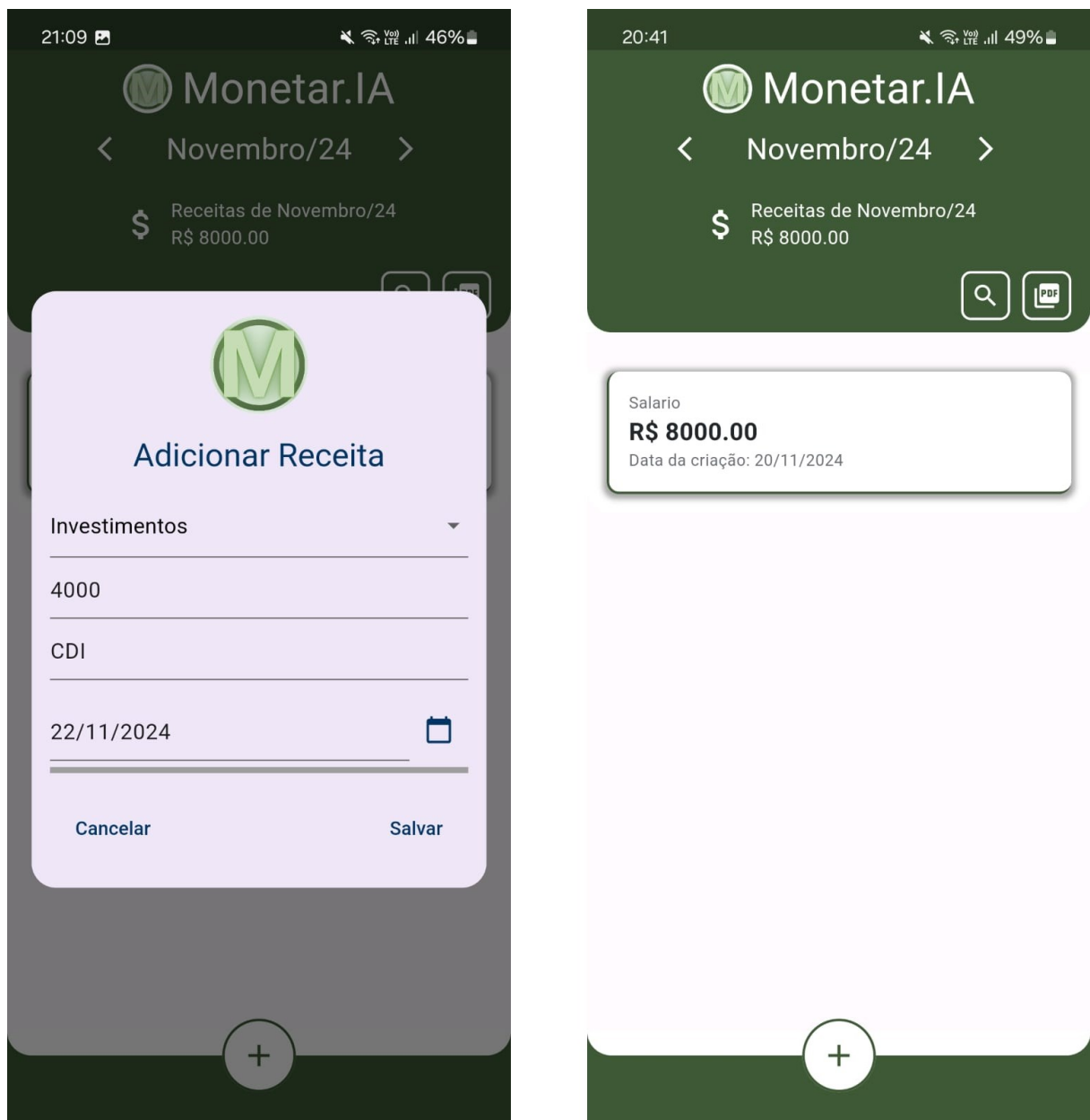


Fonte: Elaborado pelos autores (2024).

O cadastro de uma meta financeira será feita através da definição de um objetivo — como economizar para viajar, comprar um carro ou um imóvel, entre outras —, seguido pelo valor que deverá ser economizado, o valor atual, uma descrição da meta e um prazo final, definido através de calendário da aplicação.

Ademais, o cadastro de receitas segue um fluxo muito semelhante, onde cada receita terá uma categoria específica — como salário, investimentos, entre outros —, seguido pelo valor de capital da receita, uma breve descrição e a data em que a transação ocorreu, como pode-se perceber na imagem abaixo:

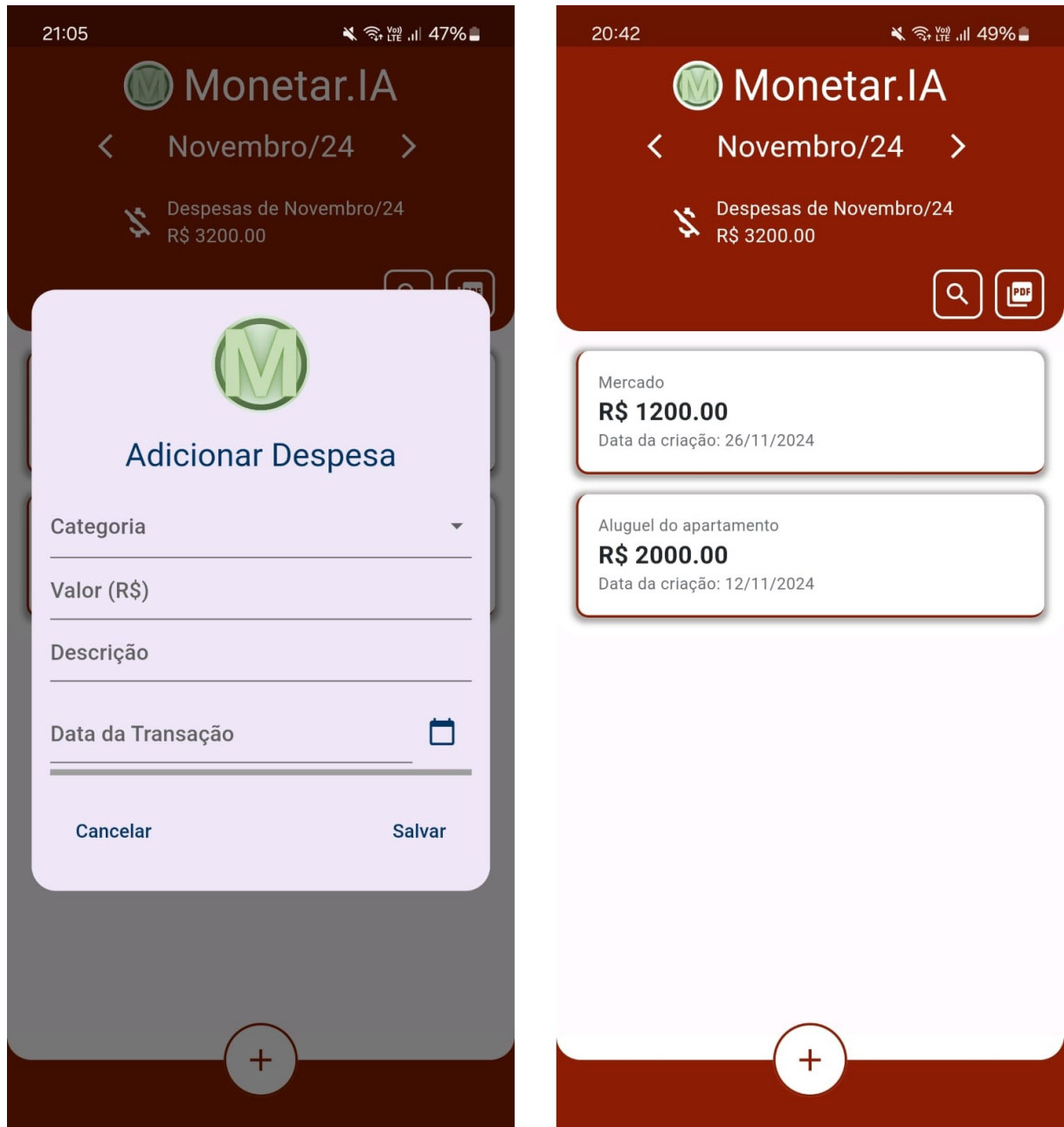
Figura 22 - Visão geral da tela cadastro de receitas da Monetar.ia



Fonte: Elaborado pelos autores (2024).

A tela de cadastro de despesas segue a mesma lógica, com uma categoria para cada despesa seguido pelo valor da despesa, a descrição e a data em que a despesa ocorreu, como visto a seguir:

Figura 23 - Visão geral da tela cadastro de despesas da Monetar.ia

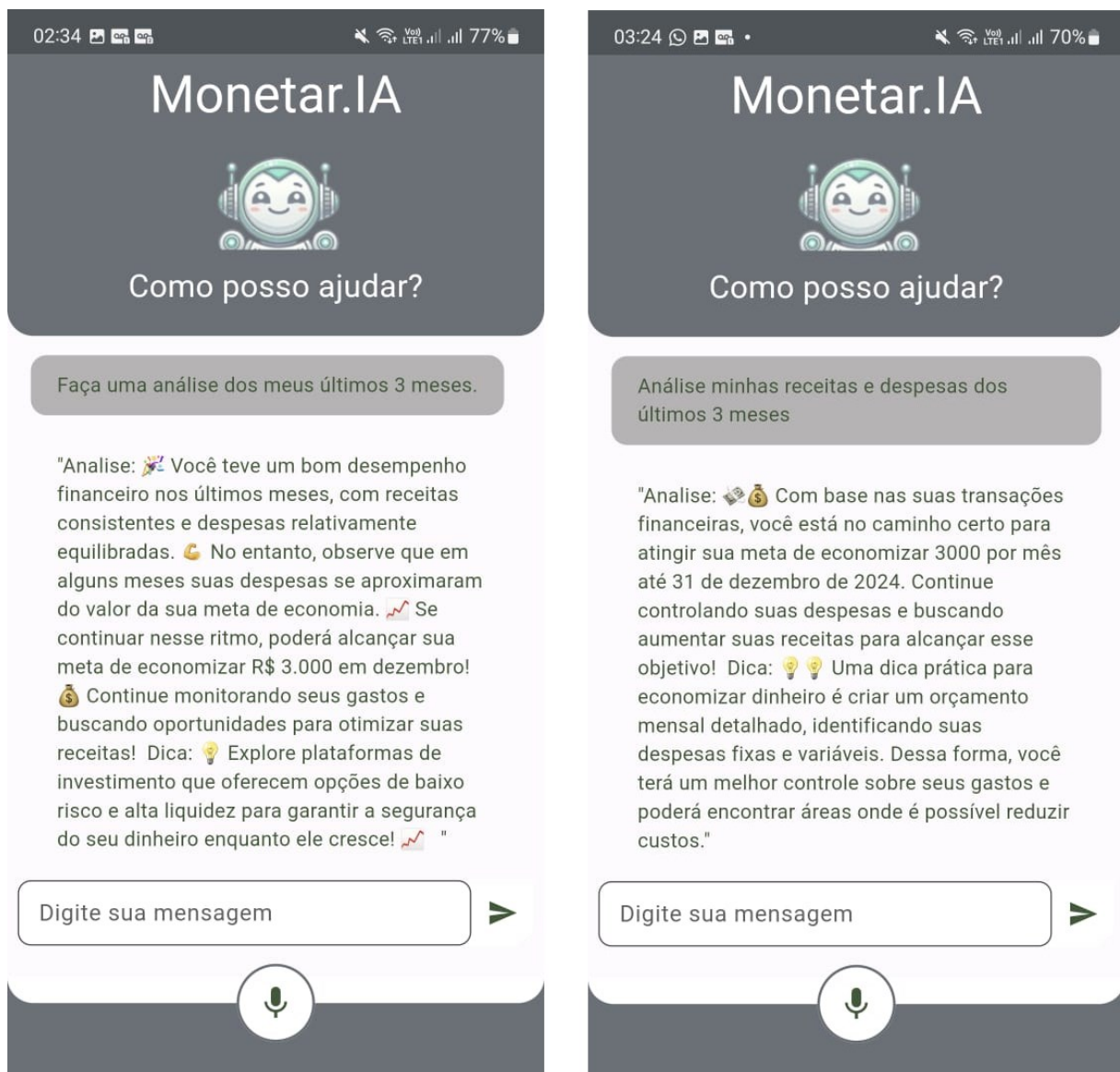


Fonte: Elaborado pelos autores (2024).

A principal funcionalidade da aplicação Monetar.ia será o assistente virtual integrado com inteligência artificial generativa. Esse assistente atuará como um

consultor financeiro pessoal, oferecendo sugestões personalizadas para aprimorar o consumo e os hábitos financeiros do usuário, além de realizar análise preditiva baseado nas informações econômicas cadastradas na ferramenta, projetando cenários e identificando tendências. Na imagem abaixo, pode-se notar como será o aspecto da funcionalidade:

Figura 24 - Visão geral da tela assistente virtual da Monetar.ia



Fonte: Elaborado pelos autores (2024).

Como descrito na imagem, o assistente virtual terá acesso as informações cadastradas na aplicação e poderá realizar uma gama de cálculos e respostas baseadas nas perguntas efetuadas pelo usuário. As respostas geradas pelo

assistente terão um alto nível de precisão, baseando-se nos dados de entrada fornecidos por meio da API e garantindo que as informações estejam sempre atualizadas e alinhadas com as necessidades do usuário. Além disso, as respostas serão cuidadosamente restritas ao contexto da gestão financeira pessoal, evitando desviar para tópicos não relacionados e mantendo o foco em áreas como orçamento, planejamento de despesas e metas financeiras.

Baseado nas funcionalidades propostas, nota-se que a Monetar.ia cumpre o objetivo proposto de ser uma ferramenta de fácil utilização e acessibilidade, que se propõe em ser um facilitador da gestão financeira pessoal dos usuários, com a assistência de um consultor financeiro personalizado com o uso da inteligência artificial generativa.

5 CONCLUSÃO

O objetivo desse trabalho científico foi propor uma abordagem inovadora para simplificar a gestão financeira pessoal por meio de uma aplicação móvel. A Monetar.ia propõe a inclusão de inteligência artificial generativa como uma assistente virtual personalizada que auxilia e educa o usuário sobre a condição financeira através da análise preditiva das informações inseridas no aplicativo, promovendo uma relação consciente e estratégica da gestão econômica.

A escolha por uma solução em formato de aplicativo móvel foi motivada pelo aumento significativo nos últimos anos do uso de aparelhos móveis para a gestão financeira e bancária, impulsionado principalmente pelo advento do pix e da crescente do *mobile banking*. Esse cenário reflete a preferência dos usuários por soluções práticas, acessíveis e intuitivas para a gestão financeira pessoal.

O desenvolvimento da Monetar.ia foi estruturada com uma API robusta baseada em uma arquitetura de microsserviços, projetada para oferecer alta confiabilidade e escalabilidade. Essa arquitetura permite que a aplicação mantenha desempenho, suporte um alto número de usuários ativos, facilita a manutenção e atualização ao longo da vida útil da aplicação sem comprometer o funcionamento da aplicação, garantindo longevidade.

As interfaces da ferramenta foram pensadas para serem o mais simples possível, com o objetivo de proporcionar uma experiência de uso altamente intuitiva, além disso, a implementação de uma interface de voz foi identificada como uma maneira eficaz de garantir acessibilidade e facilitar o uso da aplicação.

Por último, foram testados os modelos de inteligência artificial generativa Gemini e ChatGPT para a integração na Monetar.ia. No geral, o ChatGPT destacou-se como a solução mais adequada ao projeto, fornecendo respostas mais consistentes, sem apresentar anomalias ou imprecisões. A Gemini, em contrapartida, apresentando respostas demasiadamente prolixas e cálculos incorretos das informações cadastradas no aplicativo. No momento que a imprecisão dos cálculos foi notada, durante o desenvolvimento do produto, tomou-se pela decisão de alterar o modelo de inteligência artificial generativa usado pela Monetar.ia, pois, falhas como essa comprometem a confiabilidade do produto danificando a experiência do usuário e a reputação que uma aplicação de gestão financeira necessita zelar.

REFERÊNCIAS

- BANCO CENTRAL DO BRASIL. Endividamento de Risco no Brasil: Atualização: impacto no Sistema Financeiro Nacional e qualificação dos indicadores. **Série Cidadania Financeira**, Brasília, 8 ed., novembro 2023. Disponível em: https://www.bcb.gov.br/content/cidadaniafinanceira/documentos_cidadania/serie_cidadania/serie_cidadania_financeira_8_endividamento_risco_2ed.pdf. Acesso em: 20 ago. 2024.
- BANCO CENTRAL DO BRASIL. Gestão de Finanças Pessoais. **Caderno de Educação Financeira**, Brasília, 2013. Disponível em: https://www.bcb.gov.br/content/cidadaniafinanceira/documentos_cidadania/Cuidando_do_seu_dinheiro_Gestao_de_Financas_Pessoais/caderno_cidadania_financeira.pdf. Acesso em: 15 set. 2024.
- BANCO CENTRAL DO BRASIL. Relatório de Cidadania Financeira. **Série Cidadania Financeira**, Brasília, 2021. Disponível em: https://www.bcb.gov.br/content/cidadaniafinanceira/documentos_cidadania/RIF/Relatorio_de_Cidadania_Financeira_2021.pdf. Acesso em: 20 ago. 2024.
- BARROS, Isabelle Guimarães de; SANTOS, Carlos Felipe Araujo dos. **Apostila de Introdução ao CSS**. Niterói: Petele, 2008.
- BENTO, Victor de Oliveira; ABREU, Pollyanna Miranda de. Interface De Voz Do Usuário: Os Impactos E Metodologias Utilizadas Para O Começo Da Construção De Uma Excelente Interface De Voz. **II Jornada Científica De Arquitetura e Engenharia**, [s. l.], 2022. Disponível em: <http://repositorio.unis.edu.br/handle/prefix/2487>. Acesso em: 17 set. 2024.
- CAMPOS, Marcos Vinícius Souza. **Desenvolvimento de APIs Baseadas em Rest Para Integração e Construção de Aplicações**. Orientador: Alberto Martins Júnior. 2013. Trabalho de Conclusão de Curso (Bacharelado em Análise de Sistemas e Tecnologia da Informação) - Faculdade de Tecnologia de Americana, Americana, 2013. Disponível em: https://ric.cps.sp.gov.br/bitstream/123456789/1596/1/20131S_CAMPOSMarcosViniciusSouza_CD1589.pdf. Acesso em: 20 out. 2024.
- CAPROVER. **Scalable, Free and Self-hosted PaaS**, [S. l.], 2024. Disponível em: <https://caprover.com/>. Acesso em: 19 out. 2024.
- CARDOSO, Leandro da Conceição. **Design de Aplicativos**. 1. ed. Curitiba: Intersaberes, 2022. *E-book*. Disponível em: <https://plataforma.bvirtual.com.br>. Acesso em: 17 set. 2024.
- CHACON, Scott; STRAUB, Ben. **Pro Git: Everything you need to know about git**. 2. ed. [S. l.]: Apress, 2024. Disponível em: <https://git-scm.com/doc>. Acesso em: 27 ago. 2024.
- CONCEIÇÃO, Melissa Tidori da; PINTO, Giuliano Scombatti. Arquitetura De Microserviços. **Interface Tecnológica**, São Paulo, v. 18, n. 2, 31 dez. 2021.

Disponível em: <https://revista.fatectq.edu.br/interfacetecnologica/article/view/1538>. Acesso em: 20 out. 2024.

COSTA, Carlos Alberto. **A aplicação da Linguagem de Modelagem Unificada (UML) para o suporte ao projeto de sistemas computacionais dentro de um modelo de referência**. [S. l.]: Scielo Brasil, 2001. Disponível em: <https://www.scielo.br/j/gp/a/RRQQ7mKTFztQXK9Sz7BKtWQ/#>. Acesso em: 7 nov. 2024.

CRUZ, Vitor da Silva; PETRUCELLI, Erick Eduardo; SOTTO, Eder Carlos Salazar. A linguagem Javascript como alternativa para o desenvolvimento de aplicações multiplataforma. **Interface Tecnológica**, [S. l.], 30 dez. 2018. Disponível em: <https://revista.fatectq.edu.br/index.php/interfacetecnologica/article/view/476>. Acesso em: 10 nov. 2024.

DART. **Paint Your UI to Life**, [S. l.], 2024. Disponível em: <https://dart.dev/>. Acesso em: 20 out. 2024.

DIGITAL OCEAN. **Learn here. Dream here. Develop here**, [S. l.], 2024. Disponível em: <https://code.visualstudio.com/>. Acesso em: 19 out. 2024.

DOCKER. **Develop faster. Run anywhere**, [S. l.], 2024. Disponível em: <https://code.visualstudio.com/>. Acesso em: 18 out. 2024.

DOURADO, Marcos Antonio Durães; CANEDO, Edna Dias. Usability Heuristics for Mobile Applications: A Systematic Review. **Scitepress**, Brasília, 2018. Disponível em: <https://www.scitepress.org/papers/2018/67814/67814.pdf>. Acesso em: 9 set. 2024.

ESKELSEN, Marina Krause. **Heurísticas de Usabilidade para Dispositivos Móveis**. Orientador: Marcos Alexandre Rose Silva. 2023. Trabalho de conclusão de curso (Técnico em Sistemas para a Internet) - Universidade Federal de Santa Maria, Santa Maria, 2023. Disponível em: https://repositorio.ufsm.br/bitstream/handle/1/32583/Eskelsen_Marina_Krause_2023_TCC.pdf?sequence=1. Acesso em: 9 set. 2024.

FASTAPI. **FastAPI Frameworkm High Perfomance, Easy to Learn, Fast to Code, Ready for Production**, [S. l.], 2024. Disponível em: <https://www.python.org/about/>. Acesso em: 20 out. 2024.

FEBRABAN. Pesquisa Febraban de Tecnologia Bancária. **Febraban**, [S. l.], 2024. Disponível em: <https://cmsarquivos.febraban.org.br/Arquivos/documentos/PDF/Pesquisa%20Febraban%20de%20Tecnologia%20Banc%C3%A1ria%202024.pdf>. Acesso em: 8 set. 2024.

FELTRIN, Larissa. **APIs REST: O Impacto das APIs REST na Evolução Digital das Empresas**. Orientador: Diógenes de Oliveira. 2023. Trabalho de Conclusão de Curso (Tecnólogo em Análise e Desenvolvimento de Sistemas) - Faculdade de Tecnologia de Americana, Americana, 2023. Disponível em:

https://ric.cps.sp.gov.br/bitstream/123456789/15349/1/20232S_Larissa%20Feltrin_OD1789.pdf. Acesso em: 20 out. 2024.

FIGMA. **Think bigger. Build faster.** [S. l.], 2024. Disponível em: <https://www.figma.com/>. Acesso em: 1 out. 2024.

FLUTTER. **Build for any Screen**, [S. l.], 2024. Disponível em: https://flutter.dev/?gad_source=1&gclid=Cj0KCQjw99e4BhDiARIsAISE7P96m7qrR-wLqL43Wlyvict9lcaPrAYzQb6iTs3JvWbjd6XiyIFoOL8aAkNREALw_wcB&gclidsrc=aw.ds. Acesso em: 20 out. 2024.

FONSECA, José Wladimir Freitas da. **Administração financeira e orçamentária**. 2. ed. Curitiba: IESDE Brasil, 2013.

FREITAS, Ed. **Flutter: Succintly**. Morrisville: Syncfusion, 2019. Disponível em: <https://pergamum.sc.senac.br/pergamumweb/vinculos/000035/0000355b.pdf>. Acesso em: 17 set. 2024.

FREITAS, Guilherme Dutra Diniz de. **Integração do Github com uma Ferramenta de Gerenciamento de Projetos de Software**. Orientador: Dennis Sávio Martins da Silva. 2017. Trabalho de conclusão de curso (Bacharel em Sistemas de Informação) - Universidade Federal do Piauí, Picos, 2017.

GITHUB. **Let's build from here**. [S. l.], 2024. Disponível em: <https://github.com/>. Acesso em: 1 out. 2024.

GOMES, Rafael; SOUZA, Rodrigo. **Docker - Infraestrutura como código, com autonomia e replicabilidade**. [S. l.]: [s. n.], 2015.

GRADIN, Felipe Amaro. **Um Estudo Sobre o Desenvolvimento Mobile Utilizando Flutter**. Orientador: Alexandre Lazzaretti. 2019. Trabalho de Conclusão de Curso (Tecnólogo em Sistemas para Internet) - Instituto Federal Sul-rio-grandense, Passo Fundo, 2019. Disponível em: <https://painel.passofundo.ifsul.edu.br/uploads/arq/20210420155132992831095.pdf>. Acesso em: 20 out. 2024.

HABBEMA, Hugo. API's com Python: Desenvolvimento Rápido de APIs com FastAPI. **Medium**, [S. l.], jul. 2023. Disponível em: <https://medium.com/@habbema/apis-com-python-8e792f2e40e4>. Acesso em: 20 out. 2024.

HALFELD, Mauro. **Investimentos: como administrar melhor seu dinheiro**. São Paulo: Fundamento Educacional, 2007.

KNECHTEL, Maria do Rosário. Metodologia da pesquisa em educação: uma abordagem teórico-prática dialogada. Curitiba: Intersaberes, 2014. *E-book*. Disponível em: <https://plataforma.bvirtual.com.br>. Acesso em: 18 ago. 2024.

LEVINO, Natallya de Almeida; SANTOS, Anderson Moreira A. dos. **Finanças pessoais para iniciantes**. Maceió: Edufal, 2019.

MENDES, Jadner Santiago. **Scrumban: Uma Sugestão de Método de Gestão de Trabalho Aplicado à Residência em TI da JFRN**. Orientador: Manoel Veras de Sousa Neto. 2022. Trabalho de conclusão de curso (Bacharel em Administração) - Universidade Federal do Rio Grande do Norte, Natal, 2022. Disponível em: <https://repositorio.ufrn.br/handle/123456789/48832>. Acesso em: 16 out. 2024.

NEWMAN, Sam. **Criando Microsserviços: projetando sistemas com componentes menores e mais especializados**. 2. ed. São Paulo: Novatec, 2022. 685 p. ISBN 978-65-86057-88-1.

NGINX. **Nginx**, [S. l.]: [s. n.], 2024. Disponível em: <https://nginx.org/en/>. Acesso em: 19 out. 2024.

NOTION. **A Próxima Geração de Anotações e Documentos**. [S. l.], 2024. Disponível em: <https://www.notion.so/pt/product/docs>. Acesso em: 16 out. 2024.

OLIVEIRA, Ruy Flávio de. **Inteligência Artificial**. Londrina: Educacional, 2018.

OZELIN, Welinton. **Desenvolvimento Front-End Para WEB: HTML e CSS**. Centro Universitário de Adamantina, Adamantina, 2017. Disponível em: <https://camda.com.br/assets/arquivos/informativos/Apostila%20-%20HTML%20e%20CSS.pdf>. Acesso em: 10 nov. 2024.

PALESTINO, CAROLINE MUNHOZ CORRÊA. **Estudo de Tecnologias de Controle de Versões de Softwares**. Orientador: José Simão de Paula Pinto. 2015. Trabalho de conclusão de curso (Bacharel em gestão da informação) - Universidade Federal do Paraná, Curitiba, 2015.

PEDROSO, Robertha Pereira. **Apostila de HTML**. Nitorói: Petele, 2007.

PELEGRINI, Leonardo; CORINO, Marcos Vissoto; SILVA, Roger Sá da. **Um Estudo de Caso Sobre a Utilização de Containers Para Aplicações em Nuvem de Alta Disponibilidade**. [S. l.], 2022.

PEREIRA, Lucas Gabriel Nerys; DINIZ, Luciana Mara Freitas. **Comparativo entre arquitetura monolítica e arquitetura de microsserviços**. 2022. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) - Pontifícia Universidade Católica de Minas Gerais, Betim, 2022.

POSTGRESQL. **What is PostgreSQL?** [S. l.], 2024. Disponível em: <https://www.postgresql.org/about/>. Acesso em: 16 out. 2024.

PYTHON. **Python is Powerful... and Fast**, [S. l.], 2024. Disponível em: <https://www.python.org/about/>. Acesso em: 20 out. 2024.

RANE, Nitin Liladhar; CHOUDHARY, Saurabh; RANE, Jayesh. Gemini versus ChatGPT: applications, performance, architecture, capabilities, and implementation. **Journal of Applied Artificial Intelligence**, Mumbai, v. 5, p. 69 - 93, 20 mar. 2024. Disponível em:

https://www.researchgate.net/publication/379177267_Gemini_versus_ChatGPT_applications_performance_architecture_capabilities_and_implementation. Acesso em: 21 out. 2024.

REHKOPF, Max. Kanban vs. Scrum: que tipo de ágil é você? **Atlassian**, [s. l.], 2024. Disponível em: <https://www.atlassian.com/br/agile/kanban/kanban-vs-scrum>. Acesso em: 14 out. 2024.

REIS, ANA PAULA SANTOS. **Os principais fatores que influenciam no endividamento e inadimplência das famílias brasileiras**. Orientador: Andson de Freitas Viana. 2020. Trabalho de conclusão de curso (Bacharelado em Administração) - Unifametro, Fortaleza, 2020. Disponível em: <http://repositorio.unifametro.edu.br/handle/123456789/164?mode=full>. Acesso em: 29 ago. 2024.

RODRIGUES, Marcelo Felisberto. **Modelagem de software**: um estudo de caso em uma empresa de distribuição de frutas. Orientador: André Luís Silva. 2013. Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Produção) - Universidade Federal de Ouro Preto, Ouro Preto, 2013. Disponível em: https://www.professor.ufop.br/sites/default/files/andre/files/marcelo_felisberto_rodrigues.pdf. Acesso em: 7 nov. 2024.

RODRIGUES, Tatiane Daby de Fatima Faria; OLIVEIRA, Guilherme Saramago de; SANTOS, Josely Alves dos. As pesquisas qualitativas e quantitativas na educação. **Revista Atlas**, Rio de Janeiro, v. 2, ed. 1, p. 154-174, 27 dez. 2021. Disponível em: <https://revistaprisma.emnuvens.com.br/prisma/article/view/49>. Acesso em: 18 ago. 2024.

ROSÁRIO, Fábio Chassot do. **Desenvolvimento de aplicativos móveis multiplataforma**. Orientador: Márcio Angelo Matté. 2015. Trabalho de Conclusão de Curso (Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) - Universidade Tecnológica Federal Do Paraná, Medianeira, 2015. Disponível em: https://repositorio.utfpr.edu.br/jspui/bitstream/1/13394/1/MD_COADS_2015_2_05.pdf. Acesso em: 17 set. 2024.

RUSSELL, Stuart; NORVIG, Peter. **Artificial Intelligence**: a modern approach. 4. ed. Hoboken: Pearson, 2021.

SAMPAIO, Rafael Cardoso; NICOLÁS, Maria Alejandra; JUNQUILHO, Tainá Aguiar; SILVA, Luiz Rogério Lopes; FREITAS, Christiana Soares de; TELLES, Márcio; TEIXEIRA, João Senna; ESCÓSSIA, Fernanda da; SANTOS, Luiza Carolina dos. ChatGPT e outras IAs transformarão a pesquisa científica: reflexões sobre seus usos. **SciELO**, [S. l.], 2024. Disponível em: <https://www.scielo.br/j/rsocp/a/rfSfWXpWqJWgrbRktcpXq9v/#>. Acesso em: 21 out. 2024.

SANTOS, Erik Correa. **Técnicas de Modelagem de Banco de Dados**. Orientador: Alex Sandro Romeo de Souza Poletto. 2015. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Fundação Educacional de Assis, Assis, 2015. Disponível em:

<https://cepein.femanet.com.br/BDigital/arqTccs/1011330353.pdf>. Acesso em: 6 nov. 2024.

SAVOIA, José Roberto Ferreira; SAITO, André Taue; SANTANA, Flávia de Angelis. Paradigmas da educação financeira no Brasil. **SciELO**, Rio de Janeiro, v. 41, n. 6, 26 fev. 2008.

SCHENINI, Paulo Henrique; BONAVIDA, José Ricardo. **Finanças para não financistas**: princípios de economia para profissionais em mercados competitivos. 3. ed. Rio de Janeiro: Senac Rio, 2012.

SCRUM. What is Scrum? **Scrum.org**, [S. l.], 2024. Disponível em: <https://www.scrum.org/learning-series/what-is-scrum/>. Acesso em: 16 out. 2024.

SILVA, Júlio Cesar Leoncio da. **Um estudo de avaliação e documentação de arquiteturas de software na indústria**. Orientador: Uirá Kulesza. 2016. Dissertação (Mestrado em Engenharia de Software) - Universidade Federal do Rio Grande do Norte, Natal, 2016. Disponível em: <https://repositorio.ufrn.br/handle/123456789/22488>. Acesso em: 20 out. 2024.

SILVA, Pedro Igor Oliveira. **Banco de Dados**: análise comportamental de desempenho entre bancos de dados relacionais e não relacionais para aplicações web. Orientador: Maurício Serrano. 2023. Trabalho de conclusão de curso (Bacharel em Engenharia de Software) - Universidade de Brasília, Brasília, 2023. Disponível em: <https://bdm.unb.br/handle/10483/31>. Acesso em: 16 out. 2024.

SOARES, Josiel Ferreira. Análise preditiva: importância e vantagens para os negócios. **Runa**, [S. l.], 2018. Disponível em: <https://repositorio.animaeducacao.com.br/items/8816b6e3-2be8-46c5-9a75-0ba4a8ac1e8c>. Acesso em: 15 set. 2024.

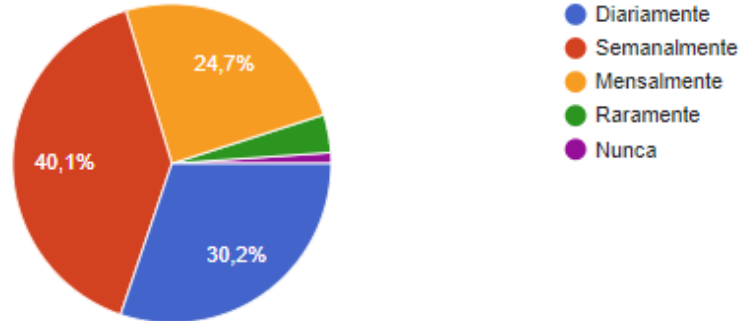
STADZISZ, Paulo César. Projeto de Software Usando a UML. **Aa**, [S. l.], 2002. Disponível em: <http://www.etelg.com.br/paginaete/downloads/informatica/apostila2uml.pdf>. Acesso em: 7 nov. 2024.

VENÂNCIO, Vinícius de Carvalho; MALAQUIAS, Rodrigo Fernandes. Tecnologia e planejamento financeiro pessoal: uma análise da percepção de alunos do curso de graduação em Ciências Contábeis. **RECSA - Revista Eletrônica de Ciências Sociais Aplicadas**, Garibaldi, v. 6, n. 1, 2017. Disponível em: <https://revista.fisul.edu.br/index.php/revista/article/view/73>. Acesso em: 24 nov. 2024.

VISUAL STUDIO CODE. **Code fast with AI**, [S. l.], 2024. Disponível em: <https://code.visualstudio.com/>. Acesso em: 17 out. 2024.

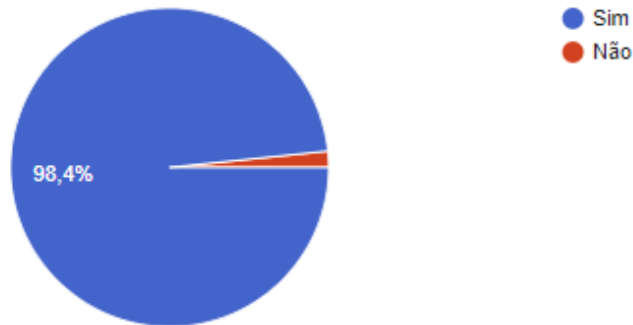
APÊNDICE A – Formulário de pesquisa semiestruturada com respostas

Gráfico 1 - Com que frequência você monitora suas finanças pessoais?



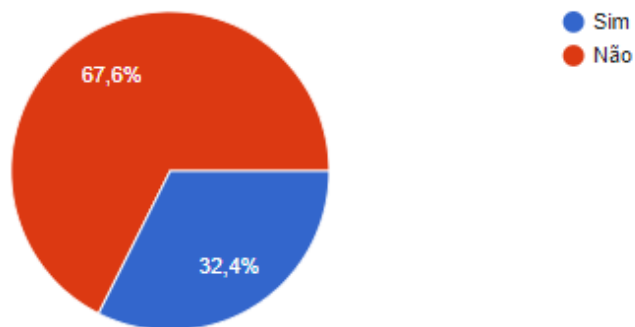
Fonte: Elaborado pelos autores (2024).

Gráfico 2 - Você acha útil ter uma visão geral dos seus gastos e recebimentos em um único lugar?



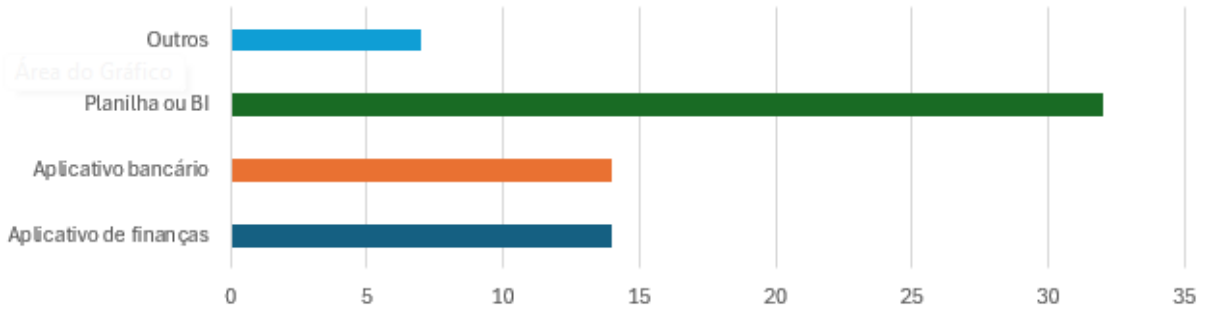
Fonte: Elaborado pelos autores (2024).

Gráfico 3 - Você utiliza alguma ferramenta ou aplicativo para gestão financeira atualmente?



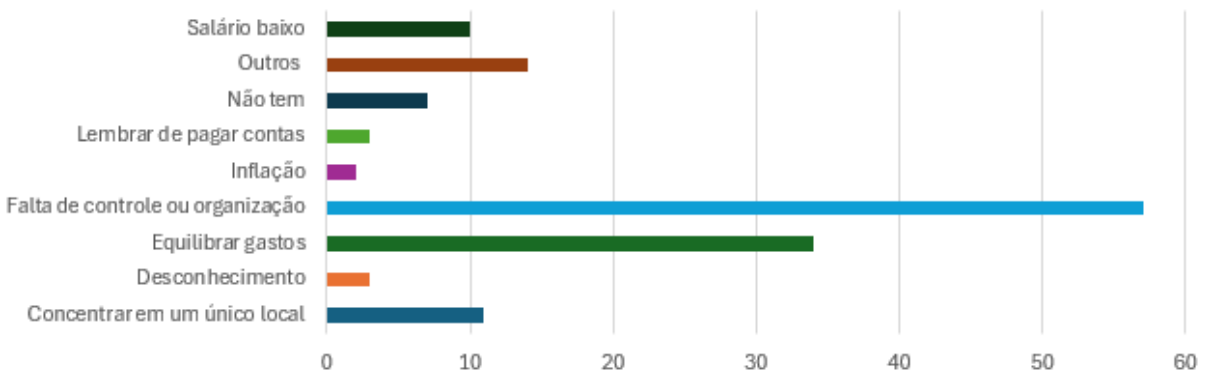
Fonte: Elaborado pelos autores (2024).

Gráfico 4 - Se sim, qual(is) ferramenta(s) ou aplicativo(s) você utiliza?



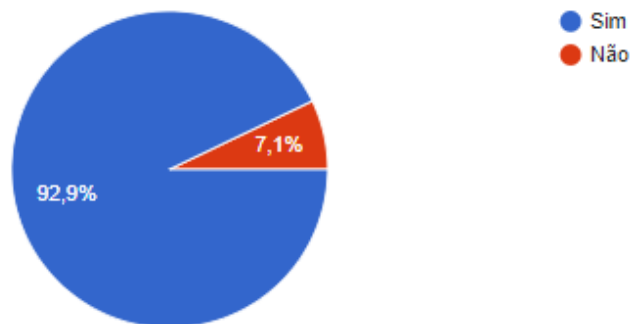
Fonte: Elaborado pelos autores (2024).

Gráfico 5 - Quais são os principais desafios que você enfrenta na gestão das suas finanças pessoais?



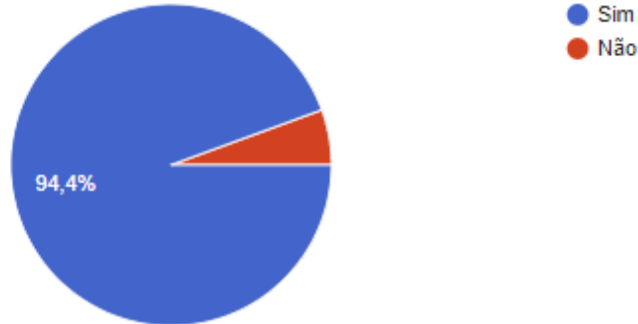
Fonte: Elaborado pelos autores (2024).

Gráfico 6 - Você gostaria de criar e acompanhar metas financeiras, como poupança, pagamento de dívidas e aumento de receitas, dentro do aplicativo?



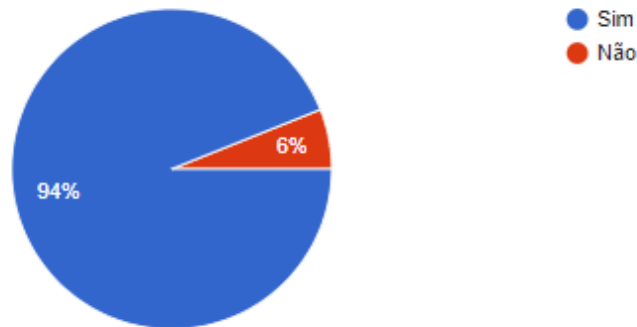
Fonte: Elaborado pelos autores (2024).

Gráfico 7 - Você consideraria útil um aplicativo que analise seus hábitos de consumo de forma inteligente (utilizando inteligência artificial) e forneça sugestões personalizadas de economia e otimização financeira?



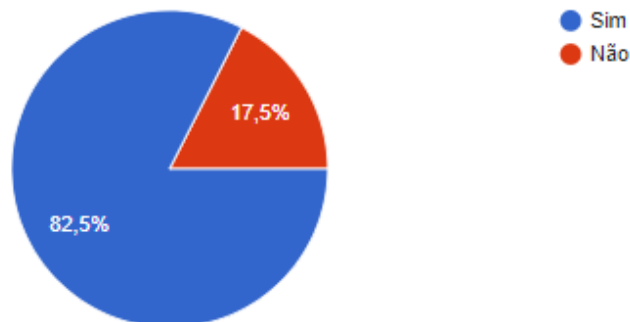
Fonte: Elaborado pelos autores (2024).

Gráfico 8 - Você acharia útil receber notificações e alertas personalizados sobre limites de gastos, vencimentos de contas e recebimentos esperados?



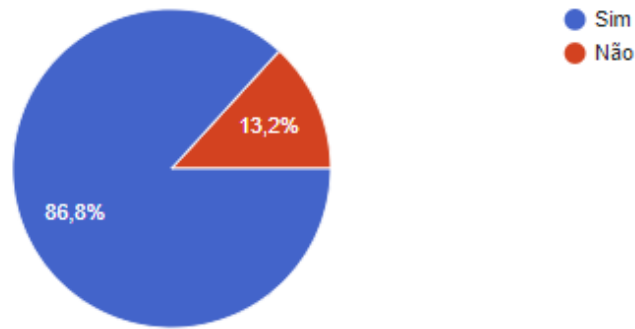
Fonte: Elaborado pelos autores (2024).

Gráfico 9 - Você gostaria de poder registrar suas despesas e receitas utilizando comandos de texto e voz em um aplicativo?



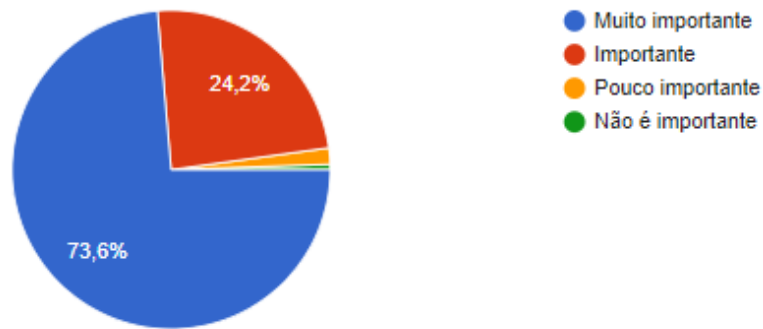
Fonte: Elaborado pelos autores (2024).

Gráfico 10 - Você acharia útil poder gerar e exportar relatórios detalhados de suas despesas e receitas em formatos como PDF ou Excel?



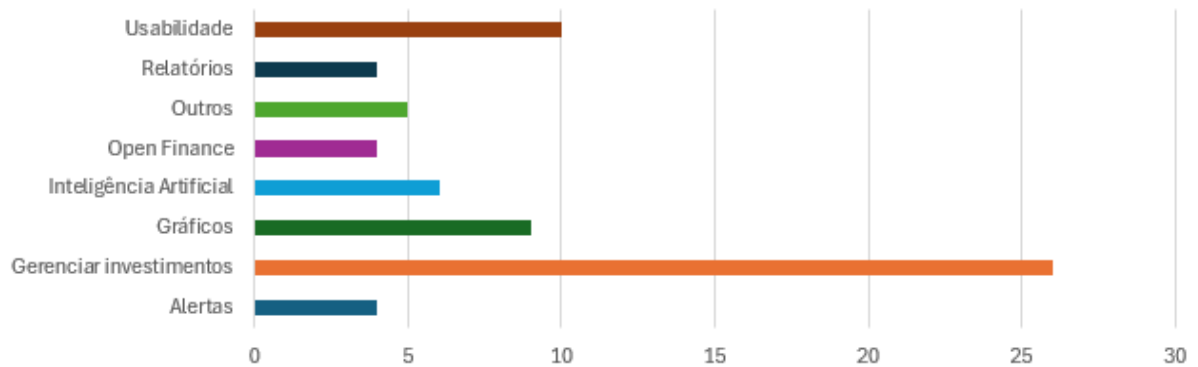
Fonte: Elaborado pelos autores (2024).

Gráfico 11 - Quão importante é para você que o aplicativo seja intuitivo e fácil de usar?



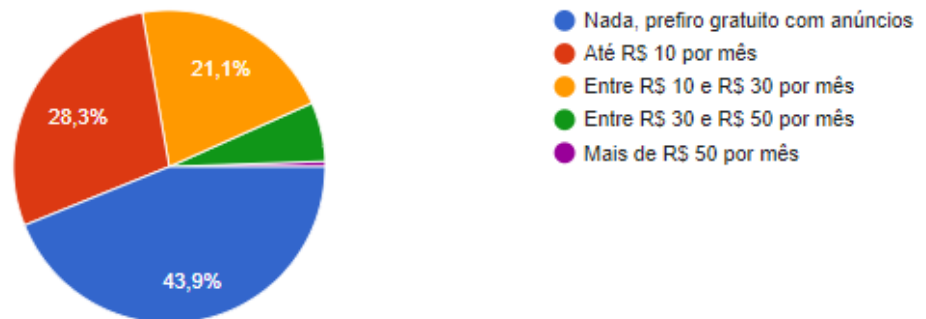
Fonte: Elaborado pelos autores (2024).

Gráfico 12 - Há alguma funcionalidade ou característica que você gostaria de ver no aplicativo?



Fonte: Elaborado pelos autores (2024).

Gráfico 13 - Quanto você estaria disposto a pagar por um aplicativo com essas funcionalidades?



Fonte: Elaborado pelos autores (2024).