UNIVERSIDADE FEDERAL DO CEARÁ

INSTITUTO UNIVERSIDADE VIRTUAL

CURSO DE GRADUAÇÃO EM SISTEMAS E MÍDIAS DIGITAIS

JOÃO LEVY CRISTIAN DOS ANJOS

DESENVOLVIMENTO DE UM APLICATIVO MÓVEL PARA AUXILIAR NO GERENCIAMENTO FINANCEIRO DE JOVENS E ADULTOS

FORTALEZA

2023

JOÃO LEVY CRISTIAN DOS ANJOS

DESENVOLVIMENTO DE UM APLICATIVO MÓVEL PARA AUXILIAR NO GERENCIAMENTO FINANCEIRO DE JOVENS E ADULTOS

Relatório Técnico apresentado ao Curso de Graduação em Sistemas e Mídias Digitais do Instituto Universidade Virtual da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Sistemas e Mídias Digitais.

Orientador: Prof. Dr. Henrique Barbosa Silva

FORTALEZA

2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

A619d Anjos, João Levy Cristian dos.
Desenvolvimento de um aplicativo móvel para auxiliar no gerenciamento financeiro de jovens e adultos / João Levy Cristian dos Anjos. – 2023.
60 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Instituto UFC Virtual, Curso de Sistemas e Mídias Digitais, Fortaleza, 2023.
Orientação: Prof. Dr. Henrique Barbosa Silva.

1. Educação financeira. 2. Desenvolvimento mobile iOS. 3. Gerenciamento financeiro. I. Título.
CDD 302.23

JOÃO LEVY CRISTIAN DOS ANJOS

DESENVOLVIMENTO DE UM APLICATIVO MÓVEL PARA AUXILIAR NO GERENCIAMENTO FINANCEIRO DE JOVENS E ADULTOS

Relatório Técnico apresentado ao Curso de Graduação em Sistemas e Mídias Digitais do Instituto Universidade Virtual da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Sistemas e Mídias Digitais.

Aprovada em: 17 de julho de 2023

BANCA EXAMINADORA

Prof. Dr. Henrique Barbosa Silva (Orientador)
Universidade Federal do Ceará (UFC)

Profa. Dra. Maria de Fátima Costa de Souza
Universidade Federal do Ceará (UFC)

Prof. Dr. Leonardo Oliveira Moreira
Universidade Federal do Ceará (UFC)

AGRADECIMENTOS

Primeiramente à Deus, por sempre me mostrar que eu sou capaz de superar cada desafio que me foi imposto nesta vida.

À minha mãe, Francisca, por ter lutado todos esses anos para criar sozinha seus 4 filhos e por ter me ensinado a ser essa pessoa que luta pela vida. Hoje, seu filho mais novo irá se formar na Universidade Federal do Ceará e se tornar o primeiro de nossa família a ter uma graduação de nível superior.

À minha avó, Palmira,por ter me amado tanto na breve passagem que tivemos nesta vida. Todo o seu amor germinou, cresceu e virou uma linda flor que levo comigo para sempre no coração.

Ao meu companheiro, Marley, por ter estado ao meu lado durante todo esse desafio. Sendo esta pessoa incrível que você é, me energizando quando eu estava cansado, me ajudando a encontrar o caminhos quando estava perdido e por sempre cuidar tanto de mim.

Aos meus amigos, por todo apoio e cuidado durante toda a graduação. Obrigado por terem tido presentes, por me ajudarem a entender melhor a vida e como ela é linda de viver ao lado de vocês.

Aos meus amigos e mentores do Apple Developer Academy IFCE, por terem sempre acreditado em mim e me ajudado a se tornar esse profissional completo que hoje sou. Eu não seria nada sem vocês, obrigado por tudo.

Agradeço ao meu orientador, Prof. Henrique Silva, por todo o auxilio durante o desenvolvimento deste trabalho e por sempre ter acreditado em mim.

Por fim, agradeço à mim mesmo por nunca ter desistido de nenhum desafio. Obrigado por aprender com cada erro, por acreditar na bondade, por lutar através de cada desafio até que ele seja vencido. Por ter me mostrado que eu sou capaz de superar minhas próprias limitações.

"Your time is limited, so don't waste it living someone else's life."

(SJ.)

RESUMO

Com o advento da Tecnologia da Informação (TI), os computadores pessoais se tornaram tão comuns que hoje em dia cabem na palma da sua mão. Os dispositivos móveis e seus milhares de aplicativos fizeram com que o contexto se tornasse mais fácil para o acesso à informação. Apesar disso, muitos conhecimentos ainda se mostram de acesso muito restrito, como a Educação Financeira. A falta de educação financeira pode ter como uma das principais causas a falta de contato com informações claras e acessíveis relacionadas a esta questão, o que pode levar a uma vida adulta sem saber lidar com dinheiro. Saber poupar não é um dom, é uma habilidade que precisa ser desenvolvida como desenhar ou andar de bicicleta. Tendo isso em mente, a necessidade de educar-se financeiramente nunca foi tão necessária. No Brasil, o currículo básico da educação financeira não dispõe de ementas que abordam essa temática, o que acaba sendo sentido quando os jovens estão adentrando no mercado financeiro, onde a falta de conhecimento relacionado ao gerenciamento de recursos pode levar a tomar decisões incorretas.

A falta de educação financeira na infância e adolescência, a constante necessidade de se sentir parte de um determinado grupo e a lógica do próprio mercado financeiro são alguns dos fatores que têm responsabilidade indireta nesse cenário de endividamento. Para contribuir para uma melhoria na qualidade de vida dessa população, é importante discutir modelos de educação financeira de fácil acesso e que abordem essa temática. Tendo em vista esse contexto, foi pensado em um aplicativo que facilitasse o aprendizado de jovens e adultos: o *PennyWise*. A proposta desse aplicativo é organizar as movimentações de seus usuários para que eles possam visualizar melhor suas finanças.

Portanto, neste presente relatório é apresentado o contexto para o desenvolvimento de uma aplicação móvel *iOS* que se utiliza de uma estratégia de organização de gastos para ajudar na organização financeira, aspectos da ideação do aplicativo, informações sobre seu público-alvo, *design* da interface, detalhes da codificação do aplicativo e seu processo de publicação na *App Store*. Por fim, é apresentado o aplicativo finalizado e seu *link* para *download* na loja.

Palavras-chave: Educação Financeira. Gerenciamento Financeiro. Desenvolvimento *Mobile iOS*.

ABSTRACT

With the rapid advancement of Information Technology (IT), personal computers have become so commonplace that they now fit in the palm of your hand. Mobile devices and their countless applications have made accessing information easier than ever before. However, despite this progress, certain areas of knowledge remain highly restricted, including Financial Education. The lack of financial literacy can be attributed, in part, to the absence of clear and accessible information on the subject, resulting in a lack of essential money management skills in adulthood. Saving money is not an innate talent; it is a skill that needs to be cultivated, much like drawing or riding a bicycle. Given this reality, the importance of financial education has never been more apparent. In Brazil, the basic education curriculum fails to address this critical aspect, leaving young individuals ill-equipped to navigate the financial landscape and make informed decisions about managing their resources.

The lack of financial education during childhood and adolescence, coupled with the constant pressure to fit into a particular social group and the dynamics of the financial market itself, are contributing factors to the prevailing issue of excessive indebtedness. To improve the quality of life for this population, it is crucial to explore easily accessible models of financial education that address these challenges. In response to this need, the concept of a mobile application called "PennyWise"was conceived, aiming to facilitate financial learning for both young people and adults. This application's proposal is to streamline users' financial transactions and provide a clearer overview of their financial situation.

Therefore, this report presents the context for developing an iOS mobile application that leverages an expenditure organization strategy to enhance financial management. It encompasses various aspects, including the app's conceptualization, insights into the target audience, user interface design, details of the app's coding, and the process of publishing it on the App Store. Finally, the completed application is showcased, along with the link for downloading it from the store.

Keywords: Financial education. Financial Management. iOS development.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE CÓDIGOS-FONTE

LISTA DE ABREVIATURAS E SIGLAS

AppStore	Loja de aplicativos da empresa *Apple Inc.*
ARM	*Advanced RISC Machines*
BNCC	Base Nacional Comum Curricular
CEO	Chefe do Escritório Executivo
DCN	Diretrizes Curriculares Nacionais
ENEF	Estratégia Nacional de Educação Financeira
IHC	Interação Humano-Computador
MVC	*Model View Controller*
MVVM	*Model View ViewModel*
NLU	*Natural Language Understanding*
PCN	Parâmetros Curriculares Nacionais
SPC	Serviço de Proteção ao Crédito
TI	Tecnologia da Informação
TT	Temas Transversais

SUMÁRIO

1 INTRODUÇÃO

A educação é um dos grandes pilares da nossa sociedade. Com o advento da TI, a educação tornou-se cada vez mais acessível na vida das pessoas. Nesta nova era, a área de finanças foi uma das que mais se beneficiou nescom bancos digitais descomplicados e linhas de crédito facilitadas, cada vez mais comuns no dia a dia dos jovens.

Segundo dados do Serviço de Proteção ao Crédito (SPC), um total de 62,08 milhões de consumidores estão negativados. O número equivale a algo como 40,2% da população adulta. No Sudeste, região que abriga a maior fatia da população, o número de negativados chegou a Nordeste (16,69 milhões, ou 40,9% da população adulta local). No Sul, são 8,31 milhões de consumidores, ou 36,4% da população adulta local, a menor entre as regiões. Já no Centro-Oeste, o contingente de negativados foi de 5,00 milhões, ou 42,1% da população adulta local. Por fim, no Norte, os negativados somam 5,62 milhões, a maior proporção adulta local: 46,2% (SPC Brasil, 2019).

Na abertura do dado por faixa etária, o destaque é da população com idade entre 30 e 39 anos. Nessa faixa, mais da metade (51,1%) está com restrição no nome, totalizando 17,61 milhões de pessoas. Também merece destaque o fato de porcentagem significativa da população com idade entre 25 a 29 (43,7%) estar negativada. Entre os mais jovens, com idade de 18 a 24 anos, a proporção cai para 16,8%. Na população idosa, considerando-se a faixa etária entre 65 a 84 anos, a proporção é de 32,8% (SPC Brasil, 2019).

A raiz deste problema é proveniente de inúmeras causas. Dentre elas, a falta de educação financeira na infância e adolescência, que se mostra como uma das principais. Se na nossa criação ou na nossa formação básica não tivermos contato com informações claras e acessíveis relacionadas a questões financeiras, a tendência é que a gente cresça sem saber lidar com dinheiro. Além disso, o acesso facilitado ao crédito para jovens universitários ou que acabaram de ingressar no mercado de trabalho são fatores que têm responsabilidade nesse cenário de endividamento. Por mais positiva que pareça, a facilidade de contratar empréstimos ou ter um cartão de crédito pode se tornar uma grande vilã nas mãos de quem ainda não sabe cuidar das próprias finanças.

Neste prisma, a necessidade de educar-se financeiramente nunca foi tão aparente. O currículo básico da educação financeira do Brasil não dispõe de ementas que abordam essa temática, impactando jovens estão adentrando no mercado financeiro, onde a falta de conhecimento relacionado ao gerenciamento de recursos ajuda na tomada de decisoes incorretas

podendo levar este jovens ao endividamento precoce.

1.1 Objetivos

Tendo como pressuposto a grande necessidade ao acesso à educação financeira, é levantada a questão: como um aplicativo móvel pode auxiliar os jovens entre 18 e 29 anos a educarem e gerirem seus recursos financeiros? Para responder tal questionamento, o presente trabalho tem como objetivo geral desenvolver um aplicativo movel que auxilie no gerenciamento financeiro de jovens e adultos.

Como objetivos específicos, este trabalho visa: (1) definir um conjunto de funcionalidades que sejam relevantes para o usuário final; (2) Especificar o uso das técnicas e padrões utilizados no desenvolvimento do aplicativo; (3) e, por fim, explanar todas as funcionalidades implementadas no aplicativo iOS;

2 FUNDAMENTAÇÃO TEÓRICA

Nesse capítulo, reúnem-se todos os conceitos necessários para a compreensão deste trabalho. São explicados conceitos de educação financeira, seu impacto na sociedade, sua disseminação no ensino de base e a tentativa de adesão da mesma de forma permanente. Também, é levantada questões relacionadas à inadimplência de jovens.

2.1 Educação Financeira

Atualmente, nossa sociedade enfrenta uma lacuna na compreensão e conhecimento dos métodos envolvidos na educação financeira. É essencial que as pessoas saibam como se comportar em diferentes situações e tomem decisões acertadas de acordo com cada contexto.

A educação financeira é um processo de aprendizagem essencial para lidar com as finanças pessoais. Por meio dela, a sociedade tem a oportunidade de desenvolver uma visão crítica sobre o uso do dinheiro. Prover as pessoas com esse conhecimento não traz benefícios somente para o indivíduo, mas também para a sociedade como um todo, pois contribui para uma economia mais saudável. Nesse contexto, o endividamento deixa de ser uma solução para obter dinheiro e passa a ser reconhecido como um problema a ser enfrentado. Essa importância da educação financeira está consagrada na Constituição brasileira, como evidenciado por Cordeiro *et al.* (2018) no trecho a seguir:

> A atual Constituição brasileira vincula a educação ao pleno desenvolvimento da pessoa e a seu preparo para o exercício da cidadania. Desta forma a educação financeira entra com essa participação cidadã, uma vez que esta viabiliza o entendimento da sociedade sobre as finanças pessoais e nacionais.

Até o ano de 2010, eram escassas as iniciativas voltadas para a educação financeira no Brasil. Podemos dizer que seu surgimento se deu com a criação da Estratégia Nacional de Educação Financeira (ENEF), por meio do Decreto 7397/2010, publicado no Diário Oficial da União em 22 de dezembro de 2010 (CORDEIRO *et al.*, 2018). Desde então, a importância da educação financeira tem ganhado destaque em diversos setores, inclusive na educação escolar. Quando se trata do uso de recursos pedagógicos com foco no ensino da educação financeira, Negri (2010) afirmou que:

> Educacao Financeira e um processo educativo que, por meio de aplicacao de metodos proprios, desenvolve atividades para auxiliar os consumidores a orcar

> e gerir a sua renda, a poupar e a investir; sao informacoes e formacoes significativas para que um cidadao exerca uma atividade, trabalho, profissao e lazer, evitando tornarem-se vulneraveis as armadilhas impostas pelo capitalismo.

2.1.1 Base Nacional Comum Curricular

Embora a educação financeira esteja presente no currículo escolar brasileiro, é importante destacar que ainda não foi devidamente incluída nos Parâmetros Curriculares Nacionais nem nas Diretrizes Curriculares Nacionais (DCN). No entanto, existem esforços em andamento para incorporar essa temática por meio dos Parâmetros Curriculares Nacionais (PCN), que possibilitam a inserção de novos temas, chamados de Temas Transversais (TT), no currículo escolar. Essa iniciativa visa promover a abordagem da educação financeira como parte integrante da formação dos estudantes.

Segundo o documento de apresentacao dos TT, eles sao "questões sociais consideradas relevantes", "problemáticas sociais atuais e urgentes, consideradas de abrangência nacional e ate mesmo de caráter universal" (MINISTÉRIO DA EDUCAÇÃO, 1997).

Como resultado de todos esses esforços, foi desenvolvido um material didático específico para o ensino médio, com o objetivo de promover a disseminação dos conhecimentos nessa área. Esse material foi implementado em um projeto piloto que abrangeu 891 escolas públicas de ensino médio, distribuídas em seis unidades federativas.

> Tal material tem objetivos de promover e fomentar a cultura de Educacao Financeira no pais, ampliar a compreensao do cidadao, para que seja capaz de fazer escolhas conscientes quanto a administracao de seus recursos e contribuir para a eficiencia e a solidez dos mercados financeiros, de capitais, de seguros, de previdencia e de capitalizacao (CORDEIRO *et al.*, 2018).

Infelizmente, atualmente, ainda existe uma resistência relacionada à inserção dessa temática na Base Nacional Comum Curricular (BNCC) – Ensino Fundamental, de acordo com (MARTINS, 2004):

> O aluno não estuda noções de comércio, economia, finanças ou impostos. O sistema educacional ignora o assunto "dinheiro", algo incompreensível, já que a alfabetização financeira é fundamental para ser bem-sucedido em um mundo complexo. [...] Não tenho dúvida de que essa falha é responsável por muitos fracassos pessoais e familiares.

No entanto, acredita-se que a incorporação da educação financeira no contexto escolar pode desempenhar um papel fundamental no fortalecimento da cidadania. Ao fornecer e apoiar iniciativas que auxiliem a população a tomar decisões financeiras mais autônomas,

a escola assume a responsabilidade de formar cidadãos conscientes e comprometidos. Dessa forma, a educação financeira no ambiente escolar tem o potencial de capacitar os indivíduos para lidar de forma mais eficaz com suas finanças e promover um desenvolvimento socioeconômico mais equilibrado (CORDEIRO *et al.*, 2018).

2.2 Inadimplência dos jovens

Ao abordarmos a problemática da inadimplência entre os jovens atualmente, é importante mencionar uma das principais causas dessa questão: o consumismo. De fato, o consumismo tem sido amplamente explorado pelo marketing agressivo, especialmente voltado para o público jovem, com a intenção de transmitir a mensagem de que o que se possui não é suficiente e que sempre haverá algo melhor para ser comprado, levando a uma constante e excessiva utilização dos recursos financeiros.

Outro grande fator que pode ser apontado como um vilão nessa problemática são as instituições financeiras, que oferecem financiamentos, cheques e cartões de crédito de forma facilitada, incentivando o consumo desde cedo. Isso ocorre porque esse público não foi devidamente preparado e educado para o planejamento financeiro (MOTA *et al.*, 2016). As consequências dessa situação são claramente observáveis: o endividamento, devido às altas e abusivas taxas de juros.

É importante destacar que os endividados e os inadimplentes são duas situações distintas. Os endividados são aqueles que contraem dívidas e comprometem uma parcela significativa de sua renda para honrá-las, como, por exemplo, os indivíduos que utilizam cartão de crédito e pagam a fatura integral. Já os inadimplentes são aqueles que deixam de cumprir um contrato ou uma cláusula específica, adquirindo dívidas que não são pagas, o que muitas vezes resulta em um estado de endividamento (MOTA *et al.*, 2016).

Frequentemente, esses problemas financeiros surgem devido à falta de planejamento e organização das finanças.

2.2.1 Finanças Pessoais

As finanças pessoais vão além de simplesmente ter recursos suficientes para pagar contas. Estão diretamente relacionadas à forma como uma pessoa administra seus recursos em várias áreas. Uma das principais áreas é o orçamento doméstico, que auxilia no planejamento

financeiro presente e futuro, equilibrando receitas e despesas pessoais.

Outra estratégia altamente eficaz é o planejamento antes de realizar uma compra. Isso oferece a oportunidade de considerar diversas opções e facilita a negociação com os prestadores de serviços. Por exemplo, ao efetuar uma compra à vista, é possível solicitar um desconto no momento da compra. Isso amplia as opções disponíveis e proporciona uma negociação mais favorável.

Essas estratégias podem ser divididas em duas categorias: curto prazo e longo prazo. Os planos financeiros de curto prazo são ações planejadas para um período de um a dois anos, levando em consideração os reflexos financeiros esperados (GITMAN, 1997). Podemos destacar os seguintes exemplos de planejamento nessa categoria:

> (i) determinar o quanto do orçamento vai para alimentação e moradia; (ii) suprir gastos desnecessários; (iii) buscar pesquisar produtos, serviços e lazer que sejam mais em conta e (iv) designar um percentual a ser destinada a uma poupança (MOTA *et al.*, 2016).

No que diz respeito ao planejamento de longo prazo, podemos afirmar que se trata de ações financeiras projetadas para um futuro distante, levando em consideração os reflexos financeiros previstos. Esses planos abrangem um período de dois a dez anos (GITMAN, 1997). Portanto, esse tipo de planejamento está mais relacionado aos objetivos de vida que se deseja alcançar. Alguns exemplos desse tipo de planejamento são:

> (i) compra ou troca de imóvel; (ii) aposentadoria; (iii) obter um automóvel de luxo e (iv) previdência privada. O planejamento financeiro pessoal deve ser compatível com a capacidade de geração de renda do planejador, pois de outra forma poderá gerar frustração e abandono dos objetivos (MOTA *et al.*, 2016).

2.3 Aplicações Relacionadas

Nos dias de hoje, a gestão financeira é uma habilidade crucial para alcançar estabilidade econômica e alcançar metas de longo prazo. Felizmente, a era digital nos trouxe uma gama diversificada de aplicativos de educação financeira que tornam esse processo mais acessível, informativo e até mesmo divertido. Esses aplicativos oferecem uma variedade de recursos projetados para ajudar os usuários a entender, planejar e tomar decisões mais conscientes em relação ao seu dinheiro. Alguns desses aplicativos são:

Mint O Mint é um aplicativo de educação financeira que oferece aos usuários uma visão detalhada de suas finanças pessoais. Ele ajuda a acompanhar gastos, criar orçamentos personalizados e fornece sugestões para economizar.

YNAB (You Need A Budget) YNAB é um aplicativo de orçamento baseado em metas. Sua abordagem educativa incentiva os usuários a atribuírem um propósito a cada real gasto e priorizarem suas metas financeiras.

PocketGuard Este aplicativo oferece uma visão clara dos saldos e transações financeiras, categorizando automaticamente as despesas.

Personal Capital Focado em investidores e planejadores financeiros, o Personal Capital oferece uma visão abrangente dos investimentos e patrimônio líquido dos usuários.

Goodbudget Baseado na técnica de orçamento de envelopes, o Goodbudget ajuda os usuários a alocar dinheiro para categorias específicas de gastos.

Cada um dos aplicativos apresentados oferece uma série de funcionalidades que podem auxiliar os usuários no gerenciamento de suas finanças. No entanto, nenhum deles se concentra em fornecer uma visão simplificada do uso do dinheiro ao longo do tempo, o que pode dificultar a compreensão das finanças pessoais de maneira holística.

Com essa necessidade em mente, surge o aplicativo *PennyWise*. O principal objetivo do *PennyWise* é proporcionar uma clara categorização das receitas e despesas, tornando o processo de tomada de decisão mais acessível e permitindo visualizar facilmente como o dinheiro está sendo utilizado em relação ao orçamento total disponível.

Diferentemente dos demais aplicativos, o *PennyWise* coloca o usuário no centro da experiência, focando em fornecer uma visão detalhada do fluxo de dinheiro. O aplicativo permite com que seu usuário saiba para onde o seu dinheiro está indo, possibilitando a identificação de padrões de gastos e ajudando na elaboração de estratégias para economizar e investir de forma mais eficiente.

Uma das principais vantagens do *PennyWise* é sua abordagem completa e personalizável para categorização das finanças. O usuário tem total liberdade para criar suas próprias categorias de receitas e despesas, adaptando o aplicativo às suas necessidades específicas. Além disso, o *PennyWise* oferece gráficos e relatórios detalhados, permitindo uma análise profunda das finanças e auxiliando na definição de metas financeiras realistas e alcançáveis.

3 CONTEXTO DE DESENVOLVIMENTO

Este trabalho será dividido em três grandes áreas que englobarão todo o desenvolvimento do aplicativo: (1) apresentação do contexto de desenvolvimento do aplicativo; (2) modelagem e prototipagem da solução; e (3) codificação da aplicação desejada.

Neste capítulo, abordaremos o primeiro item dessas áreas: o contexto do desenvolvimento do aplicativo móvel *iOS* para auxiliar a educação financeira de adultos. Será apresentado um resumo sobre a evolução da utilização de aplicativos móveis no Brasil, um breve contexto histórico sobre o *iPhone* - dispositivo móvel que utiliza o sistema operacional *iOS*, que será utilizado neste trabalho - e serão apresentados os dispositivos, programas e plataformas utilizados para o desenvolvimento do aplicativo, incluindo suas respectivas versões e configurações.

3.1 Aplicativos móveis

Durante os primeiros anos da *internet* no Brasil, o acesso era feito majoritariamente por computadores físicos. Com o advento da tecnologia móvel e a popularização do acesso à *internet*, essa realidade foi alterada, com uma grande adesão das massas aos dispositivos móveis, que oferecem acesso de forma mais rápida e simplificada. De acordo com o Instituto Brasileiro de Geografia e Estatística (2019), em pesquisa realizada no quarto trimestre deste ano, cerca de 82,7% dos domicílios brasileiros possuem acesso à *internet* atualmente, ou seja, 8 em cada 10 famílias.

Nesse contexto, o celular desponta como o equipamento mais utilizado para o acesso à *internet*. Com base em pesquisas realizadas, observou-se um aumento significativo no número de pessoas com 10 anos ou mais de idade que acessam a *internet* por meio de celular, enquanto a porcentagem daqueles que utilizam microcomputador ou *tablet* para acesso à *internet* diminuiu. Esse fenômeno pode ser visualizado na tabela 1:

Tabela 1 – Equipamento utilizado para acessar *internet*

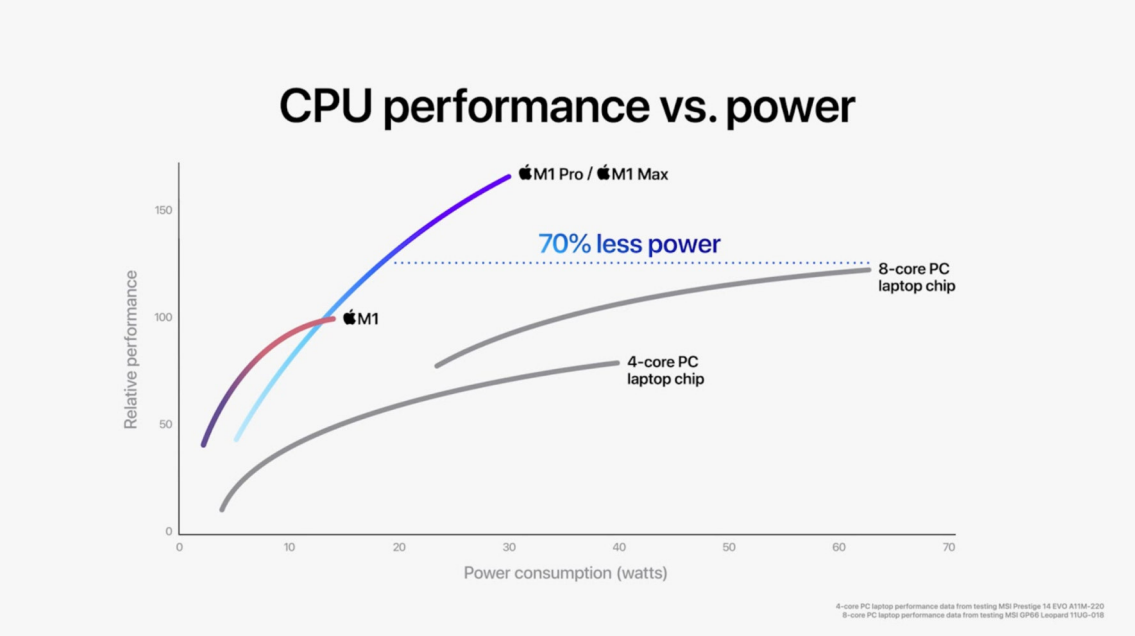
Dispositivo	2018	2018
Telefone móvel celular	98,1%	28,6%
Microcomputador	50,7%	46,7%
Tablet	12,0%	10,9%

Fonte: IBGE, Diretoria de Pesquisa, Coordenação de Trabalho e Rendimento, Pesquisa Nacional por Amostra de Domicílios Contínua 2018/2019.

A partir desses dados, é possível constatar que o uso de dispositivos móveis é o método mais popular de acesso à internet, tornando-se assim um tema de grande interesse a ser abordado. Atualmente, empresas como a *Apple Inc*. estão constantemente desenvolvendo novas tecnologias para compartilhar o poder inclusivo dos aplicativos desenvolvidos para o seu sistema operacional para dispositivos móveis, o iOS, que é utilizado em uma ampla gama de dispositivos, como *laptops*, *tablets*, relógios e até mesmo *smart TVs*.

Isso só é possível devido a um movimento massivo de mudança arquitetônica que está ocorrendo globalmente nos microcomputadores. Novos processadores, baseados na arquitetura *Advanced RISC Machines* (ARM), a mesma utilizada nos telefones móveis, estão surgindo diariamente e conquistando uma parcela significativa do mercado. De fato, existem diversas vantagens associadas à utilização desses processadores, sendo uma delas o fato de oferecerem um maior poder de processamento com uma menor demanda de recursos da máquina, conforme ilustrado na Figura 1.

Figura 1 – Performance vs Bateria



Fonte: *Apple Inc.* (2021).

Além disso, devido ao uso da mesma arquitetura dos smartphones, esses novos dispositivos têm a capacidade de executar os mesmos programas desenvolvidos para esses dispositivos. Isso significa que atualmente é possível desenvolver um aplicativo para um smartphone e ele poderá ser executado em *laptops*, *tablets*, relógios e *smart TVs*, sem a necessidade de adaptações significativas. Isso traz uma maior flexibilidade e facilidade para os desenvolvedores, bem como uma experiência mais consistente para os usuários em diferentes dispositivos.

3.2 Desenvolvimento móvel para iOS

A história por trás do desenvolvimento *mobile* para iOS teve início em 9 de janeiro de 2007, quando Steve Jobs, então Chefe do Escritório Executivo (CEO) da *Apple*, anunciou o lançamento do primeiro *smartphone* da empresa, o *iPhone*. Naquela época, a *Apple* inovou ao ser a primeira empresa a combinar três dispositivos comuns em um só: um telefone, um *player* de músicas e um dispositivo conectado à *internet*. Essa fusão revolucionária de funcionalidades foi um marco significativo no desenvolvimento dos *smartphones* e teve um impacto duradouro na indústria de dispositivos móveis. Desde então, o iOS tem sido constantemente aprimorado e a plataforma continua a desempenhar um papel importante no cenário *mobile*.

Levando isso em consideração, é evidente que o *software* acompanhou a evolução do *hardware* da empresa. O *Xcode*, o programa utilizado para o desenvolvimento de aplicativos, começou de forma bastante simples, com recursos limitados e utilização das linguagens *C* e*Objective-C* para viabilizar o desenvolvimento de aplicações. Ao longo do tempo, algumas ferramentas foram mantidas, como o *Objective-C*, enquanto outras foram removidas, como a *linguagem C*, abrindo espaço para uma série de serviços que possibilitam a criação de aplicativos simples e extremamente complexos.

Atualmente, o *Xcode* oferece suporte a uma variedade de recursos avançados, como computação gráfica, aprendizado de máquina e realidade aumentada. Esses recursos permitem que os desenvolvedores criem aplicativos que entregam experiências únicas aos usuários. A plataforma evoluiu para atender às demandas do mercado e às necessidades dos desenvolvedores, proporcionando ferramentas mais poderosas e abrangentes para a criação de aplicativos iOS.

3.3 Configurações do ambiente de desenvolvimento

Na seção a seguir, são apresentadas informações relacionadas aos programas e plataformas *online* utilizadas para o desenvolvimento, conforme descrito nas Tabelas 1 e 2, respectivamente. Além disso, são fornecidos os atributos de *hardware* do equipamento utilizado, conforme descrito na Tabela 3.

Tabela 2 – *Softwares* utilizados

Software	Descrição
Sistema Operacional	MacOS Monterey 12.4 (21F79)
Xcode	Versão 13.4 (13F17a)

Fonte: Elaborado pelo autor. (2023).

Tabela 3 – Ferramentas *online*

Ferramenta	Endereço
GitHub	github.com
Google Drive	drive.google.com
Draw.io	draw.io
Figma	figma.com

Fonte: Elaborado pelo autor. (2023).

Tabela 4 – *Hardware* utilizado

Item	Valor
Nome do Modelo	MacBook Pro
Identificador do Modelo	MacBookPro 18,2
Nome do Processador	Apple M1 Max
Velocidade do Processador	0,6 - 3,2 GHz
Número de Processadores	1
Número Total de Núcleos	10 (8 performance e 2 eficiência)
Cache L2 (por Núcleo)	<=28 MB
Cache de L3	<=48 MB
Memória	64 MB
Gráficos	Apple M1 Max 32-Core GPU
Armazenamento	1TB flash
Monitor	16,1 polegadas (3456 x 2234)

Fonte: Elaborado pelo autor. (2023).

4 MODELAGEM E PROTÓTIPOS

O aplicativo proposto, *PennyWise*, foi desenvolvido pelo autor em março de 2023 e será lançado na *AppStore* no mesmo ano. A versão abordada neste trabalho é a 1.0, que já conta com as principais funcionalidades implementadas. PennyWise é uma solução inovadora que visa auxiliar seus usuários no gerenciamento eficiente de suas finanças pessoais. Através do aplicativo, é possível ter um controle detalhado de despesas e receitas, permitindo uma melhor compreensão de como o dinheiro está sendo utilizado.

Além disso, será realizada uma análise aprofundada dos possíveis usuários do aplicativo. Serão identificados perfis de pessoas que podem se beneficiar do *PennyWise*, como estudantes, profissionais autônomos e famílias, cada um com suas necessidades específicas em relação ao gerenciamento financeiro. Com base nessa análise, poderemos direcionar melhorias e personalizações futuras para atender às demandas desses diferentes usuários.

Outro aspecto importante a ser discutido é o protótipo de média fidelidade do *PennyWise*. Será apresentado um modelo visual que representa a aparência e a usabilidade do aplicativo, permitindo uma prévia de como os usuários irão interagir com as funcionalidades disponíveis. Esse protótipo servirá como base para ajustes e refinamentos antes do lançamento final.

Por fim, faremos uma lista das funcionalidades já desenvolvidas e totalmente funcionais no aplicativo disponível na loja. Entre as principais funcionalidades, podemos destacar o registro de despesas e receitas, a geração de relatórios detalhados, o estabelecimento de metas financeiras e a integração com serviços bancários para importar extratos e facilitar a categorização das transações.

4.1 Modelo conceitual do sistema

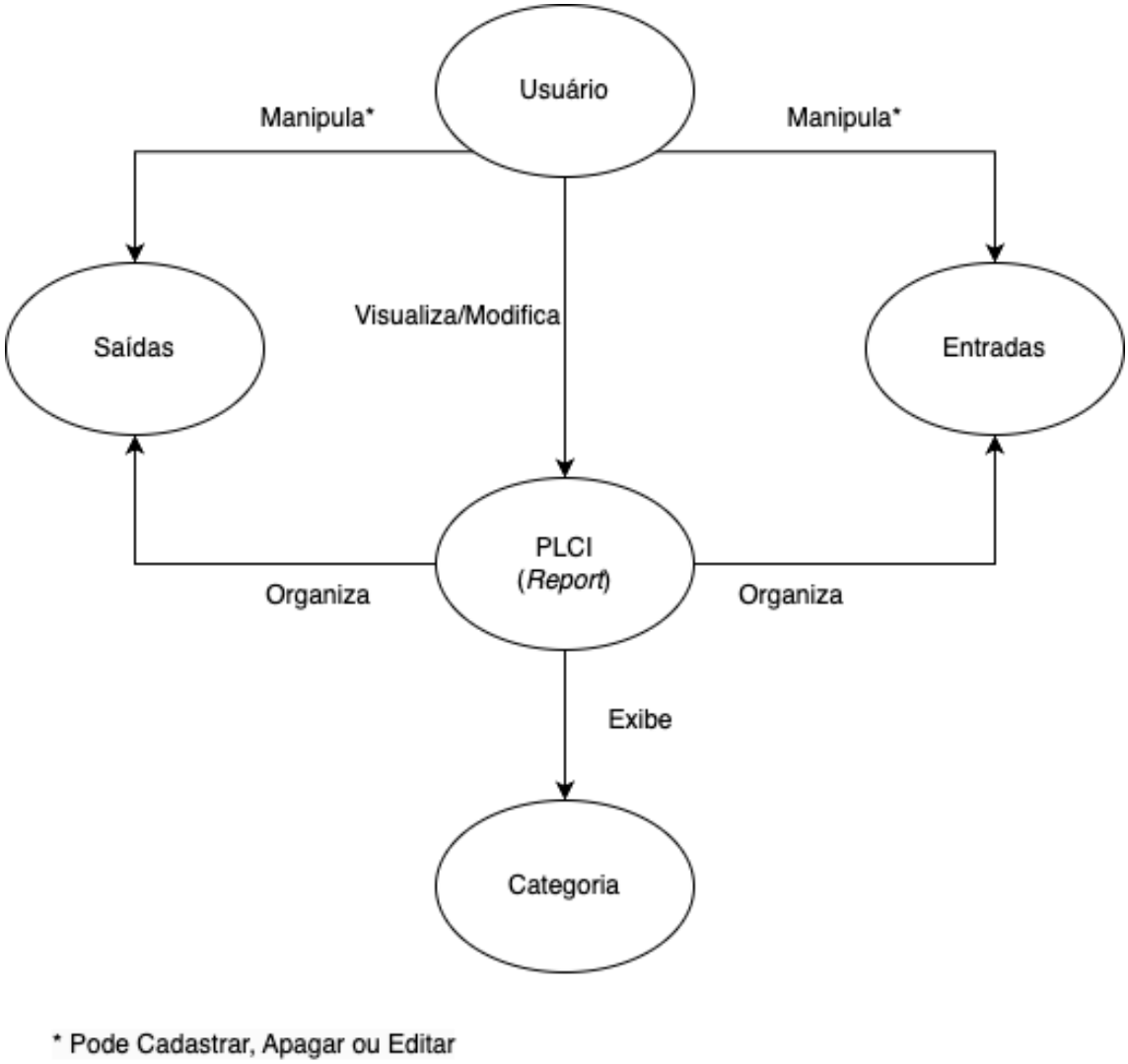
O modelo conceitual do sistema é a representação lógica de funcionamento do sistema que será construído como concebido pelo designer. Este modelo deve se basear em tarefas, requisitos, capacidades e experiência do usuário. Deve considerar também as capacidades e limitações dos mecanismos de processamento de informação do usuário, em particular limitações nos recursos de processamento e de memória de curto prazo. Por sua vez, o modelo conceitual é a compreensão que o usuário possui sobre como interagir com o sistema. Tudo o que o designer construir na imagem do sistema pode auxiliar ou prejudicar essa interpretação, tal como:

elementos de interface (*widgets*) para entrada e saída de dados; documentação, instruções, ajuda *online* e mensagens de erro (BARBOSA *et al.*, 2021).

De forma mais simples, um modelo conceitual é uma representação visual ou descritiva de um sistema, conceito ou ideia, que ajuda a compreender e organizar as relações entre seus elementos principais. Geralmente, utiliza diagramas, gráficos ou descrições verbais para ilustrar as conexões e interações entre os componentes de um sistema.

Tendo isso em mente, Na Figura 2 é está a proposta para o Modelo Conceitual Semântico do aplicativo *PennyWise* no momento de sua concepção.

Figura 2 – Modelo conceitual



Fonte: Elaborado pelo autor. (2023).

4.2 Análise do usuário

A fim de tornar o aplicativo mais relevante, sentiu-se a necessidade de entender melhor o usuário final, compreendendo suas necessidades, objetivos e contexto de uso ao procurar um aplicativo de finanças.

Para atender às necessidades do usuário, foram desenvolvidos dois artefatos extremamente importantes na Interação Humano-Computador (IHC): Persona e Contexto de Uso. Esses artefatos foram elaborados para garantir que as interfaces e funcionalidades do aplicativo fossem concebidas de forma realmente relevante para os usuários finais.

Uma persona é um personagem fictício, um arquétipo hipotético de um grupo de usuários reais, criada para descrever um usuário típico (COOPER *et al.*, 2014). Na Persona é a representação fictícia que incorpora características demográficas, comportamentais e objetivos do público-alvo. Essa representação ajuda a equipe de *design* a visualizar e compreender melhor as necessidades e motivações dos usuários, permitindo que eles projetem interfaces e recursos que atendam a essas expectativas.

Por sua vez, o Contexto de Uso é basicamente uma história sobre pessoas realizando uma atividade (ROSSON; CARROLL, 2002). É uma narrativa, textual ou pictórica, concreta e rica em detalhes contextuais, de uma situação de uso da aplicação, envolvendo usuários, processos e dados reais ou potenciais. Ele leva em consideração fatores como o ambiente físico, as tarefas a serem realizadas, as restrições e as preferências do usuário. Com base nessas informações, a equipe de *design* pode adaptar a interface e as funcionalidades para se adequarem ao contexto específico de uso, proporcionando uma experiência mais fluida e eficaz.

Portanto, ao utilizar os artefatos de Persona e Contexto de Uso, o objetivo é criar um aplicativo de finanças que atenda às necessidades e expectativas dos usuários de forma relevante e eficiente.

4.2.1 Persona

Nome Maria Oliveira
Idade 26 anos
Ocupação Estagiária de Economia
Estado civil Solteira
Mora Com a Mãe

Ganha R$ 1.500/mês

Foto:

Figura 3 – Foto da persona

Fonte: Imagem de nakaridore no Freepik. (2023).

Maria é uma jovem estudante universitária que está no último ano do curso de Economia. Ela vive em uma cidade movimentada e está sempre em busca de maneiras de gerenciar melhor suas finanças pessoais. Maria tem o desejo de alcançar a estabilidade financeira e adquirir mais conhecimentos sobre o assunto.

Como estudante, Maria tem uma rotina agitada, dividida entre aulas, estágio e atividades extracurriculares. Ela está constantemente buscando formas de otimizar seu tempo e organizar suas despesas para garantir que tudo esteja sob controle. No entanto, ela enfrenta algumas frustrações ao longo do caminho.

Uma das frustrações de Maria é a insegurança em relação às suas finanças. Ela gostaria de ter mais tempo para se dedicar aos estudos e aproveitar momentos de lazer, mas muitas vezes se sente preocupada com suas finanças. Além disso, Maria não possui uma visão clara de seus gastos e receitas mensais, o que dificulta o planejamento financeiro eficiente. Maria, estava passando por um período desafiador em sua vida. Ela se sentia constantemente sobrecarregada pela pressão do estágio, o acúmulo de tarefas acadêmicas e a preocupação com

suas finanças pessoais. Frustrada com a falta de tempo para si mesma e com o estresse financeiro de sempre que recebia seu salário, utiliza-lo todo para pagar contas e ficar o resto do mês tendo que comprar no cartão novamente. Ficando assim, presa no ciclo de endividamento financeiro.

Os objetivos de Maria são claros: ela deseja ter mais tempo para viajar, concluir sua graduação em Economia e alcançar estabilidade financeira. Ela entende que uma gestão financeira eficiente é essencial para alcançar essas metas e está em busca de ferramentas que possam ajudá-la nesse processo.

Nesse contexto, Maria está interessada em utilizar o aplicativo *PennyWise* para auxiliá-la no gerenciamento de suas finanças pessoais. Ela busca uma solução que seja intuitiva, fácil de usar e que forneça informações claras sobre seus gastos, permitindo que ela tome decisões financeiras mais informadas e alcance seus objetivos de forma mais eficiente.

4.2.2 Contexto de Uso

Maria, após passar o dia todo trabalhando, chegou em casa e percebeu que não havia pago a conta de luz do mês anterior. Decidiu que era hora de organizar sua vida financeira de uma vez por todas e, para isso, baixou um aplicativo recomendado por um amigo em seu *smartphone*. Ao explorar suas funcionalidades, Maria percebeu que o *PennyWise* poderia ser a ferramenta ideal para transformar sua situação financeira.

Ela abriu o aplicativo, cadastrou todas as suas entradas de renda e também todas as suas despesas, categorizando-as dentro da aplicação. Após essa etapa inicial, Maria pôde visualizar o quanto havia gasto em cada situação e como seu dinheiro estava sendo utilizado, permitindo-lhe redirecioná-lo para coisas que realmente desejava, como viagens.

Dessa forma, Maria deu o pontapé inicial em sua vida financeira. Nos meses seguintes, ela continuou a cadastrar todas as suas movimentações no *PennyWise*, já conhecendo bem a plataforma. Ela também começou a projetar quanto poderia eventualmente receber e quanto deveria destinar às suas despesas, a fim de não comprometer sua renda. Finalmente, Maria alcançou a tão desejada visualização de sua situação financeira.

4.3 Descrição das funcionalidades

Após análise dos artefatos apresentados, foi pensando em algo que fosse capaz entregar aos usuários funções com utilidade e valor. As funcionalidades do aplicativo na versão

1.0 são:

[F-01] Selecionar mês atual

O usuário tem a capacidade de selecionar o mês atual como base para visualizar todas as transações associadas a esse mês. Isso afeta a visualização de todas as funções subsequentes, que utilizam essas transações em seus cálculos internos.

[F-02] Visualizar balanço geral

O usuário tem a possibilidade de visualizar um resumo que apresenta o balanço geral do valor de todas as entradas e saídas. Além disso, é mostrado o total utilizado das saídas para cada categoria cadastrada no aplicativo.

[F-03] Adicionar entradas

O usuário é capaz de adicionar suas **entradas** com seus associados valores, datas e descrições. As "entradas"de cada usuário são equivalente a receitas que significa a remuneração do mesmo.

[F-04] Adicionar saídas

O usuário é capaz de adicionar suas **saídas** com seus associados valores, datas e descrições. As "saídas"equivalente as despesas.

[F-05] Visualizar resumo de transações

O usuário pode visualizar a lista de transações do mês selecionado. Essa lista inclui todas as entradas e saídas registradas durante esse período e é apresentada de forma resumida, exibindo apenas o tipo da transação, valor, data e título.

[F-06] Apagar ou Editar transações

O usuário tem a capacidade de apagar ou editar qualquer transação desejada diretamente pela tela inicial do aplicativo.

[F-07] Visualizar relatório

O usuário tem acesso a um relatório completo de seus gastos por meio de um botão chamado *recap* na tela inicial do aplicativo. A tela de relatório deve conter um compilado de todas as despesas e saídas daquele usuário no respectivo mês selecionado.

[F-08] Visualizar categorias de entrada

O usuário pode visualizar a lista de categorias de entrada registradas no aplicativo por meio da tela de relatório.

[F-09] Apagar ou Editar categorias de entrada

O usuário tem a capacidade de apagar ou edita qualquer categoria de **entrada** através da tela de relatório e adicionar entradas.

[F-10] Adicionar categorias de entrada

O usuário é capaz de adicionar qualquer categoria de entrada através da tela de relatório.

[F-11] Visualizar categorias de saída

O usuário pode visualizar a lista de categorias de saída registradas no aplicativo por meio da tela de relatório. Cada categoria será apresentada separadamente, juntamente com sua categoria pai associada. Além disso, um gráfico de pizza será exibido, mostrando a proporção de utilização de cada categoria em relação ao total de gastos. Isso proporciona uma melhor organização e compreensão das categorias de saída, permitindo uma análise visual rápida.

[F-12] Adicionar categorias de saída

O usuário tem a capacidade de adicionar categorias de saída por meio da tela de relatório, bem como adicionar transações de saída associadas a essas categorias. Cada categoria deve ser vinculada a uma categoria pai específica e é necessário informar a porcentagem de uso correspondente para cada categoria. Isso permite uma personalização detalhada das categorias de saída e sua distribuição proporcional dentro do orçamento. Como por exemplo, Salário ou Bolsa Acadêmica.

[F-13] Apagar ou Editar categorias de saída

O usuário tem a capacidade de apagar ou editar qualquer categoria de **saída** através da tela de relatório.

[F-14] Visualizar detalhes de categoria

O usuário pode visualizar os detalhes de cada categoria, incluindo o total de gastos registrados, a porcentagem correspondente em relação ao total de entradas e o

limite de gastos previstos para aquela categoria, com base na porcentagem de uso informada durante a criação da categoria. Além disso, será exibida a lista de transações associadas a cada categoria, permitindo uma análise detalhada das despesas relacionadas a cada uma delas. Essas informações fornecem ao usuário um melhor entendimento e controle sobre seus gastos em cada categoria específica.

4.4 Prototipagem da interface

Após listar as funcionalidades do aplicativo, a interface passou por uma fase de prototipagem, na qual foi desenvolvido o protótipo de média fidelidade juntamente com o diagrama de fluxo de tela. Nessa seção, é apresentado o resultado desta etapa.

4.4.1 Protótipo de telas

Os protótipos de média-fidelidade são implementações computadorizadas de aplicações com funcionalidades limitadas, contendo apenas as funções essenciais para avaliar cenários específicos. Em outras palavras, esses protótipos combinam técnicas de prototipagem de baixa-fidelidade, como esboços e *storyboards*, com suporte computacional para simular o comportamento de um protótipo de alta-fidelidade. Em geral, os protótipos de média-fidelidade apresentam características que combinam as vantagens dos protótipos de baixa-fidelidade (facilidade e rapidez na construção e edição) com as dos protótipos de alta-fidelidade (simulação, reutilização e teste de usabilidade), enquanto eliminam as desvantagens inerentes a ambos os tipos de protótipos (ALVES, 2007).

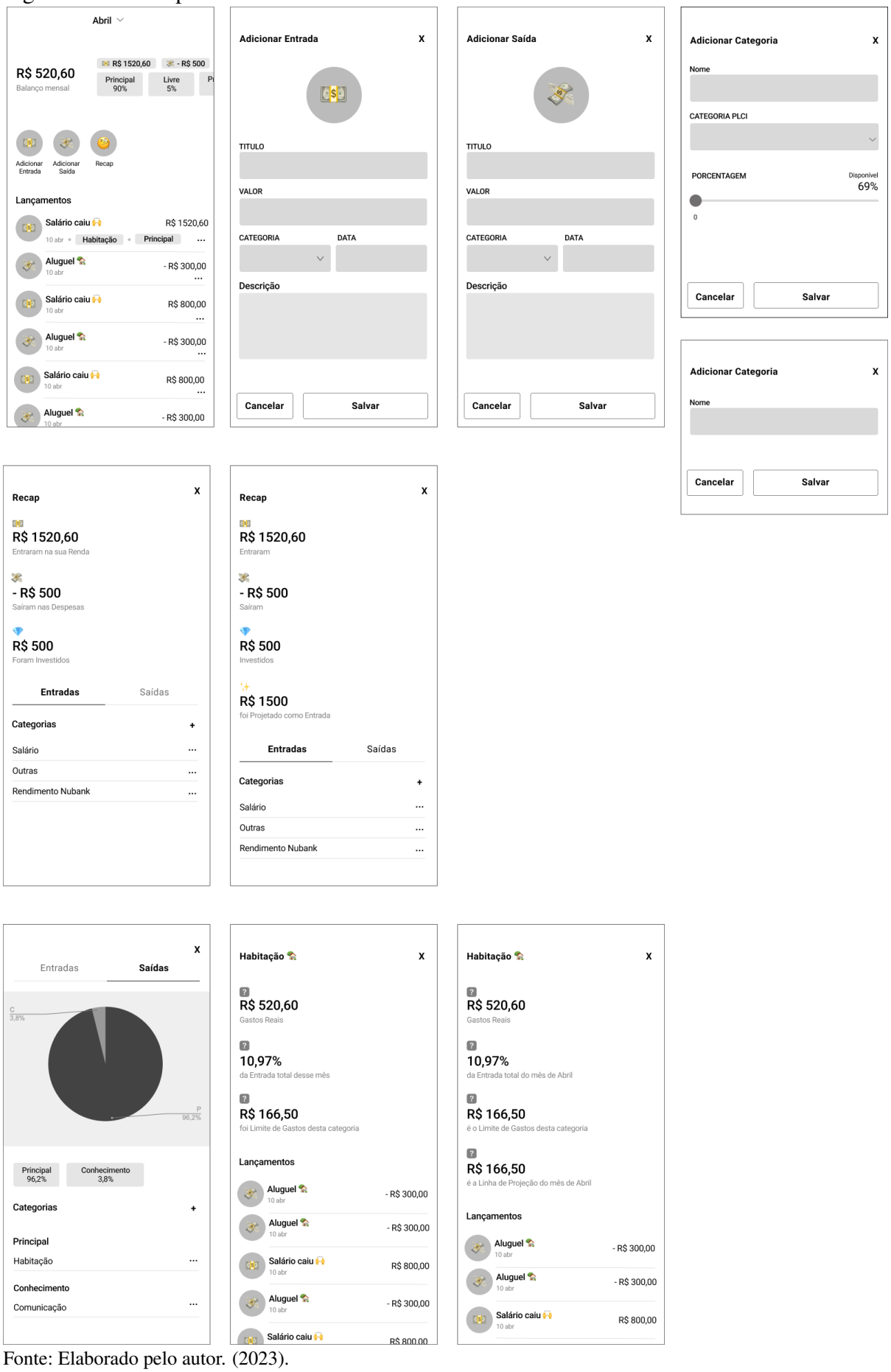
Tendo isso em mente, foi desenvolvido um protótipo de média fidelidade que permite uma visualização clara de todas as funcionalidades descritas anteriormente. O artefato resultante dessa etapa pode ser observado na Figura 4. O protótipo de média fidelidade proporciona uma representação visual que simula o comportamento e as interações do sistema de forma aproximada, permitindo uma avaliação e compreensão mais tangíveis das funcionalidades propostas.

4.4.2 Fluxo de telas

O fluxo de telas representa o caminho que o usuário deve percorrer para alcançar um determinado objetivo (SILVA; PETRUCELLI, 2018). Para proporcionar uma boa experiência,

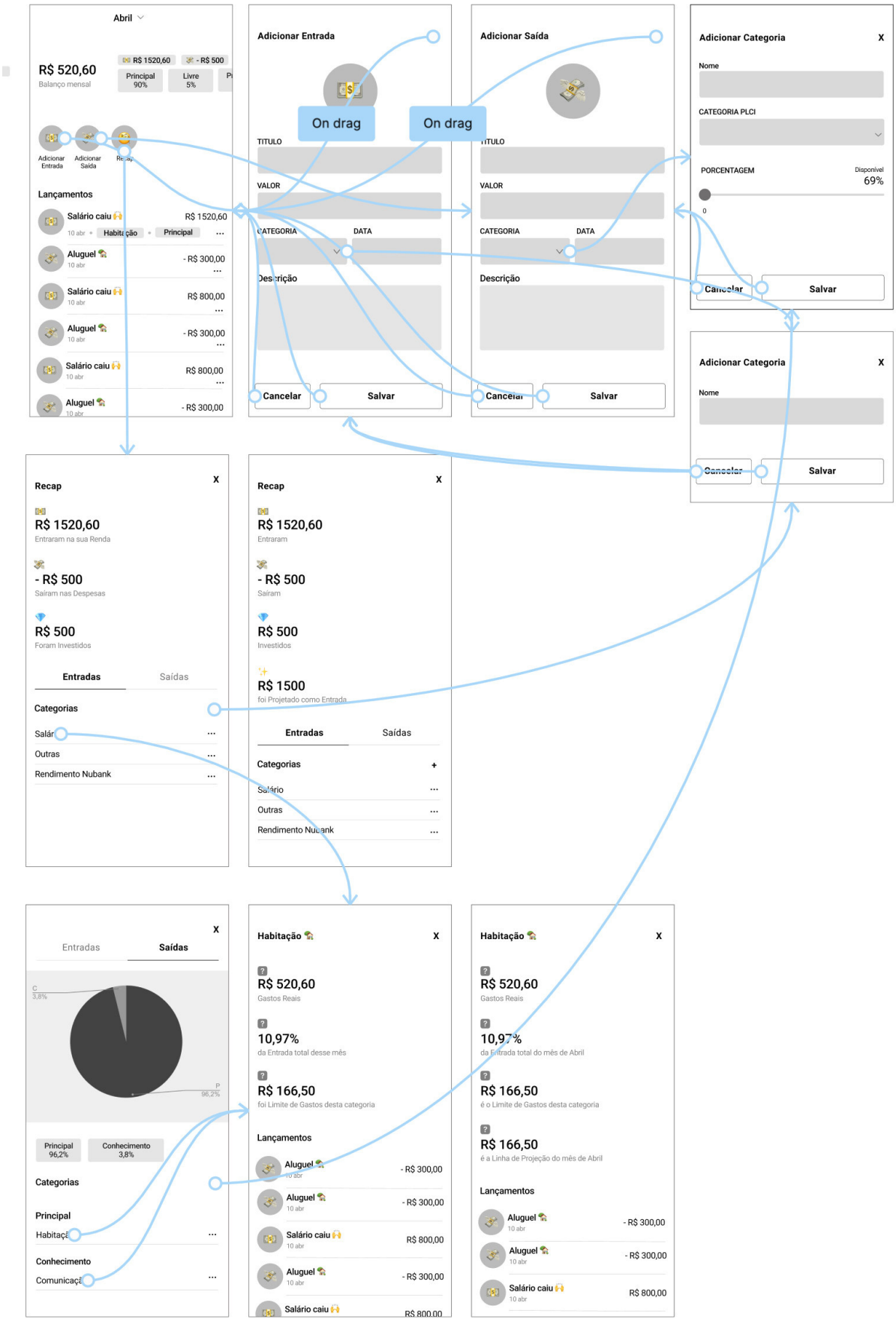
é recomendado e considerado ideal que seja possível realizar o objetivo com poucos cliques, tornando visível o comportamento de cada elemento na página quando acionado. Dessa forma, os usuários podem realizar suas ações de forma intuitiva e eficiente, sem a necessidade de navegar por múltiplas telas ou enfrentar obstáculos desnecessários. Um fluxo de telas bem projetado facilita a interação do usuário, aumentando a usabilidade e a satisfação geral com o sistema ou aplicativo. O resultado desta etapa pode ser observado na Figura 5.

Figura 4 – Protótipo de média fidelidade



Fonte: Elaborado pelo autor. (2023).

Figura 5 – Fluxo de telas



Fonte: Elaborado pelo autor. (2023).

5 IMPLEMENTAÇÃO

Nesta seção, apresenta-se uma descrição técnica dos aspectos abordados durante a implementação do PennyWise. Os aspectos abordados serão: a arquitetura do *software*, o padrão de projeto Singleton, o desenvolvimento de UI com SwiftUI, o uso da reatividade com *Combine*, o uso do CoreData como camada de dados, o CloudKit sincronizado com camada de dados, o processo de internacionalização do aplicativo e o versionamento do código.

5.1 *Padrões de projeto de software*

Gamma *et al.* (1995) descrevem que um padrão de projeto nomeia, abstrai e identifica os aspectos-chave de uma estrutura de projeto comum para torná-la útil para a criação de um projeto orientado a objetos reutilizável. Dentre os vários padrões de arquitetura existentes, os seguintes foram utilizados neste trabalho e serão abordados posteriormente.

- Padrão de projeto Model View ViewModel (MVVM)
- Singleton

5.1.1 Arquitetura do Software

A arquitetura do *sofware* utilizada para o desenvolvimento da aplicação foi o MVVM. Atualmente, o MVVM vem se tornando um dos padões de projeto mais utilizado em todo o mundo, pois resgata elegantemente ao distribuir estritamente os papéis e responsabilidades entre objetos (ALJAMEA; ALKANDARI, 2018).

Na década de 80, o MVVM foi introduzido no *Smalltalk* para superar as limitações do padrão *Model View Controller* (MVC) e aproveitar algumas de suas vantagens. O MVVM introduziu um novo objeto chamado ViewModel. Este objeto possui a lógica para preparar os dados que serão utilizados pelo objeto de visualização - View, que era responsabilidade do objeto Controlador no MVC (LO, 2015). Além disso, semelhante ao MVC, os objetos de Visualização e Controlador trabalham em par, mas o MVVM os fundiu em um único objeto chamado ViewModel.

Portanto, os três principais componentes do MVVM são:

ViewModel O Model representa a camada de dados do aplicativo. Ele é responsável por encapsular a lógica de negócios, manipulação e validação de dados. Isso pode incluir a obtenção e gravação de dados em um banco de dados, a comunicação com serviços web, a realização

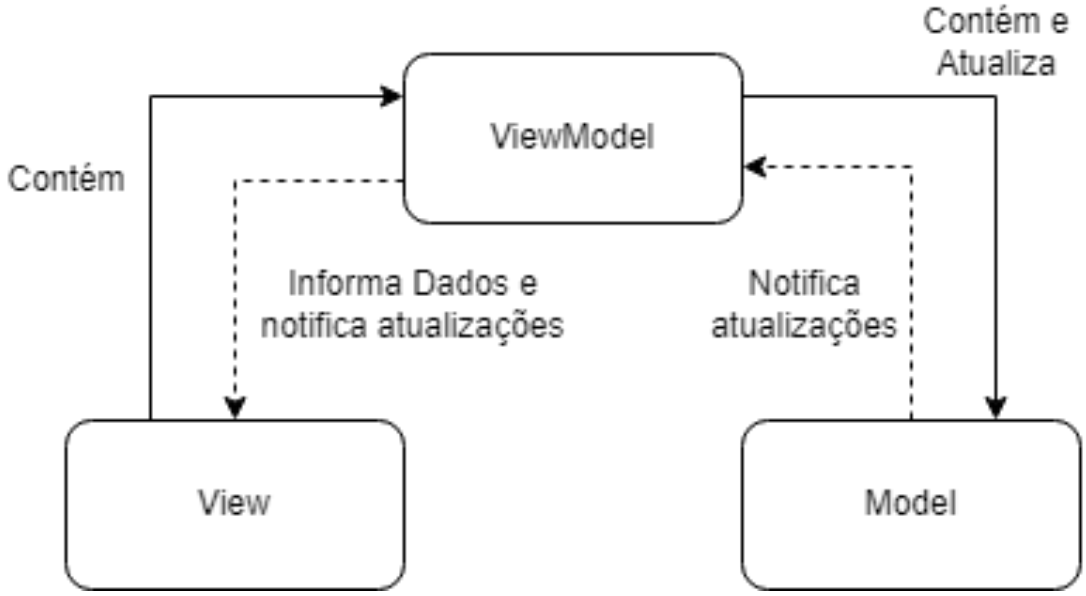
de cálculos e outras operações relacionadas aos dados.

View A View é a camada de interface do usuário. Ela é responsável pela apresentação visual dos dados e pela interação do usuário. Pode ser uma tela, uma página ou qualquer outro elemento de interface com o qual o usuário interaja. A View exibe os dados do ViewModel e também pode enviar comandos para o ViewModel, como ações do usuário, como cliques em botões.

ViewModel O ViewModel age como um intermediário entre o Model e a View. Ele contém a lógica de apresentação dos dados e a lógica necessária para manipular as ações do usuário. O ViewModel se comunica com o Model para obter e atualizar os dados e, em seguida, notifica a View sobre as alterações ocorridas. Além disso, o ViewModel expõe propriedades e comandos que são vinculados à View para atualizar a interface do usuário e responder às ações do usuário.

O diagrama na Figura 6, mostra cada componente do MVVM e suas respectivas relações:

Figura 6 – Padrão de projeto MVVM.



Fonte: Elaborado pelo autor. (2023).

5.1.1.1 Aplicação do MVVM no PennyWise

A utilização do MVVM no desenvolvimento do *PennyWise* tem se mostrado um grande aliado. Considerando o fluxo de telas do aplicativo disponível na Figura 5, é possível perceber que existem várias ações executadas pelo usuário, diversos fluxos interligados e compar-

tilhamento de muitas informações entre eles. Nesse contexto, é de extrema importância contar com uma arquitetura que possa manter essa enorme quantidade de dados atualizados.

Sem um controle adequado, a aplicação estaria sujeita a inconsistências nos estados de dados. O MVVM oferece uma abordagem que ajuda a evitar esses problemas, pois separa claramente as responsabilidades entre o Model, View e ViewModel.

O Model é responsável pela lógica de negócios e manipulação dos dados, garantindo a consistência e integridade dos mesmos. A View cuida da apresentação visual e interação com o usuário, proporcionando uma experiência amigável. O ViewModel atua como um intermediário entre o Model e a View, mantendo os dados atualizados e fornecendo a lógica de apresentação necessária.

Dessa forma, o MVVM permite que o PennyWise lide eficientemente com a complexidade do aplicativo, garantindo a consistência e atualização dos dados em diferentes telas e fluxos de forma organizada. Isso contribui para uma melhor experiência do usuário e facilita a manutenção e evolução do aplicativo no longo prazo.

5.1.2 Padrão de projeto Singleton

Para garantir a integridade dos dados voláteis compartilhados entre vários componentes e fluxos de telas no PennyWise, é necessário utilizar o padrão de projeto Singleton. Esse padrão garante que uma classe tenha apenas uma instância e fornece um ponto global de acesso a ela (GAMMA *et al.*, 1995). A sua implementação é bastante simples, basta seguir 2 regras fundamentais: O objeto só poderá ser acessado através de unica instância e inicializador por objeto tem de cer privado a fim de possibilitar aos clientes usar uma instância estendida sem alterar o seu código. Tendo isso em mente, veja a demostração da implementação de um Singleton com Swift.

Para garantir a integridade dos dados voláteis compartilhados entre vários componentes e fluxos de telas no PennyWise, é necessário utilizar o padrão de projeto Singleton [1]. Esse padrão garante que uma classe tenha apenas uma instância e fornece um ponto global de acesso a ela (GAMMA *et al.*, 1995). A sua implementação é bastante simples, basta seguir duas regras fundamentais: (1) o objeto só poderá ser acessado através de uma única instância; (2) o inicializador do objeto deve ser privado, a fim de possibilitar aos clientes usar uma instância estendida sem alterar o seu código. Tendo isso em mente, veja a demonstração da implementação de um Singleton com Swift.

Código-fonte 1 – Exemplo de Singleton

```
class Singleton {
    // primeira regra
    static let instance: Singleton = Singleton()

    // Segunda regra
    private init() {

    }
}
```

Ao utilizar o padrão Singleton, podemos assegurar que os dados são mantidos consistentes e que todas as partes do sistema têm acesso à mesma instância da classe responsável por gerenciar esses dados. Isso evita inconsistências causadas por múltiplas instâncias com diferentes estados.

Portanto, ao aplicar o padrão Singleton na camada de dados do PennyWise, estamos garantindo a coesão e a consistência dos dados em todo o sistema, facilitando a comunicação e a manipulação desses dados entre os componentes e fluxos do applicativo.

5.2 SwifUI

Nos últimos cinco anos, a programação declarativa tem sido utilizada para implementar interfaces de usuário em aplicativos móveis. A adoção dessa abordagem traz uma lista de benefícios, que incluem os seguintes recursos: uma convenção de codificação simplificada, uma curva de aprendizado mais suave, a capacidade de executar em diferentes plataformas móveis, uma prévia em tempo real das alterações durante a implementação sem precisar reconstruir todo o aplicativo, entre outros aspectos. Além disso, *frameworks* recentemente introduzidos, como SwiftUI, Flutter e React Native, contribuem para um processo de desenvolvimento mais eficiente ao utilizar um estilo de programação declarativa (IMBUGWA *et al.*, 2021).

SwiftUI é um framework de interface do usuário (UI) desenvolvido pela Apple para a criação de aplicativos iOS, macOS, watchOS e tvOS. Introduzido em 2019, o SwiftUI oferece uma abordagem declarativa e moderna para a construção de interfaces de usuário.

Ao contrário das abordagens anteriores como a utilização do editor do *Interface Builder* que gera arquivos como *XIBs* e *Storyboards*, que se baseavam em uma abordagem imperativa, o SwiftUI permite que os desenvolvedores descrevam a aparência e o comportamento da interface de usuário em um estilo declarativo. Isso significa que, em vez de escrever código procedimental para criar e atualizar elementos de interface, os desenvolvedores podem usar uma sintaxe simples e concisa para definir a estrutura e o *design* da interface.

O SwiftUI se baseia em um conjunto de controles e visualizações reutilizáveis, combináveis e personalizáveis, permitindo a construção rápida e flexível de interfaces complexas. Ele oferece uma ampla gama de recursos, como suporte para gestos, animações, gráficos vetoriais, tipografia dinâmica, localização e internacionalização.

Outra característica importante do SwiftUI é a capacidade de visualização em tempo real, permitindo que os desenvolvedores vejam imediatamente as alterações feitas na interface durante o desenvolvimento, sem a necessidade de compilação ou execução do aplicativo.

O SwiftUI também se integra perfeitamente com o Swift, a linguagem de programação utilizada para o desenvolvimento de aplicativos iOS e macOS. Isso permite que os desenvolvedores aproveitem os recursos poderosos desta linguagem, como segurança de tipo, inferência automática de tipos, tratamento de erros e muito mais.

Em resumo, o SwiftUI é um *framework* moderno e intuitivo que simplifica e acelera o desenvolvimento de interfaces de usuário para aplicativos *Apple*, proporcionando uma experiência mais agradável e eficiente para os desenvolvedores.

5.2.1 Componentização da Interface

A Componentização da Interface é um conceito que se baseia na divisão de uma interface de usuário em componentes independentes e reutilizáveis. Cada componente é responsável por uma parte específica da interface e pode ser combinado para criar interfaces complexas. Isso promove a reutilização de código e facilita a manutenção e evolução do aplicativo.

No contexto do SwiftUI, a Componentização da Interface é aplicada ao dividir a interface em pequenos componentes, como botões, campos de texto, listas, etc. Esses componentes podem ser definidos de forma independente e reutilizados em várias partes do aplicativo. Isso permite uma maior modularidade e flexibilidade no desenvolvimento da interface de usuário, facilitando a criação de aplicativos elegantes e escaláveis. Na figura 7 é possível observar alguns componentes criados no aplicativo e o respectivo código-fonte de um componente criado:

Figura 7 – Componentização da Interface.

Fonte: Elaborado pelo autor. (2023).

Código-fonte 2 – Exemplo de Componente

```
1  struct EmojiView: View {
2      let emoji: TransactionKind
3
4      var body: some View {
5          GeometryReader { geometry in
6              Text(emoji.rawValue)
7                  .font(.system(size:(32*geometry.size.width)/70))
8                  .padding(16)
9                  .background(
10                     Circle()
11                         .fill(Color.lightGray)
12                         .frame(width: geometry.size.width,
                               height: geometry.size.height)
13                 )
```

```
14              }
15          }
16  }
```

5.3 Combine

Os sistemas atuais possuem uma vasta quantidade de eventos em tempo real, sendo muito mais interativos se comparados aos do passado. Portanto, lidar com interações em tempo real vem se tornando cada vez difícil através de paradigmas de programação baseados em controladores, como o MVC (MATOS; ZUCHI, 2021). Neste prisma, surge a programa reativa como um novo paradigma de programação orientada a fluxos de dados e propagação de mudanças de forma assíncrona. Isso quer dizer que, de acordo com algumas ações, outras serão executadas como forma de reação (TSVETINOV, 2015).

O Combine é um *framework* de programação reativa fornecido pela Apple para desenvolvimento de aplicativos iOS, macOS, watchOS e tvOS. Ele foi introduzido pela primeira vez no *iOS* 13 e no *macOS* 10.15 (Catalina) como parte do ecossistema do SwiftUI. Ele oferece uma série de operadores e tipos que facilitam a composição, transformação e manipulação de fluxos de eventos assíncronos. Ele permite que os desenvolvedores trabalhem com valores assíncronos de forma declarativa e reativa.

Os principais conceitos e recursos do Combine são:

Publishers Representam a origem dos eventos ou valores assíncronos. Podem ser combinados, transformados e manipulados para criar novos fluxos de eventos.

Subscribers Observam e reagem aos eventos gerados pelos *publishers*. Eles recebem os eventos e podem executar ações, como processamento de dados, atualização da interface do usuário ou tomada de decisões.

Operadores São métodos disponíveis no *framework* Combine que permitem transformar, filtrar, mesclar ou combinar fluxos de eventos. Eles ajudam na composição de *pipelines* de manipulação de eventos assíncronos.

Cancellables Representam uma assinatura de uma subscrição. Podem ser usados para cancelar ou encerrar a subscrição quando não são mais necessários, evitando vazamentos de memória.

Subjects São uma combinação de um *publisher* e um *subscriber*. Permitem a criação de fluxos

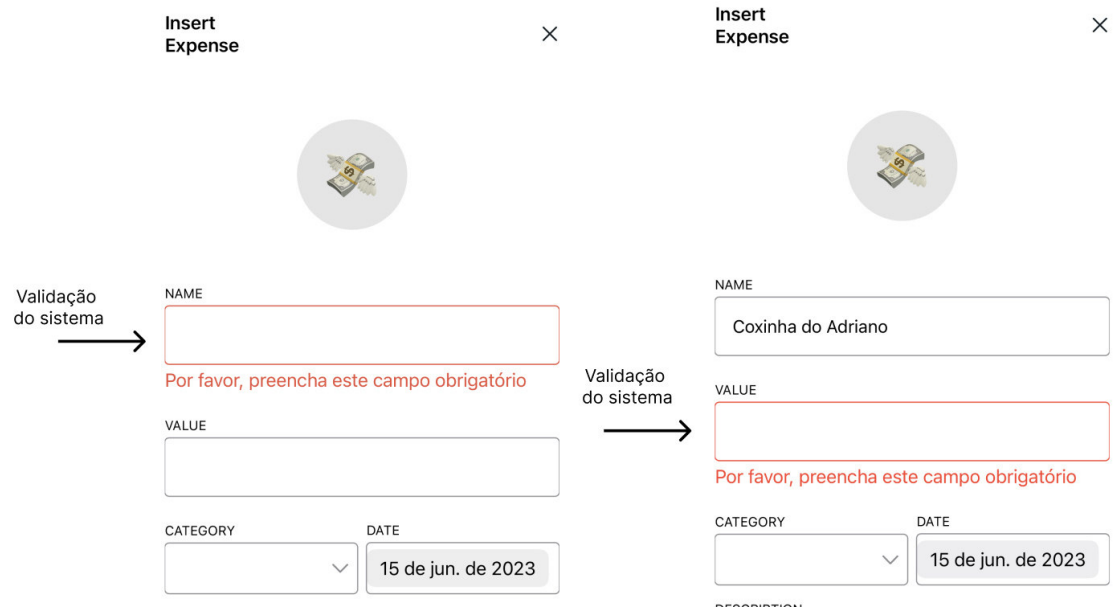
de eventos personalizados e podem ser usados para alimentar valores manualmente em um fluxo.

5.3.1 Aplicação do Combine no PennyWise

Devido à grande quantidade de componentes iterativos (Figura 4) e fluxos de tela (Figura 5), foi necessário utilizar o Combine no processo de criação das conexões que serviriam como notificações de atualização entre o Model, ViewModel e a View de cada componente principal. O Combine foi de extrema importância, pois proporcionou uma vasta biblioteca de opções para que essas conexões ficassem sutis e extremamente poderosas.

Um exemplo prático disso foi a implementação em tempo real das funções responsáveis pela validação dos campos nas telas de Criar Entrada e Criar Saída, como mostrado a figura 8:

Figura 8 – Validação do Sistema.



Fonte: Elaborado pelo autor. (2023).

No Combine, os modificadores responsáveis por criar essas conexões são chamados de *Published*. Foi necessário usar uma variável de controle que continha cada campo e seu respectivo estado, a fim de determinar quais campos estavam válidos ou não. Essa implementação pode ser encontrada no trecho de código-fonte 3:

Código-fonte 3 – Exemplo de Validação com Combine

```
1 class CreateViewModel: ObservableObject {
```

```
2
3       //Campos a serem validados
4       @Published var valdiator: [String: Bool?] = ["name": nil, "value"
            : nil, "category": nil]
5
6       ...
7       //Campo informado pelo usuário
8       @Published var name: String = "" {
9           willSet {
10              self.valdiator["name"] = !newValue.isEmpty ? false : true
11          }
12      }
13  }
```

No exemplo fornecido, a variável ‘validator‘ armazena os estados de validação de cada campo do formulário. Através do modificador ‘willSet‘, a lógica de pré-validação é executada sempre que uma ação é realizada pelo usuário e um novo valor é atribuído ao campo. O resultado dessa validação é então armazenado no respectivo objeto dentro do dicionário de validação.

No trecho de código fornecido, é verificado se o campo ‘name‘ está vazio ou não. Se o campo estiver vazio, a validação falha e a entrada correspondente no dicionário de validação é marcada como ‘true‘ (indicando uma falha na validação). Caso contrário, se o campo não estiver vazio, a validação é bem-sucedida e a entrada correspondente no dicionário de validação é marcada como ‘false‘ (indicando uma validação bem-sucedida).

No swiftUI, o suporte completo ao Combine facilita a ligação entre os componentes de interface do usuário e os dados subjacentes, permitindo uma atualização automática da interface sempre que os dados forem modificados. No código-fonte 4, é mostrada uma forma simplificada de como essa ligação pode ser implementada para o componente responsável por coletar o nome da transação em um cadastro:

Código-fonte 4 – Integração com *View*

```
1  import SwiftUI
2
3  struct CreateView: View {
4      @ObservedObject public var viewModel: CreateViewModel
```

```
5
6
7       var body: some View {
8           PWTextField(componentTitle: "name", input: self.$viewModel.
                name, isRequired: (self.viewModel.valdiator["name"] ??
                false) ?? false)
9       }
10  }
```

Neste exemplo, a classe CreateViewModel é uma classe observável (ObservableObject) que contém as propriedades validator e name. A propriedade name é marcada como @Published, o que permite que ela seja observada para atualizações. Quando o valor de name é alterado, o código dentro de willSet é executado do CreateViewModel é executado.

No corpo da CreateView, o componente TextField está vinculado ao valor da propriedade name do viewModel por meio do $viewModel.name. Isso garante que qualquer alteração no campo de texto seja refletida automaticamente na propriedade name. Além disso, o componente Text exibe o resultado da validação com base no valor de validator["name"].

Dessa forma, sempre que o usuário digitar algo no campo de texto, a propriedade name será atualizada, o willSet será acionado, a validação será realizada e a interface do usuário será atualizada de acordo, fornecendo um *feedback* visual em tempo real sobre o estado de validação do campo.

5.3.1.1 Combine na camada de Model

A fim de criar a estrutura necessária para a atualização da camada de *ViewModel* a partir da camada de *Model*, foi também utilizado o combine para fazer todo o processo de notificação entre cada objeto responsável por implementar essa parte da arquitetura. No exemplo abaixo, é possível visualizar a utilização do combine na camada de *Model*:

Código-fonte 5 – Integração com *Model*

```
1  import Combine
2
3  class StorageManager: NSObject, ObservableObject {
4      //instÃ¢ncia de acesso do singleton
5      static let shared: StorageManager = StorageManager()
```

```
6
7       var incomes = CurrentValueSubject<[Income], Never>([])
8
9   }
```

No exemplo fornecido, é mostrado o uso do Combine na camada de modelo por meio da classe ‘StorageManager‘. Esta classe é um singleton e implementa a classe ‘NSObjec‘t e ‘ObservableObject‘.

Dentro da classe ‘StorageManager‘, há uma propriedade chamada incomes, que é do tipo ‘CurrentValueSubject<[Income], Never>‘. O ‘CurrentValueSubject‘ é uma classe do Combine que armazena um valor atual e emite esse valor para os assinantes sempre que ocorre uma alteração.

No caso específico, incomes é um ‘CurrentValueSubject‘ que armazena uma matriz de objetos Income. Essa propriedade pode ser utilizada para armazenar e gerenciar uma lista de receitas. Sempre que ocorrer uma alteração na lista de receitas, o ‘CurrentValueSubject‘ emitirá o novo valor para os assinantes.

Essa abordagem permite que a camada de ‘ViewModel‘ se inscreva no ‘Current-ValueSubject‘ e receba notificações sempre que houver uma atualização na lista de receitas, permitindo que a interface do usuário seja atualizada em tempo real com os dados mais recentes.

Além disso, o ‘CurrentValueSubject‘ foi configurado com o tipo de erro ‘Never‘, indicando que não há erros associados a ele. O exemplo a seguir ilustra a forma como o ‘ViewModel‘ implementa a chamada de ações oferecidas pelo modelo:

Código-fonte 6 – Integração com *ViewModel*

```
1   import Combine
2
3   class DashboardViewModel: ObservableObject {
4
5       @Published var totalIncome: Double = 0
6
7       private var subscribers: [AnyCancellable] = []
8
9       init() {
10          StorageManager.shared.incomes.sink { [weak self] value in
11              self?.totalIncome = value.reduce(0, +)
```

```
12          }
13          .store(in: &subscribers)
14      }
15
16  }
```

No exemplo fornecido pelo Código-fonte 6, a classe ‘DashboardViewModel‘ representa a camada de ‘ViewModel‘ e se integra à camada de ‘Model‘ por meio do uso do Combine. Essa integração permite que o ‘ViewModel‘ observe as alterações nos valores do objeto ‘Income na camada de ‘Model e atualize seus próprios dados, para que a ‘View‘ possa ser notificada e exibir os dados atualizados.

Na inicialização do ‘DashboardViewModel‘, é feita a subscrição (*sink*) ao ‘CurrentValueSubject‘ ‘StorageManager.shared.incomes‘. O operador *sink* permite receber os valores emitidos pelo ‘CurrentValueSubject‘ e executar um bloco de código sempre que um novo valor for emitido.

No bloco de código dentro do *sink*, ‘[weak self]‘ é usado para evitar referências fortes e possíveis vazamentos de memória. Dentro do bloco, é atualizado o valor da propriedade ‘totalIncome‘ do ‘ViewModel‘ com base nos valores do *array value*, que representa a lista de receitas. Nesse exemplo, o ‘totalIncome‘ é calculado somando todos os valores do *array*.

Em seguida, o operador ‘store(in: subscribers)‘ é utilizado para armazenar o objeto ‘AnyCancellable‘ retornado pelo *sink* na propriedade ‘subscribers‘. Essa ação é necessária para manter a subscrição ativa enquanto o ‘ViewModel‘ estiver em uso.

Dessa forma, sempre que houver uma alteração na lista de receitas na camada de ‘Model‘, o ‘CurrentValueSubject‘ emitirá o novo valor, o bloco de código dentro do sink será executado e o ‘totalIncome‘ será atualizado. Essa atualização será automaticamente refletida na ‘View‘, permitindo que os dados sejam exibidos corretamente.

5.4 Core Data

O Core Data é um *framework* da *Apple* para persistência de dados em aplicativos. Ele fornece uma maneira fácil de gerenciar o modelo de dados e oferece recursos poderosos para armazenar, buscar, atualizar e excluir objetos persistentes. Com recursos como mapeamento objeto-relacional, gerenciamento de contexto e persistência transparente, o Core Data simplifica a manipulação de dados persistentes e é amplamente utilizado no desenvolvimento de aplicativos

Apple.

No *PenyWise*, o Core Data foi utilizado para implementar um banco de dados relacional eficiente. Ele permite que qualquer alteração em um dado de uma classe específica seja notificada ao objeto responsável pela interação com o banco de dados. Esse objeto, por sua vez, notifica cada objeto que esteja observando os dados daquela classe. Por exemplo, ao adicionar uma nova transação (entrada ou saída), o objeto responsável pelo banco de dados é notificado, atualiza os dados e notifica os observadores. Os observadores recebem os novos dados e atualizam suas respectivas visualizações. Dessa forma, o Core Data facilita a atualização automática da interface do usuário com os dados mais recentes, garantindo uma experiência de usuário fluida e consistente como ilustra o código-fonte 7:

Código-fonte 7 – Implementação da Camada de Dados

```
1  import Combine
2
3  class StorageManager: NSObject, ObservableObject {
4      //instÃ¢ncia de acesso do singleton
5      static let shared: StorageManager = StorageManager()
6
7      private lazy var incomesFetchRequest: NSFetchRequest<Income> = {
8          let fetchRequest: NSFetchRequest<Income> = Income.
               fetchRequest()
9          fetchRequest.sortDescriptors = [NSSortDescriptor(key: "date",
                ascending: true)]
10         fetchRequest.returnsObjectsAsFaults = false
11         return fetchRequest
12     }()
13     var incomes = CurrentValueSubject<[Income], Never>([])
14
15     private override init() {
16         super.init()
17
18         incomesFetchController = NSFetchedResultsController(
19             fetchRequest: incomesFetchRequest,
20             managedObjectContext: context,
21             sectionNameKeyPath: nil,
22             cacheName: nil
23         )
```

```
24
25          incomesFetchController.delegate = self
26
27          do {
28              try incomesFetchController.performFetch()
29              incomes.value = incomesFetchController.fetchedObjects ??
                    []
30          } catch {
31              NSLog("Error: could not fetch objects")
32          }
33
34      }
35  }
36
37  extension StorageManager: NSFetchedResultsControllerDelegate {
38      func controllerDidChangeContent(_ controller:
            NSFetchedResultsController<NSFetchRequestResult>) {
39          if let incomes = controller.fetchedObjects as? [Income] {
40              self.incomes.value = incomes
41          }
42      }
43  }
```

No Código-fonte 7, é adicionado um objeto do tipo ‘NSFetchRequest‘ chamado ‘incomesFetchRequest‘, que é responsável por buscar os dados da entidade ‘Income‘ no banco de dados.

No inicializador do singleton ‘StorageManager‘, o ‘incomesFetchRequest‘ é configurado com as opções desejadas, como a ordenação dos resultados pelo campo "date"em ordem crescente e a exclusão de objetos em *cache*.

Em seguida, é criado um ‘NSFetchedResultsController‘ chamado ‘incomesFetchController‘, que realiza a busca dos dados usando o ‘incomesFetchRequest‘ e um objeto de contexto de banco de dados chamado ‘context‘.

O objeto ‘StorageManager‘ implementa a extensão ‘NSFetchedResultsControllerDelegate‘, que inclui o método ‘controllerDidChangeContent‘. Esse método é chamado pelo ‘incomesFetchController‘ sempre que houver alterações nos dados buscados. No método, é verificado se os objetos retornados são do tipo ‘Income‘, e em seguida, os valores da propriedade

‘incomes‘ (do tipo ‘CurrentValueSubject‘) são atualizados com os novos objetos.

No geral, esse código estabelece a integração entre o Core Data e o ‘StorageManager‘, onde as alterações nos dados do Core Data são refletidas na propriedade ‘incomes‘ do ‘StorageManager‘. Isso permite que os observadores sejam notificados das alterações e atualizem adequadamente a interface do usuário com os dados mais recentes.

5.5 CloudKit

CloudKit é um serviço fornecido pela *Apple* que permite armazenar e sincronizar dados em nuvem para aplicativos. Ele oferece um conjunto de *APIs* que facilitam o gerenciamento e o compartilhamento de dados entre dispositivos e usuários.

CloudKit é uma solução completa de *backend* que lida com a infraestrutura de armazenamento, autenticação, segurança e sincronização de dados em nuvem. Ele oferece um modelo de dados flexível baseado em registros e campos, onde você pode definir tipos de registro personalizados para armazenar e organizar os dados do seu aplicativo.

Uma das vantagens do CloudKit é que ele oferece armazenamento escalável e confiável, com recursos de segurança robustos integrados. Ele também fornece recursos avançados de consulta e notificações baseadas em mudanças nos dados, permitindo que os aplicativos respondam em tempo real às atualizações do servidor.

No PenyWise, foi utilizado a integração entre CloudKit e Core Data, onde foi possível usar o Core Data como uma camada de persistência local para armazenar e recuperar dados em *cache*, enquanto o CloudKit lida com a sincronização desses dados com a nuvem e com outros dispositivos. Isto permitiu com que o aplicativo só se preocupasse com seus dados em *cache* enquanto os dados dos usuários estavam seguros e salvos na nuvem, sem depender única e exclusivamente dos dados locais. Então, se o usuário conter mais de um dispositivo, todos os dados entre esses dispositivos estarão sincronizados em uma só fonte. Pro exemplo, adicionando um dado no aplicativo do *iPhone* reflete imediatamente nos dados exibidos no *iPad* do usuário.

Esta sincronização ocorre através do uso de zonas do CloudKit, que representam uma coleção de registros dentro de um banco de dados. O Core Data mapeia as entidades e atributos do modelo de dados para registros e campos do CloudKit. As alterações feitas nos dados locais do Core Data são propagadas para o CloudKit, e as atualizações feitas em outros dispositivos são sincronizadas de volta para o Core Data local. Esta integração permite que os

aplicativos aproveitem os benefícios do armazenamento em nuvem do CloudKit, mantendo um *cache* local com o Core Data para um acesso mais rápido aos dados.

5.6 Internacionalização

No PenyWise, foi utilizado o serviço de internacionalização oferecido pelo *Xcode* para fornecer suporte total a dois idiomas: inglês americano e português brasileiro. A internacionalização permite que o aplicativo seja localizado e exibido em diferentes idiomas, atendendo às preferências de idioma dos usuários.

Ao utilizar o serviço de internacionalização do *Xcode*, os textos exibidos no aplicativo são colocados em arquivos de strings separados, um para cada idioma suportado. Esses arquivos são chamados de ‘arquivos de localização‘ e possuem uma extensão .strings.

Para cada idioma suportado, é criado um arquivo de localização correspondente. Por exemplo, para o inglês americano, é criado um arquivo chamado "en.lproj/Localizable.strings", e para o português brasileiro, é criado um arquivo chamado "pt-BR.lproj/Localizable.strings". Esses arquivos contêm pares de chave-valor, em que as chaves são identificadores únicos e os valores são os textos traduzidos para cada idioma.

Código-fonte 8 – Exemplo de *Localizable.strings* para pt-BR

```
1  "monthlyBalance" = "Balanço mensal";
2  "addCategory" = "Adicionar\nCategoria";
3  "addIncome" = "Adicionar\nEntrada";
4  "addExpense" = "Adicionar\nSaída";
5  "addInvest" = "Adicionar\nInvestimento";
6  "recap" = "Recap";
7  "transactions" = "Lançamentos";
8  "actions" = "Ações";
9  "delete" = "Apagar";
10 "edit" = "Editar";
```

Durante a execução do aplicativo, o sistema operacional determina o idioma preferido do dispositivo e carrega automaticamente o arquivo de localização apropriado. O Xcode lida com a troca de idiomas e garante que os textos corretos sejam exibidos com base no idioma selecionado.

A integração destas chaves com o código é bastante simples com o swiftUI. A

maioria dos componentes já possuem suporte nativo ao ‘NSLocalizedString‘, então na maioria das vezes é possível emitir esse construtor e utilizar diretamente a chave nos componentes como no código-fonte 9:

Código-fonte 9 – Exemplo de utilização da chave no componente de Texto

```
1    // Será exibido o valor "Balanço mensal" para essa chave
2    Text("monthlyBalance", comment: "monthlyBalance")
```

No exemplo fornecido, a chave "monthlyBalance"é usada diretamente como argumento para o construtor do componente ‘Text‘. O ‘NSLocalizedString‘ é implicitamente aplicado para localizar o valor correspondente a essa chave no arquivo de localização apropriado (nesse caso, "pt-BR.lproj/Localizable.strings").

Quando o aplicativo é executado em um dispositivo com idioma definido como português brasileiro, o componente ‘Text‘ exibirá o valor "Balanço mensal", que é a tradução para o idioma selecionado.

Essa abordagem simplifica a integração das chaves de localização com o código, permitindo que você se concentre nas traduções e na organização dos arquivos de localização, enquanto o SwiftUI lida com a resolução das chaves e a exibição dos textos corretos com base no idioma selecionado pelo usuário.

Além disso, vale ressaltar também que o aplicativo funciona com a localização da moeda do usuário, assim é possível cadastrar entradas e saídas independente da moeda que o usuário utiliza.

5.7 Versionamento de Código

No contexto do versionamento de código no PenyWise, foi adotado o padrão Git e a metodologia Gitflow. O Git é um sistema de controle de versão distribuído amplamente utilizado que permite rastrear as alterações feitas no código-fonte ao longo do tempo.

O Gitflow é uma metodologia de fluxo de trabalho baseada no Git, que define uma estrutura para organizar as *branches* e o processo de desenvolvimento. Ele estabelece duas *branches* principais: ‘master‘ e ‘develop‘.

A *branch* ‘master‘ é usada para armazenar as versões de produção estáveis do código. Ela representa o código que está atualmente em produção e é mantido em um estado confiável e

funcional.

A *branch* ‘develop‘ é usada como a ramificação principal para o desenvolvimento contínuo. Ela é usada para integrar as diferentes funcionalidades desenvolvidas pelos membros da equipe e serve como base para a criação de versões futuras.

Além dessas *branches* principais, o Gitflow define as seguintes *branches* secundárias:

Feature Utilizada para desenvolver novas funcionalidades. Cada funcionalidade é desenvolvida em uma *branch* separada, derivada da *branch* ‘develop‘.

Release Utilizada para preparar uma nova versão para lançamento. As correções finais e os ajustes são feitos nesta *branch* antes de serem mesclados à ‘master‘ para lançamento.

Hotfix Utilizada para correções de *bugs* críticos na versão de produção. Essa *branch* é criada a partir da ‘master‘, e as correções são mescladas tanto na ‘master‘ quanto na ‘develop‘.

Essa abordagem do Gitflow facilita o gerenciamento das diferentes fases do desenvolvimento de software e permite que as equipes trabalhem de forma colaborativa e organizada. Cada funcionalidade é desenvolvida em uma *branch* separada, evitando conflitos e facilitando a revisão de código. As *branches* secundárias são mescladas de volta para a branch ‘develop‘ após a conclusão, garantindo que o código seja integrado de forma incremental e mantendo a estabilidade da *branch* ‘master‘ para lançamentos. As *branches* ‘release‘ e ‘hotfix‘ permitem correções e preparação adequada para lançamentos sem interromper o fluxo de desenvolvimento.

No PenyWise, o padrão Git e a metodologia Gitflow foram adotados para garantir um fluxo de trabalho eficiente e controle de versões adequado.

6 APLICAÇÃO

Neste capítulo é apresento o resultado do processo de implementação do aplicativo PennyWise. Com imagens das telas e descrições de como utilizar suas funcionalidades, o usuário pode interagir com a aplicação pela primeira vez através do painel inicial, onde encontra acesso fácil a todas as outras funcionalidades do sistema, tais como cadastrar despesas, registrar entradas, lista de categorias, criar categorias, visualizar relatórios, entre outras.

6.1 *Dashboard*

Esta tela é responsável por conter um compilado de todos os dados contidos na aplicação. Nela, o usuário pode adicionar as suas despesas e entradas, visualizar uma prévia das despesas cadastradas já processadas pelo aplicativo e a lista de despesas ordenas a partir da data de cadastro das mesmas, como ilustrado pela Figura 9.

6.2 Cadastro de Transações

Conforme ilustrado na Figura 10, nesta tela o usuário pode cadastrar suas transações: Entradas e Saídas. Para fazer isso, ele deve informar os dados associados àquela transação, como Título, valor, data, associá-la a uma categoria e adicionar uma descrição, se necessário.

6.3 Lista e Registro de Categorias

As telas de listar e criar categorias servem como suporte para dois fluxos principais: a criação das transações e a exibição das categorias na tela de *Recap*. Nessas telas, é possível visualizar todas as categorias existentes no *app* e adicionar, se necessário, uma nova categoria, informando seu nome e porcentagem de uso para as categorias de saída. Essas telas podem ser observadas na Figura 11.

6.4 Visualizar Relatório

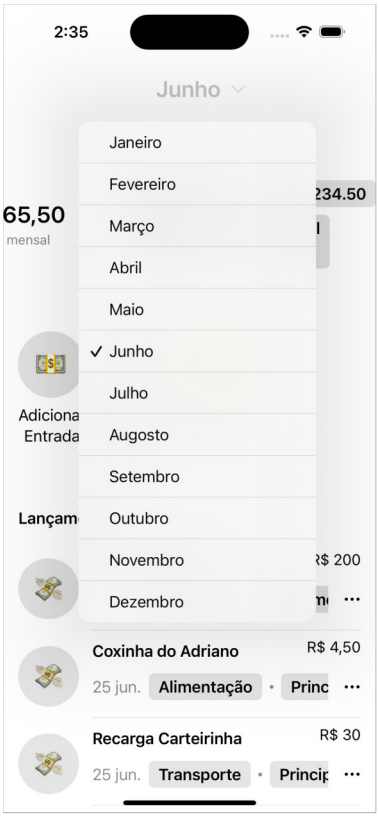
A tela de visualização de relatório é responsável por mostrar um resumo geral das finanças do usuário. Nela, é possível obter dados a respeito do balanço mensal, com o total de entradas e saídas cadastradas para aquele mês. Além disso, é possível ver a lista de categorias cadastradas no *app* e, na aba de despesas, visualizar um gráfico de setores que ilustra

a distribuição das despesas por categoria.

Além disso, também é possível acessar os detalhes de uma categoria específica e observar uma análise com base no total de entradas por categoria, juntamente com sua respectiva porcentagem relativa em relação ao total do mês. É possível ainda realizar uma análise dos dados de saída, que mostra o total de saída por categoria, sua respectiva porcentagem em relação ao total de saídas do mês e o valor esperado como limite de gastos, com base na porcentagem informada durante o cadastro da categoria.

Todos esses dados são processados diretamente no aplicativo. Qualquer alteração em uma transação associada, seja entrada ou saída, gera uma atualização em todos os cálculos matemáticos relacionados. Você pode observar essas telas na Figura 12.

Figura 9 – Telas do *Dashboard*.



1. Seleção do mês

2. Rolagem lateral do componente

3. Estado da tela com dados cadastrados

4. Estado da tela com uma categoria adicionada

Fonte: Elaborado pelo autor. (2023).

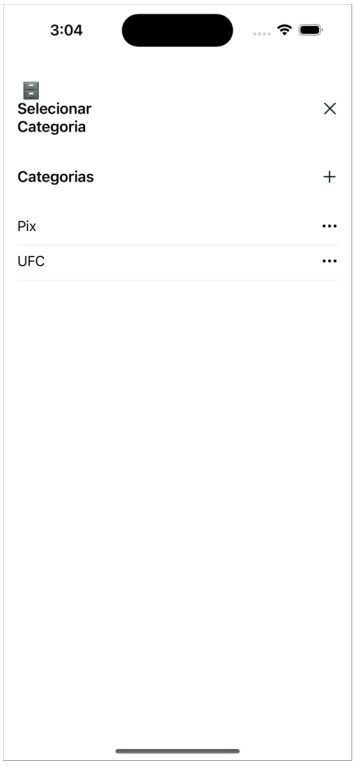
Figura 10 – Telas do Cadastro de Transações.

1. Tela de adicionar Entrada

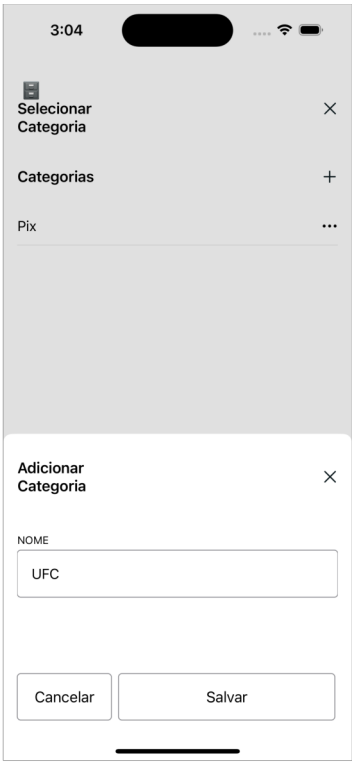
2. Tela de adicionar Saída

Fonte: Elaborado pelo autor. (2023).

Figura 11 – Telas de Lista/Cadastro de categorias.



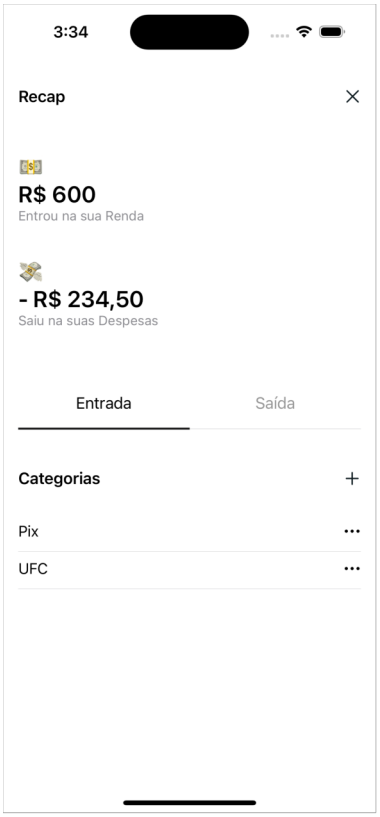
1. Tela de lista de categorias



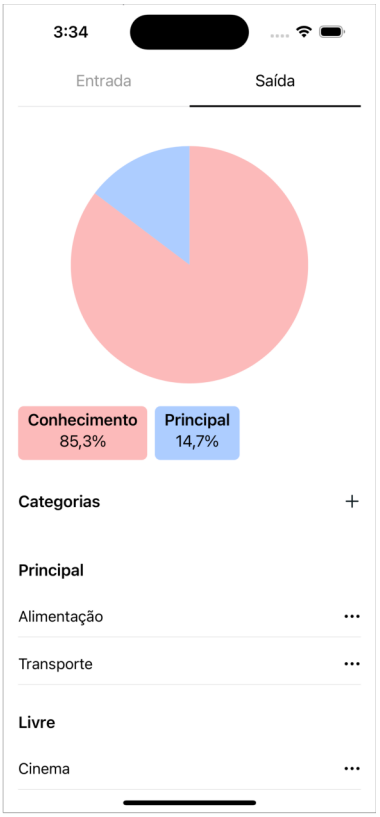
2. Tela de criação de categoria de entrada

Fonte: Elaborado pelo autor. (2023).

Figura 12 – Telas do Relatório.



1. Tela inicial do *Recap* com *categorias* de entrada



2. Tela inicial do *Recap* com *categorias* de saída

3. Tela de detalhes de categorias de entrada

4. Tela de detalhes de categorias de saída

Fonte: Elaborado pelo autor. (2023).

7 CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho apresentou o projeto e desenvolvimento de uma aplicação móvel voltada para o suporte no planejamento financeiro de jovens e adultos. O objetivo principal do projeto é fornecer aos usuários uma ferramenta simples e direta que auxilie no controle e na organização de suas finanças pessoais.

O *design* foi pensado de forma a facilitar a navegação e a compreensão das funcionalidades, mesmo para usuários sem experiência prévia em aplicativos de controle financeiro. Isso foi alcançado por meio de uma abordagem centrada no usuário, levando em consideração suas necessidades e preferências.

Foi apresentado o relatório técnico do desenvolvimento para dispositivos *iOS*, utilizando a linguagem de programação *Swift* onde os aspectos mais importantes em torno do desenvolvimento do *PennyWise* foram abordados e detalhados com suas respectivas implementações.

Foram apresentados os resultados iniciais da implementação que resultaram na criação da Versão 1.0 do aplicativo com suas funcionalidades principais totalmente desenvolvidas.

Como trabalhos futuros pretende-se adicionar as seguintes funcionalidades na aplicação:

- Ampliar para os celulares *Android*.
- Criação da Identidade Visual do aplicativo.
- Melhoria nas funcionalidades base da aplicação.
- Corrigir eventuais *bugs*.
- Publicação na .
- Criação de assistente virtual, Julius, com utilização de *Natural Language Understanding* (NLU) para criação de transações rápidas a partir da fala.

REFERÊNCIAS

ALJAMEA, M.; ALKANDARI, M. Mmvmi: A validation model for mvc and mvvm design patterns in ios applications. **IAENG Int. J. Comput. Sci**, v. 45, n. 3, p. 377–389, 2018.

ALVES, K. M. O. e Yuska Paola Aguiar e Bernardo Lula Júnior e Luiz Carlos Chaves e Gabriela Souza e Diénert Vieira e Ygor Carvalho e Jânio Lima e M. O uso de modelos e múltiplos protótipos na concepção de interface do usuário. **Revista Principia - Divulgação Científica e Tecnológica do IFPB**, v. 1, n. 15, p. 15–29, 2007. ISSN 2447-9187. Disponível em: <https://periodicos.ifpb.edu.br/index.php/principia/article/view/258>.

BARBOSA, S. D. J.; SILVA, B. S. d.; SILVEIRA, M. S.; GASPARINI, I.; DARIN, T.; BARBOSA, G. D. J. **Interação Humano-Computador e Experiência do Usuário**. [S.l.]: Autopublicação, 2021.

COOPER, A.; REIMANN, R.; CRONIN, D.; NOESSEL, C. **About Face: The Essentials of Interaction Design**. 4th edition. ed. Indianapolis, IN: Wiley, 2014.

CORDEIRO, N. J. N.; COSTA, M. G. V.; SILVA, M. N. D. Educação financeira no brasil: uma perspectiva panorâmica. p. 69–84, 2018. Acessado em 14 maio de 2022. Disponível em: <https://revistas.pucsp.br/emd/article/download/36841/25699/0>.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. **Design Patterns: Elements of Reusable Object-Oriented Software**. Addison-Wesley, 1995. (Addison-Wesley professional computing series). ISBN 9783827330437. Disponível em: <https://books.google.com.br/books?id=tmNNfSkfTlcC>.

GITMAN, L. J. Princípios da administração financeira. São Paulo: Habra, p. 588, 1997.

IMBUGWA, G. B.; ARAÚJO, L. J. P. de; KHAZEEV, M.; ENOMBE, E.; SALIU, H.; MAZZARA, M. A case study comparing static analysis tools for evaluating swiftui projects. **Journal of Physics: Conference Series**, IOP Publishing, v. 2134, n. 1, p. 012022, dec 2021. Disponível em: <https://dx.doi.org/10.1088/1742-6596/2134/1/012022>.

Instituto Brasileiro de Geografia e Estatística. **PNAD Contínua: Acesso à Internet e à televisão e posse de telefone móvel celular para uso pessoal (boletim de divulgação da pesquisa, de onde foram retirados os infográficos publicados neste Especial)**. 2019. Disponível em: <https://biblioteca.ibge.gov.br/visualizacao/livros/liv101794_informativo.pdf>. Acessado em 28 maio de 2022.

LO, D. A. Sensor plot kit: An ios framework for real-time plotting of wireless sensors. 2015.

MARTINS, J. P. Educação financeira ao alcance de todos. São Paulo, p. 05, 2004.

MATOS, E. Ferreira de; ZUCHI, J. D. Um estudo sobre programaÇÃo reativa. **Revista Interface Tecnológica**, v. 18, n. 2, p. 219–228, dez. 2021. Disponível em: <https://revista.fatectq.edu.br/interfacetecnologica/article/view/1287>.

MINISTÉRIO DA EDUCAÇÃO. **Secretaria da Educação Fundamental – Parâmetros Curriculares Nacionais: introdução aos parâmetros curriculares nacionais**. Brasília: [s.n.], 1997.

MOTA, C. T.; CHIMELO, G.; BENDER, C. M.; NESPOLO, D.; BORELLI, V. A.; FACHINELLI, A. C. OrganizaÇÃo financeira pessoal: AnÁlise dos fatores que influenciam no endividamento e inadimplÊncia dos jovens. **Revista Eletrônica de Ciências Sociais Aplicadas**, p. 46–61, 2016. ISSN 2176-5766. Acessado em 15 maio 2022. Disponível em: <http://187.103.250.244/index.php/revista/article/view/44>.

NEGRI, A. L. L. **Educação Financeira para o Ensino Médio da Rede Pública: uma proposta inovadora.** 73 p. Tese (Mestrado em educação) — Centro Universitário Salesiano de São Paulo: UNISAL, São Paulo, 2010.

ROSSON, M. B.; CARROLL, J. M. **Usability Engineering: Scenario-Based Development of Human-Computer Interaction**. [S.l.]: Morgan Kaufmann, 2002.

SILVA, G. A.; PETRUCELLI, E. E. PrincÍpios de ux design no desenvolvimento de websites: estudo de caso de um site de notícias esportivas. **Revista Interface Tecnológica**, v. 15, n. 2, p. 28–38, dez. 2018. Disponível em: <https://revista.fatectq.edu.br/interfacetecnologica/article/view/449>.

SPC Brasil. **Inadimplência de Pessoas Físicas**. 2019. Disponível em: <https://www.spcbrasil.org.br/wpimprensa/wpcontent/uploads/2019/02/AnÃąlise-PF_Janeiro_2019.pdf>. Acessado em 23 abril de 2022.

TSVETINOV, N. **Learn Reactive Programming with Java 8**. [s.n.], 2015. Disponível em: <https://bit.ly/3C4foBT>.