UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

DVDR: APLICATIVO DE RATEIO DE DESPESAS

LUCAS REINERT

BLUMENAU
2020

LUCAS REINERT

DVDR: APLICATIVO DE RATEIO DE DESPESAS

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Sistemas de Informação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

Prof. Everaldo Artur Grahl, Mestre - Orientador

BLUMENAU
2020

DVDR: APLICATIVO DE RATEIO DE DESPESAS

Por

LUCAS REINERT

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:	Prof. Everaldo Artur Grahl – Orientador(a), FURB
Membro:	Prof. Gilvan Justino – FURB
Membro:	Prof. Mauro Marcelo Mattos – FURB

Blumenau, 14 de dezembro de 2020

Dedico este trabalho ao Professor Everaldo Artur Grahl, que foi meu Orientador e me auxiliou em todas as fases do projeto.

AGRADECIMENTOS

Aos meus pais e irmã que me incentivaram em toda a minha jornada, não somente no TCC mais em todo o curso.

Ao professor Everaldo Artur Grahl, que aceitou me orientar e esteve comigo desde o início do projeto até agora.

Aos meus amigos e colegas que me ajudaram a melhorar meu TCC me dando ideias e todo apoio necessário.

A única verdade é a realidade.

Aristóteles

RESUMO

Este trabalho tem como objetivo demonstrar o desenvolvimento de um aplicativo móvel capaz de realizar a divisão das despesas de um determinado grupo de pessoas. Por meio do aplicativo é possível criar grupos e adicionar pessoas que irão participar do rateio das despesas. A arquitetura de *cliente-servidor* do aplicativo funciona com *backend* construído em Java 11, utilizando o framework Spring Boot e está presente na parte do servidor. A aplicação *mobile* foi construída com a linguagem de programação Dart e *framework* Flutter. Para realizar a autenticação dos usuários, armazenamento de arquivos e chat em tempo real, foi utilizado o serviço do Firebase, sendo o banco de dados PostgreSQL. Por fim, foi realizada uma avaliação de usabilidade do aplicativo com a metodologia, System Usability Scale (SUS). O aplicativo recebeu a avaliação de "muito bom" e permite um rateio de despesas eficiente, com interação e uma forma efetiva de validação dos pagamentos.

Palavras-chave: Rateio de despesas. Aplicativo móvel. Usabilidade

ABSTRACT

This paper aims to demonstrate the development of a mobile application capable of dividing the expenses of a certain group of people. Through the application it is possible to create groups and add people who will participate in the apportionment of expenses. The client-server architecture of the application works with a backend built in Java 11, using the Spring Boot framework and is present on the server side. The mobile application was built with the Dart programming language and Flutter framework. To perform user authentication, file storage and chat in real time, the Firebase service was used, being the PostgreSQL database. Finally, an usability evaluation of the application was carried out using the methodology, System Usability Scale (SUS). The application received the rating of “very good” and allows an efficient apportionment of expenses, with interaction and an effective way of validating payments.

Keywords: Apportionment of expenses. Mobile App. Usability.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ABNT – Associação Brasileira de Normas Técnicas

API – Application Programming Interface

ARM - Advanced RISC Machine

FK – Foreign Key

GPS - Global Positioning System

IDE – Integrate Development Environment

JPA – Java Persistence API

MER – Modelo de Entidade e Relacionamento

OEM - Original Equipment Manufacturer

PK – Primary Key

RF – Requisito Funcional

RNF – Requisito Não Funcional

RSI – Redes sociais da Internet

SGBD – Sistema Gerenciador de Banco de Dados

SUS – System Usabillity Scale

UC – Use Case

UML – Unified Modeling Language

UID – User Identifier

WIFI - Wireless Fidelity

SUMÁRIO

1 INTRODUÇÃO

O rateio de despesas tem importância para a tomada de decisão e planejamento financeiro. É através dos rateios que se consegue alocar as parcelas dos custos indiretos aos diferentes produtos ou às diferentes funções de acumulação de custos, quer eles sejam fixos ou variáveis (DUTRA, 1994). De acordo com Dutra (1994), rateio é uma divisão proporcional por uma base que tenha dados conhecidos em cada uma das funções em que se deseja apurar custos.

Para que o indivíduo possa ter controle sobre os seus recursos, faz-se necessário que ele saiba fazer uma boa gestão de suas finanças pessoais. Por isso, tratar sobre finanças pessoais pode ser visto como uma necessidade. Conforme cita Pires (2006), tratar as finanças pessoais como uma área de conhecimento sistemático e transmissível no âmbito da ciência econômica, é uma necessidade contemporânea.

A busca por uma divisão correta, influencia tanto o meio corporativo quanto o pessoal. Por meio desta, que aplicações como Splitwise (2019) e Evenfy (2019) se destacam, pois, conseguem proporcionar aos seus usuários maneiras justas de divisão no valor das despesas e garantir que cada pessoa pague um valor correto. Ambas as aplicações possuem um conjunto de maneiras de dividir uma ou várias despesas. Desse modo garante que o valor a ser pago por cada usuário, seja um valor pertinente ao que foi usufruído.

Diante deste cenário, este trabalho propõe o rateio de despesas, procurando auxiliar a realização dele de forma adequada e buscando auxiliar a realizar a divisão na maior quantidade de cenários possíveis. Foram adotadas diferentes estratégias de rateio, para conceder a cada usuário, opções diferenciadas de ratear as despesas que cada um consumiu. Pretende-se desta forma contribuir com o planejamento financeiro pessoal dos usuários, e agregar segurança para informar o pagamento de cada usuário.

1.1 OBJETIVOS

O objetivo principal deste trabalho é disponibilizar um aplicativo móvel para gerenciar o rateio de despesas para determinados grupos. Os objetivos específicos são:

a) possibilitar de forma prática a divisão de despesas entre as partes de um grupo;
b) agregar segurança na confirmação do pagamento de cada membro do grupo;
c) desenvolver uma aplicação para facilitar a visualização dos gastos individuais e do grupo.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção explora fundamentos importantes para o desenvolvimento do tema proposto. A subseção 2.1 trata sobre o desenvolvimento de plataformas móveis, incluindo conceitos de aplicativo nativo, *webapp* e híbrido. A subseção 2.1 discorre sobre a temática voltada para desenvolvimento para plataformas móveis. Na subseção 2.2 é abordado o tema das redes sociais. A subseção 2.3 aborda sobre a avaliação de usabilidade. Por fim, na subseção 2.4 são apresentados os trabalhos correlatos.

2.1 DESENVOLVIMENTO PARA PLATAFORMAS MÓVEIS

O desenvolvimento para plataformas móveis possui uma enorme gama de linguagens no mercado. Pelo fato de existir uma grande demanda para aplicações que sejam capazes de executar tanto no sistema operacional Android quanto iOS, foram criados frameworks para o desenvolvimento híbrido que tem como objetivo executar em ambos os sistemas operacionais.

Para cumprir a necessidade das aplicações híbridas, atualmente existem vários *frameworks*, pode-se citar Xamarin, React Native e Flutter. O mercado de desenvolvimento de apps móveis necessitava que fosse possível diminuir o tempo de desenvolvimento destas aplicações e atingir mais de uma plataforma rapidamente. Essa é a proposta base das aplicações híbridas. (SILVA, 2019).

Conhecendo os diferentes tipos de aplicações, pode-se afirmar que a aplicação desenvolvida neste trabalho é uma aplicação híbrida. O *framework* Flutter foi utilizado no desenvolvimento da aplicação. Criado pela Google em 2017, o *framework* utiliza como linguagem de programação o Dart. De acordo com Gažo (2018), a utilização do Dart que é uma linguagem compilada, evita um problema comum de desempenho que é a ponte de compilação. Sendo que a linguagem Dart é capaz de realizar a compilação antes do tempo, garantindo uma inicialização mais rápida.

De acordo com Stackoverflow (2020), os aplicativos desenvolvidos em Flutter são escritos na linguagem de programação Dart e podem se conectar com as linguagens de programação Java, Kotlin, Swift e Objective-C. Diferente das opções para desenvolvimento mobile híbrido, o Flutter não utiliza WebView e nem widgets Original Equipment Manufacturer (OEM). Segundo Stackoverflow (2020), o *framework* possui o seu próprio mecanismo de renderização de alto desempenho e estrutura para desenhar os widgets na tela. Segundo Flutter (2020), os widgets incorporam diferenças críticas de plataforma, como rolagem, navegação, ícones e fontes, e seu código é compilado para código de máquina Advanced RISC Machine (ARM) nativo utilizando Dart.

2.2 REDES SOCIAIS

Os elementos que constituem a criação desta aplicação contemplam características de uma rede social, trazendo consigo aspectos cujo qual possibilita dentro de sua estrutura virtual a possibilidade conexão e interação entre pessoas sobre interesses comuns. Tratamos das redes sociais na internet (RSI) como espaço coletivo e colaborativo para a comunicação, troca de informação, aprofundamento de um determinado tema, pesquisa. (ALLEGRETTI *et al*., 2012). Deste modo, o intuito é conectar pessoas e compartilhar de maneira ágil informações que necessitam de alternativas para tomada de decisões, tais como a divisão de despesas entre grupos de pessoas.

> O uso das redes sociais tem se intensificado, pois cresce a cada dia o número de usuários que querem estar conectados e gostam de compartilhar todo tipo de informações e materiais digitais. As redes se expandem e são alocadas para o lazer, para uso social, para uso comercial, para a cultura, para a educação etc. (ALLEGRETTI *et al*., 2012, p. 59).

Dentro do contexto das redes sociais, existem vários produtos diferentes que são consideradas redes sociais ou que possuem características. Portanto, cada produto possui os seus diferenciais tendo em vista os seus objetivos com o determinado produto. É mencionado por Koehler (2016) as características que os sites de redes sociais ofertam diferem um dos outros, pois assim como a informação a ser compartilhada no Instagram existe uma prioridade de compartilhamento de imagens e vídeos de poucos segundos. O Facebook possibilita o compartilhamento de fotos, vídeos, arquivos, textos, parágrafos e links. Tendo em vista as funcionalidades destas redes sociais é possível entender que as redes sociais têm como buscam conectar relações entre os usuários com o objetivo de compartilhar várias informações da forma que desejarem.

As redes sociais não só proporcionam facilidades na troca de informações como também mantém todo um histórico de tudo que já aconteceu na rede. Conforme citado por Recuero (2014), a rede sempre mantém as interações e ali elas permanecem, sendo assim elas podem ser replicadas facilmente caso desejado. O histórico que é gerado com o tempo nas redes sociais, pode ser útil para identificar ou relembrar algum conhecimento deixado para trás. Na aplicação desenvolvida neste trabalho por exemplo, o histórico das trocas de mensagens pode auxiliar os usuários a compreender uma decisão que foi tomada no passado para adicionar ou não uma despesa ou uma pessoa no grupo.

2.3 AVALIAÇÃO DE USABILIDADE

No desenvolvimento de aplicativos para plataformas móveis, sites e até mesmo aplicações desktop. Uma preocupação muito grande é em como tornar a aplicação agradável e deixando claro o propósito de seu uso. Para alcançar estes objetivos os desenvolvedores podem seguir alguma heurística que contém vários princípios a respeito do que deve ser feito ou não ser feito, contribuindo assim para que o resultado da aplicação possa ter a melhor usabilidade possível.

A usabilidade de uma aplicação independente da plataforma, é muito importante para que o usuário consiga navegar de forma fluída e compreender o propósito da aplicação. Para Nielsen (2006), usabilidade é um atributo de qualidade que avalia a facilidade de uso das interfaces de usuário. A palavra “usabilidade” também se refere a métodos para melhorar a facilidade de uso durante o processo de design. Nielsen (2006) aponta 5 importantes componentes de usabilidade, que são aprendizagem, eficiência, memorização, erros e satisfação.

Para avaliar a usabilidade da aplicação deste trabalho foi utilizada a metodologia SUS. Segundo Barboza (2019), a metodologia de avaliação de usabilidade foi criada no ano de 1986 por John Brooke, a metodologia permite avaliar uma variável de produtos e serviços de uma forma cientifica e que não seja longa para o usuário e nem para o pesquisador. De acordo com Barboza (2019), o SUS possui três pilares divididos entre efetividade, eficiência e satisfação. A efetividade tem como objetivo descobrir se os usuários conseguem completar seus objetivos. A eficiência busca entender quanto esforço e recursos são necessários para que os usuários consigam completar os seus objetivos.

De acordo com Geraldes, Martins, Afonseca (2019) está métrica é constituída por um questionário padronizado e considerado o mais utilizado do mundo por sua praticidade e rapidez. Lewis (2018) afirma que, quando os pesquisadores e profissionais precisam de uma medida de usabilidade, estes certamente consideram seriamente o uso do SUS.

De acordo com Barboza (2019), para calcular o índice de usabilidade do SUS, é necessário verificar as respostas das questões ímpares e subtrair 1 da pontuação que o usuário atribuiu como resposta. Para as respostas pares, deve ser feito o cálculo 5 – resposta atribuída pelo usuário. Depois de realizado esse procedimento, deve ser somado todos os valores das dez perguntas e multiplicar por 2,5. Chegando assim na pontuação do índice de usabilidade SUS. Quando a pontuação for abaixo de 60 pontos atribui-se que a aplicação possui uma usabilidade inaceitável. A pontuação entre 60 e 70 pontos, é considerada Ok. Para a pontuação entre 70 e

80 pontos, a usabilidade da aplicação é considerada Boa. Caso a aplicação alcance uma pontuação entre 80 e 90 pontos, a usabilidade é considerada excelente. A melhor usabilidade possível pode ser alcançada somente para aplicações que atingem uma pontuação superior a 90.

2.4 TRABALHOS CORRELATOS

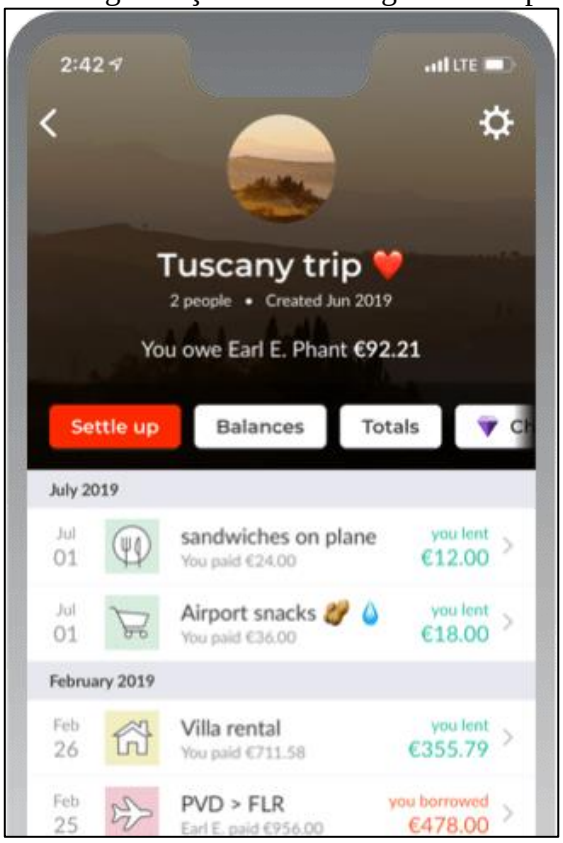
Nesta sessão são apresentados trabalhos que possuam relação com o trabalho proposto nesta pesquisa. Na subseção 2.4.1 demonstra-se a aplicação de Splitwise (2019), uma ferramenta gratuita que auxilia na divisão de contas por grupo de pessoas. Na subseção 2.4.2, é apresentado o aplicativo de Evenfy (2019) que permite a criação dos eventos, registro dos custos e destaca-se pela interação entre os membros do grupo. A subseção 2.4.3 apresenta a aplicação Passa Régua de Oasys(2013), que possui um layout que lembra uma nota fiscal. Trata-se de uma proposta que busca uma divisão justa entre o grupo de pessoas e possibilita compartilhar a cobrança em redes sociais.

2.4.1 Splitwise

O Splitwise é uma aplicação capaz de dividir contas, disponibilizada para os sistemas operacionais Android, iOS ou acessível no ambiente *web*. Para garantir que as despesas sejam divididas da maneira mais justa possível, Price (2019) relata que as despesas podem ser dívidas da maneira que o usuário desejar. Pode ser divido igualmente, por percentual ou por um valor específico.

Ao lidar com o rateio de despesas, é necessário garantir que a parte de cada membro do grupo seja paga. Gutoskey (2019) cita que é possível fazer o upload de imagens de recibos, ou configurar cobranças recorrentes e pagar pelos aplicativos Venmo ou Paypal. A Figura 1 ilustra a aplicação de Splitwise (2019) e destaca a organização das despesas de uma viagem.

Figura 1 - Organização de uma viagem com Splitwise



Fonte: Splitwise (2019).

Splitwise (2019), proporciona uma integração com o seu sistema através da disponibilização de uma Application Programming Interface (API), garantindo uma maior flexibilidade da aplicação. Conforme é apresentado no Quadro 1, a documentação oferecida por Splitwise (2019) possui SDK's desenvolvida por terceiros para as linguagens: Javascript, Ruby, Python, Elixir e Java.

Quadro 1 – Lista de SDK's disponibilizadas

Linguagem	Link
Javascript	https://github.com/keriwarr/splitwise
Ruby	https://github.com/divyum/splitwise-ruby
Python	https://github.com/namaggarwal/splitwise
Elixir	https://github.com/matiasdelgado/ex_splitwise
Java	https://github.com/sritejkv/splitwise-java

Fonte: Splitwise (2019).

De acordo com Splitwise (2019), a documentação da API apresenta funcionalidades de autenticação, informações e atualizações da conta do usuário logado, gerenciamento do grupo do usuário logado, informações e gerenciamento da lista de amigos, despesas e notificações.

2.4.2 Evenfy

O Evenfy é um software que permite organizar contas em grupo de forma colaborativa e gratuita (FERRARI, 2017). A plataforma é fácil de usar e basta fazer o cadastro, organizar um grupo com os amigos envolvidos no evento, listar os gastos e efetuar os pagamentos. A aplicação pode ser acessada por várias plataformas. Preuss (2016) destaca que o aplicativo é disponibilizado em Android, iOS e na plataforma *web*.

Conforme apresentado na Figura 2, os participantes dos eventos podem interagir entre si para ajudar e facilitar a organização do evento, além disso podem inserir suas despesas, acompanhar as informações com transparência e trocar mensagens para ajudar na organização (Evenfy, 2019).

Figura 2 - Troca de mensagens entre participantes

Fonte: Coletiva (2019).

Por ser uma ferramenta colaborativa, é possível a troca de mensagens entre os participantes do grupo, inserir novas despesas e acompanhar em tempo real todas as informações relacionadas ao grupo. Na Figura 3 é apresentado um exemplo de participação dos usuários na inserção de despesas e interação com o grupo.

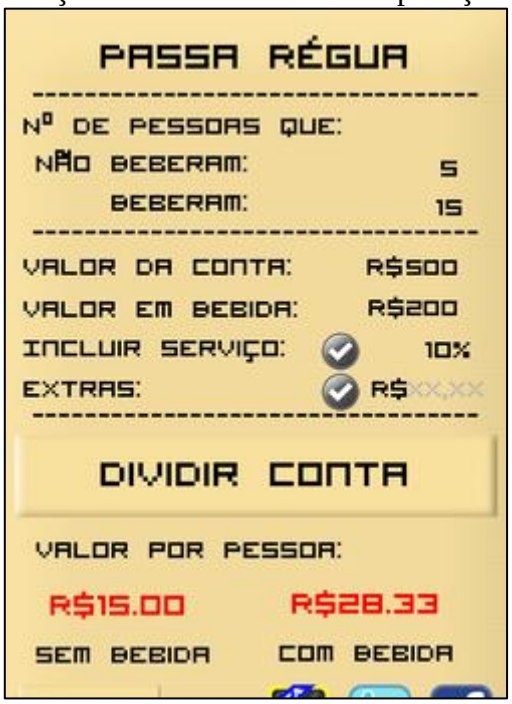
Figura 3 - Participação dos usuários no grupo

Fonte: Evenfy (2019).

2.4.3 Passa Régua

Com um layout muito similar ao de uma nota fiscal, a aplicação desenvolvida pela empresa Oasis Innovation. A proposta de Oasys(2013), é facilitar a divisão de despesas, possuindo uma fórmula de cálculo simples. O aplicativo Passa Régua está disponível para smartphones que contemplem o sistema operacional iOS e pode auxiliar os seus usuários a fazer uma divisão de despesas de forma rápida e simples. Portanto, não possui diversas formas de divisão, apenas algo mais simples e direto.

Figura 4 - Ilustração de funcionamento da aplicação Passa Régua



Fonte: Ferreira (2013).

Baseando-se na Figura 4, Ferreira (2013) cita que para fazer o cálculo, deve-se escolher dividir a conta - com bebida ou sem bebida – informar o valor da conta, incluir ou não o serviço e ver o resultado. Ferreira (2013) comenta que, a opção de dividir a conta com bebida permite que quem não consumiu nenhum tipo de líquido consiga pagar o valor proporcional. É possível compartilhar a dívida com os amigos pelo Facebook ou pelo Twitter.

3 DESENVOLVIMENTO

Neste capítulo é apresentado o levantamento de requisitos funcionais e não funcionais, a especificação e a implementação do aplicativo. Na subseção 3.1 são descritos os requisitos funcionais e não funcionais do aplicativo. Na subseção 3.2 são abordadas as especificações que contemplam o diagrama de casos de uso, a matriz de rastreabilidade, diagrama de classes, diagrama de atividades e modelo de entidade e relacionamento. A seção 3.3 aborda sobre as ferramentas, conceitos e técnicas utilizadas e a operação da aplicação. Por fim, a seção 3.4 destaca os resultados obtidos com este trabalho e faz um comparativo com os trabalhos correlatos pesquisados.

3.1 LEVANTAMENTO DE REQUISITOS

O presente trabalho trata de um aplicativo que realiza o gerenciamento de despesas com foco no rateio dos valores. Para o desenvolvimento deste aplicativo foi realizado um estudo a partir dos trabalhos correlatos que agregaram ideias para as funcionalidades do aplicativo. Deste modo, o aplicativo poderá ajudar os seus usuários com suas finanças pessoais pois possui uma funcionalidade específica para realizar o rateio de despesas entre um grupo de pessoas, sendo necessário somente um integrante do grupo ter o aplicativo.

O rateio de despesas pode ser realizado por um integrante de um grupo, ao adicionar ou editar uma despesa. O usuário deve informar o título, data valor da despesa e a quantidade, gerando assim um valor total que é o resultado do cálculo “valor da despesa x quantidade”. Esse valor total, é o valor que será rateado entre todos os participantes do grupo. Para ratear a despesa o usuário pode utilizar os botões da própria tela que o auxilia a realizar um cálculo automatizado baseado nos usuários que estão marcados no *checkbox*, ou se preferir, o usuário pode fazer a divisão de uma forma customizada atribuindo um valor ou percentual que um integrante do grupo irá pagar pela determinada despesa. Quando uma despesa é gerada, apenas o administrador do grupo ou o criador da despesa tem permissão para editar ou excluir a mesma. Sempre que uma despesa é adicionada, é possível verificar no grupo quais pessoas que estão pagando por essa despesa e quanto estão pagando, assim como, o valor total de todas as despesas. Além disso, o usuário logado tem acesso a todo o seu histórico de gastos dentro da aplicação baseado em todos os grupos que já participou.

A aplicação contém em si características de uma rede social, pois permite um usuário encontrar, visualizar o perfil e adicionar outros usuários na aplicação. Os usuários que estão dentro de um mesmo grupo, podem interagir entre si, sendo assim cada um pode adicionar, editar ou excluir as despesas e trocar mensagens de texto ou imagens pelo chat do grupo.

Ademais a aplicação permite que o administrador do grupo encerre a conta e posteriormente os participantes podem anexar comprovantes de pagamento e/ou adicionar uma observação a respeito do pagamento. Após isso o administrador pode visualizar e decidir se aprova ou não o recebimento do pagamento no aplicativo, garantindo assim uma maior segurança em relação ao recebimento.

A listagem a seguir apresenta os Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF) do aplicativo desenvolvido:

a) permitir o cadastro de usuário (RF);
b) permitir o cadastro de grupos (RF);
c) permitir o registro de despesas de um grupo (RF);
d) permitir o cadastro de contatos (RF);
e) permitir o cadastro de participantes do grupo (RF);
f) permitir a visualização do histórico do grupo. (RF);
g) permitir que os participantes de um grupo enviem mensagens de texto e imagens no chat (RF);
h) permitir a visualização do histórico de gastos do perfil (RF);
i) permitir a visualização do perfil de outros usuários cadastrados na aplicação (RF);
j) permitir que o administrador do grupo encerre a conta (RF);
k) permitir que os usuários do grupo possam informar o pagamento (RF);
l) permitir que o administrador do grupo aceite o pagamento dos usuários (RF);
m) desenvolver o *backend* utilizando a linguagem Java 11, com o *framework* Spring Boot (RNF);
n) desenvolvimento *mobile* utilizado a linguagem Dart, com o *framework* Flutter (RNF);
o) autenticação de usuários deve ser utilizado o Firebase Authentication (RNF);
p) as imagens salvas devem ficar armazenadas no Firebase Storage (RNF);
q) as mensagens do *chat* devem ficar armazenadas no Google Cloud Firestore (RNF);
r) o banco de dados utilizado deve ser o PostgreSQL (RNF);
s) o aplicativo deve rodar em celulares com o sistema operacional Android (RNF).

3.2 ESPECIFICAÇÃO

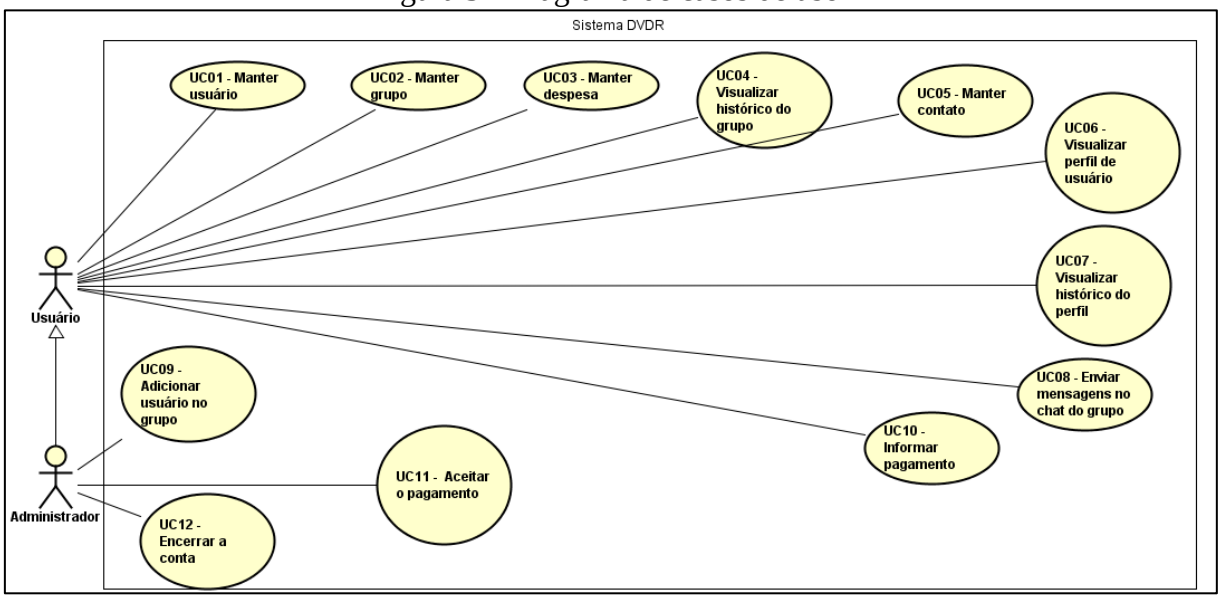
Esta seção apresenta os artefatos que foram criados a partir das necessidades da aplicação. A subseção 3.2.1 apresenta o diagrama de casos de uso. A subseção 3.2.2 apresenta a matriz de rastreabilidade que faz um mapeamento entre os casos de uso e os requisitos

funcionais do aplicativo. A subseção 3.2.3 apresenta o diagrama de classes. A subseção 3.2.4 demonstra o diagrama de atividades. Ao final desta seção, tem-se a subseção 3.2.5 que apresenta o Modelo de entidade e relacionamento (MER). Para o desenvolvimento dos diagramas da Unified Modeling Language (UML) usou-se a ferramenta Astah Community. Para a criação do MER, apesar do banco de dados da aplicação ser o PostgreSQL, foi utilizada a ferramenta do MySQL WorkBench de geração de MER.

3.2.1 Diagrama de casos de uso

Nesta subseção é apresentado o diagrama de casos de uso. É apresentado no diagrama o ator `Usuário` e todas as funcionalidades que ele tem acesso. Este usuário pode ser tanto um usuário comum, quanto o usuário administrador de um grupo. Na Figura 5 pode ser observado as ações do `Usuário` e suas respectivas interações com o aplicativo. A descrição mais detalhada de alguns casos de uso pode ser encontrada no Apêndice A.

Figura 5 - Diagrama de casos de uso



Fonte: elaborado pelo autor.

O caso de uso `manter usuário` é referente à manutenção do perfil do usuário, ou seja, ele pode cadastrar-se no aplicativo e editar as suas informações de acordo com a sua preferência, podendo alterar o nome, `foto de perfil`, `data de nascimento`, `senha` e `alterar o nome de como deseja ser encontrado por outros usuário`.

O caso de uso `manter grupo` está relacionado ao usuário poder criar, visualizar e editar um `grupo`. Na parte de edição pode-se editar a imagem, observação e título do grupo. Este caso de uso mostra-se fundamental, sendo um pré-requisito de outros casos de uso que o aplicativo possui.

O caso de uso `manter despesa` diz respeito a criação de despesas dentro de um grupo, ao atribuir um título, valor unitário, quantidade e data. Esta despesa pode ser editada pelo usuário administrador do grupo ou o usuário que criou a despesa. Dá ao administrador ou criador da despesa a possibilidade de selecionar quanto que cada usuário do grupo deve pagar pela despesa específica. Sendo possível atribuir um percentual, colocar manualmente um valor ou utilizar os botões disponíveis na tela para automatizar o rateio baseado na caixa de seleção dos integrantes do grupo. Os botões disponíveis na tela possuem duas opções de rateio, a primeira de ratear o valor total somente entre os usuários que estão marcados ou dividir o resto do valor (valor total – valor a pagar dos usuários não selecionados) entre os usuários selecionados, e uma terceira opção para zerar o valor de todos os usuários selecionados.

O caso de uso `visualizar histórico do grupo` refere-se à visualização do valor de cada despesa do grupo e quanto cada participante irá pagar por ela e quanto que cada integrante teve de gasto em todo o ciclo de vida do grupo.

O caso de uso `manter contato` se refere ao usuário conseguir enviar solicitação de contato, aceitar uma solicitação de contato ou remover um contato. Tendo adicionado um contato, é possível adicionar os contatos de um usuário ao grupo que ele em si é administrador.

O caso de uso `visualizar perfil de usuário` diz respeito a funcionalidade em que ao pesquisar por um usuário pelo seu nome exclusivo, pode-se clicar sobre o nome do usuário e uma nova tela de visualização de perfil deste usuário irá aparecer, contendo as informações que foram cadastradas em seu perfil.

O caso de uso `visualizar histórico de gastos do perfil` possibilita o usuário ter uma visão do seu ciclo de vida no aplicativo, pode-se verificar quais foram os seus gastos em cada grupo que participou.

O caso de uso `enviar mensagens no chat do grupo` refere-se à possibilidade de os usuários de um grupo interagirem entre si, sendo possível enviar mensagens de texto ou também imagens da galeria do celular.

O caso de uso `adicionar usuários no grupo` refere-se à possibilidade de o administrador do grupo adicionar um usuário no grupo que está na sua lista de contatos ou adicionar um anônimo, que seria a opção de adicionar um usuário que não é um contato ou não possui o aplicativo. Essa opção existe para que o rateio das despesas não dependa de que todas as pessoas possuam o aplicativo instalado, basta uma pessoa ter o aplicativo e ir poder adicionar contatos anônimos.

O caso de uso `informar pagamento` possibilita que, após o encerramento da conta do grupo, o usuário informe uma observação sobre o pagamento foi realizado ou adicionar uma imagem, seja uma nota fiscal, comprovante de pagamento ou qualquer outro meio.

O caso de uso `administrador do grupo aceitar o pagamento` irá aceitar o pagamento que foi mencionado no caso de uso `informar pagamento`. Após isso o aplicativo deve apresentar uma mensagem de Pago ao lado do nome do usuário, simbolizando o pagamento. No caso do usuário ser um usuário anônimo, o administrador pode simplesmente confirmar o pagamento deste quando desejar.

O caso de uso `encerrar a conta` está relacionada a funcionalidade do administrador encerrar a conta de um grupo, para que não seja mais possível inserir novos integrantes e nem novas despesas a fim de manter a integridade dos valores. Assim o administrador pode iniciar os processos do caso de uso `aceitar o pagamento` e `informar pagamento`.

3.2.2 Matriz de rastreabilidade

Nessa seção é apresentada a Matriz de rastreabilidade, conforme o Quadro 2 que objetiva relacionar os casos de uso que foram descritos na subseção 3.2.1 com os requisitos funcionais descritos na seção 3.1.

Quadro 2 - Matriz de rastreabilidade

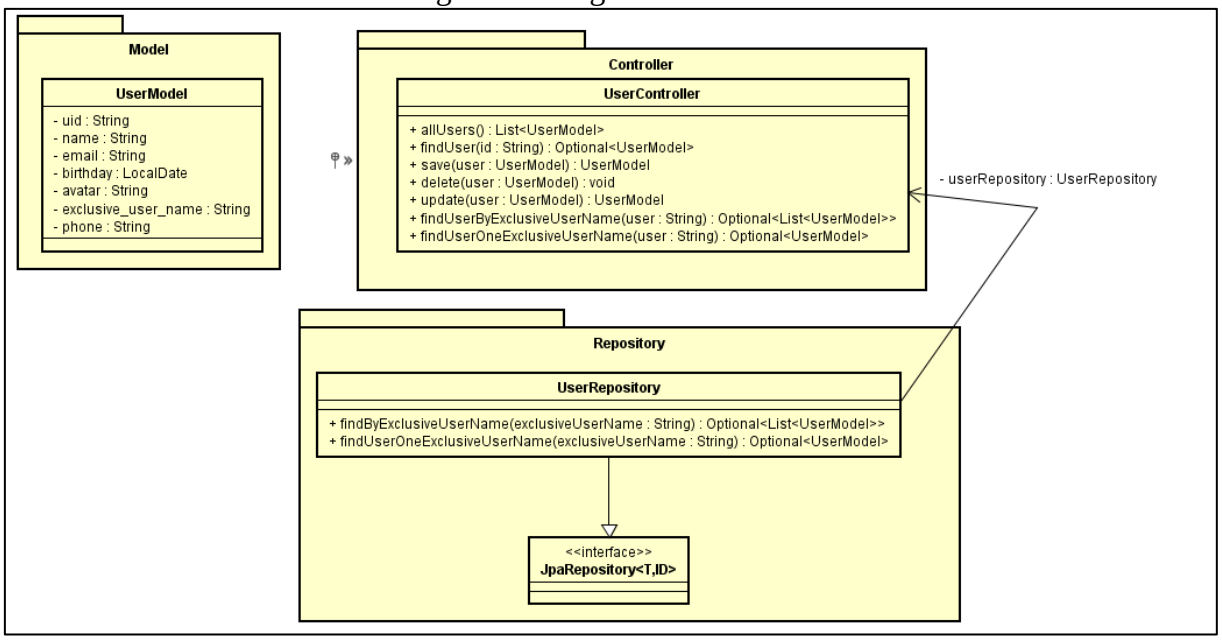
Requisitos Funcionais	UC
Permitir o cadastro de usuário	UC01
Permitir o cadastro de grupos	UC02
Permitir o registro de despesas de um grupo	UC03
Permitir o cadastro de participantes do grupo	UC09
Permitir que os participantes de um grupo consigam enviar mensagens de texto e imagens no chat	UC08
Permitir a visualização do histórico de gastos do perfil	UC07
Permitir o cadastro de contatos	UC05
Permitir a visualização do histórico do grupo	UC04
Permitir a visualização do perfil de outros usuários cadastrados na aplicação	UC06
Permitir que o administrador do grupo encerre a conta	UC12
Permitir que os usuários do grupo possam informar o pagamento	UC10
Permitir que o administrador do grupo aceite o pagamento dos usuários	UC11

Fonte: elaborado pelo autor.

3.2.3 Diagrama de classes

Essa subseção apresenta as classes no aplicativo através do diagrama de classes que está representado na Figura 6. Somente as classes relacionadas ao `Usuário` são apresentadas, pois são as classes principais para o funcionamento do aplicativo.

Figura 6 - Diagrama de classes



Fonte: elaborado pelo autor.

A entidade `UserModel` corresponde a classe de usuários. A partir das notações do Hibernate, esta classe é persistida no banco de dados criando uma tabela para os usuários com os respectivos atributos que constam no diagrama de classes `UserModel` do pacote `Model`. Para a classe `UserModel` foram omitidos os *getters* e *setters*, uma vez que são gerados pela anotação do Lombok que é uma das particularidades do SpringBoot.

A classe `UserController` situada no pacote `Controller`, responsável por criar os *endpoints* da aplicação, pode ser acessada por requisições Hypertext Transfer Protocol (HTTP). Estes *endpoints* farão a conexão entre uma aplicação externa com a aplicação *backend*. Esta classe possui os métodos `allUsers` que retorna todos os usuários do banco de dados, `findUser` que passado um ID por parâmetro a aplicação pesquisará por esse ID no banco de dados e retornará o resultado, `save` que irá receber uma entidade do tipo `UserModel` e irá salvar essa entidade no banco, o método *delete* que é responsável por deletar o usuário passado por parâmetro, também o método `update` que realizará a atualização do registro do usuário passado por parâmetro. Tem ainda `findUserByExclusiveUserName` que fará uma comparação com a coluna `exclusive_user_name` e retornará uma lista de usuários que contém um nome parecido com o que foi passado por parâmetro. Por fim o método `findUserOneExclusiveUserName` que fará uma comparação com a coluna `exclusive_user_name` e retornará apenas um registro.

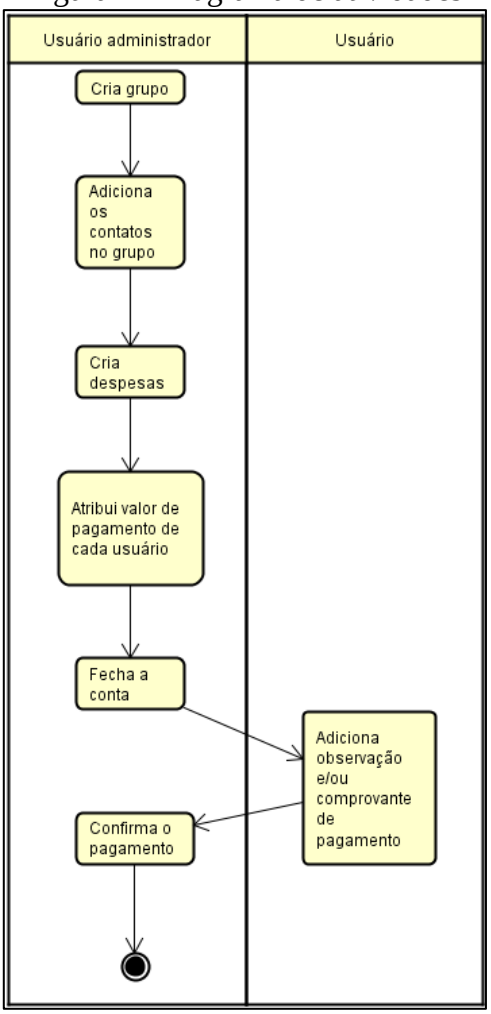
A classe `UserRepository` que estende da classe `JpaRepository` é a responsável por disponibilizar todas as pesquisas que são feitas no banco de dados. Nesta classe foram

implementados dois métodos: `findByExclusiveUserName` que realiza uma *query* de busca por uma lista de usuários que possuam na coluna `exclusive_user_name` um valor que contenha o parâmetro que foi passado, `findUserOneByExclusiveUserName` que realiza uma query busca um usuário que possua na coluna `exclusive_user_name` um valor igual ao que é passado por parâmetro.

3.2.4 Diagrama de atividades

Esta subseção apresenta o diagrama de atividades do processo de divisão de despesas. A Figura 7 representa todas as etapas do ciclo de vida de um grupo, parte principal do aplicativo. Uma característica importante a se observar, é que existem dois tipos de atores dentro de um grupo: Administrador e Usuário.

Figura 7 - Diagrama de atividades

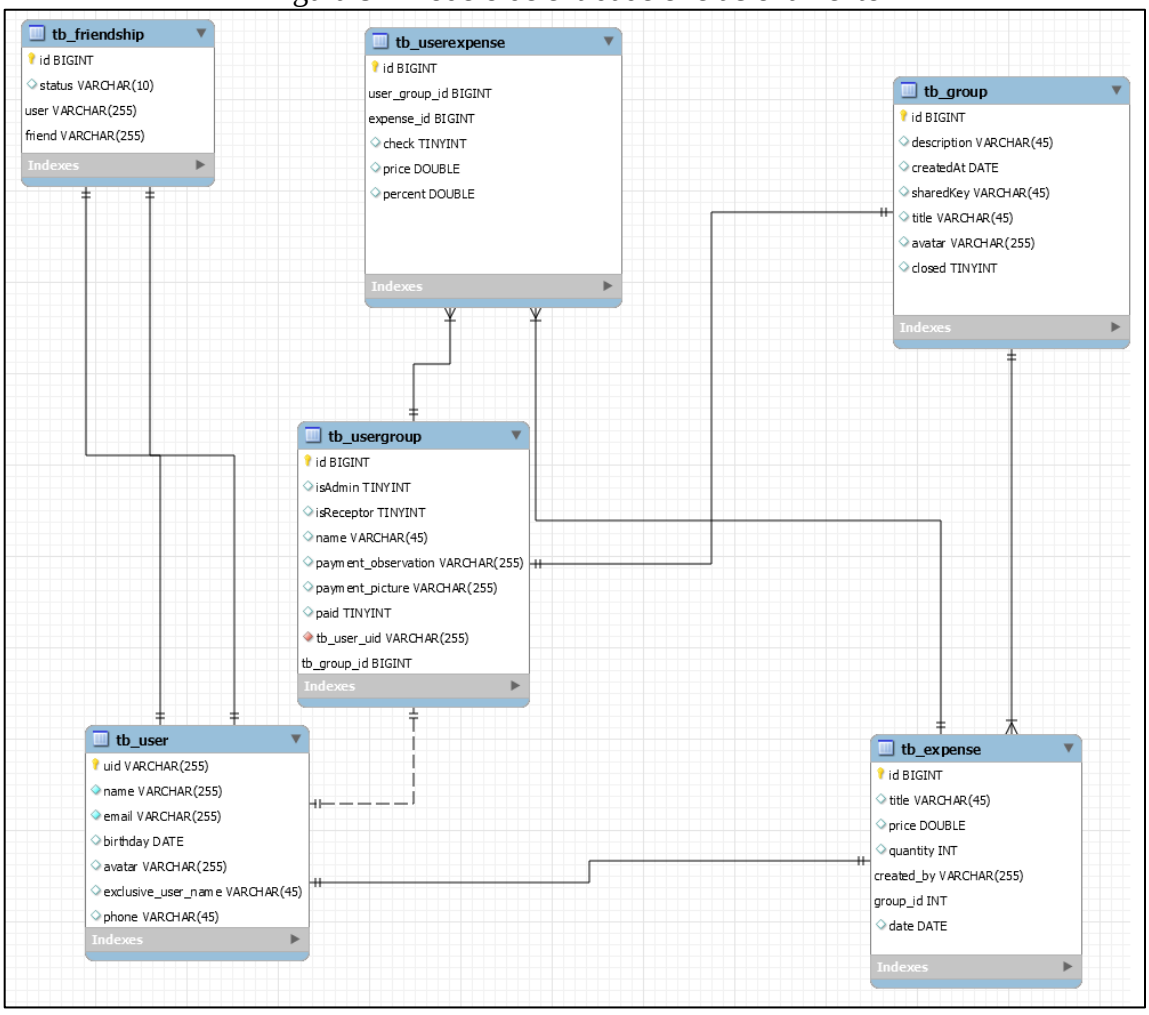


Fonte: elaborado pelo autor.

3.2.5 Modelo de Entidade e Relacionamento

Nesta subseção é apresentado o modelo de entidade e relacionamento. Aqui é possível visualizar todas as tabelas com seus respectivo atributos e relacionamentos que foram criados a partir do mapeamento de classes do Hibernate. A Figura 8 apresenta o MER da aplicação. O respectivo dicionário de dados foi apresentado no Apêndice B.

Figura 8 - Modelo de entidade e relacionamento



Fonte: elaborado pelo autor.

O detalhamento das entidades está descrito nos itens a seguir:

a) `tb_friendship`: relacionamento entre dois usuários que representam um contato no aplicativo;
b) `tb_usergroup`: relacionamento entre usuário e grupo;
c) `tb_user`: tabela que armazena os usuários do aplicativo;
d) `tb_userexpense`: tabela que faz o relacionamento entre usuário, grupo e despesa;
e) `tb_group`: tabela que armazena os grupos;
f) `tb_expense`: tabela que armazena as despesas.

3.3 IMPLEMENTAÇÃO

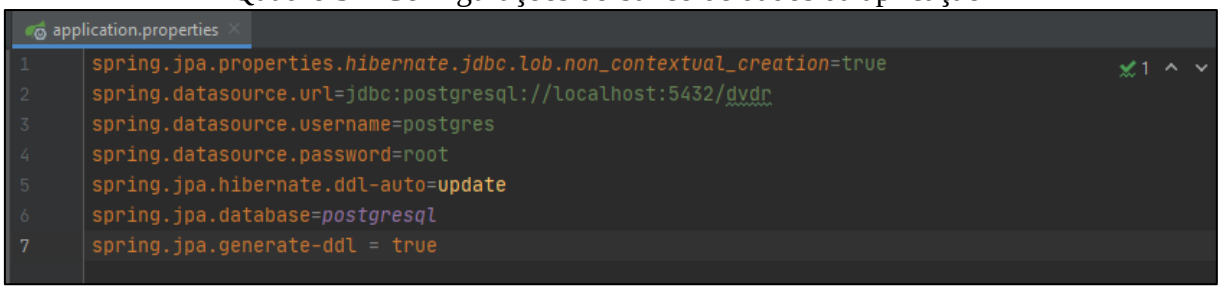
Nessa seção é demonstrado como foi realizado o desenvolvimento da aplicação, incluindo as técnicas, *framework* e linguagens adotadas.

3.3.1 Técnicas e ferramentas utilizadas

Para o desenvolvimento do *backend* utilizou-se a Integrated Development Environment (IDE) Intellij. A linguagem utilizada foi Java na versão 11 com o *framework* Spring Boot. O *backend* é responsável pela disponibilização das API que serão consumidas pela aplicação *mobile*. Esta aplicação *backend* é executada por um servidor Tomcat, responsável por receber as requisições através do protocolo Hypertext Transfer Protocol (HTTP).

O Sistema Gerenciador de Banco de Dados (SGBD) utilizado na aplicação é o PostgreSQL, pois é uma ferramenta gratuita e cumpre com todas as exigências que o projeto necessitou. Para se comunicar com o banco de dados e realizar as criações de entidade e relacionamento, e a resposta das API via JavaScript Object Notation (JSON), foi utilizado o Hibernate que é o padrão do Framework Spring Boot. A configuração entre a aplicação e o banco de dados é realizada no arquivo *application.properties* como é demonstrado no Quadro 3.

Quadro 3 – Configurações do banco de dados da aplicação



Fonte: Digitalizado pelo autor.

Quando a aplicação é executada, todos os *endpoints* que foram criados ficam acessíveis para serem consumidos por outras aplicações. Sempre que um *endpoint* é consumido por uma requisição HTTP a aplicação irá realizar todo o procedimento que foi escrito na função referente ao *endpoint* consumido, e retornar o que for pertinente para quem o consumiu. Os *endpoints* foram escritos na camada `Controller`. Esta camada é responsável por disponibilizar os *endpoints* e por conseguinte executar e fornecer uma resposta. No Quadro 4 é representado *endpoint* `/users` que faz parte da classe `UserController`. Esse *endpoint* deve fazer uma comunicação com o banco de dados através do objeto `userRepository`, e a invocação do método `findAll()`, irá retornar todos os usuários cadastrados na aplicação. Os demais

endpoints da classe estão disponíveis para visualização na Figura 9 que apresenta o Swagger da API.

Quadro 4 - API de usuário

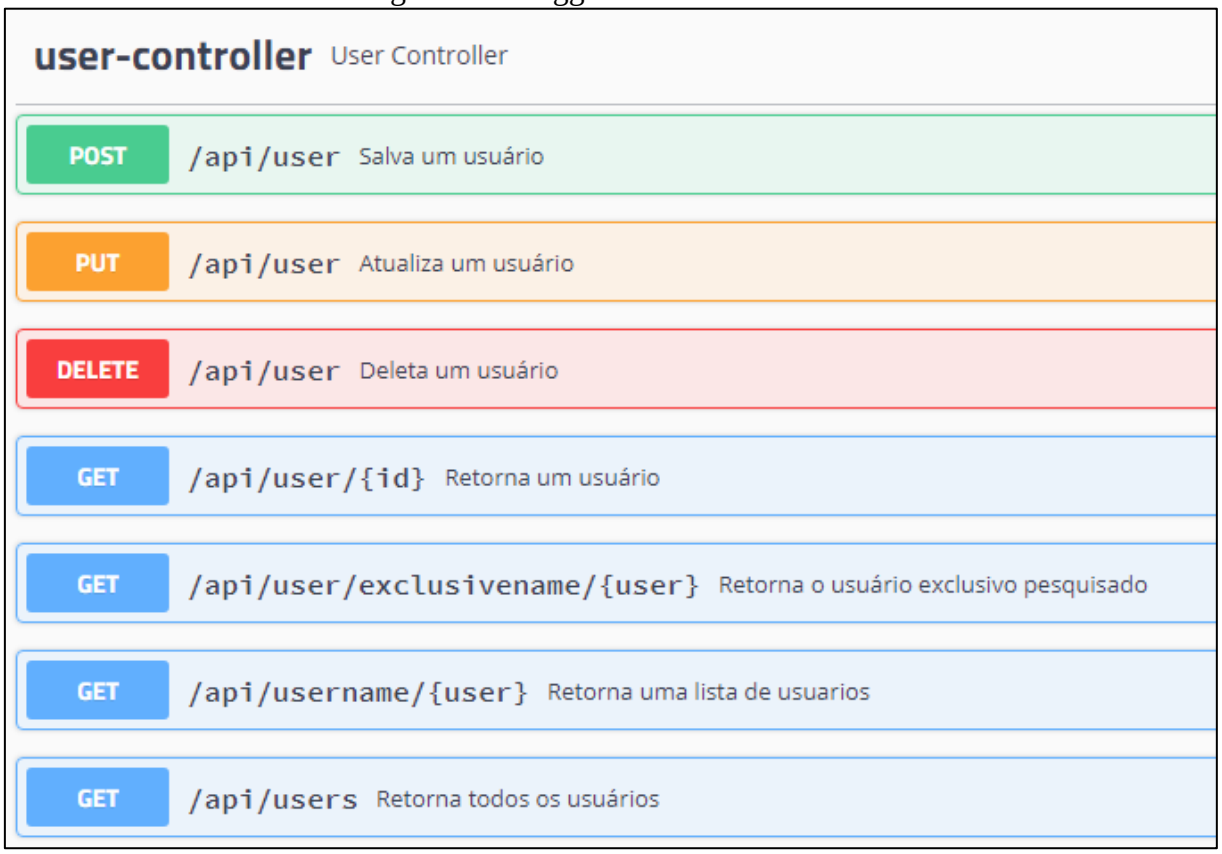
```java
@RestController
@RequestMapping(value = "api")
@Api(value = "API usuário")
@CrossOrigin(origins = "*")
public class UserController {

    @Autowired
    UserRepository userRepository;

    @GetMapping("/users")
    @ApiOperation(value = "Retorna todos os usuários")
    public List<UserModel> allUsers() { return userRepository.findAll(); }
```

Fonte: Digitalizado pelo autor.

Figura 9 - Swagger da API do usuário



Fonte: Digitalizado pelo autor.

A camada `Controller`, como citado anteriormente, disponibiliza *endpoints* para que a aplicação seja acessada. Ela comunica-se com a camada `Repository`, responsável por realizar toda a interação com o banco de dados. A camada `Repository` é composta por interfaces que estendem da interface `JpaRepository` e utilizam o conceito de Java Persistence API (JPA),

pois realiza o mapeamento da classe e realiza a persistência no banco de dados. Para encontrar um registro pela chave primária, ou para retornar todos os registros da tabela, a própria interface do `JpaRepository` já consegue realizar esses procedimentos. Porém, é necessário que os parâmetros genéricos da classe sejam passados corretamente. Ao declarar que a interface se estende do `JpaRepository`, é necessário passar como parâmetro a tabela e o tipo de dado da chave primária da tabela. Portanto, para buscar dados mais específicos é necessário executar um Structure Query Language (SQL) com base no que o usuário deseja buscar conforme o Quadro 5.

Quadro 5 - Classe UserRepository

```
public interface UserRepository extends JpaRepository<UserModel, String> {
    @Query("SELECT u FROM UserModel u WHERE UPPER(u.exclusive_user_name) like %:exclusiveUserName%")
    Optional<List<UserModel>> findByExclusiveUserName(@Param("exclusiveUserName")String exclusiveUserName);

    @Query("SELECT u FROM UserModel u WHERE UPPER(u.exclusive_user_name) = UPPER(:exclusiveUserName)")
    Optional<UserModel> findUserOneExclusiveUserName(@Param("exclusiveUserName")String exclusiveUserName);

}
```

Fonte: Digitalizado pelo autor.

O desenvolvimento *mobile* foi feito na linguagem de programação Dart utilizando o *framework* Flutter, no editor de código Visual Studio Code. Para a implementação da autenticação, utilizou-se o Firebase Authentication, que torna mais fácil a parte da autenticação e bastante segura, porque a senha do usuário fica armazenado neste serviço. Sendo assim, este tipo de dado sensível não pode ser visualizado no banco de dados e em nenhum outro lugar. A comunicação com o Firebase Authentication é feita utilizando o protocolo HTTP. Para que o Firebase Authentication interaja com a aplicação e consiga logar ou criar um usuário persistindo com os dados da aplicação, ele retorna um objeto de usuário do Firebase. Por meio dele é possível obter o atributo User Identifier (UID), independentemente se uma conta está sendo criada ou se ela está sendo logada. O aplicativo então irá validar esta informação com o campo UID da tabela `tb_user`, e caso seja um cadastro, irá criar um usuário com este UID retornado pelo Firebase. Tanto o UID do usuário no banco de dados quanto o UID do usuário no Firebase devem possuir o mesmo valor. Para que o Firebase aceite as requisições feitas pela aplicação, faz-se necessário criar um projeto no site oficial. No final da criação do projeto é fornecido um arquivo chamado *google-services.json* e o mesmo deve ser incluído no projeto. Ademais, é necessária a instalação dos pacotes do Firebase no Flutter para que seja possível fazer a leitura deste arquivo e ter acesso as funcionalidades. A implementação para criar um usuário no Firebase, pode ser vista no Quadro 6. Quando um usuário é cadastrado, é possível visualizar o seu cadastro no painel do Firebase Authentication conforme a Figura 10

Quadro 6 – Implementação de criação de conta com Firebase Authentication

```dart
Future<User> signInFirebase(String email, String password) async {
  try {
    return (await FirebaseAuth.instance
            .signInWithEmailAndPassword(email: email, password: password))
        .user;
  } catch (error) {
    Get.defaultDialog(
      title: "Erro",
      middleText: "Ocorreu um erro ao tentar realizar o login ${error.code}",
      confirm: MaterialButton(
        child: Text("Ok"),
        color: Get.theme.primaryColor,
        onPressed: () => Get.back(),
      ), // MaterialButton
    );
    return null;
  }
}
```

Fonte: Digitalizado pelo autor.

Figura 10 - Painel dos usuários cadastrados no Firebase

Fonte: Digitalizado pelo autor.

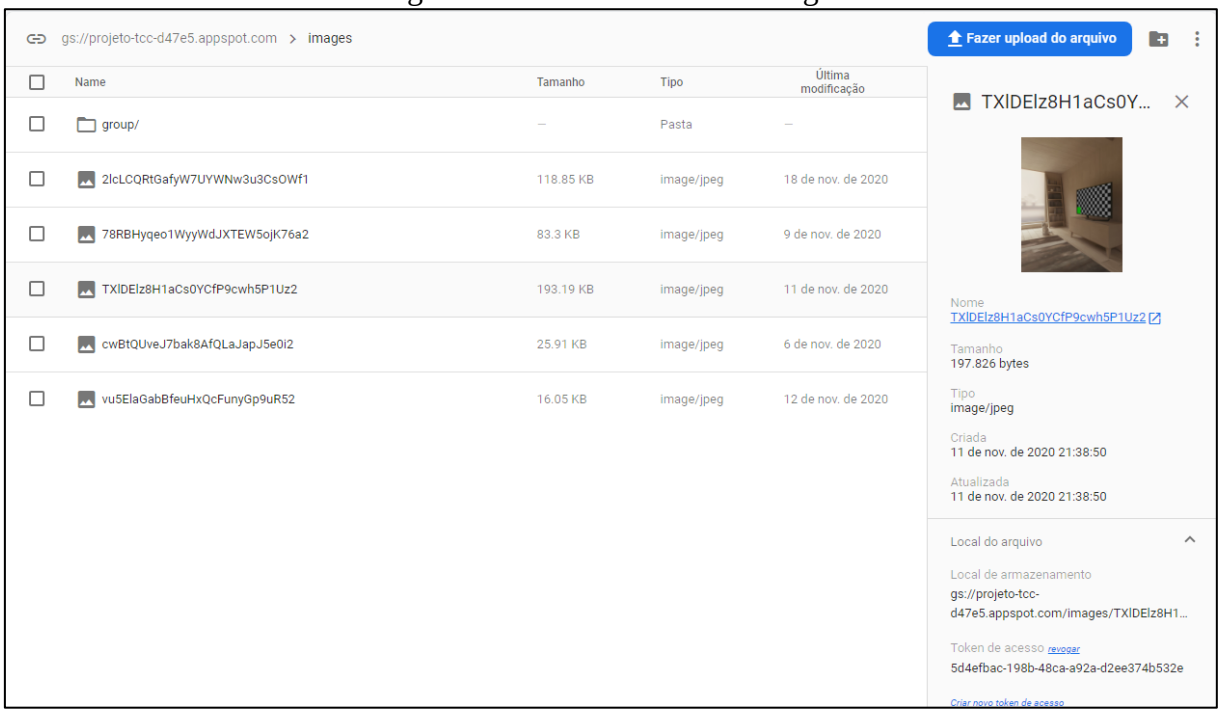
Dentre as funcionalidades do aplicativo, estão inclusas a alteração de uma imagem, seja ela para `foto de perfil`, `foto do grupo` ou `anexo de pagamento`. Para essa funcionalidade, foi utilizado o Firebase Storage, pois ele possibilita que seja feito o upload da imagem para um diretório na nuvem e que essa imagem seja acessível pela aplicação. Esta abordagem foi adotada, pois deixa o banco de dados mais leve sem a necessidade de conversão do arquivo para um outro formato para que possa ser salvo. O Quadro 7 apresenta a implementação da rotina de upload de um arquivo e retornando a URL gerada no Firebase Storage. A Figura 11 demonstra o arquivo na nuvem. Visualiza-se alguns metadados e a opção para obter o seu link para acessar de forma externa, pois trata-se da forma como a aplicação faz para demonstrar a imagem nas telas.

Quadro 7 – Implementação de upload de arquivo para o Firebase Storage

```
static Future<String> uploadImageFirebase(File image, String path) async {
  return await FirebaseStorage.instance
      .ref()
      .child(path)
      .putFile(image)
      .then((data) async {
    return await data.ref.getDownloadURL().then((value) {
      return value;
    });
  });
}
```

Fonte: Digitalizado pelo autor.

Figura 11 - Painel Firebase Storage



Fonte: Digitalizado pelo autor.

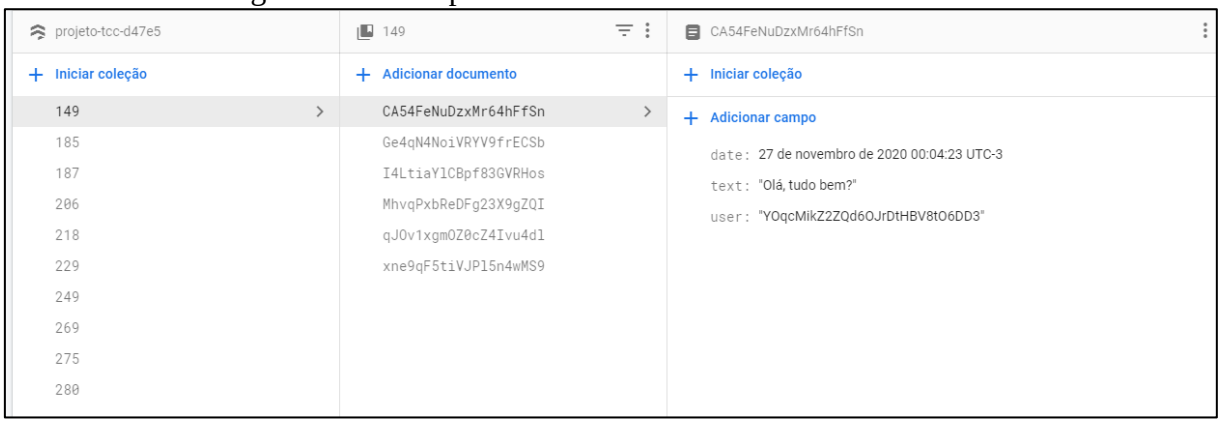
Para a implementação do *chat*, que possui uma interação em tempo real entre os usuários, foi utilizado o Firebase Cloud Firestore que é um banco de dados não relacional. Utiliza-se de uma estratégia de guardar os dados dentro de coleções e documentos. Uma forma bem similar ao conceito de diretório e arquivo dos sistemas operacionais, em que a coleção é o diretório e o documento é o arquivo. Isto está representado na Figura 12. Foi utilizada essa tecnologia, pois ela trabalha com *streams* que são atualizadas em tempo real sempre que algum dado é alterado no banco de dados, facilitando assim o desenvolvimento do *chat*. A parte da implementação de envio de dados está representado no trecho de código no Quadro 8.

Quadro 8 - Implementação de envio de dados para o Firebase Cloud Firestore

```
void sendMessage(String text) {
  if (text.isNotEmpty) {
    FirebaseFirestore.instance
        .collection("group")
        .firestore
        .collection("${this.group.value.id}")
        .add({"user": this.user.uid, "date": DateTime.now(), "text": text});
  }
}
```

Fonte: Digitalizado pelo autor.

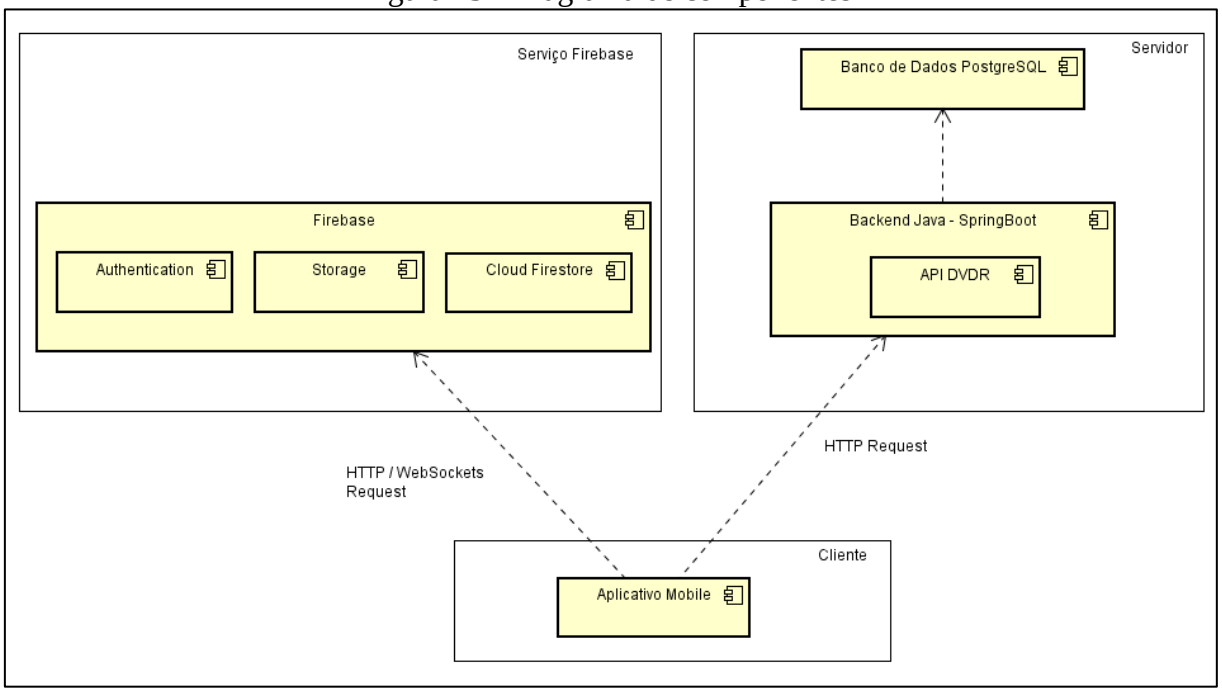
Figura 12 - Exemplo dos dados no Firebase Cloud Firestore



Fonte: Digitalizado pelo autor.

Conforme citado nessa subseção, a aplicação utiliza-se de vários recursos do serviço Firebase e do *backend* da aplicação. Desta forma, para demonstrar a arquitetura utilizada no desenvolvimento da solução, foi modelado o diagrama de componentes conforme a Figura 13.

Figura 13 - Diagrama de componentes



Fonte: Elaborado pelo autor.

3.3.2 Operacionalidade da implementação

Esta subseção deve apresentar as funcionalidades que o aplicativo possui e mostrar as telas que foram criadas. Ao iniciar a aplicação, o usuário terá acesso a tela de `Login` e a aba de `cadastre-se`. Na tela de `Login` o `usuário` que já possuir cadastro deve preencher os campos de `e-mail` e `senha`. O botão de `LOGIN` só será habilitado, caso o usuário preencher um `e-mail` válido e uma `senha` com mais de 6 caracteres. A Figura 14 representa a tela de login.

Figura 14 - Tela de login

Fonte: Elaborado pelo autor

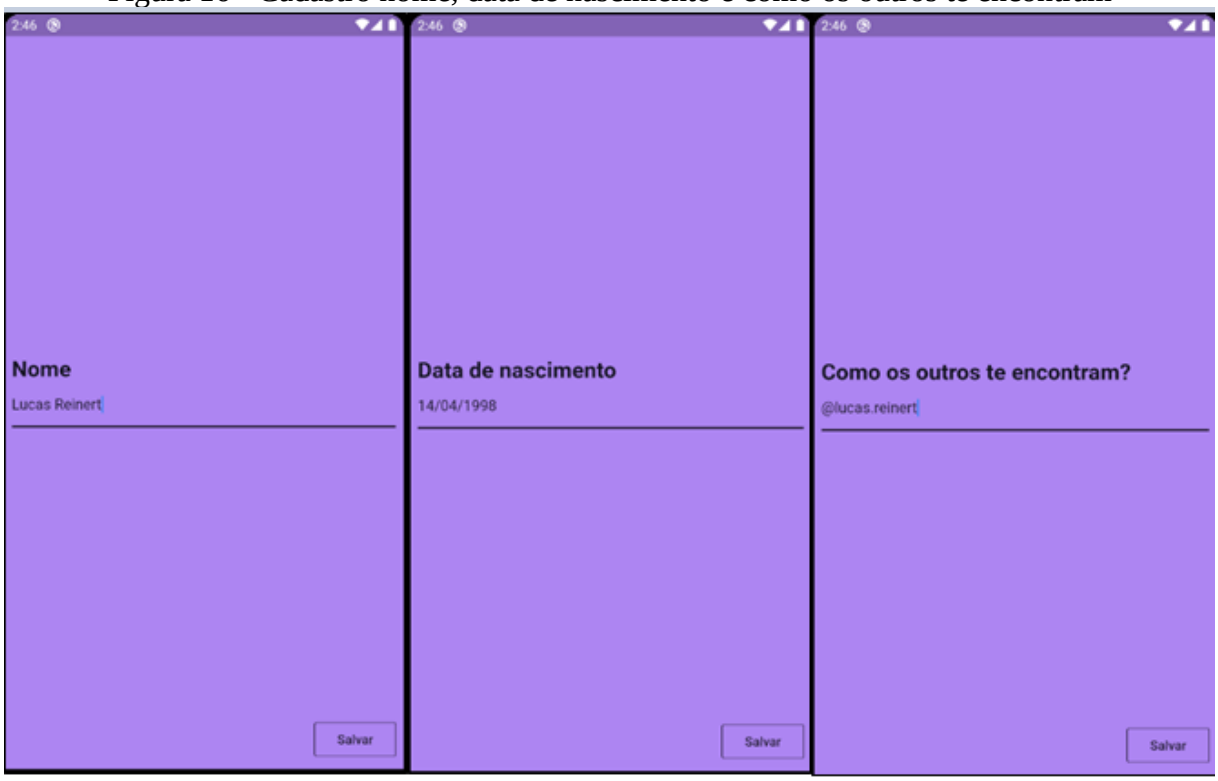
Pela tela de `Login` também é possível acessar a tela `cadastre-se`. Nessa tela o `usuário` deve informar um e-mail válido, que não tenha sido utilizado anteriormente no aplicativo e uma senha com no mínimo 6 caracteres. A Figura 15 representa a tela de cadastramento do aplicativo. Além disso, é possível redefinir a `senha` caso o `usuário` desejar. Para isso basta informar o e-mail e por conseguinte clicar em `Redefinir senha`. Um `e-mail` será enviado para que o `usuário` possa realizar a alteração da `senha`.

Figura 15 - Tela de cadastro de usuário

Fonte: Elaborado pelo autor

Caso o usuário esteja logando no aplicativo pela primeira vez, ele será submetido a algumas perguntas para completar o cadastro. Os campos que devem ser preenchidos serão os seguintes: `nome`, `data de nascimento` e `como os outros se te encontram`. A solicitação dos campos por parte do aplicativo é representada na Figura 16.

Figura 16 - Cadastro nome, data de nascimento e como os outros te encontram



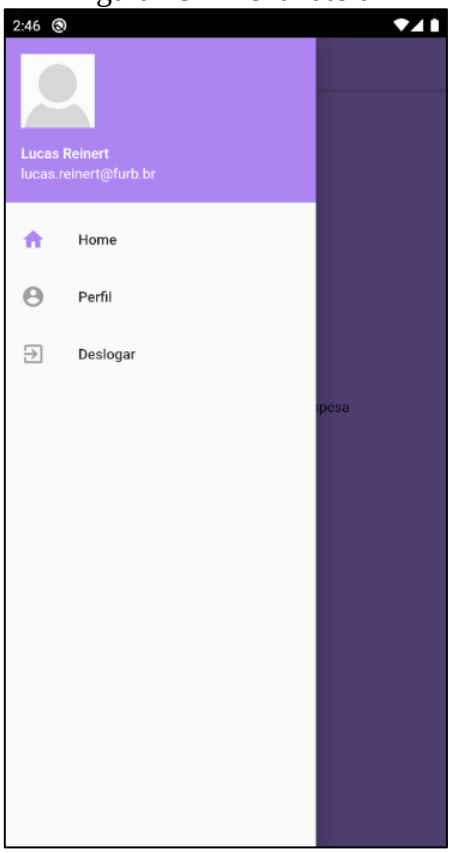
Fonte: Elaborado pelo autor.

Após o preenchimento de todos os campos solicitados, o aplicativo irá direcionar o usuário para a `tela inicial` que está representada na Figura 17. Nesta tela, o usuário conseguirá visualizar todo o seu histórico de gastos no aplicativo e terá a sua disposição dois botões na barra inferior, um deles chamado de `Meus contatos`. Este botão deve direcionar o usuário para a tela que lista os `contatos`. Existe também a possibilidade de acesso ao botão `Meus grupos`, que direciona o usuário para uma tela que lista os `grupos` entre os quais o usuário logado faz parte. Além disso, conforme a Figura 18, o usuário tem a sua disposição no topo superior esquerdo da tela, um menu que ao ser clicado mostra a sua `foto de usuário`, `nome` e `e-mail`, e a opção de ser direcionado para a `Home`, `Perfil` ou `Deslogar`.

Figura 17 - Tela inicial

Fonte: Elaborado pelo autor.

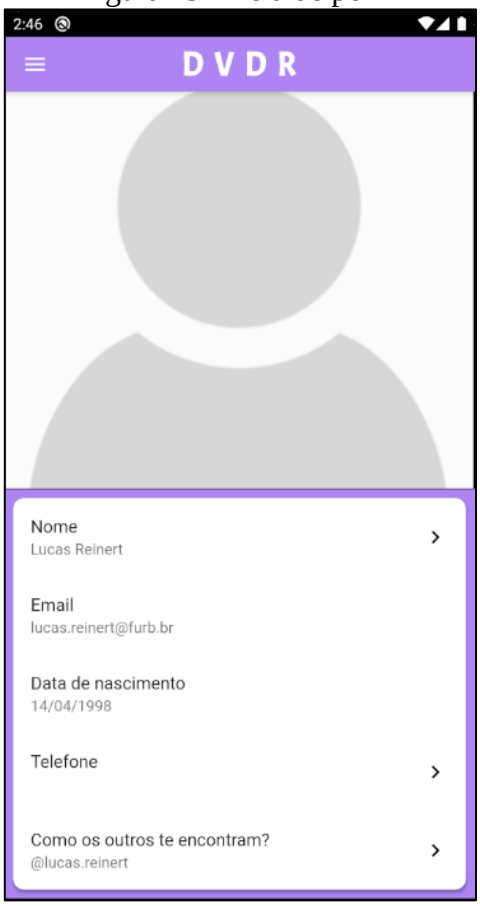
Figura 18 - Menu lateral



Fonte: Elaborado pelo autor.

Quando acessar o `Perfil`, o usuário pode alterar a `foto de perfil` com as fotos que estão na galeria do *smartphone* e alterar o `nome`, `telefone` e o `como que os outros te encontrem`. As informações que serão alteradas nessa tela ficarão disponíveis para que outros usuários visualizem posteriormente. A Figura 19 representa a tela do `perfil de usuário`, em que aparecem todas as características. Nesta figura vê-se que alguns itens possuem uma seta para o lado direito. Esta seta representa que esse registro pode ser editado e os demais registros devem permanecer conforme o cadastro no aplicativo.

Figura 19 - Tela de perfil



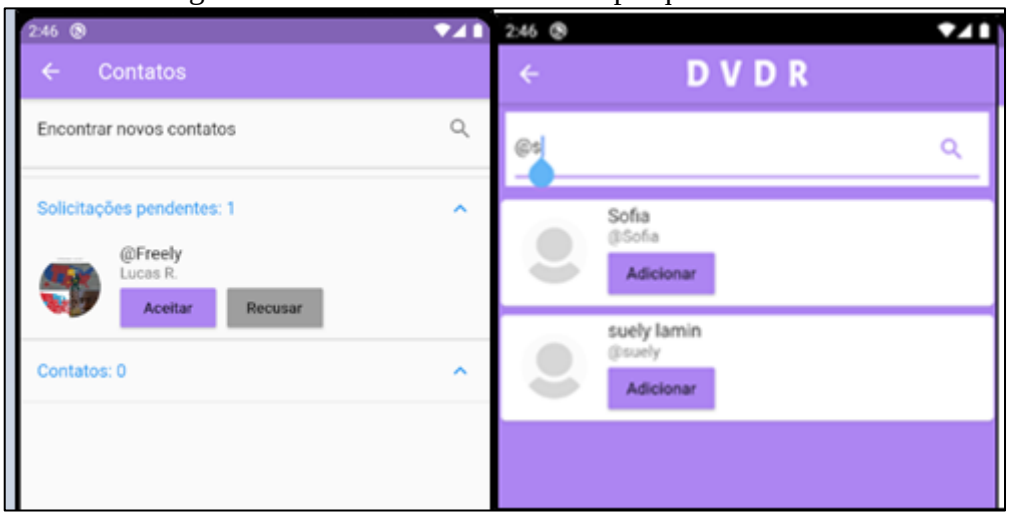
Fonte: Elaborado pelo autor.

A tela de `contatos` retratada na primeira parte da Figura 20 lista todos os contatos já existentes e solicitações pendentes que foram enviadas por outros usuários. Por meio desta tela é possível aceitar ou recusar uma solicitação que esteja pendente, ou remover um contato já existente. Essa funcionalidade é de suma importância, pois quando um grupo for criado o usuário irá adicionar os seus próprios contatos no grupo.

Existe também a possibilidade de o usuário pesquisar por novos contatos. Para isso basta ele pesquisar uma parte do nome exclusivo que o usuário cadastrou no campo `como os outros te encontram`, de um contato que ele deseja adicionar e uma lista será carregada com todos os usuários que possuem um nome exclusivo que contenha os caracteres digitados na barra de

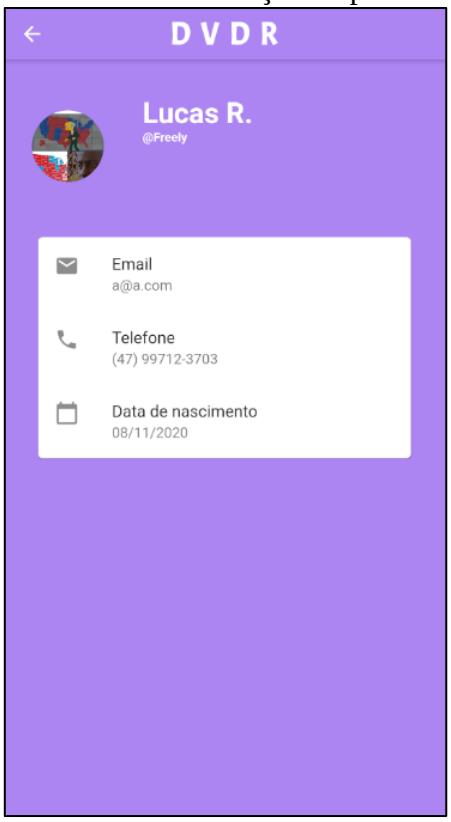
pesquisa conforme o exemplo da segunda parte da Figura 20. Nessa lista de pesquisa, para os usuários que não possuírem um vínculo de amizade com o usuário logado, deve aparecer um o botão `Adicionar`. Se o usuário logado já tiver enviado uma solicitação de amizade, deve aparecer o botão `Cancelar solicitação`, quando o usuário listado ter enviado uma solicitação irá aparecer o botão `Aceitar solicitação`. Por fim, quando o usuário listado e o usuário logado já forem um contato, nenhum botão deve aparecer. Pode-se visualizar as informações de perfil dos usuários clicando sobre eles. A Figura 21 representa esse caso de uso.

Figura 20 - Tela lista de contatos e pesquisar contatos



Fonte: Elaborado pelo autor.

Figura 21 - Tela de visualização de perfil de usuário

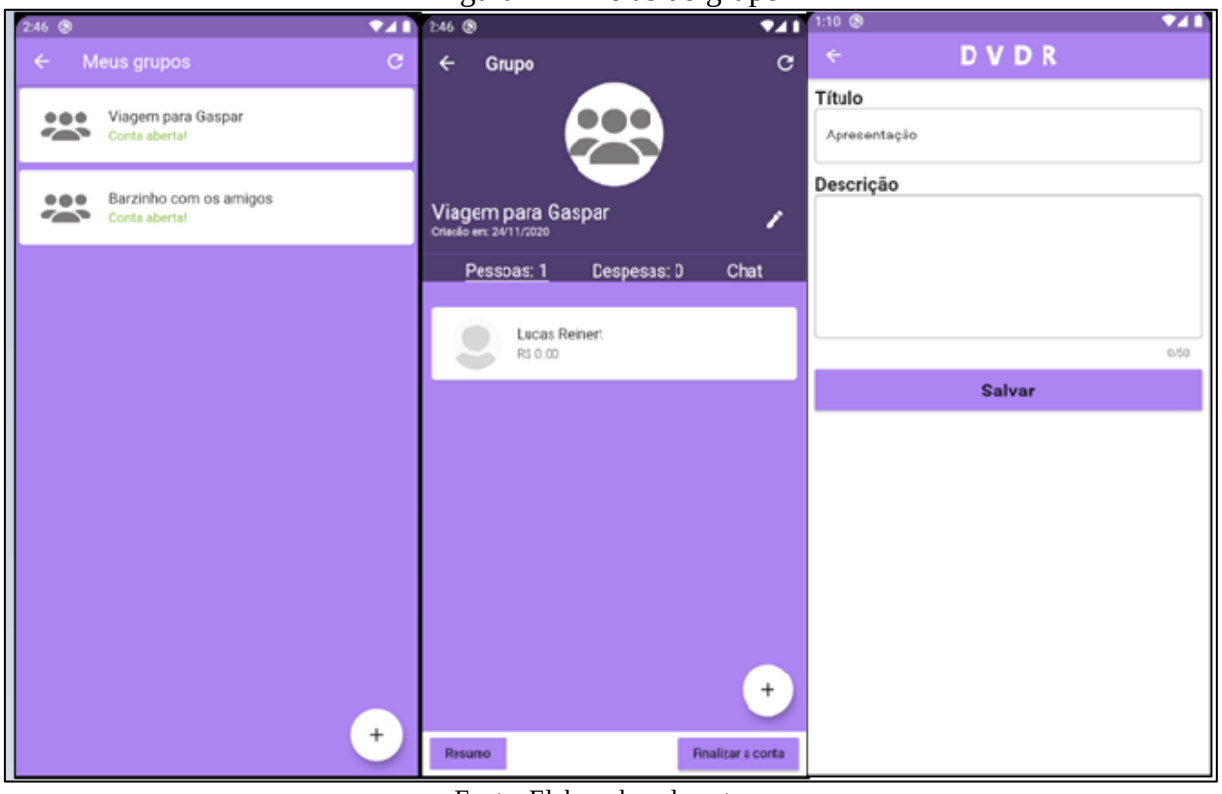


Fonte: Elaborado pelo autor.

A tela de `listagem de grupos`, deve listar todos os `grupos` em que o usuário logado está inserido. Por meio desta tela, torna-se possível criar grupos ou abrir algum `grupo` no qual o usuário está inserido. A primeira parte da Figura 22 apresenta a tela conforme as funcionalidades descritas.

Na tela do respectivo grupo, o usuário tem a opção de editar o nome e descrição, adicionar pessoas, adicionar despesas e conversar com os demais integrantes via *chat*. Para alterar o nome e descrição do grupo, basta o usuário clicar no ícone do lápis, localizado acima da palavra *chat* na segunda parte da Figura 22. A terceira parte da Figura 22 apresenta a tela para edição de título e descrição do grupo. Para editar o usuário pode simplesmente clicar sobre os campos, digitar o texto que deseja e clicar em salvar. Ainda de acordo com a segunda parte da Figura 22, pode-se alterar a foto do grupo, ao clicar sobre a imagem circular posicionada acima do título. Dessa forma, o aplicativo deve pedir permissão ao usuário para abrir a galeria de fotos do dispositivo. Após o usuário permitir a utilização, o aplicativo irá direcionar o usuário para a galeria de imagens do celular e ele irá escolher a foto que deseja colocar no grupo.

Figura 22 – Telas do grupo

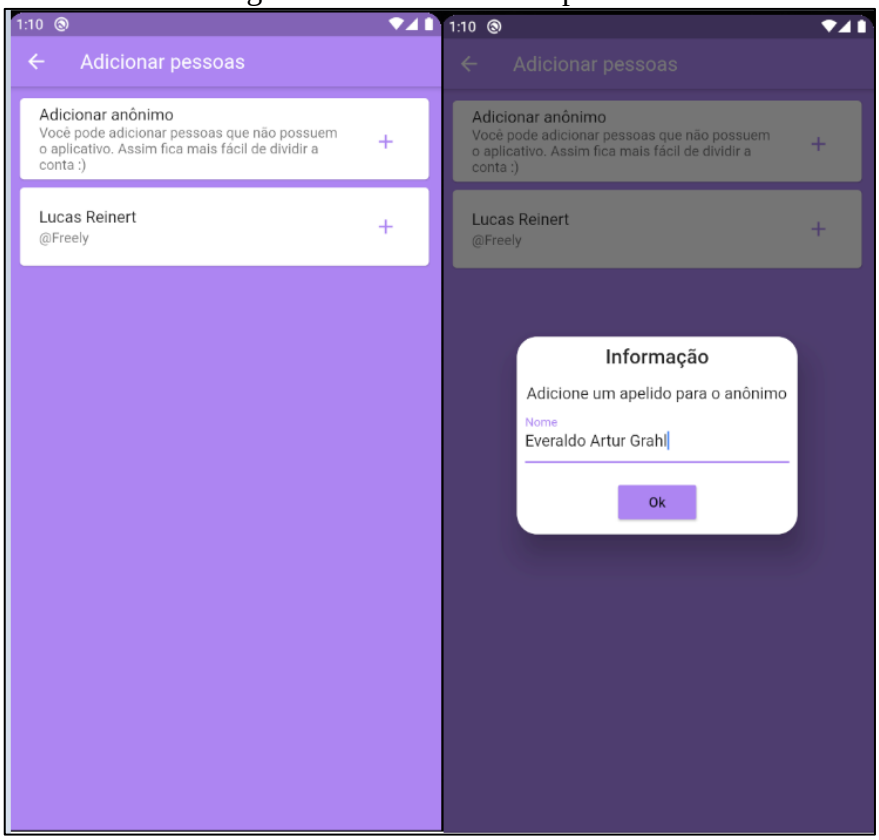


Fonte: Elaborado pelo autor.

Para realizar o rateio das despesas destaca-se como importante que o administrador do grupo adicione as pessoas cadastradas que irão participar da divisão do valor das despesas. Para adicionar novas pessoas ao grupo, basta o usuário permanecer na aba `Pessoas` conforme a primeira parte da Figura 22 e posteriormente clicar no ícone de "`+`" localizado no lado direito

inferior da tela. Após isso, será apresentado para o usuário a tela da primeira parte da Figura 23, que fornece a opção de adicionar um usuário anônimo. Este usuário anônimo existe para que não seja necessário que todas as pessoas que irão participar do rateio precisem necessariamente ter o aplicativo instalado. O administrador pode incluir pessoas sem a aplicação por meio desta opção e adicionar um apelido para elas, conforme a segunda parte da Figura 23.

Figura 23 - Tela adicionar pessoas



Fonte: Elaborado pelo autor.

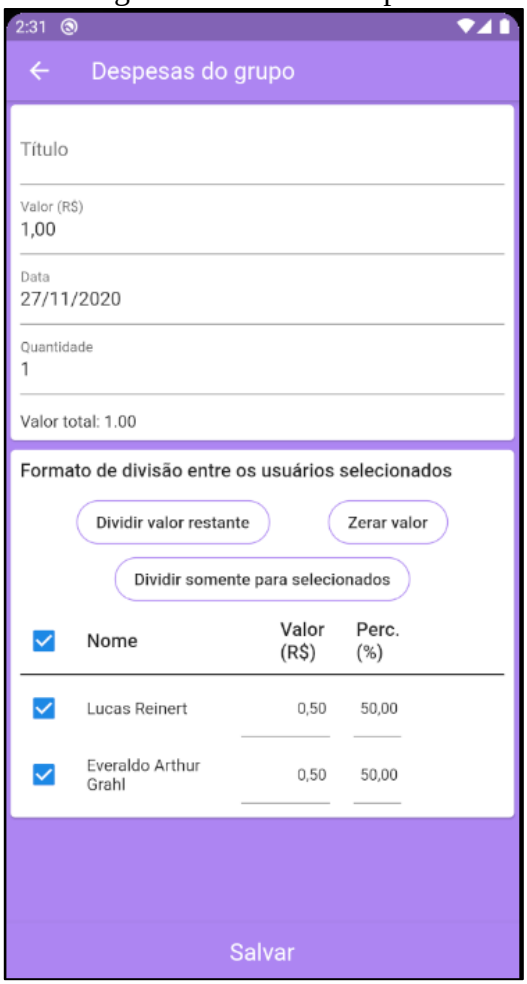
Após adicionar um novo `usuário` para ratear os valores, é necessário criar uma `despesa` para realizar o `rateio`, para isso basta voltar para a tela do `grupo` e ir na aba de `despesa`. Então deve-se clicar no botão com o ícone de + localizado no lado inferior direito da aplicação, assim abrindo a tela de `despesas` para criar uma `despesa`. Nessa tela o `usuário` deve informar para um `título`, `valor`, `data` e `quantidade`. O aplicativo calcula o `valor total` automaticamente baseado no `valor` e `quantidade` informados, utilizando a fórmula valor x quantidade.

Na parte inferior da tela é possível visualizar o formato de divisão entre os usuários selecionados. Nessa parte da tela será realizado o rateio das despesas. Se o usuário desejar atribuir um percentual para cada participante do grupo, pode-se utilizar a coluna Perc. (%) e atribuir o percentual que deseja para cada participante. Dessa mesma forma funciona a coluna

Valor (R$), o administrador do grupo ou criador da despesa pode colocar nesse campo o valor que desejar.

Além disso, o aplicativo pode auxiliar a ratear os valores entre os integrantes do grupo utilizando os botões: `Dividir valor restante`, `zerar valor` e `dividir somente para selecionados`. Estes botões utilitários possuem como objetivo levar em consideração os usuários que têm o seu *checkbox* marcado. O botão `Dividir valor restante` vai realizar o seguinte cálculo (valor total – soma dos usuários não selecionados) / quantidade de usuários selecionados. O resultado deste cálculo será atribuído para todos os usuários que estão com o seu *checkbox* marcado, além disso o percentual também deve alterar de acordo com o valor. O botão `Zerar valor`, deve zerar o valor de todos os usuários que estão com o *checkbox* marcado. Por fim, o botão `Dividir somente para selecionados` deve fazer o seguinte cálculo: (valor total / quantidade de usuários selecionados) e assim atribuir o resultado para todos os usuários que estão selecionados. Após realizado todo o rateio da despesa, basta o usuário clicar em salvar na parte inferior da tela. A Figura 24 representa toda a tela de despesa e rateio de despesa.

Figura 24 - Tela de despesa



Fonte: Elaborado pelo autor.

Existe a possibilidade de os usuários pertencentes ao grupo interagirem entre si através de um *chat*. Na aba *chat* os usuários podem conversar enviando mensagens de texto e imagens. Para enviar uma mensagem basta clicar na caixa de texto, digitar a mensagem desejada e clicar na flecha ao lado da caixa de texto. Enviar uma imagem torna-se possível através do ícone de câmera, ao clicar o aplicativo deve solicitar permissão para acessar a galeria do dispositivo. Assim que o usuário aceitar a solicitação o aplicativo vai direcioná-lo para a galeria do seu celular para selecionar a imagem desejada. Sendo assim, quando a imagem for selecionada, deve ser enviada para o chat do grupo conforme a Figura 25.

Figura 25 - Tela de chat

Fonte: Elaborado pelo autor.

3.4 RESULTADOS E DISCUSSÕES

Nesta seção são apresentados os resultados obtidos. Na subseção 3.4.1 é apresentado um comparativo entre os trabalhos correlatos e o aplicativo DVDR. Na subseção 3.4.2 são descritos os resultados da pesquisa com usuários.

3.4.1 Comparativo dos trabalhos correlatos

O Quadro 9 apresenta o comparativo entre as aplicações citadas na subseção 2.4 e o trabalho proposto.

Quadro 9 - Comparativo entre trabalhos correlatos e trabalhos proposto

Correlatos / Características	Splitwise (2019)	Evenfy (2019)	Oasys (2013)	DVDR (2020)
Plataformas da aplicação.	Web/*Mobile* (Android/IOS)	Web/*Mobile* (Android/IOS)	*Mobile* (IOS)	*Mobile* (Android)
Rateio de despesas.	Sim	Sim	Sim	Sim
Permite a criação de grupos.	Sim	Sim	Não	Sim
Possibilita a troca de mensagens.	Não	Sim	Não	Sim
Possibilita envio de imagens no chat.	Não	Não	Não	Sim
Interação de todo grupo de usuário com as despesas.	Não	Sim	Não	Sim
Comprovar o pagamento	Sim	Sim	Não	Sim

Fonte: elaborado pelo autor.

Conforme é apresentado no Quadro 9, as aplicações Splitwise (2019), Evenfy (2019), Oasys (2013) e DVDR (2020), tem como foco principal o rateio de despesas, agregando formas diferentes para a realização do cálculo. Cada aplicação possui suas próprias particularidades para atenderem o problema proposto.

A aplicação de Splitwise (2019) e Evenfy (2019), apostam em uma expansão de plataformas para acessarem suas aplicações, pois em ambas é possível o acesso por dispositivos móveis com sistema operacional iOS ou Android, e acesso ao ambiente *web* por meio dos navegadores. O aplicativo de Oasys (2013), possui a sua versão somente *mobile* e limitado para o sistema operacional iOS. Já a aplicação DVDR (2020) está disponível apenas na plataforma *mobile*, para usuários que tenham instalado em seus dispositivos o sistema operacional Android.

A funcionalidade de criação de grupos está presente nas aplicações Splitwise (2019), Evenfy (2019) e DVDR (2020). Na aplicação de Splitwise (2019), os grupos já ficam dispostos direto na tela inicial que já prove ao usuário o botão de criação ou visualização de um grupo já existente. Para adicionar um contato a um grupo é necessário vincular o mesmo a lista de contatos do aparelho celular, ou criar um usuário colocando o nome e o e-mail. Sendo assim, a aplicação deve mandar um e-mail de confirmação para esse usuário, não sendo possível adicionar usuários que não tenham o aplicativo. A aplicação de Evenfy (2019) trata os grupos com a nomenclatura de evento. Para adicionar novas pessoas aos eventos, pode-se criar um participante e futuramente vincular este participante a um usuário através de um link que pode ser enviado para as partes que desejarem fazer parte. A aplicação DVDR, possui uma tela específica para os grupos, através do Menu de grupos. Para adicionar pessoas aos grupos, é

necessário que previamente tenha sido relacionada uma amizade pelo aplicativo, ou adicionar usuários anônimos, para que não seja necessário o download da aplicação por todos os usuários.

Para facilitar a comunicação entre usuários de um grupo e auxiliar no alinhamento de quais despesas adicionar, as aplicações Evenfy (2019) e DVDR (2020) possibilitam uma comunicação por chat. A aplicação DVDR possui um diferencial, pois além de permitir envio de mensagens de texto, possibilita também o envio de imagens no grupo. Isso melhora a comunicação do grupo e pode auxiliar os usuários a manter um histórico de conversas e fundamentos para as despesas que foram adicionadas. Além dos usuários poderem interagir por chat, as aplicações Evenfy (2019) e DVDR (2020) permitem que os usuários consigam cadastrar suas despesas. A aplicação Evenfy (2019), se limita a permitir somente o usuário que criou a despesa, ter autorização para alterar a mesma. Enquanto a aplicação DVDR (2020), permite que o usuário que criou a despesa e o administrador do grupo possam interagir na edição de um mesmo registro.

A comprovação de pagamento, é fornecida pelas aplicações Splitwise (2019), Evenfy (2019) e DVDR (2020). A comprovação de um pagamento na aplicação de Evenfy (2019), possui o pré-requisito de encerrar um evento. Quando um evento está encerrado, a aplicação fornece da aba de pagamentos a opção para o administrador marcar os usuários de quem recebeu. Na aplicação de DVDR (2020), o pré-requisito de encerrar o grupo também é necessário. Portanto, os integrantes do grupo têm a possibilidade de anexar um documento e uma observação acerca do pagamento que foi realizado. O administrador do grupo irá validar os pagamentos, e pode atribuir como pago os usuários que já tiverem feito o pagamento, não sendo necessário que os integrantes tenham colocado uma observação ou anexo, pois isso é opcional. A aplicação Splitwise (2019), possui uma abordagem um pouco diferente para o pagamento das despesas, pois apesar de estarem em um grupo, o pagamento das despesas é feito do(s) usuário(s) X para um usuário Y. Portanto, não é possível o usuário que recebeu o pagamento confirmar que foi recebido, apenas o usuário que pagou informa que foi realizado o pagamento. Além disso, não é necessário que o grupo seja encerrado para que um pagamento seja realizado.

3.4.2 Resultados

Segundo Barboza (2019) uma premissa de todo produto seria atender as necessidades do usuário. No entanto, sempre restam dúvidas se isso estará, de fato, acontecendo. A partir disto, os resultados obtidos a partir do tema proposto neste presente trabalho foram aplicados em forma de pesquisa. Através de um formulário, foram desenvolvidas dez perguntas no total,

em que houve participação de oito pessoas com idades que variam entre 16 e 31 anos. As perguntas deste formulário foram baseadas no método SUS, uma métrica de avaliação da usabilidade de sistemas computacionais. Para formular e validar esta pesquisa, foi disponibilizada uma versão do aplicativo *mobile* e o link para acessar o questionário. Os participantes puderam testar a usabilidade do aplicativo no tempo que fosse necessário antes de responder as questões do formulário.

Ao analisar as respostas contidas no Apêndice C, observa-se que a partir dos resultados do questionário, em geral o aplicativo foi bem avaliado, sendo que todos os avaliadores tiveram experiências positivas e, portanto, em sua maioria não tiveram sugestões de melhorias significativas que alterassem seu escopo inicial. Verifica-se que os pontos de maior destaque se concentraram na boa integração de todas as funções do aplicativo, obtendo a pontuação máxima, bem como também em sua maioria, os participantes de acordo com o questionário não acharam que o aplicativo apresenta inconsistências.

Todos os participantes que avaliaram a aplicação, possuem pelo menos uma experiência mínima com o aplicativo, e que 12,5% das pessoas já utilizaram um aplicativo semelhante para o rateio de despesas. Pode-se colocar como destaques positivos, a facilidade no uso da aplicação que obteve a pontuação média 3,875. A mesma pontuação média foi para o item que se refere a boa integração do sistema. O item que mais se destacou e obteve a pontuação máxima era sobre se o sistema não é atrapalhado de usar. Os itens que se destacam com menor média foram os que se referem ao sistema apresentar inconsistências, pois obteve uma pontuação média de 3. O item, que se refere as pessoas que gostariam de utilizar esse sistema com frequência obteve a pontuação média de 3,125.

Por meio dos resultados coletados, pode-se dizer que a aplicação se caracteriza por ser de fácil memorização, pois independentemente do tempo que leve para o usuário experimentá-lo novamente, seu design intuitivo proporciona agilidade para executar tarefas que normalmente, de modo manual, iriam ser mais demoradas e imprecisas.

Em um contexto geral, avaliando-se os resultados obtidos por este questionário, a aplicação alcançou a pontuação 90,93 na escala do SUS. Segundo Barboza (2019) pontuações acima de 90 representam a melhor usabilidade possível.

4 CONCLUSÕES

Este trabalho apresentou o aplicativo para divisão de despesas intitulado DVDR, que auxilia pessoas de um grupo dividirem suas despesas, buscando a divisão de forma adequada. O aplicativo apresenta uma experiência bastante similar de uma rede social, pois permite o usuário adicionar contatos, visualizar o perfil de cada um deles, adicionar em um grupo e conversar via chat para uma melhor interação entre cada participante do grupo.

Os objetivos específicos foram cumpridos. A aplicação permite que o usuário crie uma despesa e faça o rateio, pois para cada participante, é possível parametrizar que percentual ele irá pagar de uma despesa, ou qual o valor real que ele vai pagar, ou caso o usuário queira ratear de uma forma automatizada, é possível utilizar as opções disponíveis. Quando nenhuma despesa deve ser mais inclusa e o pagamento deve ser feito, o administrador do grupo tem a possibilidade de finalizar a conta, sendo assim, os integrantes podem começar a anexar os seus comprovantes de pagamento e observações. O administrador irá verificar e confirmar o pagamento de cada usuário, agregando assim mais segurança a aplicação. Para entender quanto que cada usuário está gastando ou deverá pagar em um grupo, o aplicativo mostra em tempo real ao lado do nome de cada integrante, quanto que o mesmo deve pagar no total das despesas, acessando as despesas individualmente. Também é possível visualizar quanto que cada usuário deverá pagar por uma despesa específica.

O aplicativo DVDR tem como maior diferencial a forma como faz o gerenciamento dos contatos. O procedimento para adicionar um contato e interagir com ele é realizado pelo próprio aplicativo.

O aplicativo possui algumas limitações na questão de plataformas, pois só é possível o acesso por um smartphone com o sistema operacional Android. Além disso o aplicativo se limita na questão de pagamento de contas, pois não fornece nenhum método utilizando token de pagamento, apenas o usuário informa que pagou e o administrador de um grupo confirma esse pagamento.

4.1 EXTENSÕES

Para este trabalho, sugere-se como extensões os seguintes itens:

a) permitir que o pagamento do grupo seja realizado utilizando uma plataforma de pagamento digital;
b) disponibilizar um relatório mostrando gráficos dos gastos do usuário autenticado e das entradas e saídas dos grupos;
c) desenvolver um sistema web que integre com as API’s do sistema atual, aumentando

assim a abrangência da aplicação em relação a plataformas;

d) desenvolver alguma forma de convite para o grupo. Atualmente a única forma de um usuário fazer parte de um grupo é sendo convidado por um administrador.

REFERÊNCIAS

ALLEGRETTI, Sonia M. Macedo et al. Aprendizagem nas redes sociais virtuais: o potencial da conectividade em dois cenários. **Revista contemporaneidad educacion y tecnologia Revista Cet**, v. 1, n. 2, p. 54-60, abr. 2012.

BARBOZA, Anderson. **Medindo a usabilidade do seu produto com System Usability Scale (SUS).** [S.I], 2019. Disponível em: https://medium.com/design-contaazul/medindo-a-usabilidade-do-seu-produto-com-system-usability-scale-sus-3956612d9229. Acesso em: 01 dez. 2020.

DUTRA, René G. Critérios de rateio e distribuição de custos. *In*: I Congresso Brasileiro de Gestão Estratégica de Custos, 11., 1994, São Leopoldo. **Anais Eletrônicos**. São Leopoldo: Associação Brasileira de Custos, 1994. Disponível em: https://anaiscbc.emnuvens.com.br/anais/article/view/3504/3504. Acesso em: 09 nov. 2019.

EVENFY. **Uma forma colaborativa para organizar contas em grupo**. [S.I.], 2019. Disponível em: https://www.contacoletiva.com.br/pt-br/. Acesso em: 09 nov. 2019.

FERRARI, Daniela. **Conta Coletiva facilita a divisão de despesas entre amigos.** [S.I.], 2017. Disponível em: https://www.techtudo.com.br/tudo-sobre/conta-coletiva.html. Acesso em: 09 nov. 2019.

FERREIRA, Michelle. **Aplicativo ajuda a dividir a conta em restaurantes e bares.** [S.I.], 2013. Disponível em: https://epocanegocios.globo.com/Informacao/Visao/noticia/2013/02/aplicativo-ajuda-dividir-conta-em-restaurantes-e-bares.html. Acesso em: Acesso em: 09 nov. 2019.

FLUTTER. **Create faster apps**. [S.I.], 2020. Disponível em: https://flutter.dev/ Acesso em: 19 dez. 2020.

GERALDES, Wendell; MARTINS, Ernane; AFONSECA, Ulisses; **Avaliação da Usabilidade do Scratch utilizando o Método System Usability Scale (SUS).** [S.I], 2019. Disponível em: https://sol.sbc.org.br/index.php/eri-mt/article/view/8589/8490. Acesso em: 01 dez. 2020.

GUTOSKEY, Ellen. **Splitwise App Makes Splitting the Bill for Dinner, a Trip, Household Bills, or Anything Else Easy.** [S.I.], 2019. Disponível em: https://www.mentalfloss.com/article/601591/splitwise-app-makes-splitting-bills-easy. Acesso em: 09 nov. 2019.

KOEHLER, Cristiane. **Interação social em rede e nas redes**. [S.I.], 2016. Disponível em: https://www.lume.ufrgs.br/handle/10183/148300. Acesso em: 26 jan. 2021.

LEWIS, James. The system usability scale: Past, present, and future. **International Journals of Human – Computer Interaction**, v. 18, n. 7, p. 577-590, mar. 2018.

NIELSEN, Jakob. **Usability 101: Introduction to Usability**. [S.I.], 2006. Disponível em: http://www.ingenieriasimple.com/usabilidad/IntroToUsability.pdf. Acesso em: 19 dez. 2020.

OASYS, Innovation. **Passa Régua.** [S.I], 2013. Disponível em: https://apps.apple.com/br/app/passa-r%C3%A9gua/id449074589. Acesso em: 01 dez. 2020.

PIRES, Valdemir. Finanças pessoais fundamentos e dicas. **Piracicaba: Editora Equilíbrio**, 2006.

PREUSS, Juliana. **Conta coletiva, aplicativo revoluciona suas contas.** [S.I.], 2016. Disponível em: https://tambotech.com.br/internet/conta-coletiva-aplicativo/. Acesso em: 09 nov. 2019.

PRICE, Emily. **Make Splitting Trip and Household Expenses Easier With Splitwise.** [S.I.], 2019. Disponível em: https://lifehacker.com/make-splitting-trip-and-household-expenses-easier-with-1838597939. Acesso em: 09 nov. 2019.

RECUERO, Raquel. **Contribuições da Análise de Redes Sociais para o estudo das redes sociais na Internet.** [S.I.], 2014. Disponível em: http://www.revistas.unisinos.br/index.php/fronteiras/article/view/fem.2014.162.01/4191. Acesso em: 26 jan. 2021.

SILVA, Matheus. **O que é melhor, desenvolver uma aplicação móvel nativa, híbrida ou uma web app**. [S.I], 2019. Disponível em: https://medium.com/seedabit/o-que-%C3%A9-melhor-desenvolver-uma-aplica%C3%A7%C3%A3o-m%C3%B3vel-nativa-h%C3%ADbrida-ou-um-web-app-e6cc6fd23173. Acesso em: 19 dez. 2020

SPLITWISE. **Menos estresse ao compartilhar despesas.** [S.I.], 2019. Disponível em: https://www.splitwise.com/. Acesso em: 09 nov. 2019.

STACKOVERFLOW. **About Flutter**. [S.I], 2020. Disponível em: https://stackoverflow.com/tags/flutter/info. Acesso em: 19 dez. 2020.

APÊNDICE A – Descrição dos Casos de Uso

Este Apêndice apresenta a descrição de alguns casos de uso do sistema. O Quadro 10 apresenta a descrição do caso de uso `UC01 - Manter usuário`.

Quadro 10 - Descrição do caso de uso `UC01 - Manter usuário`

UC01 – Manter usuário	
Ator	Usuário
Pré-condições	N/A
Cenário principal	1) O usuário clica em cadastrar-se; 2) O aplicativo direciona o usuário para a tela de cadastro de usuários; 3) O usuário preenche os campos de e-mail e senha; 4) O usuário clica em Cadastrar; 5) O aplicativo cria um usuário no Firebase e no banco de dados relacional; 6) O aplicativo faz a autenticação do usuário após o cadastro; 7) O usuário informa os dados requeridos pela aplicação após o primeiro login; 8) O usuário informa os dados e clica em salvar; 9) O aplicativo irá salvar os dados no banco de dados relacional.
Caminho alternativo	No passo 1, o usuário opta por se autenticar no sistema e editar os seus dados pelo menu “Perfil”.
Cenário de exceção 1	No passo 3 o usuário preenche o campo de e-mail de forma incorreta, tal como, não é um e-mail válido.
Cenário de exceção 2	No passo 3, o usuário preenche o campo de senha com menos de 6 caracteres.
Pós-condição	Usuário é redirecionado para a home do aplicativo.

Fonte: elaborado pelo autor.

O Quadro 11 apresenta a descrição do caso de uso `UC02 - Manter grupos`.

Quadro 11 - Descrição do caso de uso `UC02 - Manter grupos`

UC02 – Manter grupos	
Ator	Usuário
Pré-condições	Estar logado
Cenário principal	1) O usuário acessa o menu “Meus grupos”; 2) O aplicativo apresenta todos os grupos do usuário; 3) O usuário clica no ícone de + no canto inferior direito da tela; 4) O aplicativo salva um grupo no banco de dados.
Pós-condição	O aplicativo direciona o usuário para a tela dos grupos

Fonte: elaborado pelo autor.

O Quadro 12 apresenta a descrição do caso de uso `UC03 - Manter despesas`.

Quadro 12 - Descrição do caso de uso `UC03 - Manter despesas`

UC03 – Manter despesas	
Ator	Usuário
Pré-condições	Estar Logado
Cenário principal	1) O usuário acessa o menu “Meus grupos”; 2) O aplicativo lista os grupos; 3) O usuário seleciona um grupo; 4) O aplicativo direciona para a tela de grupos; 5) O usuário abre a aba de despesas; 6) O usuário clica no botão + no canto inferior direito; 7) O aplicativo direciona o usuário para a tela de despesas 8) O usuário informa título, valor, data, quantidade e quanto que cada integrante deve pagar; 9) O usuário clica em salvar; 10) O aplicativo grava a operação no banco de dados.
Cenário alternativo	No passo 8 o usuário utiliza um dos botões de cálculo automático do valor de cada integrante.
Cenário exceção	No passo 5, o usuário informa uma despesa com quantidade zerada.
Pós-condição	O aplicativo salva as informações no banco de dados

Fonte: elaborado pelo autor.

O Quadro 13 apresenta a descrição do caso de uso `UC05 - Manter contato`.

Quadro 13 - Descrição caso de uso `UC05 - Manter contato`

UC05 – Manter contato	
Ator	Usuário
Pré-condições	Estar logado
Cenário principal	1) O usuário acessa o menu “Meus contatos”; 2) O aplicativo apresenta a tela de contatos do usuário; 3) O usuário vai em “Encontrar novos contatos”; 4) O aplicativo abre a tela de pesquisa de usuários; 5) O usuário pesquisa por um contato que ele deseja adicionar; 6) O aplicativo lista os contatos pesquisados; 7) O usuário clica em adicionar; 8) O aplicativo salva no banco de dados a operação;
Cenário alternativo	No passo 2, o usuário aceita as solicitações de amizade que recebeu.
Pós-condição	O usuário possui um relacionamento com outro usuário no banco de dados.

APÊNDICE B – Dicionário de Dados

Este Apêndice apresenta a descrição das tabelas do banco de dados e seus atributos conforme mencionado na seção 3.2.5. A primeira coluna de cada quadro deve indicar se o campo é uma Primary Key (PK), Foreign Key (FK) ou não é nada, neste caso o campo fica vazio. A segunda coluna representa o `nome`, seguido pela `descrição`, `tipo de dado` e `tamanho`.

Quadro 14 - Tabela `tb_friendship`

Tabela: `tb_friendship`				
	Campo	Descrição	Tipo	Tamanho
PK	id	ID da tabela de contatos	BIGINT	
	status	Status do contato (ACCEPT, SEND, RECEIVED, NONE)	VARCHAR	10
FK	user	Relacionamento com a tabela `tb_user`, representa vínculo com o usuário.	VARCHAR	255
FK	friend	Relacionamento com a tabela `tb_user`, representa vínculo com o usuário que será o contato.	VARCHAR	255

Fonte: elaborado pelo autor.

Quadro 15 - Tabela `tb_usergroup`

Tabela: `tb_usergroup`				
	Campo	Descrição	Tipo	Tamanho
PK	id	ID da tabela de relação entre usuário e grupo	BIGINT	
	isAdmin	Campo para verificar se o usuário é administrador do grupo	BOOLEAN	
	isReceptor	Campo para verificar se é o usuário que irá receber o pagamento. (não chegou a ser utilizado)	BOOLEAN	
	name	Campo apelido para usuários anônimos, quando o relacionamento com `tb_user` for null.	VARCHAR	45
	payment_observation	Observação do pagamento deixada pelo usuário.	VARCHAR	255
	payment_picture	Armazena a URL da foto da comprovação do pagamento.	VARCHAR	255
FK	tb_user_uid	Relacionamento com a tabela de `tb_user`.	VARCHAR	255
FK	tb_group_id	Relacionamento com a tabela de `tb_group`.	BIGINT	
	paid	Tag para verificar se o usuário já pagou ou não a conta no grupo.	BOOLEAN	

Fonte: elaborado pelo autor.

Quadro 16 - Tabela `tb_user`

Tabela: `tb_user`				
	Campo	Descrição	Tipo	Tamanho
PK	uid	ID da tabela de usuário	VARCHAR	255
	name	Nome	VARCHAR	255
	email	E-mail	VARCHAR	255
	birthday	Data de nascimento	Date	
	avatar	Armazena URL da foto de perfil	VARCHAR	255
	exclusive_user_name	Nome exclusivo	VARCHAR	45
	phone	Telefone	VARCHAR	45

Fonte: elaborado pelo autor.

Quadro 17 - Tabela `tb_userexpense`

Tabela: `tb_userexpense`				
	Campo	Descrição	Tipo	Tamanho
PK	id	ID do relacionamento da tabela de usuário, grupo e despesa.	BIGINT	
	check	Salvar o estado da tela, se o usuário está checked em determinada despesa	BOOLEAN	
	price	Preço a pagar pela despesa	DOUBLE	
	percent	Percentual a pagar pela despesa	DOUBLE	
FK	user_group_id	Relacionamento com a tabela `tb_usergroup`	BIGINT	
FK	expense_id	Relacionamento com a tabela `tb_expense`	BIGINT	

Fonte: elaborado pelo autor.

Quadro 18 - Tabela `tb_group`

Tabela: `tb_group`				
	Campo	Descrição	Tipo	Tamanho
PK	id	ID da tabela de grupo	BIGINT	
	description	Descrição	VARCHAR	45
	created_at	Data de criação do grupo	DATE	
	Shared_key	Chave do grupo para vincular um usuário (não utilizado no aplicativo).	VARCHAR	255
	title	Título	VARCHAR	45
	avatar	Armazena URL da foto do grupo	VARCHAR	255
	closed	Tag para verificar se a conta do grupo está fechada.	BOOLEAN	

Fonte: elaborado pelo autor.

Quadro 19 - Tabela `tb_expense`

Tabela: `tb_expense`				
	Campo	Descrição	Tipo	Tamanho
PK	id	ID da tabela de despesas	BIGINT	
	title	Título da despesa	VARCHAR	45
	price	Preço da despesa	DOUBLE	
	quantity	Quantidade	INT	

	created_by	Relacionamento com a tabela tb_user para identificar qual foi o usuário que criou a despesa	VARCHAR	255
	group_id	Relacionamento com a tabela tb_group para identificar em qual grupo a despea foi criada	BIGINT	
	date	Data da despesa	DATE	

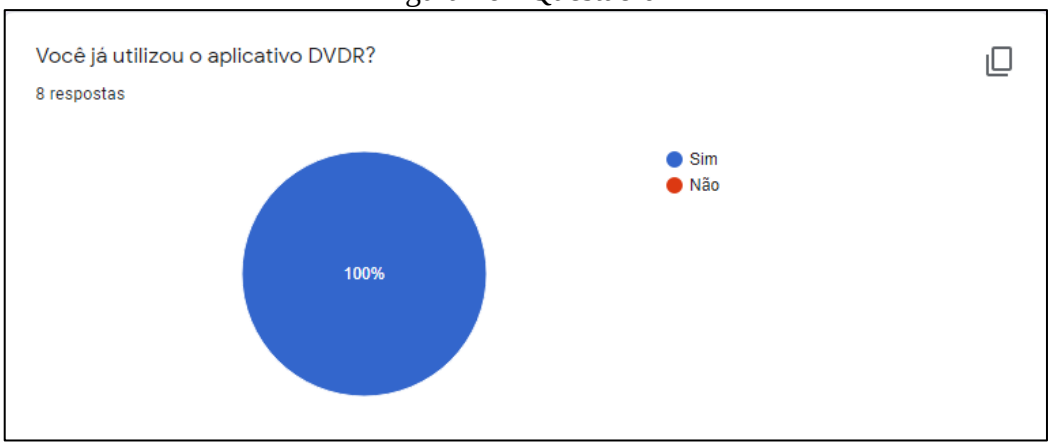
Fonte: elaborado pelo autor.

APÊNDICE C – Formulário de avaliação

Este apêndice contém os dados da pesquisa realizada pelo Google Forms, que busca identificar a quão boa é a usabilidade do aplicativo.

A Figura 26 comprova que todas as pessoas que participaram do questionário, já usaram a aplicação pelo menos uma vez.

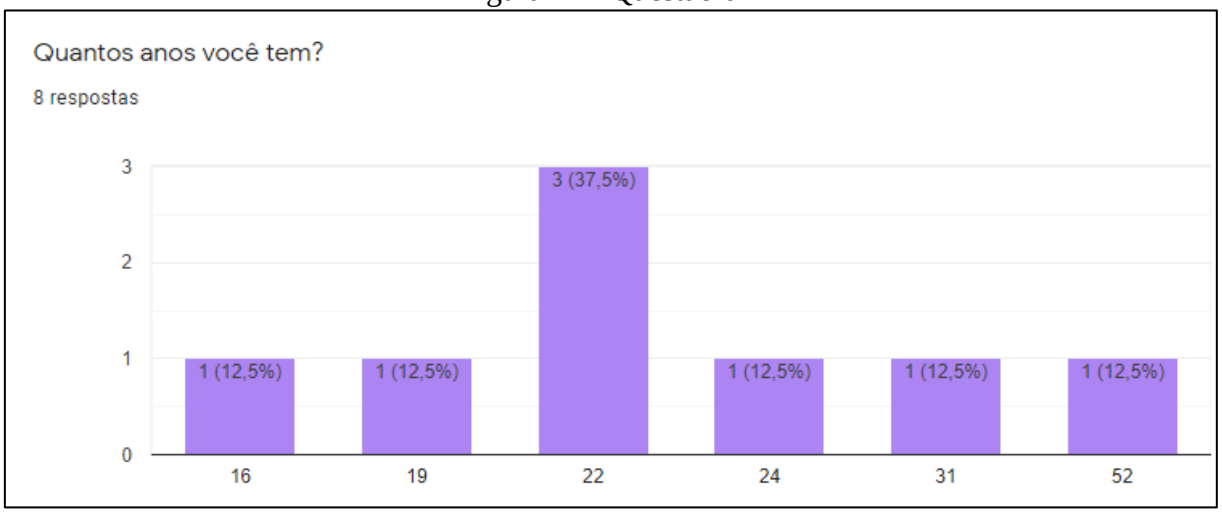
Figura 26 - Questão 01



Fonte: Digitalizado pelo autor.

A Figura 27 apresenta a idade das pessoas que responderam o questionário.

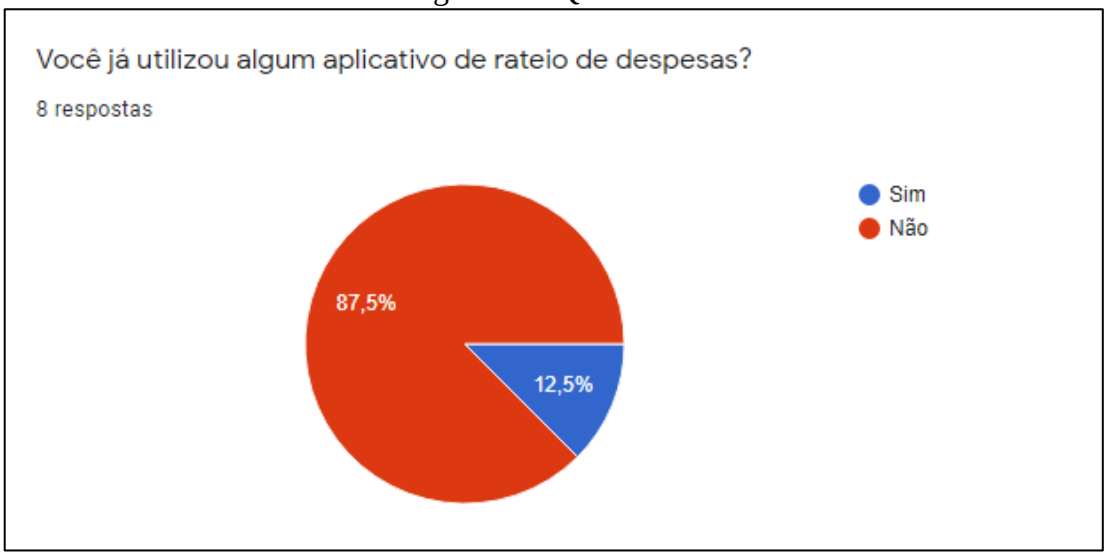
Figura 27 - Questão 02



Fonte: Digitalizado pelo autor.

A Figura 28 apresenta o percentual de pessoas que já utilizaram um aplicativo similar ao que foi proposto.

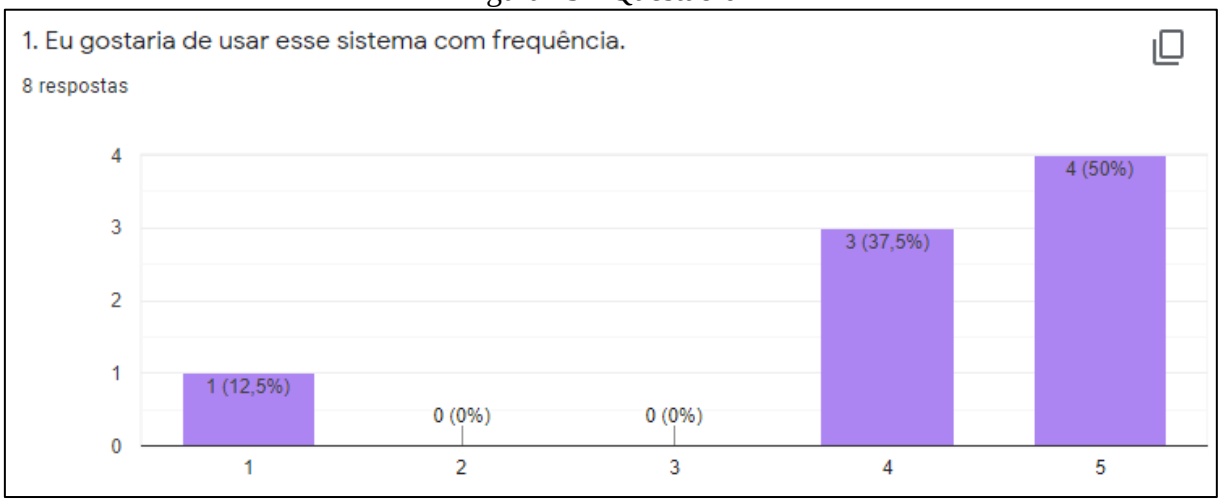
Figura 28 - Questão 03



Fonte: Digitalizado pelo autor.

A Figura 29 apresenta a pesquisa relacionada ao contínuo uso da aplicação.

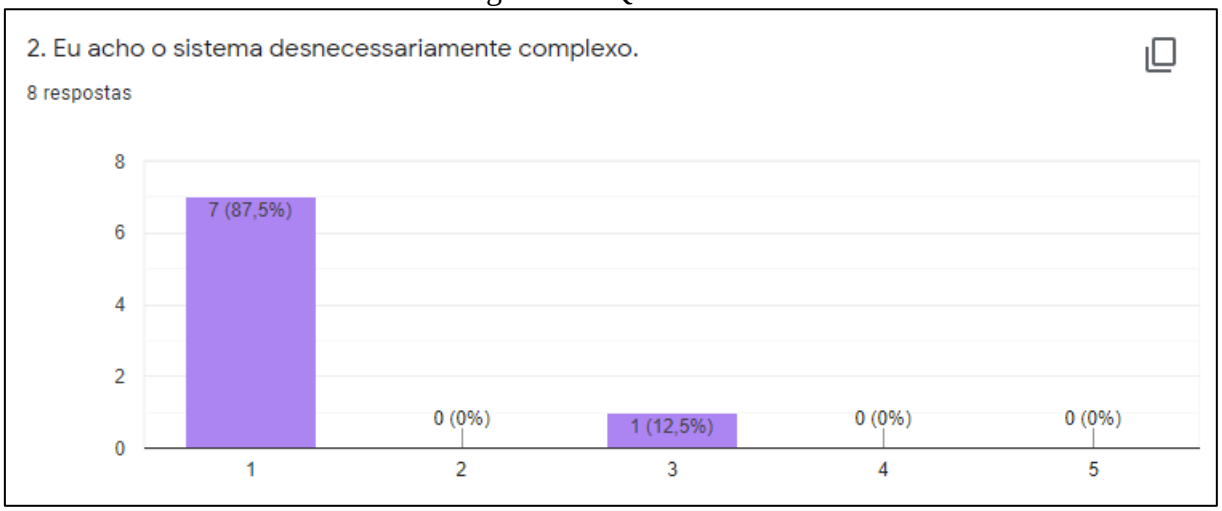
Figura 29 - Questão 04



Fonte: Digitalizado pelo autor.

A Figura 30 demonstra se as pessoas consideram o aplicativo desnecessariamente complexo.

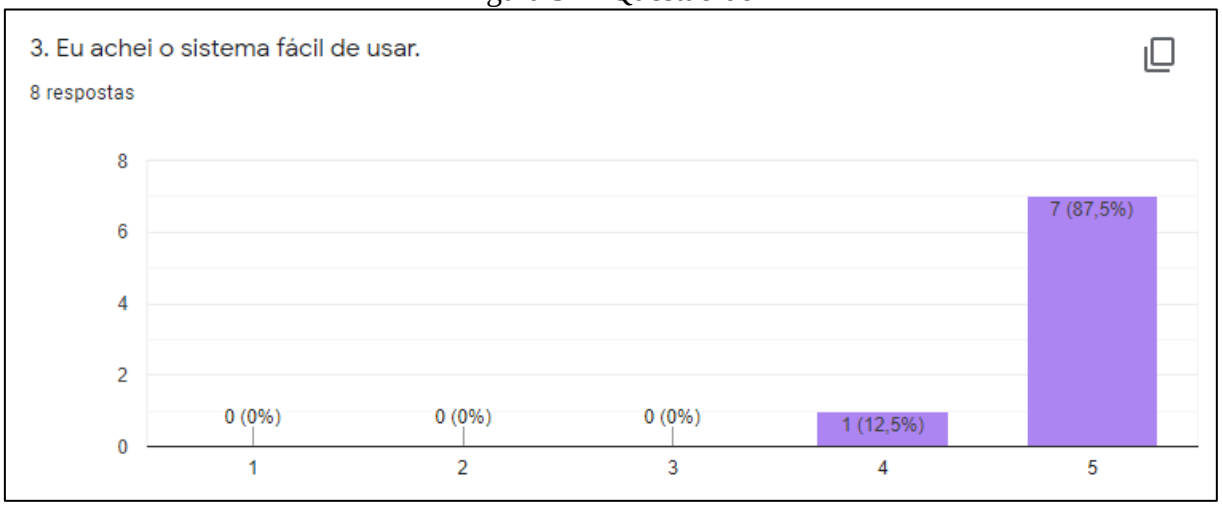
Figura 30 - Questão 05



Fonte: Digitalizado pelo autor.

A Figura 31 demonstra quantas pessoas acharam fácil utilizar a aplicação.

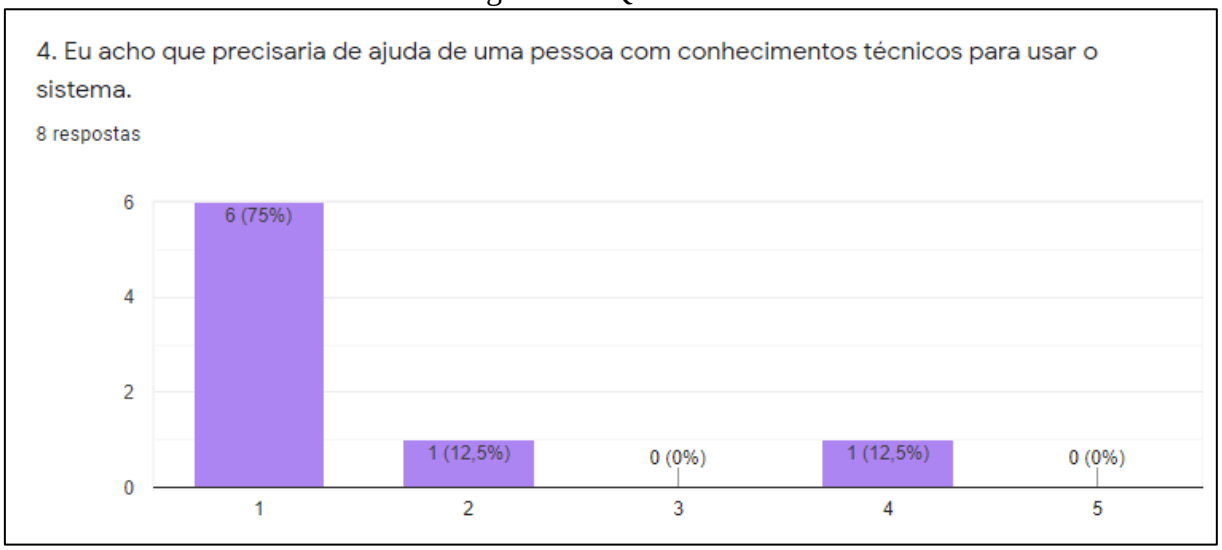
Figura 31 - Questão 06



Fonte: Digitalizado pelo autor.

A Figura 32 demonstra o resultado das pessoas que precisariam da ajuda de outras pessoas com conhecimento técnico para utilizar a aplicação.

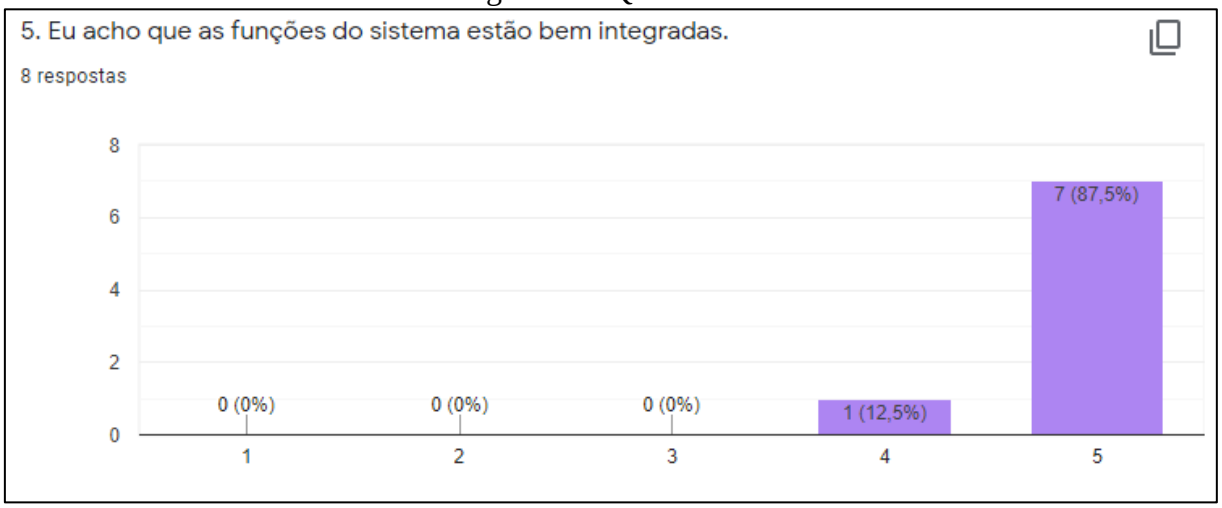
Figura 32 - Questão 07



Fonte: Digitalizado pelo autor.

A Figura 33 demonstra, quantas pessoas acham que as funções do aplicativo estão bem integradas.

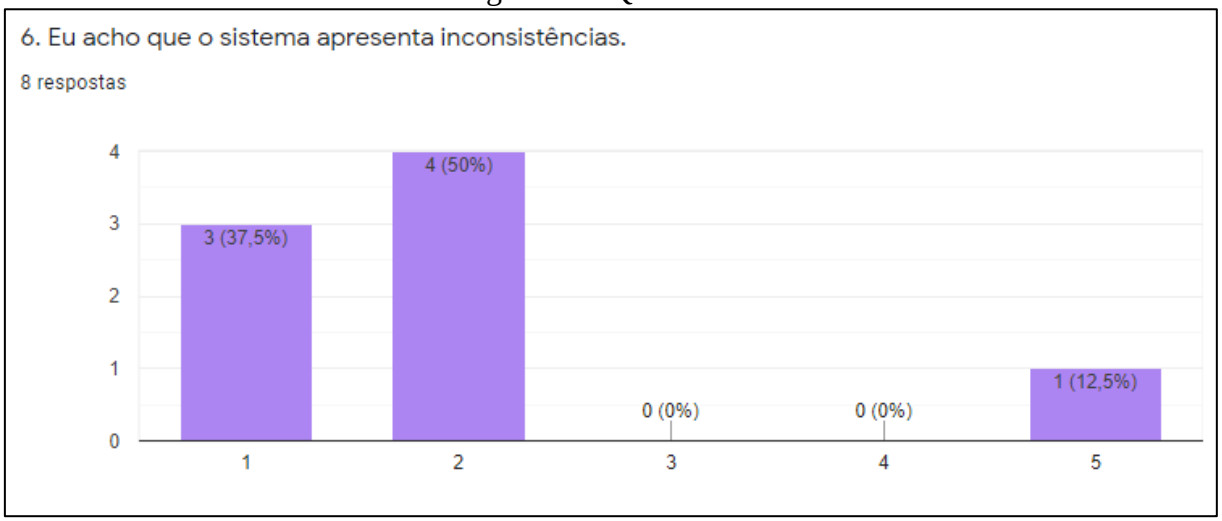
Figura 33 - Questão 08



Fonte: Digitalizado pelo autor.

A Figura 34 demonstra, o resultado das pessoas que consideram que o aplicativo apresenta inconsistências.

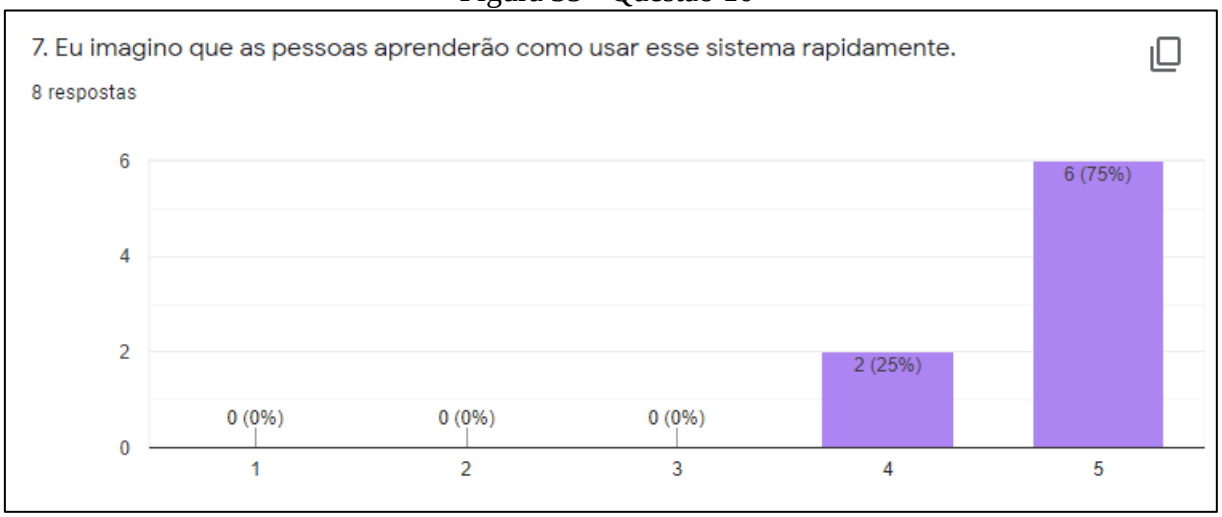
Figura 34 - Questão 09



Fonte: Digitalizado pelo autor.

A Figura 35 apresenta, quantas pessoas acham que outras pessoas irão conseguir aprender a usar o aplicativo rapidamente.

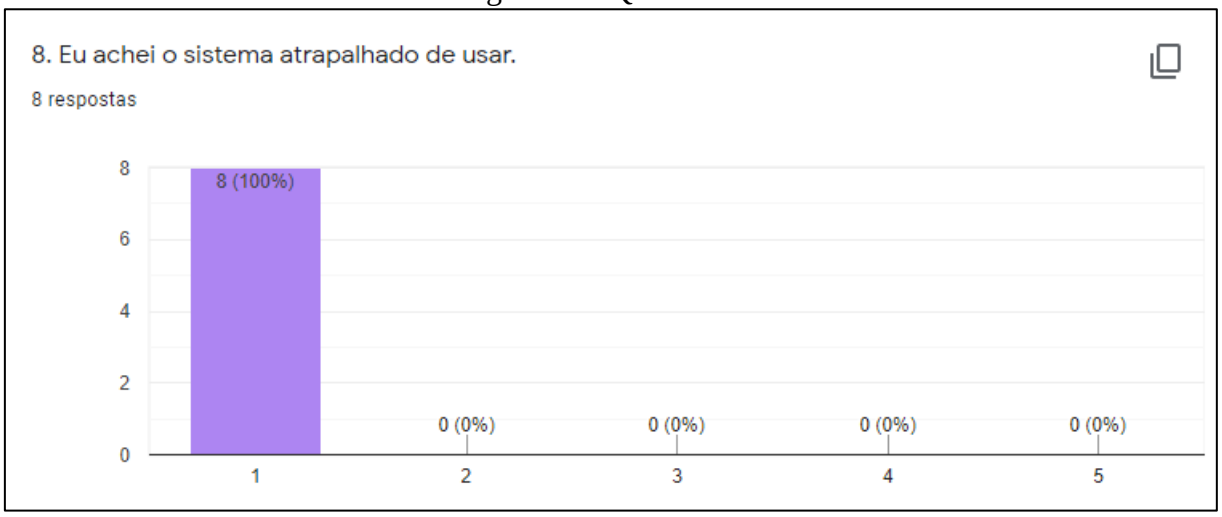
Figura 35 - Questão 10



Fonte: Digitalizado pelo autor.

A Figura 36 apresenta, o grau em que as pessoas consideram o aplicativo atrapalhado de usar.

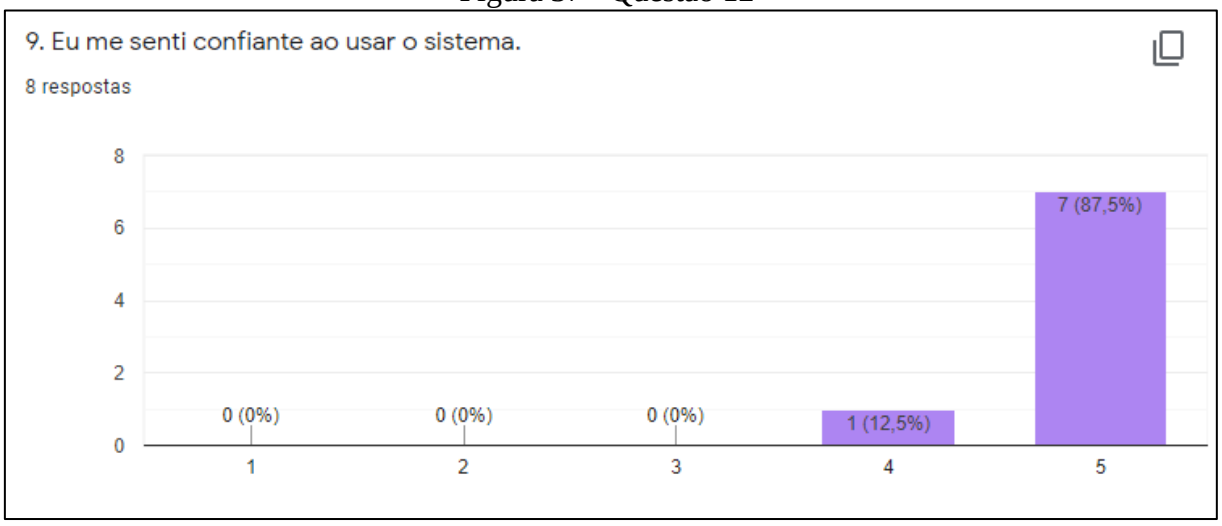
Figura 36 - Questão 11



Fonte: Digitalizado pelo autor.

A Figura 37 demonstra, o grau em que as pessoas se sentem confiantes utilizando o aplicativo.

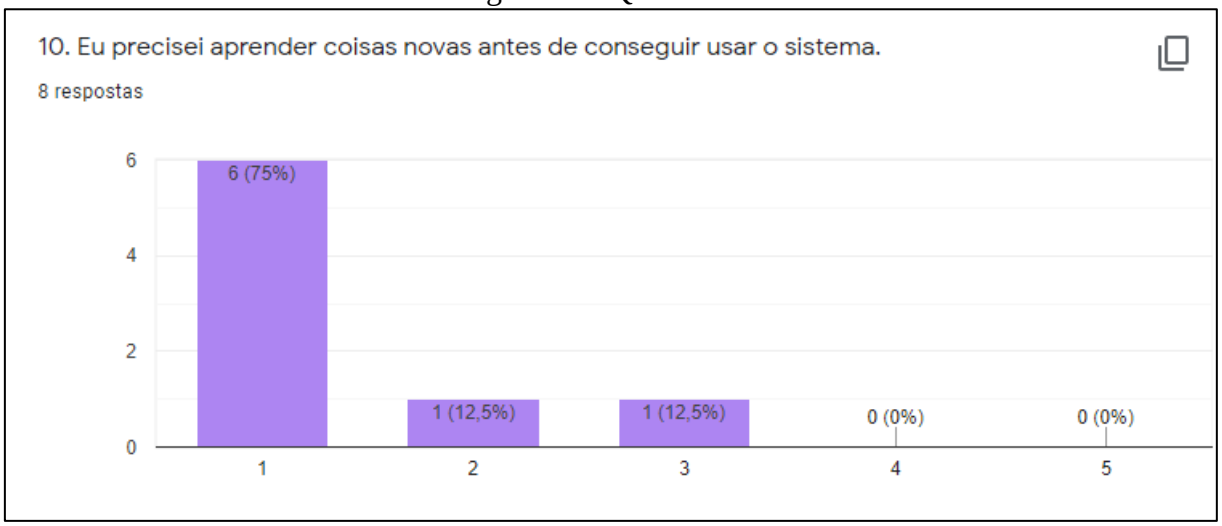
Figura 37 - Questão 12



Fonte: Digitalizado pelo autor.

A Figura 38 apresenta o grau em que as pessoas consideram que foi necessário aprender alguma coisa, para conseguir utilizar o aplicativo.

Figura 38 - Questão 13



Fonte: Digitalizado pelo autor.