Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas

MyGrana: Um Aplicativo de Auxílio à Gestão Financeira de Estudantes Universitários

Guilherme Marx Ferreira Tavares

TRABALHO DE CONCLUSÃO DE CURSO

ORIENTAÇÃO:
Alexandre Magno de Souza

COORIENTAÇÃO:
Euler Horta Marinho

Abril, 2020
João Monlevade–MG

Guilherme Marx Ferreira Tavares

MyGrana: Um Aplicativo de Auxílio à Gestão Financeira de Estudantes Universitários

Orientador: Alexandre Magno de Souza

Coorientador: Euler Horta Marinho

Monografia apresentada ao curso de Engenharia de Computação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina "Trabalho de Conclusão de Curso II".

Universidade Federal de Ouro Preto

João Monlevade

Abril de 2020

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

T231m Tavares, Guilherme Marx Ferreira .
MyGrana [manuscrito]: um aplicativo de auxílio à gestão financeira de estudantes universitários. / Guilherme Marx Ferreira Tavares. - 2021.
90 f.: il.: color., tab..

Orientador: Prof. Me. Alexandre Magno de Souza.
Coorientador: Prof. Me. Euler Horta Marinho.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Graduação em Engenharia de Computação .

1. Aplicativos móveis. 2. Controle financeiro. 3. Estudantes universitários. 4. Finanças. I. Marinho, Euler Horta. II. Souza, Alexandre Magno de. III. Universidade Federal de Ouro Preto. IV. Título.

CDU 658.15:004.41

Bibliotecário(a) Responsável: Flavia Reis - CRB6-2431

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E APLICADAS
DEPARTAMENTO DE COMPUTACAO E SISTEMAS

FOLHA DE APROVAÇÃO

Guilherme Marx Ferreira Tavares

MyGrana: Um Aplicativo de Auxílio à Gestão Financeira de Estudantes Universitários

Monografia apresentada ao Curso de Engenharia de Computação da Universidade Federal
de Ouro Preto como requisito parcial para obtenção do título de graduação em Engenharia de Computação

Aprovada em 26 de abril de 2021.

Membros da banca

Mestre - Alexandre Magno de Sousa - Orientador - Instituto de Ciências Exatas e Aplicadas
Mestre - Euler Horta Marinho - Coorientador - Instituto de Ciências Exatas e Aplicadas
Doutor - Fernando Bernardes de Oliveira - Instituto de Ciências Exatas e Aplicadas
Doutor - Filipe Nunes Ribeiro - Instituto de Ciências Exatas e Aplicadas

Prof. Alexandre Magno de Sousa, orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 06/05/2021.

Documento assinado eletronicamente por **Alexandre Magno de Sousa**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 07/05/2021, às 16:14, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0168806** e o código CRC **C00253E9**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35400-000
Telefone: - www.ufop.br

Agradecimentos

Agradeço aos meus pais, Marcos e Cintia, sem os quais seria impossível chegar onde eu me encontro e desde os tempos de criança me ofereceram todas as condições necessárias em termos humanos e educacionais.

Agradeço aos meus amigos, que me ofereceram todo o apoio possível para que eu pudesse superar as mais diversas adversidades encontradas durante esses anos de graduação.

Aos meus professores também deixo o mais sincero obrigado por todo o esforço dedicado em despertar-me o interesse pelo conhecimento.

Por fim, gostaria de deixar um agradecimento especial à todos aqueles que, em qualquer nível, contribuíram para minha formação como ser humano. Tantos professores de história, filosofia e sociologia pelo conhecimento a respeito da vida em sociedade, meus amigos por me mostrar os caminhos da empatia, minha família por me mostrar a sinceridade e importância de laços incondicionais e a todos que sempre me ofereceram uma palavra de carinho e suporte em frente aos momentos turbulentos que a vida proporciona.

“Se não puder voar, corra.

Se não puder correr, ande.

Se não puder andar, rasteje,

mas continue em frente

de qualquer jeito.”

Martin Luther King

Resumo

A má gestão financeira pode ocasionar diversos problemas. No âmbito pessoal, ela causa um grave estresse que pode levar a problemas de saúde física e mental, adoção de comportamentos prejudiciais, queda de recursos cognitivos e problemas nos relacionamentos, logo, uma boa relação com o dinheiro se relaciona diretamente a qualidade de vida do indivíduo. A organização financeira é um processo complexo e depende totalmente do grupo de pessoas ao qual ela é aplicada. Estudantes universitários possuem padrões de comportamento diversos, mas ainda assim passíveis de análise e de organização. Aplicações móveis podem ser usadas para propor soluções para problemas apresentados pelos usuários. Este trabalho se propõe a desenvolver uma aplicação móvel multiplataforma para auxiliar à gestão financeira de estudantes universitários. Para tal foi desenvolvido um aplicativo de teste a fim de verificar o engajamento dos usuários e a capacidade de auxílio dessa proposta. O teste, realizado com 102 estudantes universitários, apontou resultados positivos então, com base nas *features* presentes no teste, foi feita toda a modelagem do aplicativo multiplataforma, utilizando 2 bancos de dados guardados localmente no dispositivo e 1 em nuvem com o Firebase. O aplicativo, desenvolvido com o *framework* Flutter, é intuitivo e apresenta com clareza as principais funcionalidades necessárias para organização financeira dos estudantes. A partir do banco de dados remoto foi capaz a coleta de dados do usuário, coleta essa que pode ser usada para identificar padrões nos gastos e rendas. Esses padrões podem ser analisados para gerar soluções ainda mais específicas e com capacidade de auxílio ainda maiores.

Palavras-chaves: gerenciamento financeiro. estudantes universitários. dispositivos móveis. aplicativo.

Abstract

Poor financial management can cause several problems. On a personal level it causes severe stress that can lead to physical and mental health problems, adoption of harmful behaviors, decreased cognitive resources and problems in relationships, so a good relationship with money is directly related to the individual's quality of life. Financial organization is a complex process and depends entirely on the group of people to whom it is applied. College students have different patterns of behavior, but still amenable to analysis and organization. Mobile applications can be used to propose solutions to problems presented by users. This work proposes to develop a multiplatform mobile application to assist the financial management of college students. To this end, a test application was developed in order to verify the engagement of users and the assistance capacity of this proposal. The test showed positive results so, based on the features present in the test, all the cross-platform application modeling was done, using 2 databases stored locally on the device and 1 in cloud with Firebase. The application, developed with the framework Flutter, is intuitive and clearly presents the main functionalities necessary for students' financial organization. From the remote database it was possible to collect user data, which can be used to identify patterns in expenses and incomes. These patterns can be analyzed to generate even more specific and supportive solutions.

Key-words: financial management. college students. mobile devices. application.

Lista de ilustrações

Figura 1 – Imagens Promocionais do Whatsapp. 27
Figura 2 – Imagens promocionais do aplicativo Mobills. 37
Figura 3 – Imagens promocionais do aplicativo Guia Bolso. 39
Figura 4 – Casos de uso do aplicativo de teste. 43
Figura 5 – Telas do aplicativo de teste. 44
Figura 6 – Modelo Entidade-Relacionamento do Primeiro Banco de Dados local. . . . 48
Figura 7 – Modelo Entidade-Relacionamento do segundo banco de dados local. 49
Figura 8 – Árvore de coleta do banco de dados remoto. 52
Figura 9 – Diagrama de Casos de Uso 53
Figura 10 – Diagrama de Atividade Referente a Edição de uma Transação. 54
Figura 11 – Diagrama de Sequência. 55
Figura 12 – Protótipo de telas para o Cadastro. 56
Figura 13 – Protótipo de telas para a Adição de transação. 57
Figura 14 – Protótipo de telas para a Edição de transação. 58
Figura 15 – Protótipo de telas para a Exclusão de transação. 59
Figura 16 – Protótipo de telas para o Planejamento. 60
Figura 17 – Protótipo de telas para a Comparação. 61
Figura 18 – Protótipo de telas para a Exibição de extratos com filtros. 61
Figura 19 – Protótipo de telas para a Exibição do extrato do mês atual. 62
Figura 20 – Organização das Pastas do Projeto. 65
Figura 21 – Telas do fluxo de cadastro. 69
Figura 22 – Tela inicial do aplicativo. 70
Figura 23 – *Dialog* de Nova Transação. 71
Figura 24 – Exibição de extratos. 72
Figura 25 – Tela de Gastos e Rendas. 73
Figura 26 – Edição de orçamentos. 74
Figura 27 – Comparação de orçamento planejado e real. 75
Figura 28 – Informações. 76
Figura 29 – Exemplo de árvore de *Widgets* 82
Figura 30 – Diagrama de Atividades Referente a Adicionar uma Transação. 84
Figura 31 – Diagrama de Atividades Referente ao Cadastro de usuário. 84
Figura 32 – Diagrama de Atividades Referente a Comparação entre Orçamento Planejado e Real. 84
Figura 33 – Diagrama de Atividades Referente a Edição de Orçamento. 85
Figura 34 – Diagrama de Atividades Referente a Excluir uma Transação. 85
Figura 35 – Diagrama de Atividades Referente a Exibição de Extratos. 85

Figura 36 – Diagrama de Sequência Referente a Adicionar uma Transações. 86
Figura 37 – Diagrama de Sequência Referente ao Início do Cadastro. 87
Figura 38 – Diagrama de Sequência Referente ao Final do Cadastro. 88

Lista de tabelas

Tabela 1 – Comparação entre os dois algoritmos K-CESD-R. 31
Tabela 2 – Pontos Fortes e Fracos dos Trabalhos Relacionados. 36
Tabela 3 – Dicionário de dados da tabela Estado. 48
Tabela 4 – Dicionário de dados da tabela Cidade. 49
Tabela 5 – Dicionário de dados da tabela Universidade. 49
Tabela 6 – Dicionário de dados da tabela Profile. 50
Tabela 7 – Dicionário de dados da tabela Transaction. 50
Tabela 8 – Dicionário de dados da tabela Budget. 51

Sumário

1 INTRODUÇÃO 15
1.1 Objetivos 16
1.2 Metodologia de Desenvolvimento 16
1.3 Resultados e Contribuições 17
1.4 Estrutura da Monografia 17

I REFERENCIAIS TEÓRICOS 19

2 FUNDAMENTAÇÃO TEÓRICA 20
2.1 O Gerenciamento Financeiro Pessoal 20
2.1.1 O Público Universitário 20
2.2 O Estresse Financeiro 21
2.3 O Desenvolvimento Mobile 23
2.4 Considerações finais 27

II TRABALHOS RELACIONADOS 29

3 TRABALHOS RELACIONADOS 30
3.1 Desenvolvimento e avaliação de um aplicativo de autoavaliação diária de depressão otimizado para celular 30
3.2 Usando um gamified app para aumentar o engajamento, retenção e desempenho acadêmico dos alunos 32
3.3 RoadScan: Sistema *Crowdsourcing* para Identificação de Qualidade de Pavimentos Asfálticos 33
3.4 Considerações finais 34

III DESENVOLVIMENTO 41

4 DESENVOLVIMENTO 42
4.1 Aplicativo de Teste 42
4.2 Requisitos 45
4.3 Tecnologias e Recursos Computacionais Utilizados 46
4.4 Modelagem 47
4.4.1 Tratamento dos Dados Coletados 47
4.4.1.1 Primeiro Banco de Dados Local 48

4.4.1.2 Segundo Banco de Dados Local . . . 49
4.4.1.3 Banco de Dados Remoto . . . 51
4.4.2 Casos de Uso . . . 52
4.4.3 Diagrama de Atividades . . . 54
4.4.4 Diagrama de sequência . . . 54
4.4.5 Protótipos de Telas . . . 55
4.4.6 Arquitetura do Software . . . 62
4.5 Implementação . . . 63
4.5.1 Dados Necessários ao Cadastro . . . 63
4.5.2 Anonimidade do Usuário . . . 64
4.5.3 Interface gráfica . . . 64
4.5.4 Versionamento . . . 64
4.6 Testes . . . 65
4.7 Considerações Finais . . . 66

IV RESULTADOS 67

5 RESULTADOS . . . 68
5.1 Fluxo de cadastro . . . 68
5.2 Adicionar Transação . . . 71
5.3 Extratos . . . 71
5.4 Gastos e Rendas . . . 73
5.5 Orçamentos . . . 73
5.6 Informações . . . 75
5.7 Considerações . . . 76

6 CONCLUSÃO . . . 77
6.1 Limitações do trabalho . . . 78
6.2 Contribuições . . . 78
6.3 Trabalhos Futuros . . . 78
6.3.1 Coleta e Análise dos Dados . . . 79
6.3.2 Otimizações da Aplicação . . . 79

APÊNDICES 81

APÊNDICE A – COMPOSIÇÃO DE TELAS COM O *FRAMEWORK* FLUTTER . . . 82

APÊNDICE B – DIAGRAMAS COMPLEMENTARES 84
B.1 Diagramas de Atividades 84
B.2 Diagramas de Sequência 86

1 Introdução

A organização financeira é indispensável para uma vida saudável. Uma boa relação com o dinheiro ajuda a evitar estresses e manter uma situação de vida confortável (??). Essa organização, no entanto, não é simples, é necessário que se passe por um processo antes de ficarem claros quais são os gastos supérfluos ou os gastos passíveis de diminuição e até mesmo a possibilidade de realizar compras mais caras e lidar com endividamentos. Dessa forma, um indivíduo com boa relação com dinheiro tem uma vida mais saudável e confortável tomando como parâmetro poupanças, endividamentos e poder de compra (??).

Um bom nível de organização financeira e, portanto, uma relação saudável com o dinheiro não é um objetivo fácil de ser alcançado. Cada nicho, ou seja cada tipo de núcleo no qual será feita a organização financeira, deve se utilizar de técnicas específicas. Uma família se utiliza de técnicas de organização diferentes de um indivíduo ou de uma empresa por exemplo. Além disso, técnicas utilizadas para empresas grandes e pequenas são diferentes, assim como o gerenciamento pessoal depende do indivíduo analisado, alguém que trabalha com viagens vai ter padrões de gastos e rendas diferentes de alguém que trabalha localmente por exemplo (??).

Tratando especificamente do gerenciamento pessoal, ele depende do estilo de vida do indivíduo, como exemplificado anteriormente. Indivíduos podem identificar padrões gastos e rendas com o auxílio de uma categorização das transações. Essa categorização também depende do indivíduo, mas algumas categorias são padrões tais como gastos com moradia, alimentação, salário e transporte. As especificações dessas categorias também vão depender do indivíduo (??).

Uma vez identificados os padrões de rendas e gastos pessoais, é possível estimar os gastos e rendas esperados por um período de tempo. Assim, pode-se encontrar gastos supérfluos e aqueles passíveis de diminuição ou aumento. Essa estimativa é chamada de Orçamento, que pode ser definido como a lista de estimativas para cada categoria e, como descrito anteriormente, depende totalmente do padrão de gastos de cada indivíduo.

Especificando-se o nicho para estudantes universitários de graduação, os orçamentos ainda podem variar muito de um indivíduo para outro, porém, pode ser percebido certo padrão dentre os estudantes dentro de uma universidade específica. Esses padrões são a respeito de gastos de alimentação, de transporte e de moradia por exemplo, mas mesmo com esse nicho especificado, ainda se tem estudantes que trabalham e outros que só estudam, aqueles com poder aquisitivo maior ou menor, aqueles que possuem carros ou não, etc.

Toda essa especificação, portanto, não é suficiente para se criar um orçamento fixo de previsão dos gastos e rendas para todo o nicho. Dessa forma, cada indivíduo deve fazer a própria organização e esta é complexa pois se utiliza de cálculos estatísticos para identificação

dos padrões apresentados por cada categoria.

A fim de simplificar a realização desses cálculos, podem ser usadas aplicações para dispositivos móveis. Com elas é mais fácil a identificação desses padrões já que eles podem apresentar os dados de forma visual e intuitiva com todos os gráficos e indicadores necessários.

Os gastos do público universitário são frequentemente menores que o de núcleos familiares por haverem diversas iniciativas por parte da universidade com o objetivo de diminuir esses gastos como restaurantes universitários e moradias estudantis, ou incentivos de transporte público, entre outros(**??**).

1.1 Objetivos

O objetivo deste trabalho é desenvolver um aplicativo móvel com a intenção de atender às especificidades do nicho universitário, com categorias comuns a esse nicho e apresentação simples e intuitiva dos orçamentos necessários à boa gestão financeira. Para tal, os seguintes objetivos específicos devem ser realizados para se alcançar o objetivo geral:

- Revisão da literatura sobre gestão financeira para estudantes universitários de cursos de graduação;
- Realização do levantamento e análise de requisitos para o desenvolvimento da aplicação móvel;
- Criação de um protótipo de um aplicativo para realização de testes;
- Revisão dos requisitos, aplicação de melhorias na usabilidade e reestruturação da modelagem de dados.

1.2 Metodologia de Desenvolvimento

A seguir são descritas as metodologias de desenvolvimento utilizadas neste projeto:

a) Identificar as consequências causadas por uma má gestão financeira;

b) Identificar técnicas utilizadas para gestão financeira de estudantes universitários de graduação;

c) Identificar a capacidade de auxílio que o aplicativo móvel pode apresentar, por meio de:

   - Pesquisas de outras ferramentas de auxílio ao usuário que se utilizam de aplicativos móveis;
   - Desenvolvimento de uma aplicação para teste de engajamento dos usuários;

d) Identificar os requisitos da aplicação com base nas outras ferramentas de auxílio, no teste de engajamento realizado e nas técnicas para gerenciamento financeiro identificadas anteriormente.

e) Realizar a modelagem e a implementação do aplicativo MyGrana, com base nas técnicas de:

- Gerenciamento financeiro pessoal para esse público;
- Melhora da usabilidade a fim de garantir o engajamento dos usuários;
- Coleta de dados;
- Criptografia para garantir segurança dos dados coletados.

1.3 Resultados e Contribuições

Até o presente momento, a pesquisa realizada por meio da revisão da literatura não identificou outros trabalhos com o objetivo de auxílio de gestão financeira pessoal do público universitário com uso de uma aplicação móvel. Isso faz com que o MyGrana seja, além de um precedente para outras soluções similares, o ponto inicial de diversas soluções mais complexas para esse tipo de público.

A partir dos dados coletados pelo aplicativo podem ser identificados padrões de comportamento dos usuários quanto às suas respectivas gestões financeiras. Esses padrões podem ser usados como justificativa para trabalhos sociais voltados para esse público, estudos sobre as especificidades relacionadas aos seus gastos e rendas ou programas de educação financeira.

Os resultados encontrados sugerem que o MyGrana tem boa capacidade como ferramenta de auxílio à gestão financeira de estudantes universitários. O aplicativo é intuitivo e com fácil acesso do usuário à todos os recursos apresentados. A coleta de dados garante a não identificação do usuário ao mesmo tempo em que fornece uma base de dados que pode ser utilizada para realização de trabalhos futuros como os citados anteriormente sobre a identificação de padrões de comportamento.

Finalmente, a visualização fácil das informações tornam mais simples a previsão de gastos para cada categoria e demandam menos recursos cognitivos do estudante se comparados com aqueles necessários para fazer os cálculos manualmente.

1.4 Estrutura da Monografia

A estrutura deste trabalho é apresentada conforme descrito a seguir. O Capítulo 2 apresenta a descrição do estudo e pesquisa para os referenciais teóricos deste trabalho. São apresentados os conceitos e técnicas de gerenciamento financeiro pessoal, as especificidades do público universitário, as consequências da má gestão financeira e um histórico do desen-

volvimento *mobile*. No Capítulo 3 são descritos três trabalhos relacionados a este, sendo um de auxílio ao diagnóstico de depressão, um para auxílio ao estudo e o último de identificação da qualidade de pavimentos asfálticos. O capítulo também apresenta considerações a respeito de cada trabalho e como eles se relacionam ao MyGrana. Por sua vez, o Capítulo 4 apresenta todos os artefatos utilizados para modelagem e implementação do MyGrana. Também são descritos a modelagem, a implementação e os resultados do aplicativo de teste, a modelagem dos bancos de dados utilizados e das principais *features*, as tecnologias utilizadas neste trabalho e os testes que foram realizados. No Capítulo 5 são apresentados os resultados obtidos com a implementação, mostrando o funcionamento das telas do aplicativo e como foram satisfeitos todos os requisitos que foram levantados durante o desenvolvimento. Por fim, o Capítulo 6 trás uma análise da forma na qual os objetivos propostos foram alcançados, as limitações que o projeto apresenta, as contribuições que ele proporciona e os trabalhos futuros sugeridos.

Parte I

Referenciais teóricos

2 Fundamentação Teórica

Este capítulo apresenta a fundamentação teórica e descreve os principais conceitos teóricos para este trabalho. A Seção 2.1 trata do gerenciamento financeiro pessoal e como esse se relaciona com o público universitário. Na Seção 2.2 é descrito como o estresse financeiro afeta o público universitário. Já a Seção 2.3 apresenta um breve histórico dos sistemas operacionais para dispositivos móveis e, por fim, a Seção 2.4 trata das considerações sobre este capítulo.

2.1 O Gerenciamento Financeiro Pessoal

O alto fluxo de informações no mundo moderno torna a organização financeira imprescindível em todos os níveis da vida de um indivíduo. Assim, a organização financeira de uma família, de uma pessoa ou de uma empresa torna-se bastante complexa em função das especificidades e dificuldades de cada área. Um orçamento, seja pessoal ou empresarial, é necessário para um bom planejamento do futuro (**??**).

Pensando-se na instância da organização de finanças pessoais, a análise é feita através da categorização de gastos e rendas (**??**). Essa categorização depende do nicho no qual serão aplicadas e ainda podem variar de um indivíduo para outro.

Neste trabalho, é utilizada o conceito de orçamento de **??**) que trata de dois principais pontos: a alocação de recursos escassos e as entradas e saídas de dinheiro. Assim, o orçamento tem por objetivo criar condições para uma escolha certa no momento de compra ou venda e estabelecer padrões que possam ser comparados, controlados e previstos. Esse orçamento apresenta dois pilares: planejamento e controle. O planejamento evidencia tudo o que é esperado e o controle, por sua vez, trata de mudanças no planejamento, como uma comparação do real (aquilo que de fato ocorreu) *versus* o planejado e a aplicação de correções quando necessárias.

2.1.1 O Público Universitário

Na problemática do gerenciamento financeiro pessoal, o público universitário é um nicho muito complexo e cuja organização deve considerar diversos aspectos diferentes (**????**). Ao mesmo tempo que pode-se encontrar estudantes que trabalham e que tem condições de se sustentarem, existem aqueles que são sustentados e recebem suporte de familiares. Existem ainda estudantes de universidades públicas e particulares, cada uma com restaurantes universitários ou não, esse estudante morando sozinho ou em repúblicas estudantis. Tudo isso pode impactar no orçamento. Esses alunos serem sustentados ou não diz respeito ao aspecto socioeconômico de suas famílias. Ser de universidades públicas ou particulares afeta

diretamente o gasto com os estudos já que em um caso ele paga pelos estudos e em outro ele não paga. Restaurantes universitários são estabelecimentos onde os alunos podem se alimentar de forma mais barata, normalmente subsidiados pela própria universidade, e dessa forma afetam diretamente o gasto com alimentação do estudante. Repúblicas estudantis, por outro lado, são discentes que se juntam em uma só residência, diminuindo consideravelmente o gasto com aluguel e com contas da casa.

Pode-se considerar um estudante ou uma república estudantil como sendo um núcleo familiar próprio, uma vez que possui suas próprias formas de receitas, necessidades e gastos. Dessa forma, é possível compreender os gastos universitários por meio do estudo de **??**), com esse núcleo sendo um núcleo de classe C, já que suas receitas ficam normalmente entre $R\$1.200,00$ e $R\$5.174,00$, partindo do pressuposto de que a restrição de renda faz com que qualquer decisão de compra tenha grande impacto nas finanças.

A organização financeira de um núcleo de classe C se resume em dois tipos:

a) Orçamento mental: marcado por organizar mentalmente os gastos e rendas. Normalmente utilizado da seguinte forma: no momento da compra de algo, o indivíduo pensa se o restante do dinheiro duraria o tempo que deve durar e, com base na resposta, a compra é realizada ou não;

b) Gerenciamento das receitas e despesas: organização em um orçamento planejado e controlado, ideia que é utilizada neste trabalho.

A organização financeira de um universitário deve conter em sua categorização elementos cotidianos da vida desse estudante tais como materiais de estudo, mas não pode se limitar a isso uma vez que significativa parcela desse público encontra-se em cidades diferentes de suas famílias, logo, a categoria de transporte deve ter sua própria ênfase. Além disso, trata-se de um público jovem e, portanto, pode-se esperar gastos altos com entretenimento. E também devem ser considerados investimentos ou poupanças a fim de servir como um caminho em direção a uma vida financeira saudável.

??) afirma que a gestão da renda depende de quatro características pessoais de uma pessoa: autocontrole, visão de curto prazo, preferência pelo crédito (cartão de crédito e parcelamentos) e propensão a planejar. O diferencial é que o estudo trata de núcleos com idades acima de 30 anos e como este trabalho trata de um núcleo normalmente mais jovem, essas características podem ser assimiladas pelo núcleo de estudantes universitários.

2.2 O Estresse Financeiro

O estudo de **??**) é uma das justificativas desse trabalho. Ele propõe fazer medições demográficas de estresse geral e saúde de 177 estudantes e aplica os resultados em uma escala

desenvolvida pelo trabalho com o objetivo de mensurar o estresse financeiro dos participantes. Esse estudo mostra os efeitos negativos de tal estresse na vida universitária:

a) Saúde: Apesar de ainda não ser completamente confirmado, vários estudos mostram correlação entre estresse financeiro com problemas físicos e de saúde. Da mesma forma, estudos mostram que uma relação saudável com o dinheiro é associada à uma saúde melhor;

b) Comportamentos saudáveis: estudos demonstram que estresse financeiro se relaciona a hábitos não-saudáveis, tais como fumo, consumo de bebidas alcoólicas, baixos índices de massa corporal, dietas ruins e redução na prática de exercícios;

c) Saúde mental: estudos demonstraram uma relação entre altos níveis de estresse financeiro e distúrbios de ansiedade, depressão, baixa auto-estima, uma imagem ruim de si mesmo e auto-confiança;

d) Resultados Acadêmicos: estudantes em dívida acabam por não procurar por educação acadêmica. Preocupações financeiras também foram associadas ao pensamento de desistir da graduação. Esses estudantes também se dedicam a mais horas de trabalho, o que prejudica seu desempenho acadêmico. Condições de vida ruim também foram especuladas como possível consequência de alto estresse financeiro;

e) Relacionamentos: alto estresse financeiro também foi associado à conflitos em relacionamentos, desde amizades à conjugais. Além disso, filhos de pais com alto estresse financeiro foram associados com altos níveis de depressão.

De uma perspectiva cognitiva, estudantes são obrigados a dedicar altos níveis de recursos cognitivos para regular ou mitigar efeitos do estresse financeiro. Isso faz com que indivíduos com altos níveis de estresse financeiro tenham menos recursos cognitivos para alocar à outras esferas de suas vidas.

Foram recrutados estudantes universitários (um total de 177) e todos foram submetidos a cinco métricas. Questionários foram usados pra coletar as seguintes informações:

a) Questionário demográfico: objetivo de coletar informações como idade, gênero, raça, status socioeconômico, saúde geral, nível educacional, estado civil, vínculos empregatícios e salário anual;

b) Inventário de estresse diário (*Daily Stress Inventory* ou DSI): objetivo de medir o nível de estresse geral dos participantes. Ao total, são 58 itens onde os participantes marcam se aquele causador de estresse em particular ocorreu nas últimas 24 horas e, caso tenha ocorrido, classificá-lo de 1 (não causou estresse) a 7 (causou pânico). O DSI gera 3 *scores*:

   - Frequência: número total de eventos ocorridos com o participante;
   - Impacto: soma de todos os números marcados;

- Impacto médio: *score* obtido dividindo-se o impacto pela frequência.

c) *Short-Form-36* de pesquisa de saúde (SF-36): questionário com 36 itens com a intenção de medir 8 subescalas: funcionamento físico, limitações devido a problemas físicos, dores no corpo, funcionamento social, saúde mental geral, limitações devido a problemas emocionais, energia/fatiga e percepções de saúde geral;

d) Índice de qualidade de vida multidimensional (*Multidimensional Index of Life Quality* ou MILQ): a subescala *status* financeiro do MILQ é utilizada para mensurar qualidade de vida financeira. Eram utilizadas quatro questões nas quais *scores* baixos refletem baixa qualidade de vida financeira. É importante ressaltar que, na falta de uma medida de estresse financeiro, nenhum critério poderia identificar ou ser usado pra validar essa subescala;

e) Escala de Estresse Financeiro - Versão Universitária (*Financial Stress Scale - College Version* ou FSS-CV): escala criada pelo estudo. Trata-se de um questionário de 22 questões desenvolvido para medir estresse financeiro. É respondido pelo participante por uma escala de 4 pontos, de nunca a todo o tempo.

O nível de estresse geral e financeiro pode ser obtido a partir do conjunto das 5 escalas citadas anteriormente. A FSS-CV foi considerada confiável dentro do universo dos 177 participantes, porém, é uma amostra muito pequena para se generalizar para todo o público universitário, ou seja, nesse ponto ela é inconclusiva, mas ainda assim traz muitas informações relevantes e que devem ser levadas em consideração.

Tendo em vista todos os malefícios de uma vida estressante financeiramente para um estudante universitário, como foi mostrado anteriormente, a criação de uma ferramenta de auxílio à vida financeira dos estudantes é de grande importância. Esse trabalho se propõe a desenvolver essa ferramenta com o intuito de mitigar os efeitos negativos desse estresse financeiro, o que pode levar a uma vida mais saudável e produtiva para os estudantes que sofrem com esse problema.

2.3 O Desenvolvimento Mobile

A aplicação móvel proposta por esse trabalho para mitigar o problema do estresse financeiro se utiliza de importantes conceitos desenvolvidos durante toda a história dos dispositivos móveis e seus aplicativos (apps ou aplicações).

Enquanto os primeiros telefones móveis datam da década de 30 com sistemas completamente analógicos, a sua disseminação veio com a digitalização desses dispositivos. No final da década de 80 veio o sistema operacional mobile EPOC com suas especificações bem simples e, em 1998 foi lançado o Symbian, primeiro a começar a disseminação desse tipo de dispositivo,

principalmente entre jovens com os aparelhos da Nokia[1] e Sony Ericsson. O Pocket PC 2000 (chamado normalmente de Windows Mobile) foi lançado em 2000 e era baseado na interface gráfica do Windows 95[2], e esse deu mais um passo no crescimento desses dispositivos já que o Windows 95 era muito utilizado nessa época.

A disseminação massiva desses sistemas, no entanto, veio com o lançamento do iPhone[3], primeiro dispositivo com sistema iOS em 2007 e do HTC Dream[4] em 2008, primeiro aparelho com sistema Android do mercado. O fato desses dispositivos se tornarem cada vez mais acessíveis, mesmo com hardwares modernos e alta capacidade de processamento aumentou ainda mais essa disseminação. Por um tempo o Windows Mobile se manteve um forte concorrente no mercado de smartphones, mas acabou perdendo espaço e atualmente aparelhos Android e iOS dominam esse mercado, correspondendo a mais de 90% dos smartphones comercializados no mundo.

Esse domínio de mercado se deu principalmente por esses aparelhos permitirem o desenvolvimento relativamente fácil de apps. Essas aplicações, que auxiliam de alguma forma a vida das pessoas, foram também um catalisador para o sucesso desses dispositivos.

Foram citados alguns dos principais sistemas operacionais, mas também houve outros, como por exemplo o BlackBerry OS[5], que não fizeram tanto sucesso com o público geral quanto os descritos anteriormente.

O Android se propôs a ser um sistema facilmente atualizável com compatibilidade com diversos hardwares e de mútua cooperação entre o sistema e empresas de telefonia móvel e de construção de hardware. O iOS, por outro lado, se propôs a ser um sistema operacional para dispositivos específicos da Apple, os quais possuem hardware robusto e se dedicam a parcelas do público que se dispõem a pagar um valor alto por esses dispositivos. Ambos obtiveram muito sucesso e mais de uma década depois são os mais utilizados no mundo.

O desenvolvimento de aplicativos para esses dispositivos se dá a partir de ferramentas e linguagens de programação disponibilizadas pelos próprios fabricantes. No Android utiliza-se Kotlin, Java, C e C++, enquanto no iOS usa-se Objective-C e Swift. Como cada sistema se utiliza de suas linguagens específicas e não há uma linguagem em comum, o desenvolvimento de apps para as duas plataformas se torna basicamente o desenvolvimentos de dois aplicativos diferentes: um para Android e outro para iOS. Isso fica muito caro pois são necessários dois processos de desenvolvimento, uma para programação do app em uma plataforma e outra para reprogramação utilizando as linguagens da outra plataforma.

Para suprimir esse problema, foram criados diversas ferramentas (*frameworks*) capazes de, a partir de um único código-fonte em alguma linguagem específica de cada uma, criar

[1] <https://www.nokia.com/pt_int/>
[2] <https://www.microsoft.com/pt-br>
[3] <https://www.apple.com/br/iphone/>
[4] <https://www.htc.com/us/>
[5] <https://www.blackberry.com/br/pt>

aplicações para ambos os ambientes, por exemplo:

a) PhoneGap[6], Cordova[7] e Ionic[8]: HTML, CSS e JavaScript;

b) React Native[9]: JavaScript;

c) Flutter[10]: Dart;

d) Xamarin[11]: C#;

e) Rhomobile[12]: Ruby;

f) Unity[13]: C++ e C#.

Todos esses *frameworks* compilam ou interpretam o código-fonte gerando, de alguma forma, aplicações para diversos ambientes diferentes, facilitando para o desenvolvedor e para as empresas criadoras de software diminuindo o preço do desenvolvimento de apps e aumentando ainda mais o alcance dos apps criados.

Atualmente, existem vários apps com alcance global de diversas categorias, como por exemplo:

a) Redes Sociais: Facebook[14], Twitter[15], TikTok[16], Instagram[17], Snapchat[18];

b) Comunicação instantânea: WhatsApp[19], Telegram[20], Skype[21];

c) Jogos: Garena Free Fire[22], Candy Crush Saga[23];

d) Outros tipos: Uber[24], iFood[25], YouTube[26], Spotify[27], etc.

Todos os aplicativos citados anteriormente estão disponíveis pelo menos para Android e iOS, o que mostra que estar disponível nas duas plataformas é importante pro sucesso de

6 <https://phonegap.com>
7 <https://cordova.apache.org>
8 <https://ionicframework.com>
9 <https://reactnative.dev>
10 <https://flutter.dev>
11 <https://docs.microsoft.com/pt-br/xamarin/get-started/>
12 <https://www.zebra.com/us/en/support-downloads/software/developer-tools/rhoMobile-suite.html>
13 <https://unity.com/pt>
14 <https://www.facebook.com>
15 <https://twitter.com/>
16 <https://www.tiktok.com/pt-BR>
17 <https://www.instagram.com>
18 <https://www.snapchat.com/l/pt-br/>
19 <https://www.whatsapp.com/?lang=pt_br>
20 <https://web.telegram.org/>
21 <https://www.skype.com/pt-br/>
22 <https://ff.garena.com>
23 <https://www.king.com/pt_BR/game/candycrush>
24 <https://www.uber.com/br/pt-br/>
25 <https://www.ifood.com.br>
26 <https://www.youtube.com>
27 <https://www.spotify.com/br/>

um aplicativo móvel. Muitos desses aplicativos vêm de softwares para computadores ou de plataformas Web que aumentaram seu alcance com os aplicativos móveis.

O WhatsApp é um exemplo de aplicação móvel de alcance mundial que traz diversos recursos para facilitar a comunicação instantânea de pessoas independentemente do lugar em que elas estejam, desde que tenham uma conexão com a Internet. Seus principais recursos, mostrados na Figura 1, mostram o quanto um app completo e desenvolvido com objetivo claro de ajudar as pessoas em um âmbito específico de suas vidas consegue se disseminar dentre os usuários e ajudar cada vez mais pessoas.

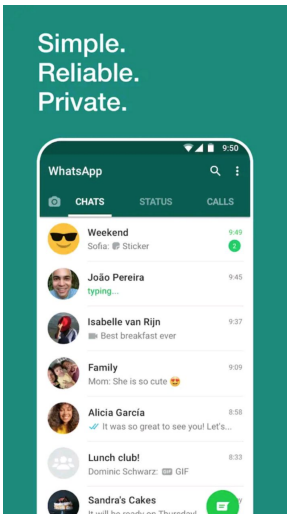
A Figura 1a mostra a tela inicial do aplicativo, onde pode-se notar todas as mensagens recebidas ou enviadas e as *tabs*[28] pra acesso à câmera, aos *status* ou às ligações, além de acesso a pesquisa e configurações. A Figura 1b é um exemplo de chat em grupo, pode-se adicionar várias pessoas a um grupo e uma mensagem enviada a este grupo chega a todos os participantes dele. A Figura 1c mostra um exemplo de uma chamada de vídeo, essas podem ocorrer entre apenas 2 usuários ou com grupos completos. Na Figura 1d pode-se ver um recurso muito interessante, que deixa disponível a um usuário a localização em tempo real de um outro usuário por um período determinado de tempo, muito utilizado por questões de segurança pessoal. Já a Figura 1e mostra um exemplo de um *status*, recurso no qual um usuário pode compartilhar imagens e textos com todos aqueles que possuem seu número (esse alcance pode ser filtrado nas configurações) e esses podem responder ao *status* apenas deslizando o dedo na tela de baixo pra cima.

Dentro de uma conversa, seja em grupo ou em particular, é possível anexar às mensagens imagens, vídeos, gifs ou *stickers*, bem como responder a uma mensagem específica ou no caso de grupos marcar um usuário em uma mensagem. Além disso, a segurança das mensagens se dá por criptografia ponta-a-ponta, o que faz com que mensagens interceptadas não possam ser decifradas e, além disso, a empresa afirma que as mensagens não são guardadas em nenhum de seus servidores, logo, ficam armazenadas apenas nos dispositivos participantes.

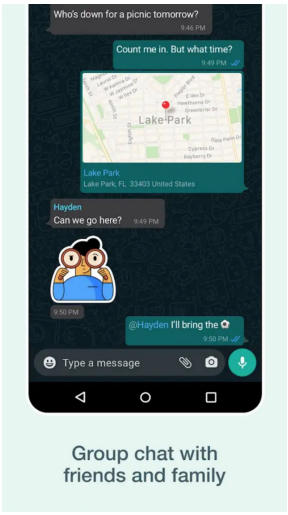
O aplicativo também apresenta protocolos próprios de compressão de áudio, imagens e vídeos, com o objetivo de melhorar a experiência do usuário.

Diante do disposto, pode-se ver que o desenvolvimento por *frameworks* multiplataforma é muito importante, já que garante um alcance maior do que o app teria se fosse apenas para uma plataforma, mas sem o custo de reprogramação necessário para se desenvolver um app pra cada plataforma.

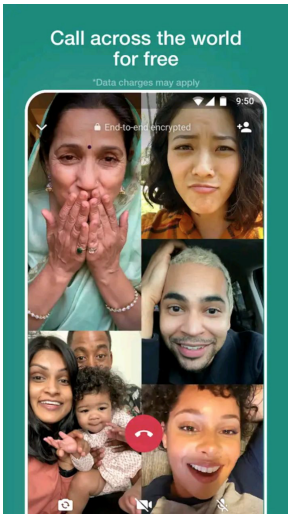
[28] Técnica utilizada por programadores para deixar visível um maior número de recursos sem sobrecarregar a interface gráfica. Ela consiste em esconder as telas das *tabs* que não estão selecionadas como subtelas que são exibidas ao se selecionar alguma *tab* específica(**??**).



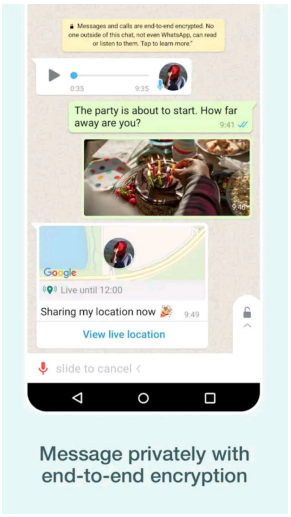
(a) Tela inicial.



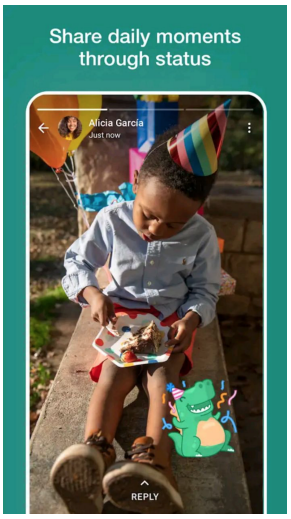
(b) Conversa em grupo.



(c) Ligação de vídeo.



(d) Localização instantânea.



(e) Status.

Figura 1 – Imagens Promocionais do Whatsapp.

Fonte: <https://play.google.com/store/apps/details?id=com.whatsapp>

2.4 Considerações finais

O gerenciamento financeiro pessoal, seja ele mental ou físico, é essencial pra uma boa vida financeira. Por mais que muitos núcleos familiares sejam muito bem organizados com o gerenciamento mental, o físico é mais recomendado. Anotar gastos e rendas e organizá-los em categorias ajuda a planejar, controlar e prever relações de causa e efeito e padrões dentro da vida financeira.

O público universitário deve ter uma organização à parte. Sua categorização de gastos e rendas deve ser de fato organizada e otimizada para suas necessidades e hábitos. Por ser um público jovem, muitas características para uma boa vida financeira podem ser adquiridas, como autocontrole, visão de curto prazo, preferência pelo crédito e propensão a planejar.

Ainda que seja um público jovem, no entanto, as consequências de um bom ou mal

planejamento financeiro tem um grande impacto na vida do universitário. A combinação de pouco autocontrole com alta preferência pelo crédito pode causar inadimplência e endividamento, por outro lado, uma boa propensão a planejar e visão de curto e médio prazo podem trazer benefícios como boa acumulação de capital, seja em poupanças ou investimentos.

O mal gerenciamento financeiro causa um estresse que dentro do público universitário pode trazer diversos males: problemas de saúde física e mental, comportamentos não-saudáveis, queda no desempenho acadêmico e problemas em relacionamentos.

Uma exposição prolongada a esse tipo de estresse, apesar de não apontada pelo estudo de **??**), pode induzir problemas ainda mais severos como de saúde física e mental e talvez deixar sequelas no indivíduo.

Desenvolver uma aplicação móvel para mitigar esse problema é importante já que, desde 2007 com o lançamento do primeiro iPhone e 2008 com o lançamento do primeiro dispositivo Android, o mercado de smartphones só cresce, com alguns aplicativos (como o Facebook ou Whatsapp) já ultrapassando a marca de bilhões de downloads. Esse desenvolvimento pode ser nativo ou multiplataforma dependendo da necessidade. Para este trabalho foi desenvolvida uma aplicação nativa pra Android a fim de teste de engajamento e de coletas de dados e um app multiplataforma a fim de poder ajudar não só os usuários Android, mas também usuários iOS, utilizando apenas um código-fonte.

Parte II

Trabalhos relacionados

3 Trabalhos Relacionados

Nesta seção são apresentados os trabalhos correlatos que formam a base de conhecimento para a construção e desenvolvimento deste projeto. A Seção 3.1 apresenta a construção de um aplicativo para detectar sintomas depressivos. A Seção 3.2 descreve um aplicativo com objetivo de aumentar engajamento de estudantes. A Seção 3.3 descreve um trabalho capaz de detectar a qualidade da estrada utilizada pelo usuário e a Seção 3.4 traz considerações sobre a importância que cada aplicativo representa dentro do nicho pro qual ele é desenvolvido.

3.1 Desenvolvimento e avaliação de um aplicativo de autoavaliação diária de depressão otimizado para celular

O estudo de **??**) é um estudo preliminar e tem por objetivo mostrar a possibilidade (ou impossibilidade) de se identificar pacientes depressivos com base em um aplicativo *mobile*. Para tal, ele se utiliza de dois algoritmos de detecção de sintomas depressivos e ao final apresenta qual desses algoritmos foi mais acurado de acordo com os testes clínicos originais.

O algoritmo utilizado é uma variação do *Center of Epidemiologic Studies Depression Scale-Revised* ou CESD-R (**??**), isso por ser o algoritmo mais utilizado atualmente, segundo o artigo, tanto no campo de epidemiologia psiquiátrica, quanto em estudos psiquiátricos, quando comparado a outros algoritmos utilizados em trabalhos similares. Nesse algoritmo, o paciente responde perguntas sobre suas últimas duas semanas, porém, esse período causa certa imprecisão nas medidas, uma vez que depende da memória do paciente.

A ideia, portanto, é desenvolver um estudo no qual o usuário responderia essas questões diariamente por duas semanas, ao invés de fazer em *retrospective recall* (respondendo sobre os 14 dias anteriores de uma só vez), atingindo um resultado mais preciso.

Por ter sido realizado em Seoul, a escala utilizada pelo estudo foi a versão Coreana do algoritmo, K-CESD-R. Os objetivos do trabalho são:

a) Desenvolver uma aplicação *mobile* para coletar os dados reportados pelos usuários de acordo com a escala K-CESD-R;

b) Desenvolver dois algoritmos para converter o resultado K-CESD-R em 4 categorias;

c) Comparar o resultado da versão original do K-CESD-R com o resultado dos dois algoritmos de conversão;

d) Determinar qual versão do algoritmo é mais adequada.

Os participantes do estudo foram escolhidos a partir de uma propaganda nos sites de

universidades de Seoul na Coreia do Sul. Foram selecionados 20 estudades de graduação jovens adultos com idade entre 19 e 29 anos.

Foram desenvolvidos dois aplicativos *mobile*, um para plataforma Android e outro para iOS, sendo os dois com interface gráfica idêntica e com 3 objetivos:

a) Guiar os usuários durante a resposta de um questionário diário de 20 perguntas de "sim" ou "não", de acordo com suas experiências nas últimas 24 horas;

b) Manter o engajamento dos participantes por 2 semanas;

c) Medir o tempo de resposta do usuário a cada pergunta.

As 20 perguntas eram aleatorizadas dia a dia para evitar vícios de respostas. Os usuários que responderam o questionário por menos de 7 dias foram desconsiderados nos resultados.

Seja Y o número de respostas "sim" dadas pelo usuário durante os 14 dias, esse valor é dividido por 2 a fim de se ter a média semanal e depois comparado com as escalas dos dois algoritmos (de *0* a *4*), A tabela Tabela 1 apresenta a comparação entre os dois algoritmos.

Tabela 1 – Comparação entre os dois algoritmos K-CESD-R.

Categoria	**Algoritmo A**	**Algoritmo B**
0	$Y = 0$	$0 \leqslant Y < 1$
1	$0 < Y \leqslant 2$	$1 \leqslant Y < 2.5$
2	$2 < Y \leqslant 4$	$2.5 \leqslant Y < 4.5$
3	$4 < Y \leqslant 6$	$4.5 \leqslant Y < 6.5$
4	$6 < Y \leqslant 7$	$6.5 \leqslant Y \leqslant 7$

Fonte: Retirado de **??**)

Com esses dois algoritmos, é possível medir a gravidade dos casos de depressão (sendo $Y = 0$ ou $Y = 1$ estados não depressivos e $Y = 2$, $Y = 3$ e $Y = 4$, estados depressivos) dos usuários com base em respostas diárias, o que faz com que a memória do usuário não seja um fator que possa levar a imprecisão na coleta dos dados dos questionários.

Após o uso do aplicativo, o usuário foi avaliado por diversos algoritmos que se utilizam do *retrospective recall*, a fim de se comparar os resultados. O Algoritmo B foi o mais preciso, sendo capaz de:

a) Manter uma avaliação diária de sintomas depressivos;

b) Identificar casos de depressão em estados iniciais ao reduzir erros de medida por não depender da memória do usuário;

c) Recomendar a ida dos usuários a clínicas psiquiátricas ainda com sintomas leves.

Por ser um estudo preliminar, ao final é recomendado a continuação do estudo e comparação com mais estados clínicos.

3.2 Usando um gamified app para aumentar o engajamento, retenção e desempenho acadêmico dos alunos

O estudo de **??**) começa com uma análise do processo de uso de aplicativos para melhoria do aprendizado dos alunos no ensino superior e quais os casos em que eles já são utilizados. Dentre os casos citados estão: facilitar transferência de conhecimento do professor ao aluno, simular situações experimentais para estudantes de medicina e prover assistência a atividades durante a aula para *feedback* imediato.

Porém, ainda falta muito na pesquisa para desenvolvimento desses aplicativos com reais impactos e, para contribuir com isso, esse estudo se propõe a investigar se o uso de um *"gamified app"*[1], que foi apenas avaliado (e não desenvolvido) por esse estudo, tem impacto positivo na performance acadêmica dos estudantes e no engajamento na disciplina avaliada.

Para avaliar esse impacto, a pesquisa teve como objetivo responder a três questões principais:

a) A introdução do app se correlaciona com o nível de retenção dos estudantes?

b) O uso do app se correlaciona com a melhora do desemplenho acadêmico dos estudantes?

c) A performance dos estudantes no app se correlaciona à performance acadêmica dos estudantes?

Aos alunos foi dada a opção de utilizar o app que havia sido desenvolvido por outro projeto na própria universidade para as plataformas Android e iOS. Para aqueles que decidiram utilizar o app, receberiam *push notifications*[2] em períodos regulares, antes e após as aulas, convidando-os a responder um questionário dos conceitos abordados na aula em um quiz de múltipla-escolha, onde teriam de 20 a 60 minutos para responder cada pergunta.

Baseado no engajamento do aluno é possível coletar alguns dados como velocidade da resposta à pergunta e número de tentativas antes de chegar a opção correta. Essas estatísticas eram usadas em um *scoreboard*[3], com um vencedor anunciado a cada semana.

As estatísticas de uso do app também também alimentavam uma base de dados da universidade, onde os alunos poderiam ver suas pontuações atuais e poderiam obter *digital badges*[4] com base em sua performance no quiz.

[1] Técnica utilizada para melhorar o engajamento de usuários. Consiste em projetar o app de forma a parecer um jogo ao ser utilizado pelo usuário(**??**).

[2] Notificações "empurradas" ou enviadas ao usuário, por exemplo, a notificação que aparece após o recebimento de uma mensagem em um aplicativo de mensagens, ou neste caso uma notificação acionada pelo professor antes e após as aulas(**??**).

[3] Quadro de pontuações mostrando as pontuações dos usuários organizadas de modo decrescente.

[4] Marcações que definem e marcam a posição no *scoreboard* dos usuários em semanas anteriores ou atual.

Os estudantes também poderiam optar por não participar das pontuações e se utilizar do aplicativo apenas como ferramenta de estudo. Dos 711 estudantes nas turmas selecionadas, 394 (55%) aceitaram utilizar o app e ter seus dados coletados pelo estudo. Caso houvesse alguma mudança de opinião nesse aspecto havia uma opção no app de parar de disponibilizar os dados, mas nenhum dos alunos do estudo se utilizou dessa opção. O app foi introduzido no segundo semestre de 2015 e foram calculadas melhoras de retenção dos alunos de $12,23\%$ quando comparado com o primeiro semestre de 2015, $9,22\%$ quando com parado com o segundo semestre de 2014 e $5,37\%$ quando comparado com o segundo semestre de 2013.

Os utilizadores do app se mostraram com uma performance acadêmica $7,03\%$ maior, comparando com os que não o utilizavam. Foi mostrada uma correlação positiva e significante entre a performance dos estudantes no app e suas performances acadêmicas.

Esse estudo tem como foco encontrar uma correlação entre o uso de um *gamified app* e o desempenho acadêmico dos estudantes e, portanto, não trata de uma relação de causa e efeito. Por esse motivo um estudo mais duradouro deve ser feito, a fim de se aprofundar nessa correlação e entendê-la por completo removendo, inclusive, estímulos externos que possam ter tido alguma relevância.

3.3 RoadScan: Sistema *Crowdsourcing* para Identificação de Qualidade de Pavimentos Asfálticos

O objetivo do trabalho de **??**) é desenvolver uma aplicação, chamada RoadScan, e um sistema Web hospedado em um servidor que cruza os dados coletados pelos usuários e é capaz de indicar os trechos de rodovias com base em sua respectiva qualidade.

Rodovias são importantes recursos econômicos e sociais, com uma grande parcela da movimentação de pessoas e cargas sendo feitas de maneira rodoviária. A qualidade de tais rodovias depende muito do clima local e também da intensidade do tráfego de veículos, principalmente veículos pesados. Isso faz com que o monitoramento da qualidade do asfalto precise ser feito de forma constante e a falta de praticidade e alto custo financeiro marcam as tecnologias atuais de monitoramento (**??**).

A implementação de uma metodologia *crowdsourcing*[5] trás grandes benefícios uma vez que uma aplicação com mínima interação com o usuário pode ser capaz de detectar via acelerômetro alterações na posição do smartphone e, com usuários suficientes, é possível medir com clareza trechos bons e ruins em rodovias.

A aplicação foi desenvolvida para ambiente Android, utilizando linguagens Java e XML, por familiaridade do autor. Os testes foram realizados em 5 smartphones e 5 carros diferentes,

[5] Metodologia que se utiliza dos dados coletados por diversos usuários para que possa ocorrer um processamento mais preciso dos mesmos.

para analisar a diferença entre os dados coletados em cada smartphone e por cada carro. O sistema Web foi desenvolvido utilizando a API GoogleMaps[6] e HTML e a comunicação com o Banco de Dados utilizou AJAX. A comunicação cliente-servidor utilizou-se do protocolo JSON.

As análises foram feitas através dos dados coletados do acelerômetro e o desvio padrão desses dados, levando em conta o carro utilizado e seu histórico recente. Os dados processados foram separados em cinco categorias de qualidade da rodovia, sendo eles:

a) Bom;

b) Normal;

c) Ruim;

d) Horrível;

e) Problema Pontual.

A coleta é realizada apenas para velocidades acima de 20km/h para evitar a captura de arrancadas e de quebra-molas. Para a aplicação fica a responsabilidade de capturar os dados e calcular o desvio padrão deles em tempo real, armazenando os dados em um Banco de Dados local. Uma vez detectada conexão com a Internet, esses dados são enviados ao servidor, onde são processados.

O algoritmo se utiliza de duas lógicas pra classificação de qualidades e uma pra detecção de buracos. Sem distinção de velocidade é possível identificar trechos bons e péssimos graças à média dos desvios padrão dos dados coletados pelos usuários. Valores baixos identificam trechos bons, valores altos identificam trechos péssimos. Para identificação de trechos médios e ruins utiliza-se a informação da velocidade. Com ela pode-se analisar picos de leitura acima de 50%, que podem ser causados por um trecho ruim ou buracos. Picos consecutivos identificam um trecho ruim e sua ausência identificam trechos médios. Picos de magnitude maior que 100% em relação a média representam problemas pontuais na pista, buracos, bueiros ou deformidades bruscas.

Após 3 meses em teste, o RoadScan apresentou resultados promissores, capaz de identificar a qualidade dos trechos de estrada e, quando errado, erra no máximo por 1 nível de diferença, o que mostra que não apresenta resultados discrepantes com a realidade.

3.4 Considerações finais

Esses trabalhos mostram o poder de flexibilidade dado aos aplicativos móveis e o quanto eles são capazes de ajudar as pessoas em diversos aspectos de suas vidas.

No trabalho de **??**), pode-se ver um aplicativo capaz de detectar sintomas depressivos, podendo ajudar tanto os usuários (sendo uma ferramenta barata de detecção da doença) quanto

[6] <https://cloud.google.com/maps-platform/maps?hl=pt-br>

os médicos, dando a eles uma possibilidade diferente e menos dependente da memória dos pacientes.

A utilização de um questionário diário facilita o uso por parte do usuário já que a única preocupação que ele deve ter é a de entrar no app e responder às perguntas sobre seu dia, fornecendo os dados que serão utilizados para o diagnóstico.

O trabalho de **??**) se utiliza de um aplicativo para melhorar o desempenho acadêmico dos usuários discentes. Isso é percebido através de uma atmosfera de competição onde os professores fornecem perguntas aos alunos e suas respostas geram pontuações. A soma dessas pontuações conferem uma posição no *scoreboard*, fazendo com que os alunos disputem entre si pelas maiores pontuações.

O professor também é auxiliado, uma vez que consegue avaliar de forma simples a evolução da turma e os pontos nos quais os alunos tiveram mais dificuldade. Logo, ao mesmo tempo em que o app incentiva os estudantes a se dedicarem mais aos estudos, também sinaliza ao professor pontos de maior dificuldade da disciplina.

Por outro lado, o trabalho de **??**) cria uma aplicação a fim de se medir a qualidade de pavimentos asfálticos. Isso é de grande ajuda para seus usuários uma vez que trechos de baixa qualidade podem ser responsáveis por diversos problemas tais como problemas no veículo, maior demora pra se chegar no destino (pois o veículo é mantido em baixa velocidade) e até acidentes.

A Tabela 2 mostra um resumo dos pontos fortes e fracos de cada trabalho.

Tabela 2 – Pontos Fortes e Fracos dos Trabalhos Relacionados.

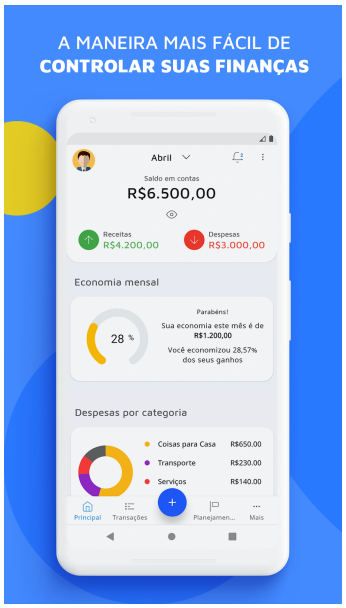
Trabalho	Tecnologias Utilizadas	Pontos Fortes	Pontos Fracos	Área de atuação
??)	*Amazon Web Services*, escala K-CESD-R	Desenvolvimento de uma nova escala de detecção de sintomas depressivos, teste não depende da memória de longo prazo do usuário	Desenvolvido 1 app pra Android e outro pra iOS, a escala criada só seria válida considerando apenas o público coreano	Saúde
??)	Técnicas Estatísticas, *Gamifing elements*	Capaz de auxiliar tanto alunos quanto professores, gera aumento na performance acadêmica	Amostra pequena, precisa de engajamento não só do usuário, mas também de outros usuários	Educação
??)	Sensores do dispositivo, cálculos estatísticos, Google Maps, técnica de *Crowdsourcing*	Alta acurácia em seus resultados, coleta de dados praticamente independente do usuário, fácil visualização dos resultados	Apenas disponível para Android, teste com poucos usuários, incapacidade de coletar dados com velocidades abaixo de 20km/h, resultados obtidos a partir de constantes predefinidas para cada modelo de carro impossibilitando generalidade	Mobilidade
MyGrana	Desenvolvimento multiplataforma, Google Firebase	Diversas ferramentas para o auxílio do usuário em sua organização financeira, disponibilidade para Android e iOS, capaz de coletar diversas informações sobre o usuário	Teste final realizado com poucos usuários, não permite múltiplos dispositivos em uma única conta, não avisa o usuário próximo ao vencimento de um pagamento agendado, somente permite usuários brasileiros	Organização

Fonte: O autor.

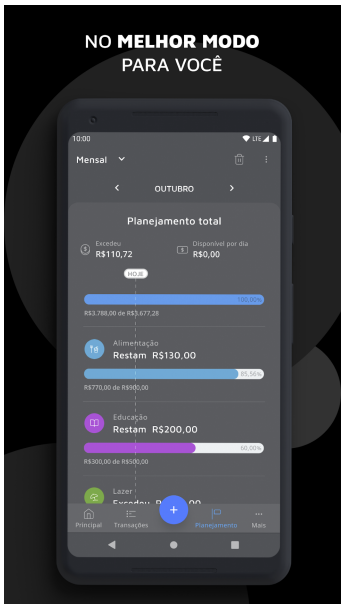
Além disso, existem outros aplicativos para gerenciamento financeiro de público geral que permitem criação de orçamentos e categorização de gastos e rendas, além de uma fácil

visualização desses dados categorizados. Dentre eles podem ser destacados os aplicativos Mobills[7] e o Guia Bolso[8] .

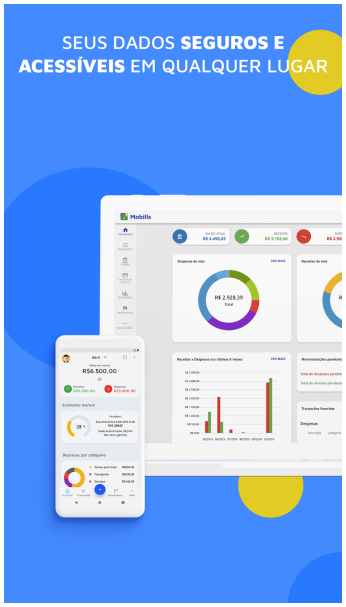
Com *features* diversas, o Mobills permite dentre outras coisas a criação de orçamentos divididos por categorias, a adição de novas categorias e subcategorias, além do planejamento de metas. As telas de publicidade do aplicativo são mostradas na Figura 2:



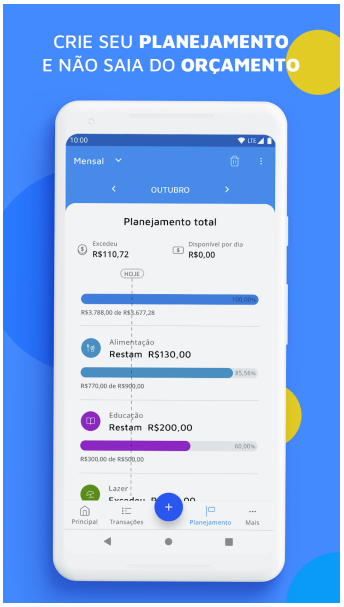
(a) Tela inicial.



(b) Modo escuro.



(c) Integração com sistema Web.



(d) Comparação entre gastos planejados e reais.

Figura 2 – Imagens promocionais do aplicativo Mobills.

Fonte: <https://play.google.com/store/apps/details?id=br.com.gerenciadorfinanceiro.controller>.

[7] <https://www.mobills.com.br>
[8] <https://www.guiabolso.com.br>

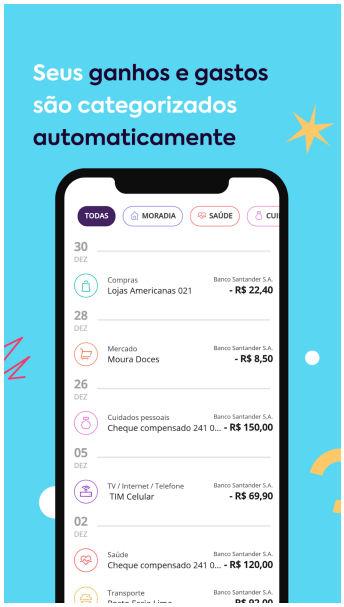
O aplicativo atende bem a um público que já possua certa educação financeira, isso porque não existem categorias pré-definidas na primeira execução do aplicativo então o próprio usuário deve adicioná-las. Isso permite alta capacidade de personalização do orçamento do usuário, porém em usuários com uma educação financeira ruim permite a criação de categorias desnecessárias que dificultam o planejamento do orçamento.

Categorias devem ser feitas de acordo com um padrão nos gastos e deve-se tomar cuidado para não se dividir gastos que possuam os mesmos padrões em categorias diferentes. A inclusão de categorias demais no orçamento cria a dificuldade de definir um limite de gastos mensal para cada categoria, fazendo com que o aplicativo não consiga ajudar na gestão financeira de alguns usuários como se propõe.

O Guia Bolso é um aplicativo mais robusto, que permite conectar contas bancárias dentro do aplicativo e aplica categorização automática para compras realizadas por cartão. Ele também permite criação de novas categorias, além de separar as categorias em grupos (Renda, Gastos Essenciais, Estilo de Vida, entre outros) e simular empréstimos. As imagens promocionais do aplicativo são mostradas na Figura 3.

(a) Tela de transferência.

(b) Tela inicial.



(c) Tela de extratos.

(d) Tela de empréstimos.

Figura 3 – Imagens promocionais do aplicativo Guia Bolso.

Fonte: <https://play.google.com/store/apps/details?id=br.com.guiabolso>.

Para usuários com certa educação financeira, o aplicativo é de grande auxílio, as integrações com os bancos torna toda a organização financeira mais dinâmica e faz com que o usuário não precise necessariamente adicionar manualmente todas as transações por ele realizadas. A adição de categorias também permite uma personalização maior do seu orçamento, mas acaba entrando no mesmo problema explicado anteriormente.

Para que um aplicativo seja capaz de auxiliar usuários de um nicho específico que não tenham boa educação financeira, portanto, é necessário que existam pré-definidas as categorias necessárias para organização financeira desse nicho. Dessa forma, no momento do cadastro

de uma nova transação basta que o usuário encontre dentre as categorias aquela que mais se aproxima da transação que está sendo cadastrada.

A impossibilidade de adicionar novas transações, apesar de limitante, ajuda usuários que ainda não são familiarizados com a forma de categorização das transações, pois adicionar novas categorias pode ocasionar em transações com difícil categorização. Um exemplo disso é a categoria de Supermercado ou Compras. Essa categoria é muito útil ao se tratar de núcleos familiares, porém em um núcleo individual é melhor separá-la em categorias específicas de Higiene e Alimentação ou até mesmo Vestimenta.

Parte III

Desenvolvimento

4 Desenvolvimento

Este capítulo trata do período de desenvolvimento do trabalho, principais decisões e problemas encontrados. A Seção 4.1 mostra o aplicativo desenvolvido para teste de engajamento dos usuários, a Seção 4.2 mostra os requisitos levantados para o projeto, a Seção 4.3 traz as tecnologias utilizadas, a Seção 4.4 trata da modelagem dos bancos de dados e do aplicativo, a Seção 4.5 mostra a implementação dos recursos do app e a Seção 4.6 descreve os testes realizados. Além disso, a Seção 4.7 traz considerações a respeito de todo o desenvolvimento do projeto.

4.1 Aplicativo de Teste

Antes do desenvolvimento deste trabalho foi necessário responder as seguintes perguntas: *"qual seria o engajamento dos usuários e se este engajamento seria capaz de auxiliar os estudantes como se propõe?"*. Para responder essas perguntas, foi criado um aplicativo de teste nativo para ambiente Android utilizando o Android Studio com as linguagens Java e XML, escolha feita por familiaridade do autor com as ferramentas.

Para simplificar o desenvolvimento dessa aplicação, foi utilizado localmente um banco de dados Realm[1] objeto-relacional e, em nuvem, foram utilizadas as seguintes ferramentas do Google Firebase[2]: Authentication para autenticação do usuário, Realtime Database para salvar os dados coletados em nuvem, Crashlytics para detectar *bugs* e Test Lab para testar o fluxo de autenticação, o que era necessário para identificar o usuário dentro do banco de dados remoto.

A análise de requisitos da aplicação levantou os seguintes pontos:

a) A aplicação deve ser capaz de permitir a adição de um novo usuário;

b) A aplicação deve ser capaz de persistir os dados do usuário em nuvem a partir de seu login;

c) A aplicação deve ser capaz de permitir a adição, remoção e edição de gastos e rendas com base em categorias;

d) A aplicação deve ser capaz de gerenciar subcategorias;

e) A aplicação deve ser capaz de exibir os gastos e rendas totais do usuário;

f) A aplicação deve requerer na primeira execução a universidade de estudo;

g) A aplicação deve ser capaz de persistir os dados na memória interna do dispositivo até que seja estabelecida uma conexão com a Internet e esses dados possam ser enviados ao Firebase;

[1] <https://realm.io>
[2] <https://firebase.google.com/?hl=pt-br>

h) A aplicação deve suportar o agendamento de gastos e rendas.

A Figura 4 mostra os casos de uso do aplicativo de teste.

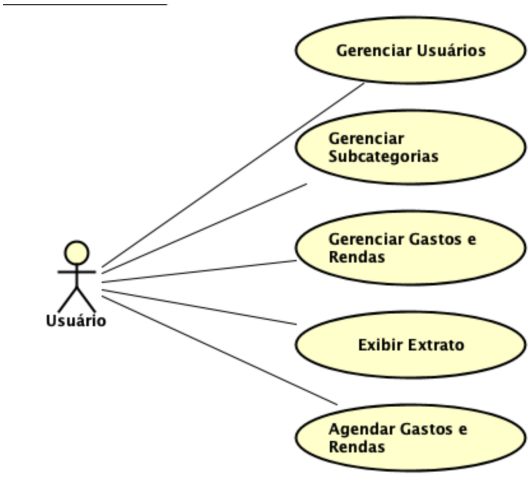


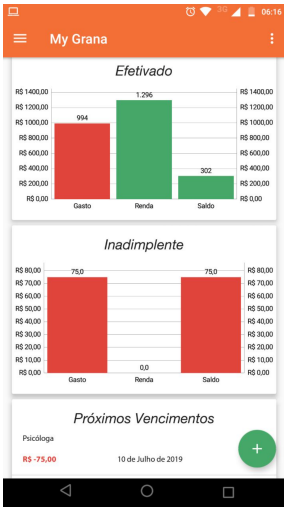
Figura 4 – Casos de uso do aplicativo de teste.

Fonte: O autor.

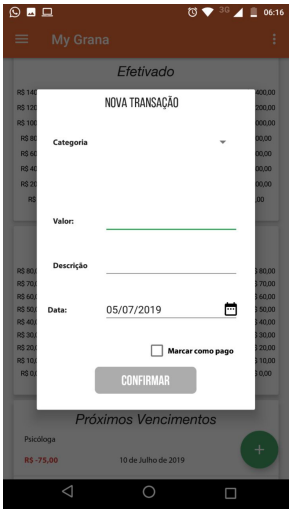
Na Figura 5 são apresentadas as telas de interface gráfica presentes no aplicativo de teste.

(a) Tela de autenticação.

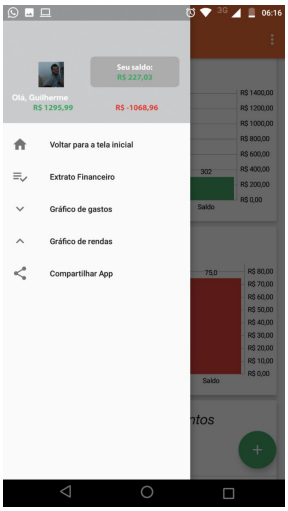
(b) Escolha da universidade.



(c) Tela inicial.



(d) Adicionar Transação.



(e) Sidebar.

(f) Tela de extrato.

Figura 5 – Telas do aplicativo de teste.

Fonte: O autor.

O processo de autenticação mostrado na Figura 5a obrigava o usuário a entrar com uma conta do Google, o que deixava fácil a identificação desse usuário dentro do banco de dados remoto. Uma vez que era preciso rastrear o engajamento, o usuário não poderia ser mantido anônimo. Quando autenticado, o usuário selecionava sua universidade na tela apresentada na Figura 5b e era redirecionado para a tela inicial do aplicativo mostrada na Figura 5c.

Na tela inicial eram apresentados dois gráficos, um com transações efetivadas e outro com transações não-efetivadas, bem como uma lista dos próximos vencimentos, além de uma *sidebar* mostrando as *features* do app, como na Figura 5e. O cadastro de uma transação também era simples, vide Figura 5d, o banco de dados remoto armazenava valor, data e categoria, mas dentro do aplicativo também havia descrição e subcategoria.

Também era possível obter uma listagem dos gastos e de rendas cadastradas conforme

mostra a Figura 5f. Não havia a opção de criar orçamentos e nem mesmo separação entre meses, pois o teste foi feito para durar pouco período de tempo.

Antes de disponibilizar o aplicativo, o fluxo principal de login e cadastro foi testado exaustivamente com a ferramenta Test Lab e, uma vez disponibilizado, o Crashlytics foi usado pra detectar erros e aplicar correções necessárias.

O aplicativo foi disponibilizado na Google Play Store e divulgado via email no ICEA. Houve engajamento dos usuários com a aplicação e ela foi bem aceita entre os usuários ao ponto em que foi feita uma matéria da Assessoria de Comunicação Institucional (ACI) da UFOP[3] descrevendo o aplicativo de gerenciamento financeiro voltado para estudantes universitários.

Ao final de pouco mais de um mês, foram cadastrados um total de 102 (cento e dois) estudantes e coletados 76KB de dados exclusivamente de transações. Foi observado um bom engajamento por maior parte dos usuários, mesmo entre os que passaram a utilizá-lo algum tempo depois do início dos testes, o que indica a sua aceitação por parte dos mesmos.

4.2 Requisitos

O aplicativo multiplataforma apresenta os mesmos requisitos do aplicativo nativo de testes, conforme descrito na seção anterior, exceto pelo fato de que:

a) não é necessário o uso de subcategorias;

b) na primeira execução de um novo usuário, devem ser requeridas:
   - cidade e estado de origem do usuário;
   - cidade e estado de estudo do usuário;
   - universidade de estudo do usuário;

c) no banco de dados remoto o usuário deve ser anônimo.

A remoção das subcategorias ocorreu pelo fato de ter sido um recurso pouco utilizado no aplicativo de testes. A inclusão de cidade e estado de origem, bem como moradia vem da necessidade de manter um maior número de dados sobre o usuário para possibilitar análises futuras. Manter os usuários anônimos aumenta a segurança dos dados, uma vez que são dados mais sensíveis no sentido de serem informações pessoais que não devem ser divulgadas publicamente, sendo assim, devem ser protegidos de acesso público. Para tal, o usuário não pode ser identificável de acordo com o Art. 5o da Lei Geral de Proteção de Dados (LGPD) n^o 13.709, de 14 de agosto de 2018 (Brasil, 2018).

[3] <https://ufop.br/noticias/pesquisa-e-inovacao/aplicativo-que-auxilia-universitarios-na-gestao-de-financas-pessoais-e>

4.3 Tecnologias e Recursos Computacionais Utilizados

A escolha de qual *framework* usar se baseou na familiaridade com as linguagens que ele utiliza. Foi necessário fazer pequenos aplicativos para testar a possibilidade de implementar todos os bancos de dados locais e conexão com o Firebase.

O primeiro *framework* a ser testado foi o Ionic[4], por familiaridade do autor com linguagens Web. O uso do mesmo foi abandonado após serem verificadas diferenças no comportamento do aplicativo em cada plataforma no momento da conexão com o Firebase e, ainda, consumo excessivo de memória RAM do aparelho ou dispositivo móvel mesmo em aplicativos simples.

O segundo teste foi feito com o React Native, baseado em JavaScript e com vasta documentação. Esse *framework* foi descartado após poucos dias de teste por dificuldade do autor com recursos robustos da linguagem.

Por fim, foi utilizado o Flutter. Apesar do autor não ter conhecimentos na linguagem Dart à época, tanto a linguagem quanto o *framework* são totalmente orientados a objetos, paradigma estudado exaustivamente durante toda a graduação. Por esse motivo, aprender Dart e Flutter foi um processo muito rápido, foi necessário aprender a sintaxe da linguagem, mas toda a teoria de orientação a objetos já era bem conhecida pelo autor. Uma breve explicação sobre o seu funcionamento pode ser encontrado no Apêndice A.

Foram utilizadas as seguintes tecnologias neste trabalho:

a) **Balsamiq Wireframes** é software utilizado para prototipagem das telas (**??**) no qual é possível acessar os recursos de:
   - prototipagem com recursos simples;
   - resultado visual de fácil compreensão;

b) **Astah**: software utilizado para diagramação UML (**??**);

c) **Firebase**: ferramenta utilizada para prover *cloud features* para o desenvolvimento do aplicativo e será utilizado o Realtime Database, que é um banco de dados NOSQL atualizado em tempo real (**??**).

d) **Flutter**: framework utilizado para desenvolver o aplicativo (**??**);

e) **Android Studio**: software utilizado para tratar de questões específicas de ambiente android no aplicativo (**??**);

f) **Xcode**: software utilizado para tratar de questões específicas de ambiente iOS no aplicativo (**??**);

g) **Visual Studio Code**: utilizado como editor de texto em todo o trabalho (**??**);

h) **SQLite**: linguagem utilizada para implementação dos bancos de dados locais (**??**).

[4] <https://ionicframework.com>

O aplicativo foi desenvolvido em um ambiente macOS através de um Macbook Pro 2015, com 8 GB de memória RAM e 128 GB de memória secundária SSD. Foram usados Flutter 1.25.0-8.1.pre e Dart 2.12.0.

4.4 Modelagem

A modelagem do aplicativo se deu com base no aplicativo de teste, porém os novos requisitos e a necessidade de criar uma interface gráfica mais intuitiva demandaram uma organização maior dos recursos disponibilizados. Os novos dados a serem coletados pela aplicação afetaram diretamente a modelagem dos bancos de dados e a garantia de anonimidade do usuário também trouxe alguns desafios a serem resolvidos e superados. Essa seção trata dos principais desafios e como o aplicativo foi organizado e projetado antes de sua implementação.

4.4.1 Tratamento dos Dados Coletados

Neste trabalho, o objetivo dos bancos de dados é armazenar dados da melhor forma possível e três definições são fundamentais para que isso seja realizado conforme descrito a seguir:

a) *Profile* que é o objeto que guarda todas as informações pessoais de um usuário, tais como:
   - Identificação;
   - Estado e cidade de origem;
   - Estado e cidade de estudo;
   - Sistema operacional utilizado;

b) *Transaction* que é o objeto que guarda as informações referentes a uma transação, tais como:
   - Categoria;
   - Valor;
   - Em alguns casos, a descrição;

c) *Budget*, que representa o Orçamento, é responsável por relacionar as categorias com seus respectivos valores planejados.

Para o *Profile*, a identificação do usuário, estados e cidades de origem e estudo e o sistema operacional são capturados no momento do cadastro.

Para a *Transaction*, a categoria e o valor são fundamentais na coleta dos dados. A descrição, no entanto, apesar de inútil na coleta, é importante para a usabilidade do aplicativo, sendo, assim, uma informação utilizada apenas pelo usuário e guardada apenas localmente.

O *Orçamento*, no entanto, é obsoleto na coleta dos dados, uma vez que as transações podendo ser facilmente acessadas e utilizadas para calcular o total de rendas e gastos de um mês. Dessa forma, trata-se unicamente de uma *feature* disponibilizada para o aplicativo, para facilitar a organização financeira do estudante.

4.4.1.1 Primeiro Banco de Dados Local

Esse banco de dados é criado e utilizado apenas no momento de cadastro a fim de coletar os dados referentes ao *Profile* do usuário. Ele possui uma tabela que representa os estados brasileiros, uma que relaciona cada estado a suas respectivas cidades e uma terceira, que relaciona cada estado às universidades presentes nele.

Na Figura 6 está mostrado o modelo Entidade-Relacionamento do primeiro banco de dados local.

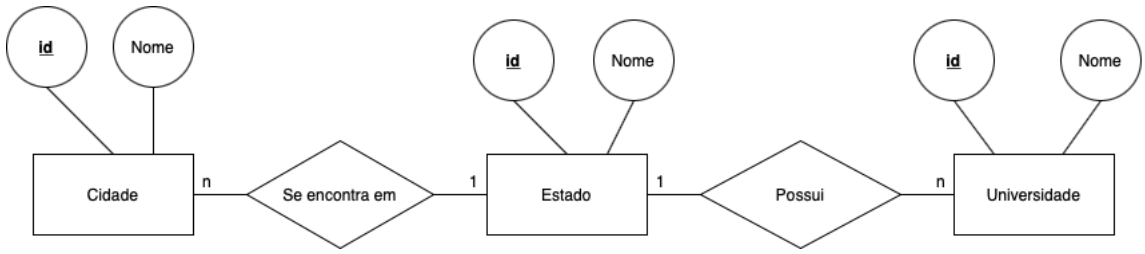


Figura 6 – Modelo Entidade-Relacionamento do Primeiro Banco de Dados local.

Fonte: O autor.

A tabela Tabela 3 apresenta o dicionário de dados da tabela Estado do banco de dados local da aplicação. A tabela contém as colunas Nome, Tipo, Tamanho e Descrição de cada dado.

Tabela 3 – Dicionário de dados da tabela Estado.

Nome	**Tipo**	**Tamanho (em Bytes)**	**Descrição**
id	int	11	Id do estado.
nome	varchar	75	Nome do estado.
uf	varchar	5	Sigla do estado.

Fonte: O autor.

Na tabela Tabela 4 é apresentado o dicionário de dados da tabela Cidade do banco de dados local do aplicativo.

Tabela 4 – Dicionário de dados da tabela Cidade.

Nome	Tipo	Tamanho (em Bytes)	Descrição
id	int	11	Id da cidade.
nome	varchar	120	Nome da cidade.
estado	int	5	Id do estado.

Fonte: O autor.

Por sua vez, a tabela Tabela 5 mostra o dicionário de dados da tabela Universidade do banco de dados local do aplicativo.

Tabela 5 – Dicionário de dados da tabela Universidade.

Nome	Tipo	Tamanho (em Bytes)	Descrição
id	int	11	Id da universidade.
nome	varchar	120	Nome da cidade.
estado	int	5	Id do estado.

Fonte: O autor.

4.4.1.2 Segundo Banco de Dados Local

Esse banco de dados é gerado ao fim do cadastro e tem como finalidade salvar os dados referentes a aplicação. Ele possui uma tabela referente ao *Profile* do usuário, uma para armazenar os dados das transações e mais uma para armazenar o orçamento planejado pelo usuário. O modelo entidade-relacionamento pode ser observado na Figura 7.

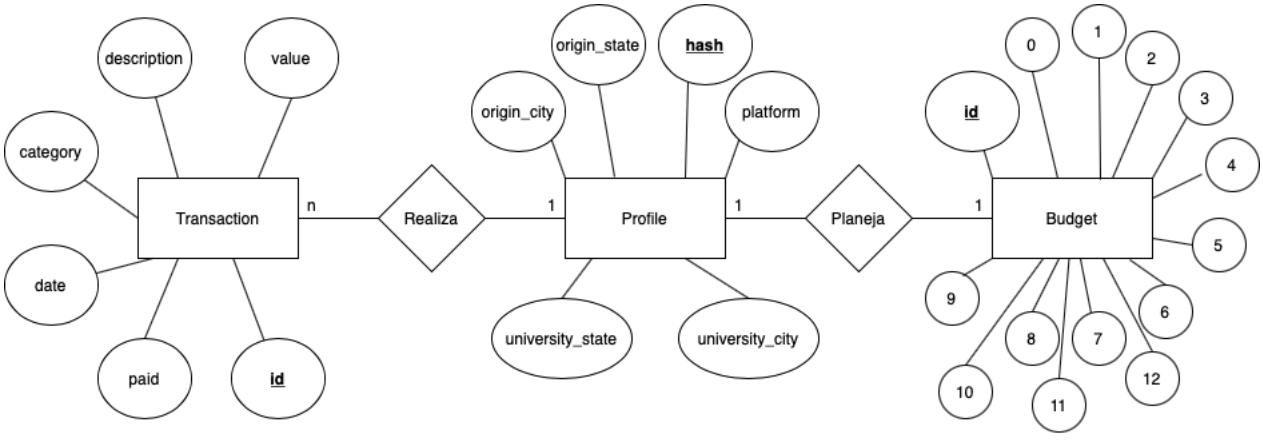


Figura 7 – Modelo Entidade-Relacionamento do segundo banco de dados local.

Fonte: O autor.

Cada aplicação possui apenas um usuário e, por isso, não é necessário o *constraint* entre *Transaction* e *Budget* com a tabela *Profile*. Pode-se observar ainda que a entidade *Budget*

tem como informações apenas números, os quais representam o *id* de cada categoria, que não possuem uma tabela no banco de dados pois podem ser tratadas como constantes dentro do aplicativo.

A Tabela 6 mostra o dicionário de dados da tabela *Profile* do banco de dados local do aplicativo. Na tabela pode-se observar quais dados do usuário são armazenados. Como identificador do usuário é guardada uma chave hash[5] ao invés de seu nome e isso é feito para que o usuário se mantenha anônimo.

Tabela 6 – Dicionário de dados da tabela Profile.

Nome	**Tipo**	**Tamanho (em Bytes)**	**Descrição**
origin_city	int	11	Id da cidade de origem do usuário.
origin_state	int	11	Id do estado de origem do usuário.
university_state	int	11	Id do estado de estudo.
university_city	int	11	Id da cidade de estudo.
hash	varchar	40	Hash que representa o estudante.
platform	varchar	10	Plataforma utilizada pelo usuário.

Fonte: O autor.

A Tabela 7 mostra o dicionário de dados da tabela *Transaction* do banco de dados local do aplicativo. Cada transação envia ao banco de dados remoto somente *id*, *category*, *date* e *value*.

Tabela 7 – Dicionário de dados da tabela Transaction.

Nome	**Tipo**	**Tamanho (em Bytes)**	**Descrição**
category	int	10	Id da categoria da transação.
date	varchar	11	Data da transação.
description	varchar	21	Descrição da transação.
id	integer	8	Id da categoria.
paid	int	10	Identifica se houve pagamento da transação.
value	int	10	Valor da transação.

Fonte: O autor.

Na Tabela 8 é apresentado o dicionário de dados da tabela *Budget* do banco de dados local do aplicativo. Como dito anteriormente, essa tabela não alimenta o banco de dados remoto com nenhum dado, ela é utilizada somente para *features* do aplicativo.

[5] Valor hexadecimal retornado por uma função Hash. Neste trabalho essa função tem por objetivo criptografar o nome do usuário em um valor hexadecimal de tamanho fixo de modo que este usuário não possa ser identificado.

Tabela 8 – Dicionário de dados da tabela Budget.

Nome	**Tipo**	**Tamanho (em Bytes)**	**Descrição**
id	int	10	Id do orçamento
0	int	10	Valor reservado pra gastos com Alimentação.
1	int	10	Valor reservado pra ganhos com Bolsa Auxílio.
2	int	10	Valor reservado pra gastos com Higiene.
3	int	10	Valor reservado pra gastos com Investimentos.
4	int	10	Valor reservado pra gastos com Lazer.
5	int	10	Valor reservado pra gastos com Moradia.
6	int	10	Valor reservado pra ganhos com Pensão.
7	int	10	Valor reservado pra ganhos com Salário.
8	int	10	Valor reservado pra gastos com Saúde.
9	int	10	Valor reservado pra gastos com Transporte.
10	int	10	Valor reservado pra gastos com Universidade.
11	int	10	Valor reservado pra gastos com Vestimenta.
12	int	10	Valor reservado pra gastos com Outros.

Fonte: O autor.

4.4.1.3 Banco de Dados Remoto

O banco de dados remoto é importante para uso dos dados em análises futuras. O primeiro problema foi a questão da *hash* do usuário. Intuitivamente pode-se pensar que ela seria a raiz da árvore, já que são os dados desse usuário que estão sendo coletados. Isso, porém, tem um problema: a colisão de *hashes* dentro do banco de dados. Esse problema é mitigado separando as *hashes* por universidade na árvore, fazendo com que dois usuários de mesma *hash*, em universidades diferentes, não causem colisão.

Cada universidade, no entanto, pode ter campus espalhados em várias cidades, dessa forma, para facilitar as consultas, cada universidade poderia estar diretamente ligada a uma cidade específica, fazendo com que as consultas por cidade sejam mais simples. De forma análoga, para simplificar a consulta por estados, é válido ligar cada cidade a um estado.

Antes do *hash*, então, tem-se Estado, Cidade e Universidade, o que diminui consideravelmente a chance de colisões. A partir do *hash* tem-se o *Profile* e as *Transactions*. Parte do *Profile* já é utilizado no caminho até o *hash*, dessa forma ele só tem atributos de estado e cidade de origem e da plataforma que ele está utilizando. As *Transactions* são organizadas com a separação de meses. Isso tem dois motivos: facilitar a consulta por um mês para cálculo de orçamento e funcionar como *flag* identificando uma possível colisão.

Para utilizar as transações como *flags*, é necessário levantar um novo requisito ao banco de dados local do aplicativo: o *id* de transação não pode em hipótese alguma diminuir. Isso significa que ao excluir uma transação, o valor de *id* dessa transação não pode ser reutilizado. Dessa forma se no momento de processamento dos dados for encontrado algum *id* duplicado significa que houve colisão de *hashes* e esses dados são inválidos.

Diante disso, seguindo essa lógica, a árvore de coleta de dados do Banco de Dados em

nuvem fica conforme descrito na Figura 8.

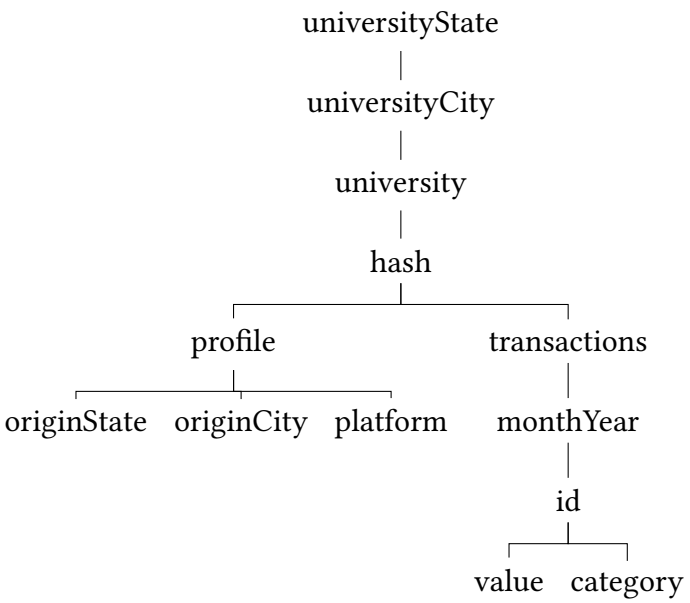


Figura 8 – Árvore de coleta do banco de dados remoto.

Fonte: O autor.

4.4.2 Casos de Uso

O primeiro caso de uso da aplicação é o cadastro do usuário. Esse será armazenado no banco de dados em nuvem, então é de grande importância. Não faz sentido, no entanto, ter um usuário no banco de dados em nuvem que não possua nenhuma transação associada, sendo assim, esse usuário só é levado pra nuvem após o cadastro de uma transação. Dessa forma, o passo de cadastro, atualização e exclusão (delete) de transações também devem ser casos de uso do aplicativo, já que esse é o elemento básico da coleta dos dados.

A listagem das transações na aplicação também representa um caso de uso, seja essa listagem com ou sem filtros. A criação, o *display* e a edição de orçamentos também é necessária para uma boa experiência do usuário dentro do aplicativo.

Na Figura 9 é apresentado o diagrama dos casos de uso do aplicativo.

O detalhamento de cada caso de uso da Figura 9 se dá conforme descrito a seguir:

- **Cadastro**: no primeiro uso do aplicativo obrigatoriamente é feito o cadastro do usuário. São requeridos nome, depois estado e cidade de origem, em seguida estado e cidade de estudo e a universidade de estudo. Depois o usuário é redirecionado para a tela inicial;
- **Adicão de transação**: para adicionar receita ou despesa, basta que toque o botão flutuante “+” na tela inicial, preencha as informações da transação a ser adicionada e selecione “Confirmar”;

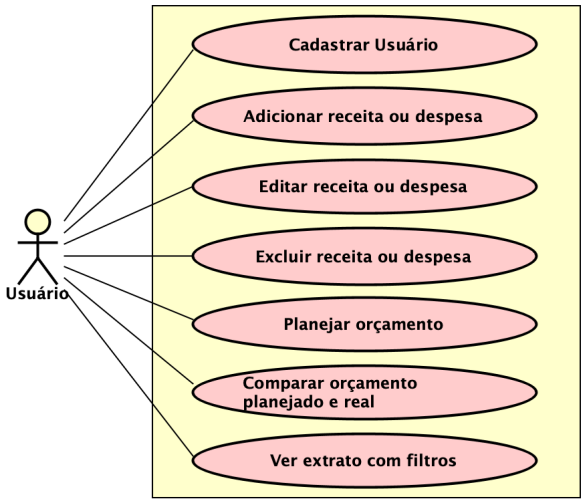


Figura 9 – Diagrama de Casos de Uso

Fonte: O autor

- **Edição de transação**: para editar uma receita ou despesa, basta tocar no botão de Extratos, selecionar os filtros para a transação a ser editada e, em seguida tocar a transação desejada, preencher as informações atualizadas e selecionar "Confirmar";
- **Exclusão de transação**: para excluir uma receita ou despesa, basta repetir os passos anteriores, porém ao invés de tocar a transação, selecioná-la, aparecerá uma caixa de diálogo pedindo pra confirmar a exclusão e clicar em "Confirmar";
- **Planejamento**: para planejar um orçamento, deve-se escolher a opção "Planejamento" na tela inicial e, em seguida, clicar em "Editar orçamento", preencher as informações desejadas e tocar em "Salvar";
- **Comparação**: para comparar os orçamentos real e planejado, deve-se escolher a opção "Orçamento" na tela inicial. Pode-se ainda ver essa comparação por categorias, para tal, basta repetir o passo anterior e clicar no botão "Ver por categoria";
- **Exibição de extratos com filtros**: para ver extrato completo com filtros de categoria, mês e ano basta escolher a opção 'Extratos' na tela inicial, escolher os filtros desejados e clicar no botão "Continuar";
- **Exibição do extrato do mês atual**: para ver extrato do mês atual com filtro de categorias deve-se escolher a opção "Gastos e Rendas" na tela inicial e em seguida clicar na categoria desejada.

4.4.3 Diagrama de Atividades

Para localizar todas as dependências do processo de editar uma transação, foi feito o diagrama de atividades mostrado na Figura 10, os diagramas referentes às outras atividades podem ser encontrados no Apêndice B.

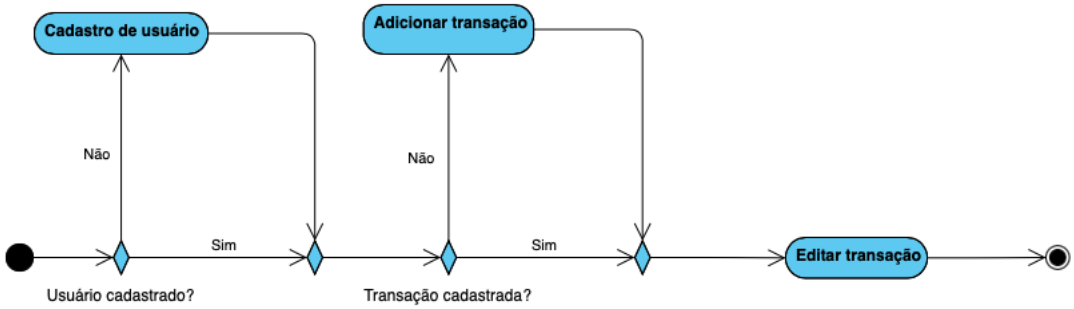


Figura 10 – Diagrama de Atividade Referente a Edição de uma Transação.

Fonte: O autor.

O diagrama mostra que para editar uma transação o usuário e a transação a ser editada já devem estar cadastrados, deixando claro que esses cadastros devem ser feitos antes do processo de edição.

4.4.4 Diagrama de sequência

Para identificar o passo a passo a ser seguido pelo aplicativo durante a edição de uma transação foi feito o diagrama de sequência mostrado na Figura 11. Outros diagramas de sequência relevantes podem ser encontrados no Apêndice B.

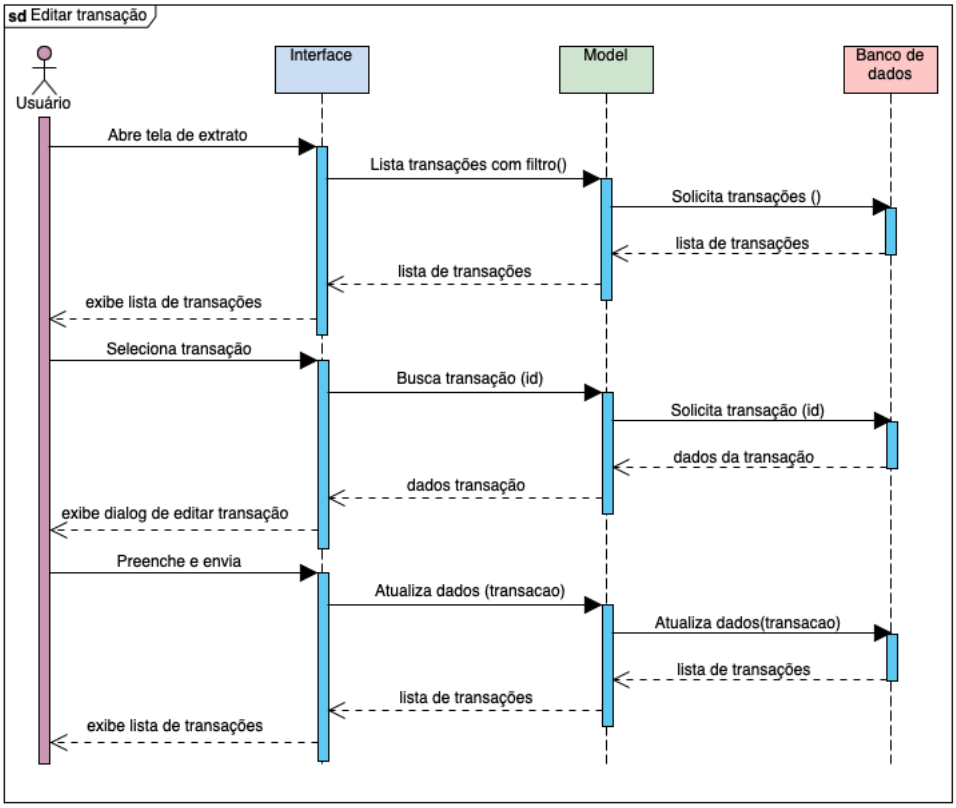


Figura 11 – Diagrama de Sequência.

Fonte: O autor.

O diagrama mostra que deve haver a impressão das transações com base no filtro escolhido pelo usuário, o usuário deve selecionar uma dessas transações e preencher os dados novos. A impressão e a exibição do *dialog* se dá após consultas ao banco de dados pela lista de usuários e pela transação selecionada respectivamente. Após o preenchimento dos dados novos, a transação escolhida é atualizada no banco de dados.

4.4.5 Protótipos de Telas

Na Figura 12 é apresentado o fluxo dos protótipos de telas referentes ao Cadastro. Esse deve ser o fluxo de telas durante a primeira execução do app. Na primeira tela o usuário deve escrever seu nome e selecionar o botão “Continuar” para ser redirecionado para a tela de escolha de estado e cidade de origem. Na segunda tela ele deve escolher um estado e o programa deve exibir apenas as cidades daquele estado e então permitir a passagem para a próxima tela. A terceira tela funciona de forma idêntica à segunda. Na quarta tela, o aplicativo deve exibir todas as universidades presentes no estado selecionado na terceira, uma vez selecionada uma universidade o usuário deve ser redirecionado à tela principal do aplicativo.

A tela principal do aplicativo deve conter o nome do app, mostrar quanto dinheiro o usuário ainda possui, o total de lançamentos ainda não pagos e os botões referentes aos

recursos do aplicativo: gastos e rendas, planejamento, extratos, orçamento e um *floating button* para adicionar novas transações.

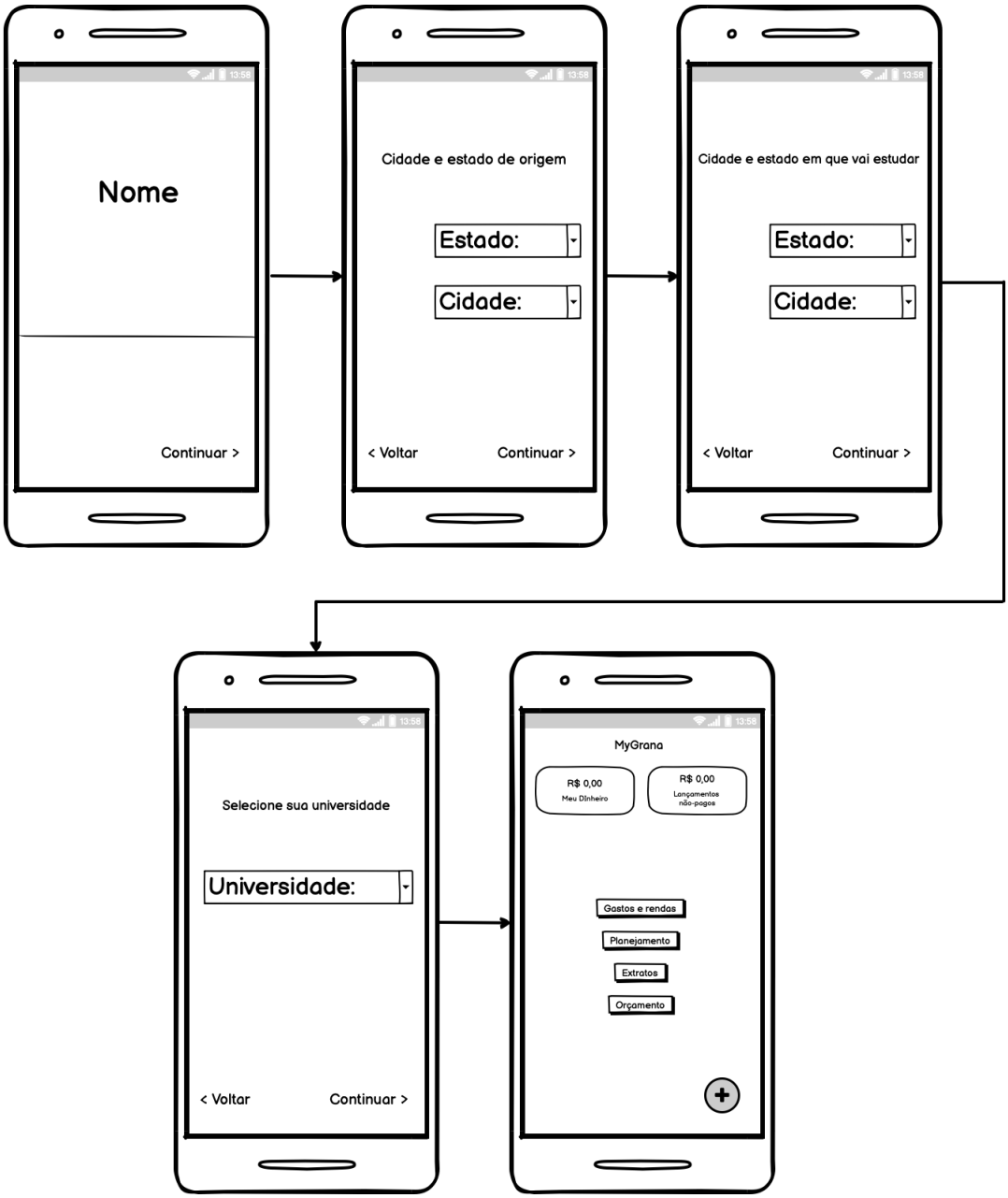


Figura 12 – Protótipo de telas para o Cadastro.

Fonte: O autor.

A Figura 13 mostra o fluxo dos protótipos de telas para a Adição de transação. Esse fluxo trata de adicionar uma nova transação. Na tela inicial do aplicativo, o usuário seleciona o *floating button* com o símbolo de "+". Um *dialog* deve se apresentar para que usuário cadastre os dados da transação: categoria, data, valor e se a transação já foi paga. Ao clicar em "Confirmar" a transação deve ser persistida na memória interna do aplicativo e enviada ao Firebase. Caso seja selecionada a opção "Cancelar" nada deve acontecer. Ao cadastrar uma nova transação o

usuário deve ser redirecionado novamente à tela inicial onde os valores de dinheiro do usuário e de lançamentos ainda não pagos devem ser automaticamente atualizados.

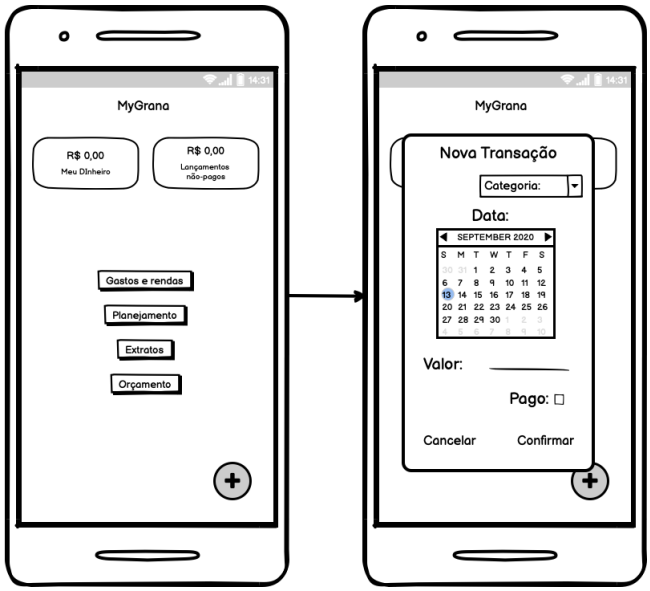


Figura 13 – Protótipo de telas para a Adição de transação.

Fonte: O autor.

Na Figura 14 observa-se o fluxo dos protótipos de telas para a Edição de transação. Esse caso de uso é referente a edição de uma transação já cadastrada. Na tela inicial o usuário deve selecionar a opção "Extratos" que o redirecionará a uma tela de filtros. Nessa tela ele deve poder escolher filtrar suas transações por categoria e/ou mês e/ou ano e em seguida clicar no botão "Continuar". Na tela seguinte, devem ser mostradas as transações que atendem ao filtro selecionado anteriormente. Ao tocar em uma transação em específico, deve aparecer um *dialog* idêntico ao de adicionar transação, porém totalmente preenchido com os dados da transação selecionada. Ao clicar em "Confirmar", o app deve persistir o valor editado, editar esse valor no banco de dados remoto e o usuário deve ser redirecionado à tela anterior, que deve estar atualizada com os valores novos que o ele preencheu no *dialog*. Caso seja selecionada a opção "Cancelar", o *dialog* fecha e nada deve acontecer.

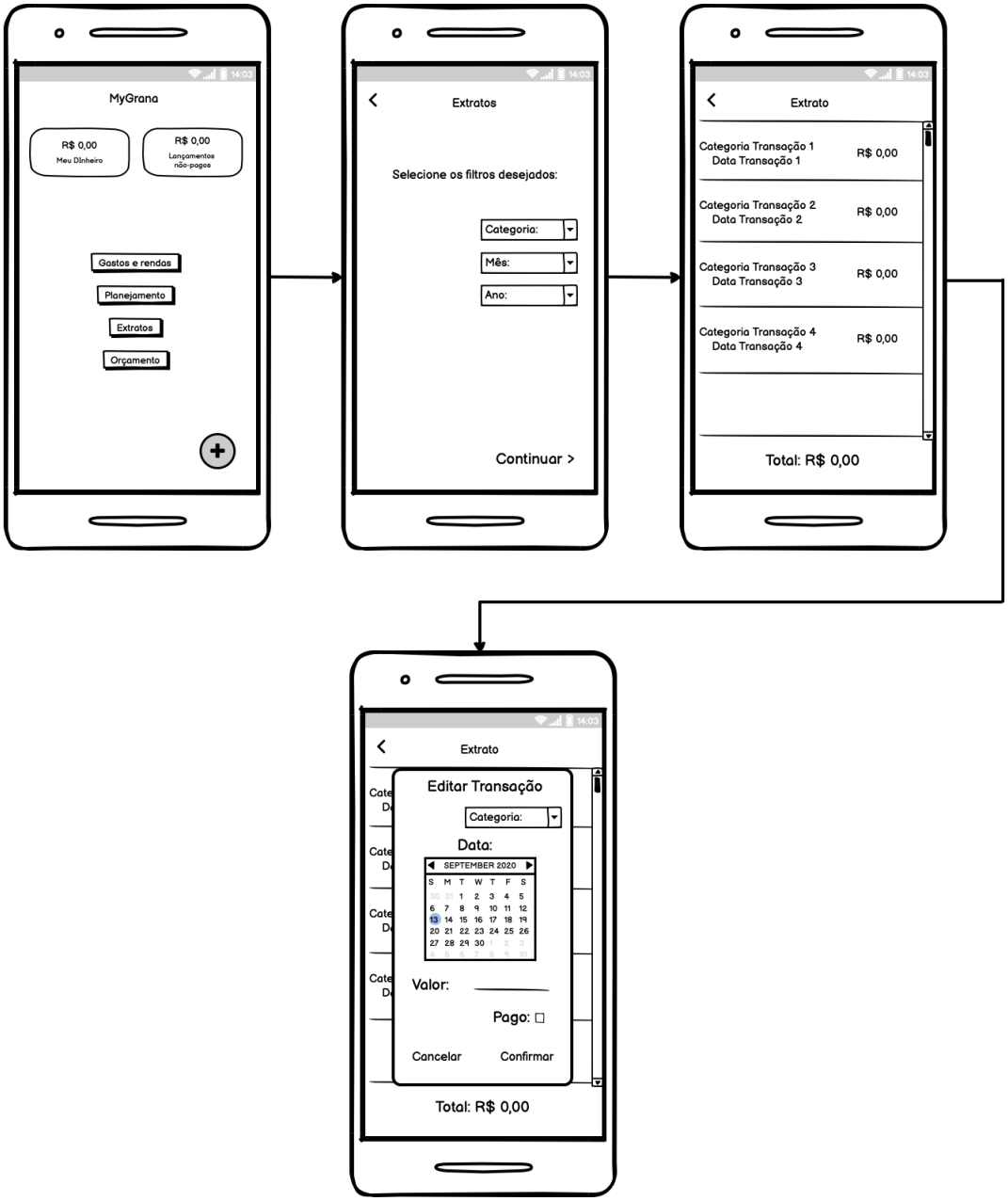


Figura 14 – Protótipo de telas para a Edição de transação.

Fonte: O autor.

Na Figura 15 é apresentado o fluxo dos protótipos de telas para a Exclusão de transação. Esse caso de uso diz respeito a excluir uma transação. O fluxo das telas é idêntico ao anterior até a tela de listagem de transações. Nessa tela caso o usuário selecione e mantenha selecionada uma transação por um tempo deve ser exibido um *dialog* perguntando se o usuário deseja deletar a transação. Caso o usuário toque na opção de "Confirmar", a transação deve ser apagada do banco de dados local. Caso a opção escolhida seja "Cancelar" ou se o usuário clicar fora do *dialog* nada deve ocorrer. Nos três casos, ao final, o *dialog* deve fechar e retornar a tela de listagem com as transações atualizada.

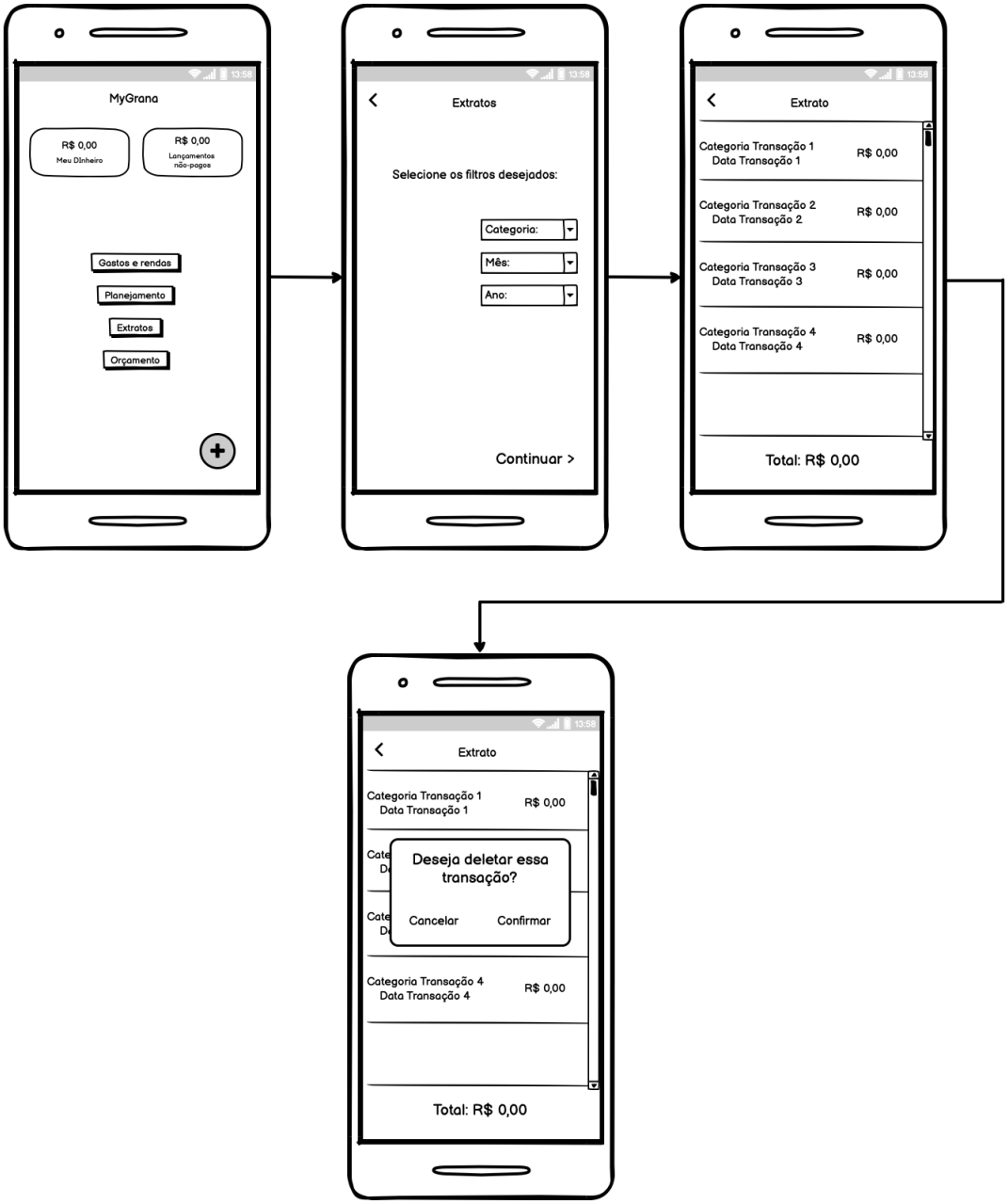


Figura 15 – Protótipo de telas para a Exclusão de transação.

Fonte: O autor.

A Figura 16 apresenta o fluxo de protótipos de telas para o Planejamento. Esse caso de uso se refere a fazer o planejamento do orçamento do mês atual. Para isso, o usuário deve escolher a opção “Planejamento” na tela inicial. Ao fazer a seleção ele é redirecionado para a tela de orçamento planejado, onde é mostrado todo o planejamento já feito e os valores totais de gastos e rendas presentes nele. Quando o usuário clicar no botão “Editar Orçamento”, ele deve ser redirecionado para a tela de planejamento. Nessa tela ele deve poder editar o valor de cada categoria e devem ser mostrados o valor total de rendas e gastos, atualizados automaticamente sempre que há alteração no valor de alguma categoria. Ao selecionar o botão “Salvar”, o usuário

deve ser redirecionado pra tela anterior, que deve estar atualizada, e o valor do orçamento do usuário deve ser persistido no banco de dados local.

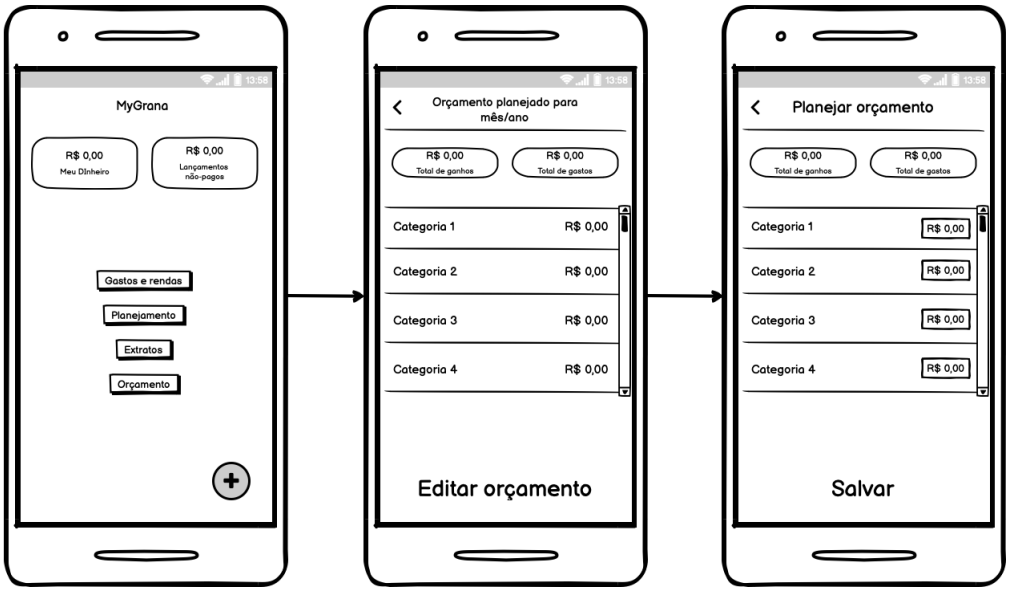


Figura 16 – Protótipo de telas para o Planejamento.

Fonte: O autor.

Na Figura 17 é apresentado o fluxo dos protótipos de telas para a Comparação. Esse caso de uso diz respeito à comparação do orçamento planejado com os gastos e rendas reais. Para tal, o usuário deve selecionar a opção “Orçamento” na tela inicial. Ao fazer isso, ele deve ser redirecionado à tela de comparação simplificada. Nessa tela são mostrados os valores de gastos e rendas planejados e reais. O usuário pode selecionar a opção “Ver por categoria” para que seja redirecionado à tela de comparação por categoria. Para cada categoria, essa tela deve mostrar o valor real, o valor planejado e um pequeno gráfico mostrando o quanto do planejado já foi utilizado.

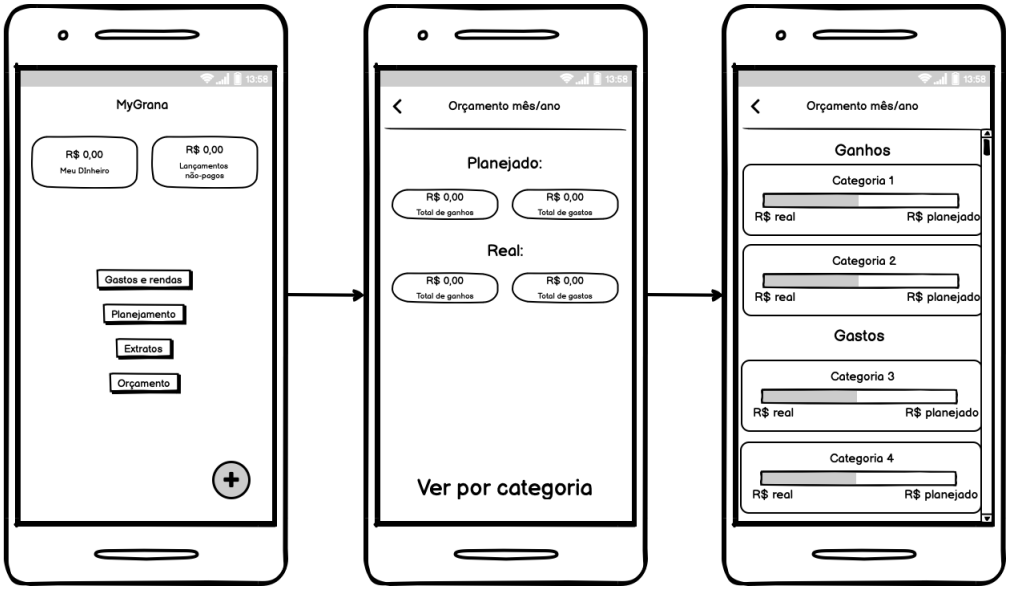


Figura 17 – Protótipo de telas para a Comparação.

Fonte: O autor.

Na Figura 18 pode-se observar o fluxo dos protótipos de telas para a Exibição de extratos com filtros. Esse caso de uso diz respeito à exibição de todas as transações com filtros. Para isso o usuário deve escolher a opção "Extratos" na tela inicial, que deve redirecioná-lo para a tela de aplicação dos filtros onde ele pode escolher filtros de categoria, mês e ano. Ao selecionar a opção "Continuar" ele deve ser redirecionado para a tela de Extratos, onde devem ser exibidas as transações que satisfazem os filtros aplicados e o valor total dessas transações.

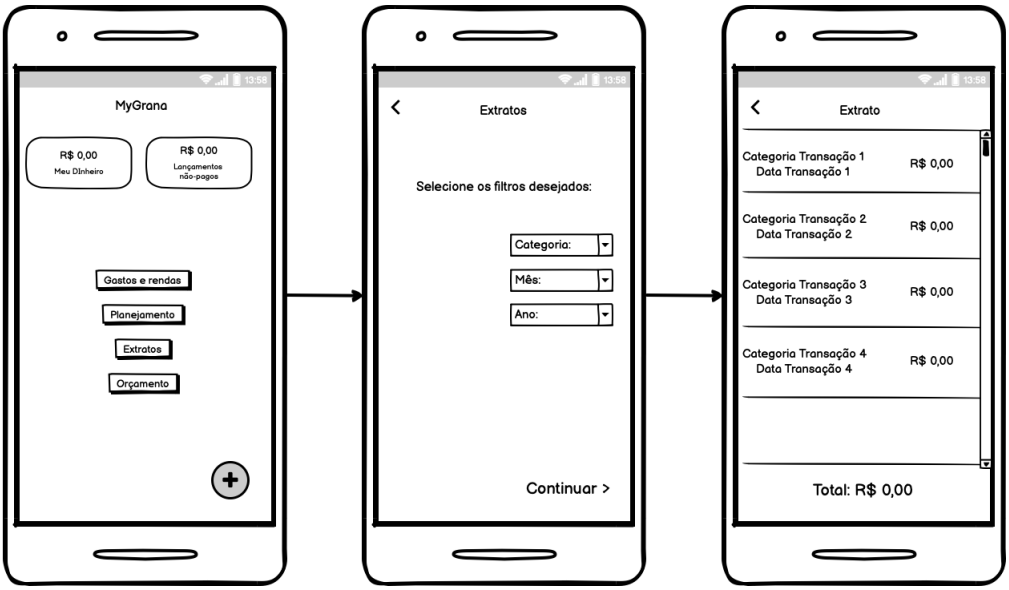


Figura 18 – Protótipo de telas para a Exibição de extratos com filtros.

Fonte: O autor.

A Figura 19 exibe o fluxo dos protótipos de telas para a Exibição do extrato do mês atual. Nesse caso de uso, o usuário deve ser capaz de ver as transações do mês atual referentes à categoria escolhida. Na tela inicial, o usuário seleciona a opção "Gastos e rendas", onde deve ser redirecionado para uma tela que mostra o total de rendas e gastos daquele mês até o momento e uma lista de todas as categorias com seus respectivos valores. Ao selecionar uma categoria deve ser exibida uma lista contendo as transações do mês atual pertencentes à categoria escolhida e o valor total dessas transações.

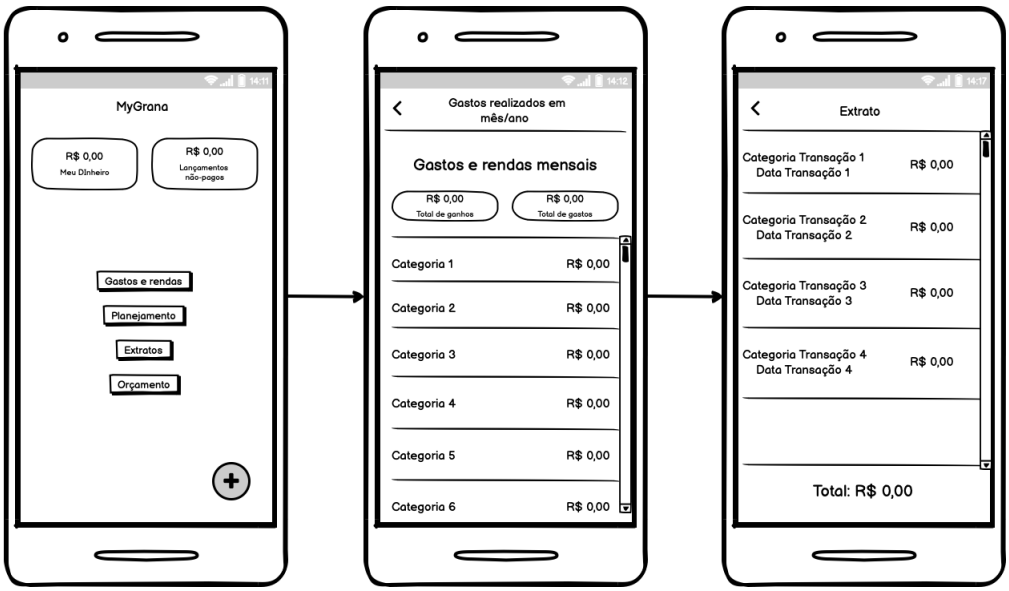


Figura 19 – Protótipo de telas para a Exibição do extrato do mês atual.

Fonte: O autor.

No projeto de interface deste projeto, todas as telas apresentadas anteriormente (com exceção do fluxo de cadastro e da tela inicial), há sempre um botão de voltar no canto superior esquerdo. Esse botão deve permitir ao usuário voltar para a tela anterior.

4.4.6 Arquitetura do Software

O uso do padrão de projeto ou da arquitetura *Model-View-Controller* (MVC) em todo o projeto foi considerado, porém a maior parte das telas de interface gráfica tem apenas botões que redirecionam pra outras telas, fazendo com o que os métodos dos *Controllers* se tornassem apenas uma chamada à *View*. Em outras telas, no entanto, como as de entradas de dados do usuário, os elementos de entrada de dados tem *Controllers* próprios do *framework*, que são utilizados pra conectar a *View* e a *Model* e definir ações a serem tomadas no período da entrada dos elementos.

No primeiro caso, portanto, a escolha de projeto foi suprimir o uso de *Controllers* e comunicar a *View* diretamente com a *Model*, enquanto no segundo a importância do *Controller*

para alterar a tela atual em tempo de execução fez com que ele precisasse ser utilizado.

4.5 Implementação

Essa seção trata da implementação do app, seus problemas e desafios. Era imprescindível que o aplicativo satisfizesse todos os requisitos e respeitasse as questões levantadas anteriormente na modelagem. Dessa forma, foi seguida uma sequência de passos que foram gerando aos poucos o resultado desejado.

4.5.1 Dados Necessários ao Cadastro

Para o momento do cadastro, era necessário que o aplicativo já tivesse alguns dados, são eles:

a) Lista de estados brasileiros;

b) Lista de cidades de cada estado;

c) Lista de universidades presentes em cada estado.

O uso das APIs do IBGE e do e-MEC para coleta dos dados de estados, cidades e universidades foi considerado, porém elas requeriam conexão com a Internet no momento do cadastro e por conta disso foram descartadas. Assim, lista de cidades e estados em formato SQL foi encontrada em <https://www.ricardoarrigoni.com/cidades-brasil-lista-de-cidades-brasileiras-em-sql/>. Não foi utilizada a tabela País, uma vez que o aplicativo seria disponibilizado apenas no Brasil. Além disso, a lista de universidades presentes por estado foi extraída com um *web-scrapping*, feito com um *script* em Python, do site <https://www.altillo.com/pt/universidades/universidades_brasil.asp>.

Com outro *script*, os dados dos estados, cidades e universidades foram convertidos para serem utilizados em um banco de dados SQLite. Nesse banco as universidades foram organizadas por estado, já que é comum o estudante morar em uma cidade e estudar em outra.

Para que o usuário não precise de conexão com a Internet no momento do cadastro, o *script* de criação e povoamento desse banco de dados foi salvo dentro do aplicativo. Na primeira execução esse *script* é executado, possibilitando o acesso do app às informações necessárias para o cadastro. As informações armazenadas na tabela *profile* são os *ids* de cada estado, cidade e universidade, a fim de economizar espaço na memória do dispositivo e no banco de dados remoto.

Ao final do cadastro, é executado um *script* de *drop* desse banco de dados e seus *scripts* de criação e povoamento recebem valores *null* para que não ocupem espaço na memória do dispositivo pois eles não são mais utilizados.

4.5.2 Anonimidade do Usuário

A anonimidade do usuário é garantida com a geração de uma *hash* ao final do cadastro. A *hash* foi gerada a partir de uma HMAC-SHA256[6] utilizando o nome de usuário como mensagem e um número aleatório como chave. Esse número não é salvo em lugar algum, somente é salva a *hash* que servirá para acesso do usuário ao banco de dados remoto, o que impossibilita o rastreio do usuário a partir da *hash*, além de diminuir a chance de colisão dentro do banco de dados remoto.

4.5.3 Interface gráfica

A interface gráfica do aplicativo foi feita de modo a garantir melhor usabilidade do aplicativo. Foram respeitados os fluxos mostrados dos casos de uso, apresentados na Seção 4.4.2, e dos protótipos mostrados na Seção 4.4.5.

Diante disso, foram criados alguns Widgets personalizados para a padronização do aplicativo. Dentre eles estão:

a) ***Dialog***: utilizado para adicionar e editar transações e emitir avisos;

b) ***Card***: utilizado para exibir dados estáticos;

c) ***Card* editável**: utilizado nos locais onde haverá entradas do usuário;

d) **Botão de fluxo de telas**: utilizado para os botões de continuar, confirmar, cancelar e voltar;

e) **Botão de tela inicial**: utilizado para os botões presentes na tela inicial do aplicativo.

Outros Widgets mais complexos foram criados utilizando esses como base, porém, uma vez que os elementos de interface estavam padronizados, o desenvolvimento das telas se tornou simples. Foi utilizada uma paleta de cores verde e amarelo, sendo o amarelo para botões e o verde pro restante dos elementos da interface.

4.5.4 Versionamento

O controle de versão foi feito utilizando o Github[7]. A Figura 20 mostra a organização utilizada para desenvolvimento do trabalho.

[6] keyed-Hash Message Authentication Code - Secure Hash Algorithm. Criptografia feita a partir de uma mensagem, aqui representada pelo nome do usuário, e uma chave, aqui representada por um número aleatório(**??**).

[7] <https://github.com>

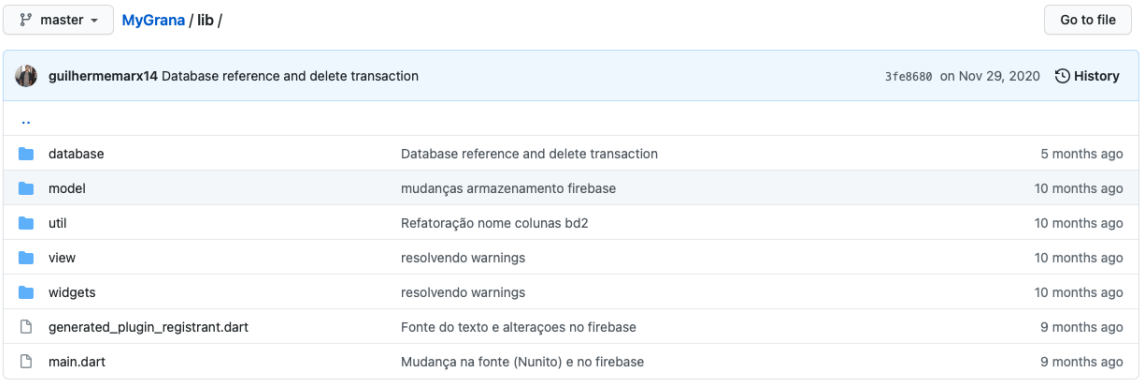


Figura 20 – Organização das Pastas do Projeto.

Fonte: O autor.

O pacote database possui os controladores dos bancos de dados locais, model possui as classes referentes a cada objeto do programa, util tem os arquivos auxiliares do trabalho (scripts, funções de uso comum, etc), view guarda o código referente às telas e em widgets se encontram os widgets desenvolvidos para padronização da interface do aplicativo.

4.6 Testes

Os testes foram realizados com 6 usuários por 2 meses (de 17 de maio a 16 de julho de 2020). Dos usuários, 4 utilizavam Android puro, 1 utilizava Android modificado pela Xiaomi e 1 utilizava sistema iOS.

O primeiro mês se dedicou à correção de *bugs* e melhorias na interface para facilitar a usabilidade. Alguns fluxos de telas foram modificados, como por exemplo o fato de que os fluxos de planejamento e comparação de valores reais e planejados era um fluxo só que, para facilitar o uso dos recursos, foi dividido em dois. Os nomes e ícones dos botões e a apresentação em uma tela de tamanho reduzido também foram aprimorados, melhorando a usabilidade dos usuários.

No primeiro mês foram obtidos resultados muito bons, com diversas alterações até que todos os usuários ficassem totalmente cientes de como acessar todos os recursos a partir da tela inicial. Ao final desse mês também se mostrou necessária uma categoria referentes a “outros” gastos, uma vez que alguns gastos pontuais não categorizados, apesar de não serem frequentes, podem ocorrer.

O segundo mês foi de uso livre do aplicativo, visando verificar a coleta de dados e como ela se comportaria em um período de tempo. Após 2 semanas de uso, um dos usuários do Android puro reclamou de lentidões e travamentos na aplicação. Para verificar o erro, os dados referentes a esse usuário no banco de dados remoto foram extraídos e injetados no emulador

de Android do Android Studio, onde o Memory Profiler identificou a causa da lentidão e travamento como sendo um alto consumo de memória em algumas telas. Com o *debug* pôde ser verificado que o pico de memória era atingido quando o resultado de consultas aos bancos de dados continham um grande número de resultados, uma vez que o aplicativo não faz paginação desses resultados.

Não foram observados problemas na coleta de dados, ela funcionou exatamente como fora planejado.

4.7 Considerações Finais

Esse capítulo tratou das decisões de projeto, modelagem e implementação do app MyGrana. Ao início foi desenvolvido um aplicativo de testes a fim de verificar qual seria o engajamento da aplicação e se ela seria capaz de ajudar o usuário da forma como se propõe. O teste foi bem sucedido, mostrando um bom engajamento por meio de um grande número de usuários e de transações cadastradas.

A partir dos resultados desse teste, foram levantados os requisitos que o aplicativo final deveria satisfazer. Os requisitos cobrem as *features* que deveriam ser implementadas a fim de se atingir os objetivos do MyGrana. A partir desses requisitos foi feita a modelagem da aplicação visando o cumprimento de todos os requisitos e simplificar a implementação.

A implementação foi feita com o *framework* multiplataforma Flutter, para garantir um maior alcance pro aplicativo, seguindo os fluxos de tela propostos na modelagem. A anonimidade do usuário foi garantida com uma função Hash, deixando o aplicativo de acordo com a Lei Geral de Proteção de Dados Pessoais L13709/2018.

O controle de versão foi feito utilizando o GitHub e os testes foram realizados por 6 usuários que utilizaram o aplicativo por 2 meses. Dessa forma, com base nos artefatos criados durante o processo de modelagem, o aplicativo atinge seu objetivo de ajudar os estudantes, além de coletar dados que podem ser analisados posteriormente.

A facilidade de aprendizado do *framework* Flutter mostrou um poder por parte do *framework* de disponibilizar recursos robustos de forma simples. O desenvolvimento da interface gráfica do aplicativo foi relativamente simples se comparado a outros *frameworks*. Além disso, a utilização de múltiplas Threads deixa o aplicativo mais rápido e com alta reatividade.

Versões mais recentes do Flutter, ainda em versão beta, são capazes de compilar aplicações para ambientes web, macOs, Linux e Windows, com o mesmo código fonte. Isso pode aumentar ainda mais o alcance do MyGrana, permitindo que os usuários o utilizem em seus computadores.

Parte IV

Resultados

5 Resultados

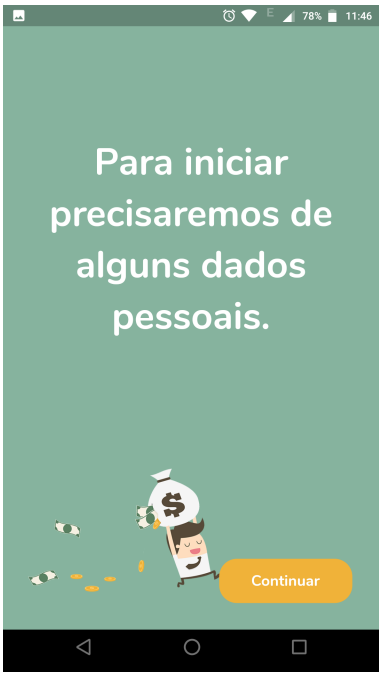
Este capítulo apresenta os resultados atingidos a partir do disposto na modelagem e implementação do MyGrana. A Seção 5.1 mostra o fluxo de cadastro do aplicativo, a Seção 5.2 mostra o fluxo de adicionar transações, a Seção 5.3 traz as telas referentes aos extratos, a Seção 5.4 identifica as telas referentes a gastos e rendas, a Seção 5.5 trata das telas de orçamento e a Seção 5.6 mostra a tela de informações do aplicativo.

As telas e transições foram feitas de modo a serem idênticas em dispositivos Android e iOS.

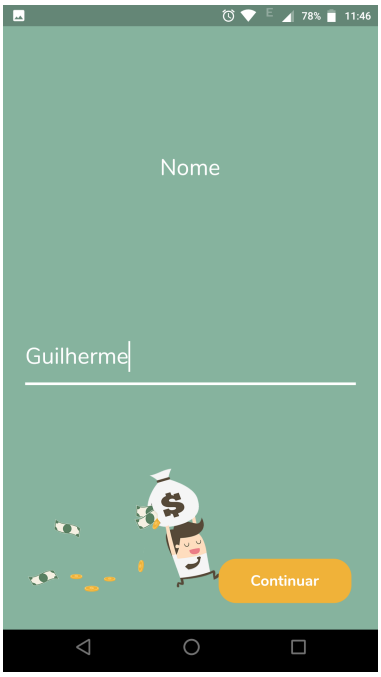
5.1 Fluxo de cadastro

As telas presentes no fluxo de cadastro referente ao Cadastro são apresentadas em sua versão final na Figura 21.

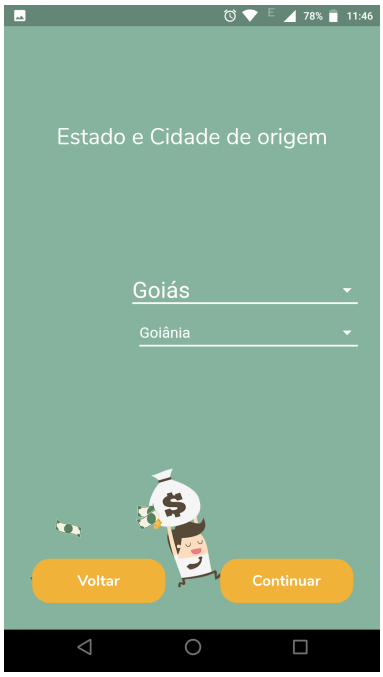
(a) Tela de *splash*.



(b) Tela de Boas-vindas.



(c) Cadastro do Nome.



(d) Cadastro de *profile* 1.

(e) Cadastro do *profile* 2.

(f) Tela de extrato.

Figura 21 – Telas do fluxo de cadastro.

Fonte: O autor.

Na Figura 21a pode-se observar a tela de *splash* do aplicativo. Essa tela é exibida sempre ao início da execução do aplicativo e tem como finalidade carregar os dados necessários para o funcionamento do aplicativo. Já na Figura 21b é apresentada a tela em que é feita as boas-vindas ao usuário e indica que alguns dados sobre ele devem ser preenchidos antes que ele possa fazer uso do aplicativo.

A Figura 21c exibe a tela na qual o usuário irá escrever seu nome, sabendo que esse nome será utilizado para criação da *hash* de identificação do usuário, o aplicativo não deixa o usuário prosseguir sem o preenchimento dessa informação.

As figuras Figura 21d, Figura 21e e Figura 21f mostram a forma na qual são coletados os dados referentes ao *Profile* do usuário, são eles: estado e cidade de origem e de estudo e universidade em que o usuário estuda. Nessas telas já possível identificar a padronização da paleta de cores, do formato dos botões, da cor da fonte e também a logo do aplicativo.

Após esse fluxo de telas, o usuário é redirecionado para a tela inicial, mostrada na Figura 22:

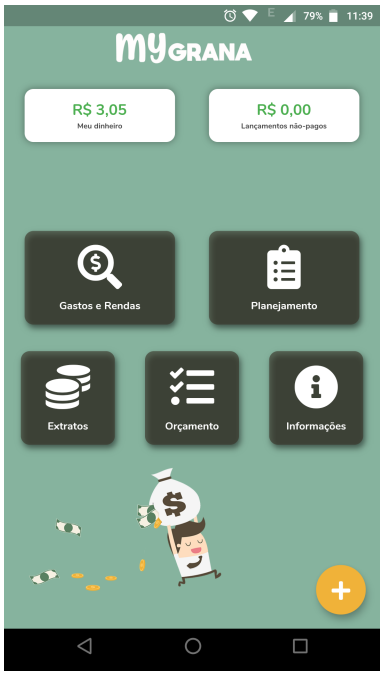


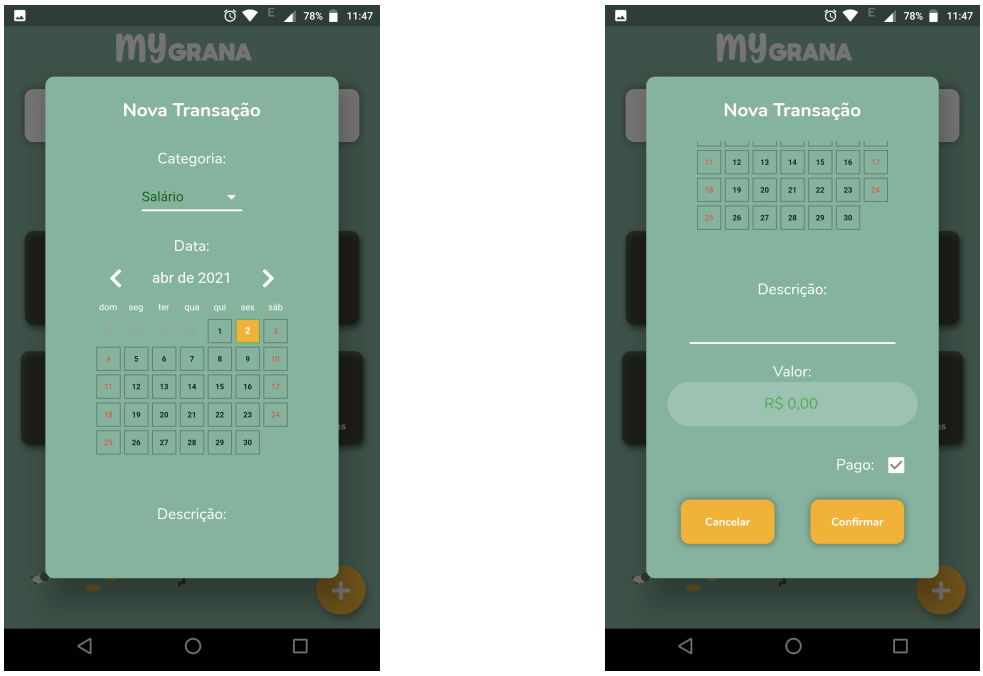
Figura 22 – Tela inicial do aplicativo.

Fonte: O autor.

Pode-se observar na tela inicial olhando de cima para baixo (vide Figura 22), a logo do aplicativo, o total de dinheiro do usuário e dos lançamentos não pagos (valores que ficam verdes quando positivos e vermelhos quando negativos), o botão flutuante para adicionar nova transação e os quatro botões presentes que foram definidos no projeto da tela prototipada na subseção 4.4.5. Além disso, foi adicionado o botão de Informações, que será descrito e apresentado mais à frente no texto deste capítulo.

5.2 Adicionar Transação

Ao clicar no botão flutuante mostrado na Figura 22 é apresentado o *dialog* mostrado na Figura 23 referente à Adição de transação:



(a) Dados de Nova Transação 1. (b) Dados de Nova Transação 2.

Figura 23 – *Dialog* de Nova Transação.

Fonte: O autor.

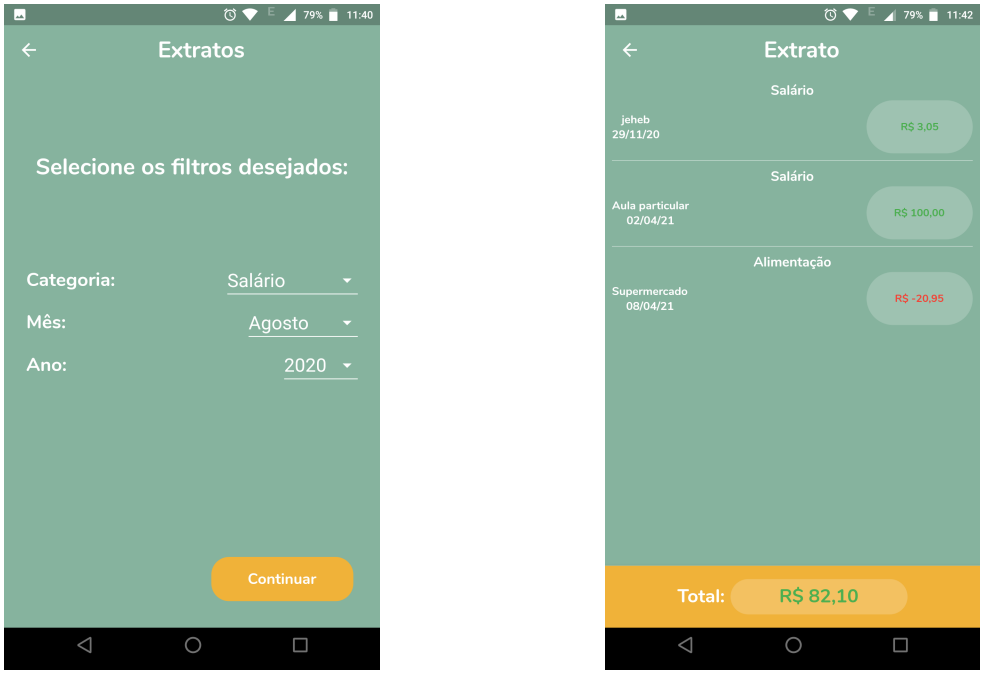
A primeira informação a ser fornecida é a categoria da transação, categorias de renda são mostradas em verde e as de gastos são mostradas em vermelho. Em seguida escolhe-se a data no calendário, podendo ser o dia atual, uma data passada ou futura, o que cobre o agendamento de transações. Após isso, pode ser adicionada uma descrição, a qual não é obrigatória. Por fim, é escrito um valor para a transação e escolhido se ela já foi paga ou não. Ao selecionar o botão Confirmar, a transação é persistida no banco de dados. Se for selecionado o botão Cancelar, a transação é descartada.

Esse mesmo *dialog* é utilizado na Edição de transação, porém, aqui com a alteração no título do *dialog* e com a diferença que todos os dados já vêm preenchidos com os dados da transação a ser editada.

5.3 Extratos

Como um *dialog*, essa tela também é utilizada em mais de um lugar no aplicativo. Pode-se verificar na Figura 24 a exibição de extratos com filtro, utilizada pela Exibição de extratos com filtros. Partindo da tela inicial apresentada na Figura 22 e selecionando-se o botão Extratos

é exibida em primeiro lugar a tela de filtros (apresentada em Figura 24a) e em seguida a de extratos (exibida na Figura 24b).



(a) Escolha dos filtros. (b) Lista de transações.

Figura 24 – Exibição de extratos.

Fonte: O autor.

Na tela de escolha de filtros, o usuário pode escolher uma categoria, um mês e um ano e ao clicar em Continuar, as transações que atendem a esse filtro são mostradas na tela de extrato. Para cada transação listada são apresentados: categoria, descrição, data e valor, sendo que categorias de rendas mostram valores verdes e categorias de gastos mostram valores vermelhos. Na parte inferior da tela é mostrado o total da soma dessas transações listadas, valores positivos são mostrados em verde, valores negativos em vermelho.

Essa tela é reutilizada em outros casos de uso a saber:

a) Edição de transação: Estando nessa tela de extratos, ao tocar uma transação é aparecido o *dialog* da figura Figura 23, sendo que o título alterado para Editar Transação e todos os dados da transação já vêm preenchidos;

b) Exclusão de transação: Para excluir uma transação basta selecionar por um tempo uma transação que aparecerá o mesmo *dialog*, porém, ao invés dos campos para serem preenchidos, é exibida apenas uma mensagem de confirmação e os botões de Confirmar e Cancelar;

c) Exibição do extrato do mês atual: Ao selecionar o botão Gastos e Rendas, o fluxo de telas será mostrado a seguir, ma,s ao final, o usuário escolhe uma categoria e

os filtros de mês e ano são aplicados automaticamente de acordo com a data e as transações são listadas.

5.4 Gastos e Rendas

Ao clicar no botão Gastos e Rendas conforme é apresentado na Figura 22 referente à Exibição do extrato do mês atual, o usuário é redirecionado para uma tela mostrando o total de gastos e rendas daquele mês e o valor total por cada categoria, isso pode ser visto na Figura 25.

Figura 25 – Tela de Gastos e Rendas.

Fonte: O autor.

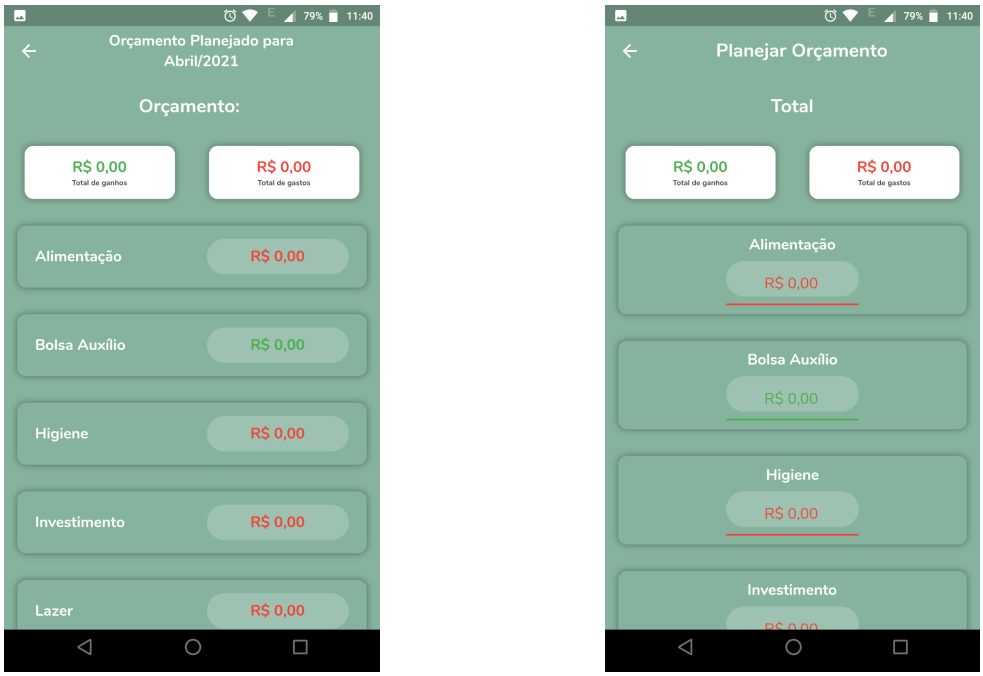
Na parte superior da Figura 25 são apresentados o mês e ano atuais com os valores todos calculados com base em transações para essa data. Na parte de baixo, são apresentados o total de gastos e rendas do mês e por último é mostrado o total de cada categoria.

Selecionando uma categoria, o usuário é redirecionado para a tela mostrada na Figura 24b, na qual o mês e o ano aplicados são os atuais e a categoria é a escolhida pelo usuário.

5.5 Orçamentos

As *features* referentes à orçamentos podem ser acessadas por intermédio do botão Planejamento ou do botão Orçamento na tela inicial (Planejamento e Comparação). O primeiro é referente ao planejamento do orçamento e o segundo a comparação dele com os gastos e rendas reais.

A Figura 26 apresenta o fluxo de telas do aplicativo ao ser escolhida a opção Planejamento na tela inicial.



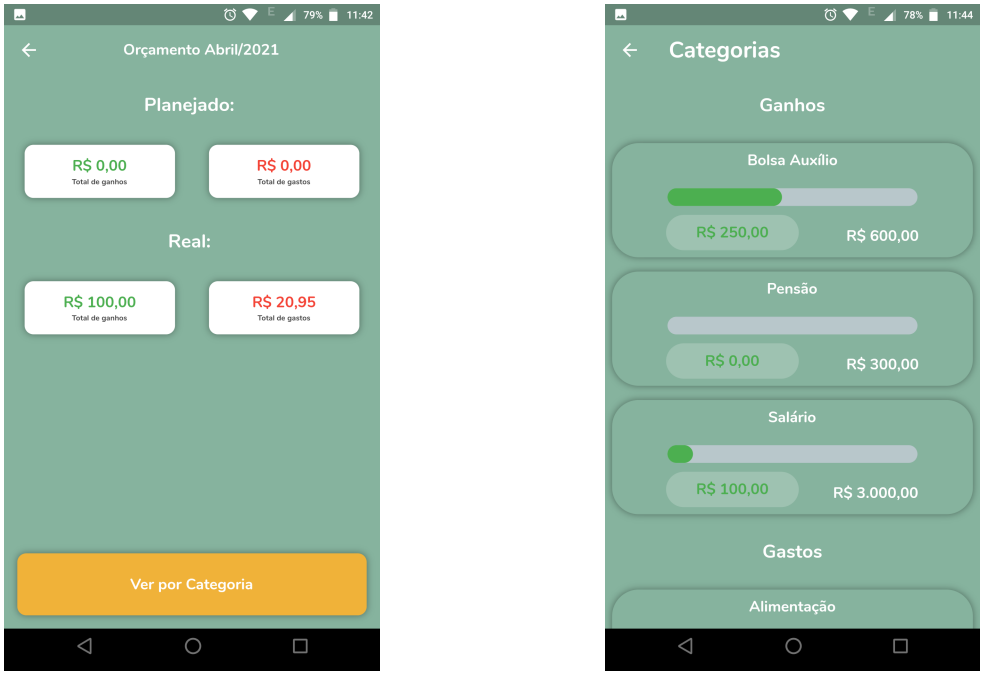
(a) Exibição do orçamento do mês atual. (b) Editar orçamento.

Figura 26 – Edição de orçamentos.

Na Figura 26a pode-se visualizar a tela de exibição do orçamento que foi planejado. O título da tela identifica o mês e o ano desse orçamento, em seguida é apresentado o valor total de gastos e rendas previstos nesse orçamento para cada categoria e ao final é mostrado um botão para editar o orçamento mostrado.

A Figura 26b apresenta a tela de edição do orçamento, nessa figura é exibido o total de gastos e rendas no orçamento que está sendo planejado e uma caixa de texto para cada categoria, onde o usuário digita o valor a ser destinado para aquela categoria. Ao final do planejamento existe um botão para salvar esse orçamento.

De acordo com a Figura 27, são apresentadas as telas de comparação entre o orçamento planejado e os gastos e rendas reais, acessível a partir da tela inicial com o botão Orçamento.



(a) Comparação entre valores totais.

(b) Comparação por categorias.

Figura 27 – Comparação de orçamento planejado e real.

Na Figura 27a é apresentada a comparação entre os valores totais de gastos e rendas e o total planejado. No título é exibido o mês e ano da comparação, em seguida o total de rendas e gastos planejados, por fim, rendas e gastos reais e por fim um botão para visualização desses valores com base em cada categoria.

A Figura 27b mostra a tela de comparação por categorias. Cada categoria é exibida como um *card* contendo um simples gráfico para comparar os valores reais com os planejados de forma que a cada categoria o usuário possa ver o quanto dos gastos e rendas planejados já foram utilizados até o presente momento.

5.6 Informações

A última tela de interface gráfica do aplicativo desenvolvida é a de informações do aplicativo. O propósito dessa tela é mostrar que o aplicativo foi desenvolvido em um Trabalho de Conclusão de Curso e colocar o nome do desenvolvedor, do orientador e do coorientador. Ela pode ser visualizada na Figura 28.

Figura 28 – Informações.

Fonte: O autor.

5.7 Considerações

Pode-se observar a partir das telas de interface gráfica presentes no aplicativo que todos os casos de uso foram satisfeitos. O Cadastro é apresentado na Figura 21, a Adicão de transação é satisfeita conforme construção apresentada na Figura 23, a Edição de transação pode ser visualizada na Figura 23, na Figura 24 a Exclusão de transação é atendida, os casos de uso de Planejamento e Comparação são satisfeitos na Figura 26 e Figura 27, enquanto os casos de uso de Exibição de extratos com filtros e Exibição do extrato do mês atual podem ser vistos através das telas apresentadas na Figura 24.

O fato de os elementos de interface serem padronizados dão ao app identidade e deixam seu uso mais prazeroso para o usuário, principalmente se comparado com o aplicativo utilizado para testes que foi desenvolvido no início do trabalho. Além disso, essa padronização também facilita o desenvolvimento das telas, uma vez que margens, *paddings*, formas, cores, fontes, estilos e outros já estão pré-definidos. Assim, a complexidade de desenvolvimento de cada tela se torna basicamente a lógica de funcionamento e não a interface gráfica.

6 Conclusão

Nesse trabalho foi desenvolvida a aplicação *mobile* MyGrana, um aplicativo de auxílio à gestão financeira de estudantes universitários. O aplicativo foi desenvolvido utilizando o *framework* Flutter, que é uma ferramenta para desenvolvimento multiplataforma, com o objetivo de aumentar o alcance e difusão do aplicativo para diferentes tipos de usuários.

A fundamentação teórica do trabalho se deu a partir de conceitos de gerenciamento financeiro pessoal e familiar e de desenvolvimento de software multiplataforma. O estudo desses conceitos deixou fundamentado o planejamento para o desenvolvimento do aplicativo de maneira a cumprir seu objetivo e identificar as *features* necessárias para sua realização.

Os trabalhos relacionados mostraram a capacidade que aplicativos móveis tem de auxiliar em aspectos importantes da vida do usuário. Com eles ficou clara a importância do engajamento e da busca de novas soluções para atender as mais diversas demandas que o usuário possui. Eles mostraram a forma com a qual um aplicativo auxilia o usuário, deixando dicas importantes de desenvolvimento para que o aplicativo possa de fato ser relevante.

Foi feito o levantamento de requisitos para desenvolvimento de uma aplicação a fim de testar qual seria o engajamento dos estudantes e para verificar se esse aplicativo seria capaz de auxiliar o usuário como se propõe a fazer. Esse aplicativo foi desenvolvido exclusivamente para ambiente Android por familiaridade do autor com as ferramentas. Foi utilizado um banco de dados simplificado do que seria o banco de dados final, além de contar com as ferramentas do Firebase para ajudar a melhorar a experiência do usuário, desde o rastreamento de *bugs* até a possibilidade de sincronização do banco de dados remoto para que o usuário não perca as transações cadastradas em caso de limpeza de *cache* ou reinstalação do aplicativo. O teste foi bem sucedido, os resultados apresentaram um bom engajamento dos usuários com o aplicativo pois um grande número de transações foram cadastradas.

A modelagem do aplicativo multiplataforma se deu de forma a manter as principais *features* da aplicação de testes, para tal foram removidas as *features* que foram pouco utilizadas e adicionado um requisito de anonimidade do usuário no banco de dados remoto para que não houvesse a possibilidade de identificação do usuário a partir de seus dados. Isso se refletiu principalmente na modelagem dos bancos de dados e na forma de coleta dos dados.

Os casos de uso se deram a partir da análise desses requisitos levantados. Eles servem para mostrar o fluxo de execução que o aplicativo precisa realizar. Sendo assim, com base em cada caso de uso, foram criados protótipos de fluxos de telas a fim de deixar mais claro a sequência de passos necessárias para cumprir todos os requisitos. A modelagem utilizou de diagramas de sequência e de atividades do sistema com o objetivo de colocar em foco as relações entre o fluxo de telas e suas dependências e sequência de comportamento do aplicativo

com relação aos acessos aos bancos de dados. A arquitetura do software se deu a partir do modelo MVC.

6.1 Limitações do trabalho

Assegurar a anonimidade do usuário no banco de dados remoto acabou por remover a *feature* que o aplicativo de testes possuia de ser possível a utilização de uma mesma conta por diversos dispositivos ou sincronização das transações no momento do *login*. Essa é uma *feature* muito útil para núcleos com vários estudantes como repúblicas estudantis, já que possibilitava que vários usuários utilizassem a mesma conta, deixando os gastos e rendas de todos os usuários acessíveis.

O alto uso de memória RAM do dispositivo também é um problema da aplicação desenvolvida. Uma vez que ela não faz paginação dos resultados de buscas ao banco de dados, se houverem muitas transações nos resultados um alto custo de memória pode ser percebido.

6.2 Contribuições

Durante a pesquisa de fundamentações teóricas e trabalhos relacionados não foi encontrado, até o presente momento, trabalho que se dedicasse à gestão financeira de estudantes universitários por meio do uso de um aplicativo ou em dispositivos móveis. Essa abordagem utiliza de conceitos de gerenciamento financeiro juntamente com tecnologias móveis para se criar uma solução que pudesse mitigar os altos custos psicológicos que podem ser causados por estresse financeiro.

O uso de bancos de dados locais em conjunto com bancos de dados remotos possibilitou a coleta dos dados dos usuários, que foi feita mantendo o usuário anônimo. O uso do Firebase Realtime Database possibilitou um banco de dados remoto de tempo real e, ainda, faz com que o usuário não precise de conexão com a Internet para cadastrar uma transação já que o Firebase possui ferramentas para que a sincronização seja feita apenas quando houver conexão.

Além disso, este projeto deixa o ambiente do banco de dados remoto pronto para ser utilizado futuramente em análises de dados na tentativa de identificar melhorias no uso do aplicativo. Essa análise pode ser usada para estudar padrões de comportamento e, com isso, propor novas soluções ainda mais específicas para cada tipo de estudante, maximizando tanto o alcance dessa proposta quanto sua capacidade de auxílio.

6.3 Trabalhos Futuros

As próximas subseções apresentam possíveis trabalhos futuros. Trabalhos relacionados à coleta e análise dos dados são mostrados na subseção 6.3.1, enquanto otimizações do aplicativo

podem ser encontradas na subseção 6.3.2.

6.3.1 Coleta e Análise dos Dados

No âmbito de análise dos dados, por meio do banco de dados remoto podem ser identificados padrões no comportamento dos usuários tendo com base a universidade e cidade que o usuário estuda. Esses padrões podem ser utilizados para incentivar projetos sociais nas cidades analisadas, na predição de comportamentos dos estudantes ou mesmo como fonte de informações para autoridades públicas.

Analisando os dados podem ser identificadas questões tais como o impacto econômico que a presença dos estudantes causa nos comércios da cidade ou estado analisados. Esse impacto econômico pode ser utilizado para fazer medições a respeito do impacto de presença de restaurantes universitários, da moradia pública ou até mesmo da distribuição de bolsas por parte da universidade. Isso também pode ser utilizado em um sistema para acesso público que identifica uma média de ganhos e despesas por universidade e por cidade, o que seria bastante útil para estudantes que pretendem entrar em uma determinada instituição. Muitas vezes esse estudante está se candidatando para estudar em locais fora de sua cidade de origem, o que torna difícil o planejamento financeiro antes do deslocamento do estudante da cidade na qual reside sua família para a cidade da universidade.

Para que essas análises possam ser potencializadas, pode-se coletar mais dados, como curso e campus de cada usuário pois algumas universidades possuem mais de um campus em uma mesma cidade. Dessa forma, a especificidade dos dados analisados de acordo com o perfil dos estudantes aumenta, já que passa a ser possível observar usuários de cada curso e fazer comparações entre os diferentes cursos. Isso pode ser utilizado, por exemplo, para sugerir ao usuário orçamentos personalizados com base em seus gastos pessoais e nos gastos de outros estudantes do mesmo curso ou campus.

6.3.2 Otimizações da Aplicação

Algumas otimizações na aplicação também são sugeridas para trabalhos futuros. É possível melhorar o consumo de memória RAM do aplicativo aplicando-se técnicas de paginação e otimização nas buscas realizadas no banco de dados local.

A permissão de uso de uma mesma conta em vários dispositivos também pode ser realizada no futuro. Isso pode ajudar no uso do aplicativo por repúblicas estudantis. Além disso, pode ser permitido também o uso de diversas contas em um mesmo dispositivo com o intuito de auxiliar núcleos que compartilham aparelhos. Ambas são *features* que estavam presentes no aplicativo de testes.

Uma *feature* que pode ser feita é integração com o aplicativo de Calendário do dispositivo, isso pode ser usado para avisar o usuário de lançamentos futuros. Outra *feature* sugerida é

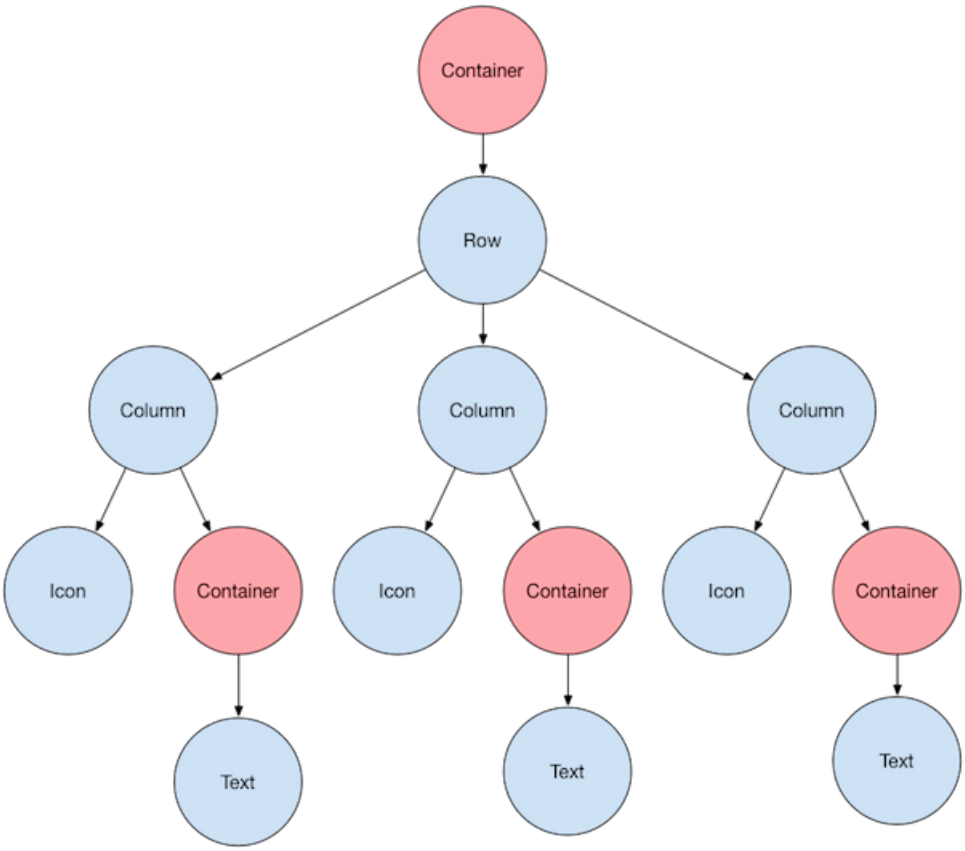
notificações de alerta ao usuário do momento em que ele estiver se aproximando do total planejado de alguma categoria.

Apêndices

APÊNDICE A – Composição de telas com o *framework* Flutter

O elemento básico da programação de uma aplicação com Flutter se chama Widget. Um *widget* nada mais é do que um elemento de interface gráfica, desde um simples texto até a tela por completo. Tudo na interface é uma combinação de *widgets* (??).

Figura 29 – Exemplo de árvore de *Widgets*



Fonte: <https://flutter.dev/docs/development/ui/layout>

Na Figura 29 vê-se um exemplo de uma árvore de *widgets*. No nível 0 há um Container, *widget* que serve como "caixa" onde se colocam outros *widgets*, ele pode possuir 1 filho, ou *child*, que também deve ser do tipo *widget*.

No nível 1 é mostrado um Row, ele serve para organizar seus *widgets* filhos dentro de uma linha na interface gráfica. Ele pode possuir uma lista filhos, ou *children*, que também devem ser do tipo *widget*. No caso da imagem ele possui 3 *children*, cada um sendo do tipo

Column.

No nível 2 temos 3 *widgets* Column, cada um deles permite organizar seus *children* em uma coluna. Nesse caso, cada um tem como *children* um *widget* Icon e um Container.

Em seguida, pode-se ver que cada Container possui um *child* do tipo Text, que também é um *widget*.

Essa alta diversidade de tipos de *widgets* se deve ao polimorfismo que a linguagem Dart proporciona (**??**). Uma classe pode se tornar um *widget* herdando as classes StatelessWidget[1] ou StatefullWidget[2].

A herança e o polimorfismo da linguagem Dart são marcas constantes nos projetos Flutter. A combinação de *widgets* simples para formar *widgets* mais complexos é fundamental para organização do código.

Quando uma classe herda StatelessWidget ou StatefullWidget basicamente deve implementar um método *build* que retorna um objeto do tipo *widget* que será apresentado em tela quando seu construtor for chamado.

Outra marca fundamental são as funções de alta-ordem e, consequentemente, funções anônimas e de *callback*. Por exemplo, se for criado um *widget* que é um botão, no seu construtor deve ser passada qual a ação que será tomada uma vez que esse botão for selecionado. Dentro do *widget* do botão, essa ação enviada no construtor se torna uma variável do tipo Function e pode, assim, ser usada no momento correto.

Outra relação importante é a entre o *widget* e os seus construtores. Na Figura 29 o construtor do Container no nível 0 recebe um *widget* como parâmetro na propriedade *child*. Esse *child* receberá o construtor de um Row, que recebe uma lista de *widgets* na propriedade *children*, cada um destes é um Column que recebe outra lista de *widgets* na propriedade *children* e assim por diante. Isso mostra que a interface gráfica de uma tela em Flutter é simplesmente uma composição de *widgets*, que sao exibidos a partir de seus construtores. Assim, o código das interfaces é bastante simples, porém a lógica de implementação dos construtores por muitas vezes pode ser bastante complexa.

[1] *Widget* sem estado, ou seja, não é possível modificar seus parâmetros sem que se atualize a tela por completo.
[2] *Widget* com estado, ou seja é possível modificar seus parâmetros sem atualizar a tela por completo

APÊNDICE B – Diagramas Complementares

B.1 Diagramas de Atividades

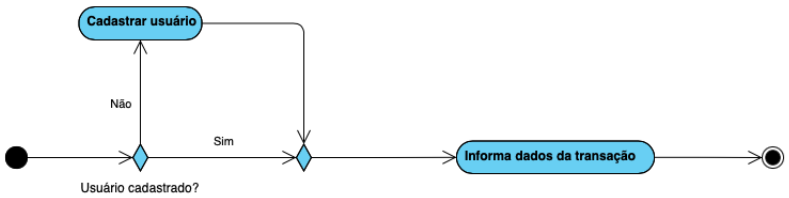


Figura 30 – Diagrama de Atividades Referente a Adicionar uma Transação.

Fonte: O autor.

Figura 31 – Diagrama de Atividades Referente ao Cadastro de usuário.

Fonte: O autor.

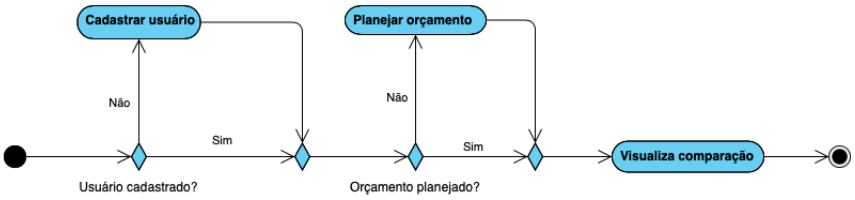


Figura 32 – Diagrama de Atividades Referente a Comparação entre Orçamento Planejado e Real.

Fonte: O autor.

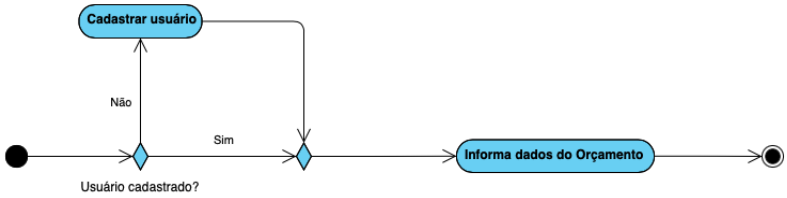


Figura 33 – Diagrama de Atividades Referente a Edição de Orçamento.

Fonte: O autor.

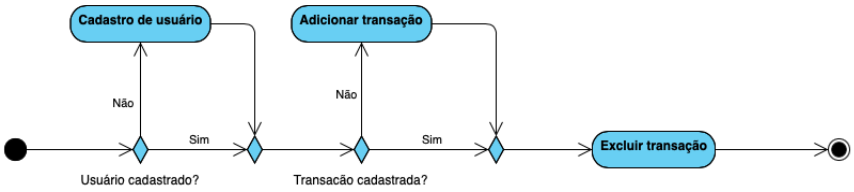


Figura 34 – Diagrama de Atividades Referente a Excluir uma Transação.

Fonte: O autor.

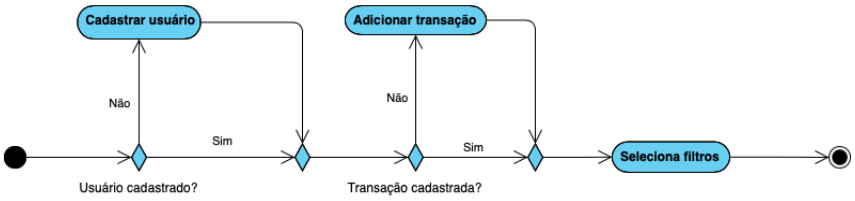


Figura 35 – Diagrama de Atividades Referente a Exibição de Extratos.

Fonte: O autor.

B.2 Diagramas de Sequência

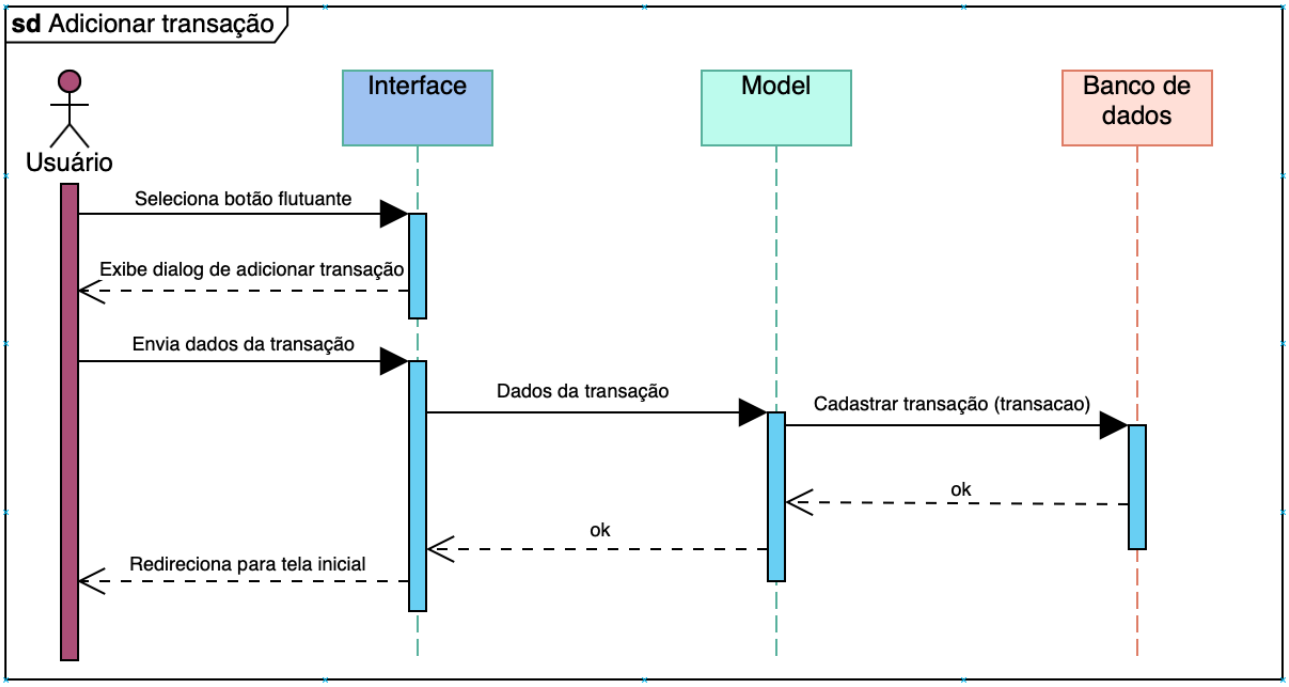


Figura 36 – Diagrama de Sequência Referente a Adicionar uma Transações.

Fonte: O autor.

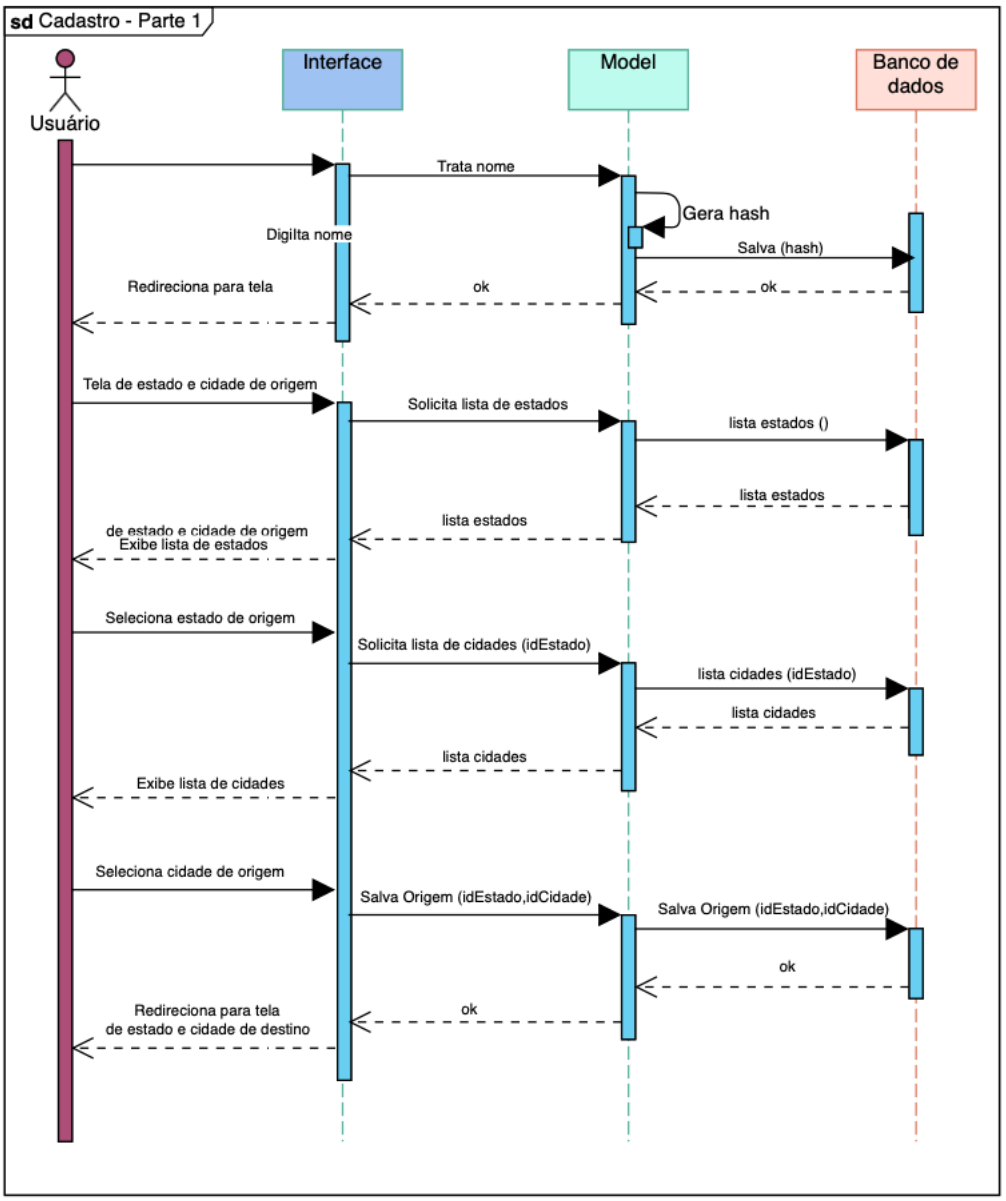


Figura 37 – Diagrama de Sequência Referente ao Início do Cadastro.

Fonte: O autor.

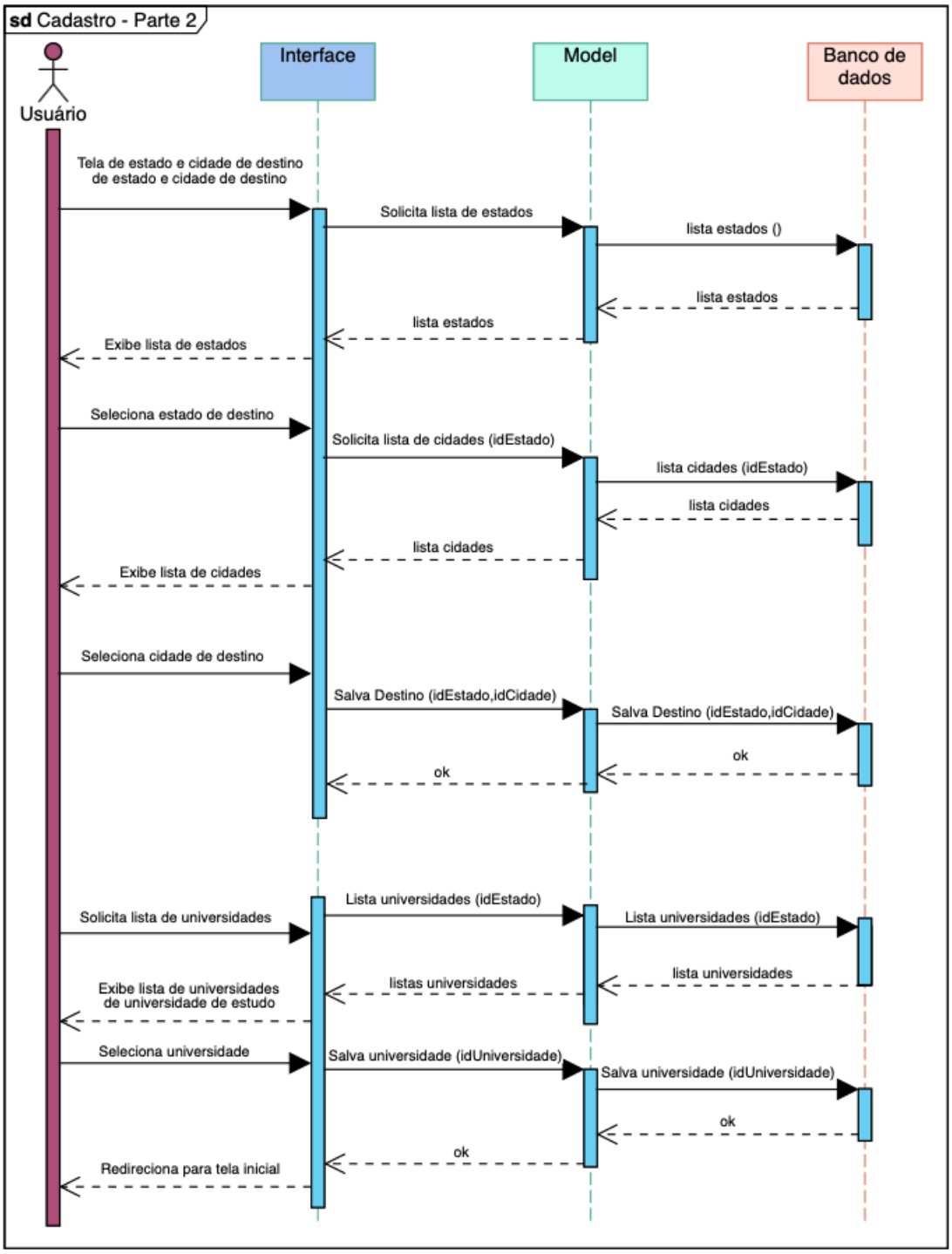


Figura 38 – Diagrama de Sequência Referente ao Final do Cadastro.

Fonte: O autor.