Kimberly Lamounier Campos Ferreira

Desenvolvimento De Um Protótipo Organizador De Moradias Compartilhadas Usando Práticas Ágeis

Formiga - MG

2023

Kimberly Lamounier Campos Ferreira

Desenvolvimento De Um Protótipo Organizador De Moradias Compartilhadas Usando Práticas Ágeis

Monografia do trabalho de conclusão de curso apresentado ao Instituto Federal de Minas Gerais - Campus Formiga, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais

Campus Formiga

Ciência da Computação

Orientador: Roger Santos Ferreira

Formiga - MG

2023

Ferreira, Kimberly Lamounier Campos
F383d Desenvolvimento de um protótipo organizador de moradias compartilhadas usando práticas ágeis / Kimberly Lamounier Campos Ferreira -- Formiga : IFMG, 2023.
73p. : il.

Orientador: Prof. MSc. Roger Santos Ferreira
Trabalho de Conclusão de Curso – Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais – *Campus* Formiga.

1. Metodologias ágeis. 2. Design Thinking. 3. OKR. 4. Kanban. 5. Interface de usuário. I. Ferreira, Roger Santos. II. Título.

CDD 004

Ficha catalográfica elaborada pela Bibliotecária Msc. Simoni Júlia da Silveira

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS
Campus Formiga
Diretoria de Ensino
Docência Área Acadêmica de Computação
Rua São Luiz Gonzaga, s/n - Bairro São Luiz - CEP 35570-000 - Formiga - MG
- www.ifmg.edu.br

KIMBERLY LAMOUNIER CAMPOS FERREIRA

DESENVOLVIMENTO DE UM PROTÓTIPO ORGANIZADOR DE MORADIAS COMPARTILHADAS USANDO PRÁTICAS ÁGEIS

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Minas Gerais - Campus Formiga, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

APROVADO em 30 de junho de 2023.

BANCA EXAMINADORA

Prof. Roger Santos Ferreira (Orientador)

Prof. Manoel Pereira Junior (IFMG)

Prof.ª Paloma Maira de Oliveira (IFMG)

Formiga, 30 de junho de 2023.

Documento assinado eletronicamente por **Roger Santos Ferreira**, **Professor**, em 30/06/2023, às 16:54, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Manoel Pereira Junior**, **Professor**, em 30/06/2023, às 21:38, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Paloma Maíra de Oliveira Lima**, **Professora**, em 02/07/2023, às 08:40, conforme Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade do documento pode ser conferida no site https://sei.ifmg.edu.br/consultadocs informando o código verificador **1598082** e o código CRC **AE48CACA**.

Agradecimentos

Acredito firmemente que cada experiência que vivenciamos ao longo da vida representa uma valiosa oportunidade de aprendizado e crescimento pessoal. Nesse sentido, é com imensa gratidão que expresso meus sinceros agradecimentos a minha família, principalmente minha mãe, ao meu companheiro Erick, aos professores e servidores do IFMG - *Campus* Formiga, aos amigos que conquistei durante minha trajetória acadêmica, especialmente Rafael, Henrique, Leandro, Fernanda e Danilo, aos colegas e amigos que colaboram no meu contexto profissional, em particular Liliane, Lorenzo, Pedro, Thiago e a todas as pessoas que acreditaram em minha jornada e ofereceram seu apoio incondicional. Sem o suporte e encorajamento dessas pessoas, certamente os desafios superados e conquistas alcançadas não teriam sido possíveis. Reconheço a importância fundamental dessas conexões e expresso minha profunda gratidão por contribuírem de maneira significativa para meu desenvolvimento pessoal e profissional.

"Compreender é o começo da aprovação" (Baruch Espinoza)

Resumo

As moradias compartilhadas são arranjos habitacionais em que grupos de pessoas dividem um imóvel, estabelecendo uma gestão autônoma baseada em regras acordadas por unanimidade. Com o intuito de aprimorar a convivência nas moradias compartilhadas, sugere-se a implementação de uma aplicação *web* responsiva, desenvolvida através de métodos ágeis, como *Design Thinking, Kanban* e *Objectives and Key Results* (OKR), integrados a uma interface de usuário que desempenha o papel de conceber o protótipo e facilitar a execução de ações pertinentes. Essa integração visa demonstrar a aplicação dos métodos ágeis ao longo de todo o ciclo de vida do projeto, desde a fase inicial de concepção até a implementação prática, apresentando resultados que exemplifique a efetividade dessas práticas ágeis e considerações da interação entre as tecnologias e metodologias empregadas.

Palavras-chave: Metodologias ágeis, Design Thinking, OKR, Kanban, Interface de usuário.

Abstract

Shared housing represents housing arrangements in which groups of individuals share a property, establishing autonomous management based on unanimously agreed rules. This form of housing is sought after due to its potential cost reduction, encompassing expenses such as rent and household bills, while also fostering a sense of community, personal growth, and socialization. To enhance the coexistence within shared housing, the adoption of agile methods such as Design Thinking, Kanban, and Objectives and Key Results (OKR), integrated with a user interface that serves the purpose of designing the prototype and facilitating relevant actions, is suggested. This integration aims to demonstrate the application of agile methods throughout the entire project lifecycle, from the initial conception phase to practical implementation, resulting in a responsive web application that exemplifies the effectiveness of these agile approaches. It highlights the benefits and outcomes achieved through the synergy between the adopted methods and the user interface.

Keywords: Agile methodologies, Design Thinking, OKR, Kanban, User interface.

Lista de ilustrações

Lista de abreviaturas e siglas

OKR	*Objectives and Key Results*
DT	*Design Thinking*
UML	*Unified Modeling Language*
BDD	*Behavior-Driven Development*
TDD	*Test-Driven Development*
SPA	*Single Page Application*
HTTP	*Hypertext Transfer Protocol*
DOM	*Document Object Model*
HTML	*HyperText Markup Language*
CSS	*Cascading Style Sheets*
IHC	Interface Humano-Computador
ES	Engenharia de *Software*
CRUD	*Create, Read, Update, Delete*
API	*Application Programming Interface*
JSON	*JavaScript Object Notation*
NPM	*Node Package Manager*

Sumário

1 Introdução

As moradias compartilhadas configuram-se em grupos de indivíduos que dividem um imóvel e adotam uma forma autônoma de gestão e administração, onde os moradores orientam-se por regras definidas por unanimidade entre eles (VINTEROVA, 2008). As pessoas procuram esse sistema de habitação por apresentar vantagens, tais como a redução das despesas, pois dividem valores de aluguel, contas da residência e até alimentação são divididos entre aqueles que habitam o mesmo ambiente, também podemos citar o senso de coletividade, agregando ao crescimento pessoal e à socialização, em que os indivíduos se tornam amigos ou já possuem a amizade concretizada antes de dividirem o lar (FARO, 2019).

Contudo, há desvantagens e dificuldades nesse formato de convívio, podendo chegar a acontecer situações como a perda de privacidade, falta de respeito ao espaço e ambiente do outro, falta de compromisso ou a ausência da definição de regras e limites, problemas com divisão e organização de custos e tarefas da moradia (UNIFOA, 2017). Segundo Godinho (2013) essa questão de morar fora do ciclo familiar é delicada e é necessário diálogo, tal como fazer rodízios de limpeza e organizar as finanças da casa, tendo fácil acesso a esses valores.

Com o objetivo de fomentar uma convivência mais harmoniosa entre os indivíduos que compartilham o mesmo espaço habitacional, independentemente de sua configuração, propõe-se a implementação de um protótipo de aplicação *web* que poderia permitir uma otimização na convivência cotidiana, utilizando métodos ágeis como o *Design Thinking*, *Kanban*, OKR (Objetivos e Resultados-Chave), com integração a uma interface de usuário para guiar a elaboração do projeto, com base nos conhecimentos teóricos e práticos da área de Ciência da Computação.

1.1 Objetivos

O objetivo geral deste projeto consiste em evidenciar as aplicações de práticas ágeis na modelagem de projetos em conjunto com a interface do usuário, resultando no desenvolvimento de uma aplicação *web* responsiva, visando a eficiência e flexibilidade do processo de implementação. Pode-se citar os objetivos específicos:

- Aplicar metodologias ágeis na construção do protótipo organizacional.
- Definir requisitos funcionais e não-funcionais através da metodologia de *Design Thinking*.

- Definir objetivos e resultados-chave da elaboração do protótipo através da metodologia de OKR.
- Elaborar o diagrama de casos de uso da Linguagem de Modelagem Unificada (UML) com os artefatos do protótipo..
- Oferecer um guia visual utilizando *Figma* e práticas de *design* da interface de usuário.
- Descrever os cenários orientados a comportamentos.
- Desenvolver um protótipo de aplicação que possibilite a organização e divisão de tarefas e despesas entre membros de grupos, com base nas informações levantadas por meio dos procedimentos mencionados anteriormente.

1.2 Estrutura do Trabalho

Esta monografia está estruturada em cinco capítulos, cada um dedicado a uma etapa relevante da pesquisa. O capítulo de Fundamentação Teórica aborda os conceitos e fundamentos que sustentam o estudo. O capítulo de Materiais e Métodos descreve os materiais utilizados e detalhada como foi construído o projeto. O capítulo de Modelagem e Implementação do Protótipo apresenta o desenvolvimento da solução proposta, com ênfase na modelagem e nas etapas para realizar a implementação. O capítulo de Resultados apresenta de forma sistemática os resultados obtidos pela utilização da modelagem do protótipo. Por fim, o capítulo de Considerações Finais resume as principais conclusões do estudo, incluindo sugestões para futuras pesquisas.

2 Fundamentação Teórica

Nesta seção serão apresentados os conceitos sobre moradias compartilhadas, Engenharia de *Software*, Interface Humano-Computador, Programacao Orientada em Fluxos de Dados Assíncronos, *Single Page Application* e trabalhos relacionados.

2.1 Moradia Compartilhada

Esse formato de moradia consiste em um lar comunitário em que todas as áreas de convivência de uma casa convencional são compartilhadas, áreas como lavanderia, cozinha, sala de estar e jantar são ambientes de uso mútuo entre todos os moradores enquanto o espaço privado se reduz ao quarto (GROZDANIC, 2016). Segundo Cavadas (2022), esse estilo de moradia envolve uma nova maneira de aproveitar os espaços residenciais disponíveis, em busca de oportunidades e benefícios para todos. Assim, várias pessoas ocupam o mesmo espaço habitacional e todos dividem direitos e responsabilidades quanto ao uso dos ambientes.

2.2 Engenharia de *Software* (ES)

A ES é uma metodologia de desenvolvimento que utiliza processos dinâmicos para gerar soluções, efetivar padrões de qualidade e produtividade nas atividades e produtos, gerindo de maneira eficiente, envolvendo questões técnicas e não-técnicas. Tem se mostrado uma grande ferramenta para auxiliar no desenvolvimento de sistemas, tendo como objetivo visar a sistematização da produção, da manutenção, da evolução e a recuperação de produtos de *software* para que o desenvolvimento ocorra dentro dos prazos estimados, utilizando ferramentas, métodos e tecnologias para promover mais qualidade na construção e realização de tarefas, definindo uma estrutura de controle sobre as atividades e os recursos envolvidos. (REZENDE, 2006). Uma abordagem comumente adotada na ES é a utilização de práticas ágeis, que proporcionam flexibilidade e adaptabilidade aos projetos.

2.2.1 Desenvolvimento ágil de *software*

No contexto atual, onde as mudanças estão ocorrendo de forma rápida, o desenvolvimento ágil de *software* se torna essencial. As organizações estão conscientes de que a agilidade no desenvolvimento e entrega de sistemas é fundamental para aproveitar oportunidades e responder às pressões competitivas. Nesse cenário dinâmico, os requisitos iniciais podem ser modificados à medida que os clientes ganham experiência e compreendem melhor suas necessidades. Além disso, fatores externos podem influenciar a evolução dos

requisitos mesmo após a entrega do sistema. Portanto, o desenvolvimento ágil de *software* permite uma abordagem flexível e iterativa, adaptando-se às mudanças e proporcionando maior agilidade e resposta às necessidades do mercado (SOMMERVILLE, 2011).

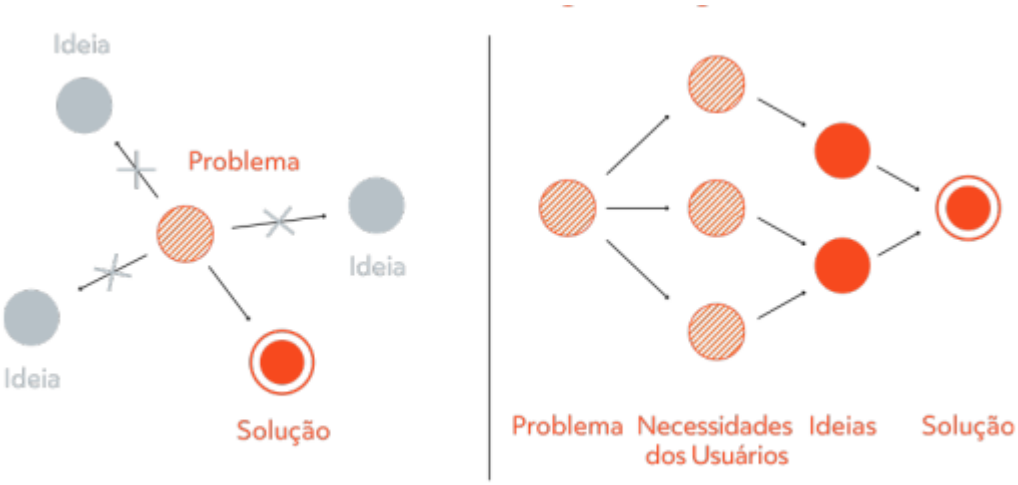
A fim de organizar um produto, os métodos ágeis advém para melhorar a comunicação, que era grande e complexa, agora mais objetiva, criando uma construção clara durante todo o processo de desenvolvimento. Esses métodos são flexíveis, se adequa conforme a necessidade e auto-evolui com o decorrer do processo, eliminando os erros na entrega (SILVA, 2016).

Nesse contexto, têm sido explorados diversos métodos e técnicas com o objetivo de aumentar a produtividade e a qualidade dos produtos finais. Cada abordagem adotada possui um enfoque específico e entre elas apresenta-se o *Design Thinking, Objetive and Key-Results, Behavior Driven Development* e *Kanban*.

2.2.1.1 *Design Thinking* (DT)

O DT é uma metodologia desenvolvida na empresa *IDEO*, que atua tanto no campo do *design* e em consultoria de inovação, em que engloba um conjunto de práticas inspiradas no *design*, visando à resolução de problemas e ao desenvolvimento de projetos, além de visar integração do que é desejável do ponto de vista do cliente com o que é economicamente e tecnicamente viável para a organização. Em sua prática utiliza-se conceitos como empatia, criatividade e racionalidade para atender às necessidades dos usuários e alinhar-se aos objetivos de quem o utiliza (BROWN; KATZ, 2010). A Figura 1 ilustra a diferença do método de solução tradicional feito de forma intuitiva, baseado em tentativa e erro, enquanto o processo orientado pelo DT é estruturado.

Figura 1 – Solução de problemas: Modelo tradicional x *Design Thinking*

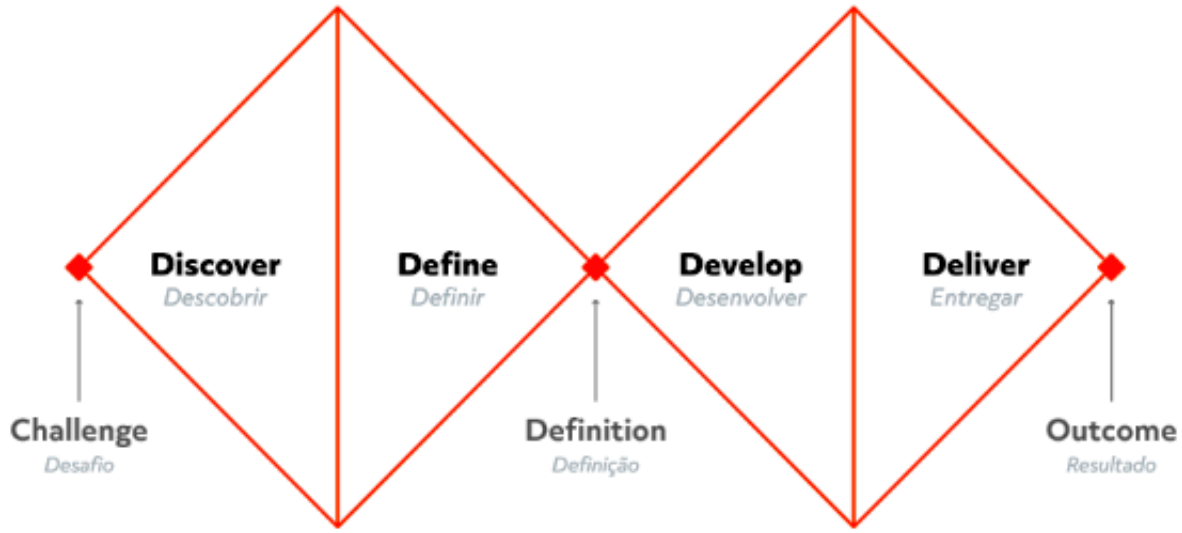


Fonte: (ESTUDIO MARTE, 2022)

Embora não exista uma fórmula predefinida, é comum a presença de alguns elementos e etapas recorrentes, como por exemplo, as fases de pensamento divergente e convergente. O pensamento divergente permite a exploração do problema e a geração de novas ideias, incentivando a criação de opções sem restrições ou julgamentos, além da compreensão de diferentes perspectivas. Por sua vez, o pensamento convergente é utilizado para refinar as ideias, selecionar entre as alternativas existentes e chegar a um consenso (PINHEIRO; FERREIRA, 2017).

Desta forma, o *framework Double Diamond* pode ser implementado, mostrando-se eficaz em termos de processo de resolução de problemas. Em cada uma de suas etapas há uma fase de exploração que amplia as possibilidades, seguida por uma fase de convergência que busca reduzir o escopo de soluções. Possui quatro fases essenciais para o sucesso do processo do DT: descoberta, definição, desenvolvimento e entrega, conforme ilustra a Figura 2 (LI; LIU, 2022).

Figura 2 – *Double Diamond*



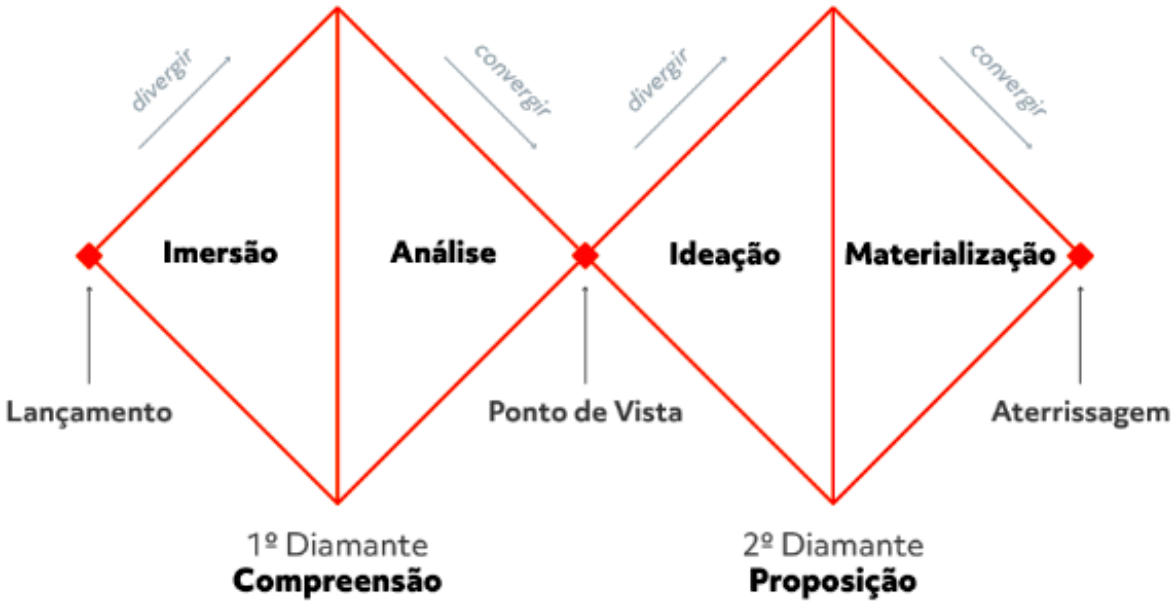
Fonte: (ESTUDIO MARTE, 2022)

O modelo de DT proposto por Vianna et al. (2012) é composto por quatro etapas: Descobrir, Definir, Desenvolver e Entregar, organizadas em estágios divergentes e convergentes do processo de *design*. Na etapa de Descobrir, ocorre a imersão inicial para entender o problema e o contexto e externalizar os problemas dos usuarios. Na etapa de Definir, são interpretados e ponderados os *insights* obtidos na fase anterior, aprofundando-se nas necessidades e oportunidades. Na etapa de Desenvolver, são criadas potenciais soluções com base nos dados obtidos, identificando os problemas mais recorrentes. Na etapa de Entregar, ocorre a convergência da solução e a validação do que foi produzido. Essas etapas permitem uma abordagem sistemática e iterativa para o desenvolvimento de soluções.

Na Figura 3 observa-se outra forma de analisar as fases do DT, sem perder o

significado das fases, de acordo com HPI Academy (2023). Independente da forma que é aplicada, seja por quatro fases ou mais, quando aplicado corretamente, esse método exploratório pode levar a ideias inesperadas que devem ser consideradas e trabalhadas como alternativas para resolver o problema em questão.

Figura 3 – *Double Diamond* aplicado no *Design Thinking*



Fonte: (ESTUDIO MARTE, 2022)

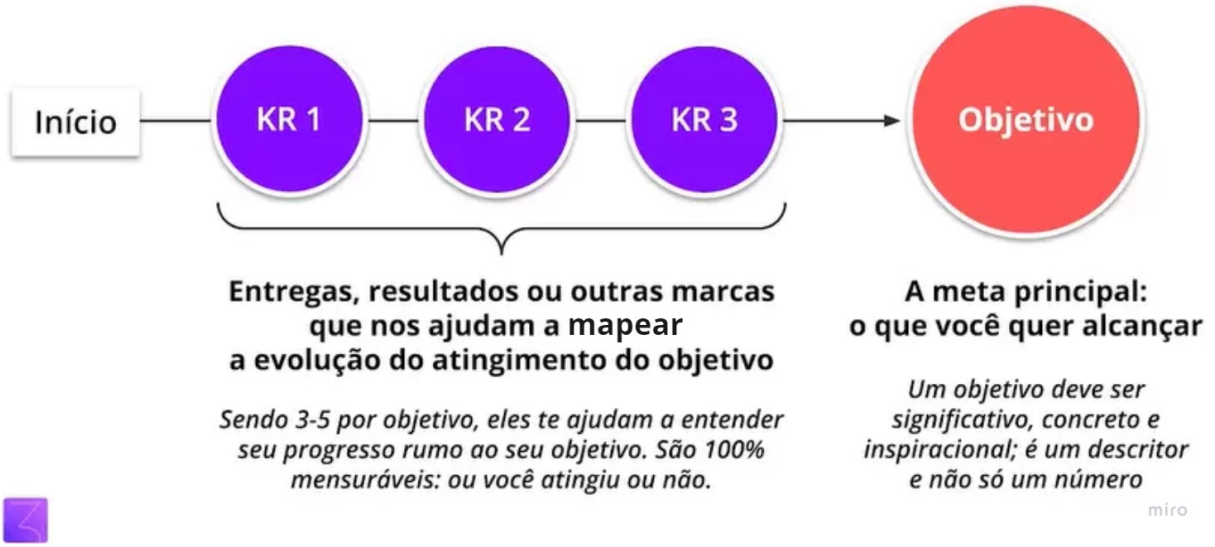
2.2.1.2 *Objectives and Key Results* (OKR)

O OKR é uma metodologia de definição de metas que surgiu originalmente na *Intel* e se disseminou para outras empresas do Vale do Silício. É uma abordagem simples que busca criar alinhamento e engajamento em torno de metas mensuráveis. Uma boa meta no OKR deve incluir o que será alcançado e a forma como será medida, sendo a medição um elemento fundamental para tornar a meta concreta. Assim, o OKR é composto por dois componentes: Objetivos, que são descrições qualitativas do que se deseja alcançar, devendo ser curtos, inspiradores e envolventes para motivar e desafiar a equipe e Resultados-Chave, que são um conjunto de métricas, quantitativas e mensuráveis, que medem o progresso em direção ao objetivo, recomendando-se ter de dois a cinco resultados principais para cada objetivo, evitando um número excessivo que dificulte a lembrança dos mesmos, como podemos observar na Figura 4 (CASTRO, 2020).

De acordo com Niven e Lamorte (2016), existem várias razões pelas quais o OKR é uma ferramenta eficaz de planejamento, podendo citar esses benefícios como: a agilidade proporcionada pelos ciclos mais curtos de metas permitindo ajustes rápidos e melhor adaptação às mudanças, alinhamento e colaboração entre equipes, comunicação clara das prioridades gerando engajamento e autonomia que faz com que cada indivíduo responsável

pelos objetivos, foco e disciplina nas iniciativas devido ao número reduzido de metas, promovendo um direcionamento organizacional mais efetivo e a possibilidade de estabelecer metas mais ambiciosas e desafiadoras.

Figura 4 – OKR



Fonte: Adaptado de PM3 (2022)

2.2.1.3 *Behavior-Driven Development* (BDD)

Desenvolvimento Orientado a Comportamento (BDD) é uma técnica ágil de desenvolvimento que incentiva a comunicação e colaboração entre todas as partes envolvidas, independentemente de serem técnicas ou não, em um projeto de *software*. Proposto originalmente por Dan North como uma resposta às limitações do Desenvolvimento Orientado a Testes (TDD), o BDD tem como objetivo integrar verificação e validação a uma técnica que visa começar pela parte do software percebida pelo usuário, estabelecendo os critérios de aceitação dos clientes como ponto de partida e, em seguida, projetando cada componente de forma separada (TAVARES et al., 2010).

Segundo North (2006), o comportamento de uma história em um sistema é definido pelos seus critérios de aceitação. Se o sistema atende a todos os critérios de aceitação, ele está se comportando corretamente. Para capturar esses critérios de aceitação, é necessário criar um modelo que seja flexível porém estruturado, sendo dividido em fragmentos, separados por funcionalidades, seguindo a forma padrão para escrever cenários executáveis: dado um contexto inicial, quando um evento ocorre, então garanta alguns resultados, utilizando o dado-quando-então (*template given-when-then*). Assim, quando finalizado, esses cenários se transformam em testes abrangentes.

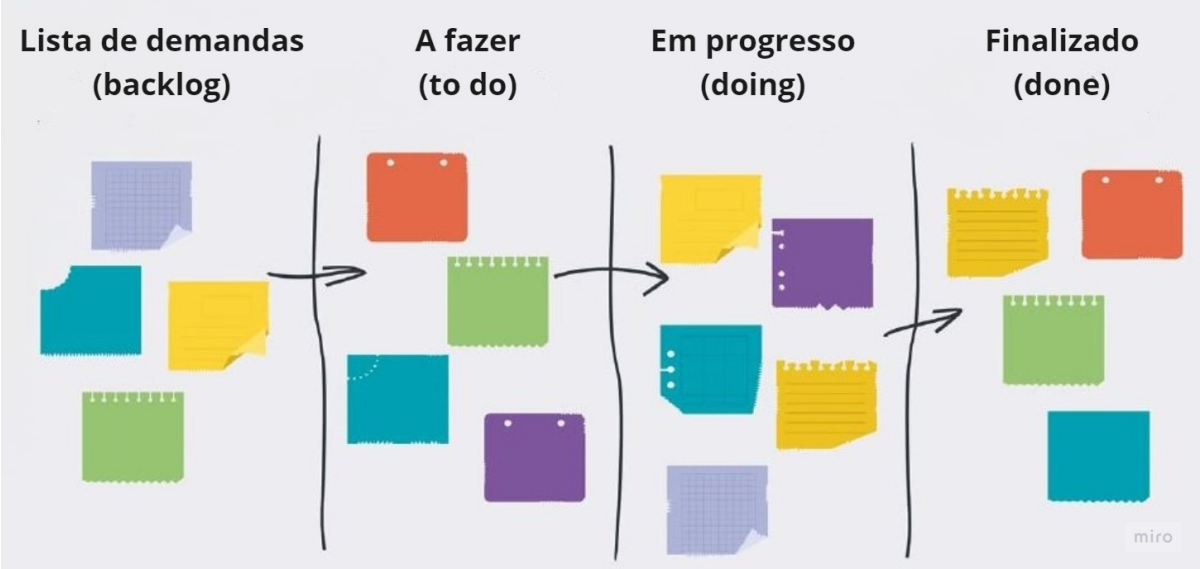
O processo gradual dessa técnica permite reduzir custos e aprimorar o sistema, concentrando-se na conexão direta dos requisitos de *software* com o código-fonte, o artefato

responsável por implementá-los (TAVARES et al., 2010).

2.2.1.4 Kanban

A metodologia *Kanban*, que em tradução literal significa "anotação visível" ou "sinal", foi inicialmente concebido na *Toyota* em 1947 com o objetivo de aprimorar o fluxo de produção, mas só em 2004 que o método foi introduzido como um processo de desenvolvimento de *software* por David J. Anderson. (ALAIDAROS et al., 2021)

Figura 5 – Quadro *Kanban*



Fonte: Adaptado de (POSCH, 2019)

O *Kanban* utiliza um mecanismo de controle visual, como um quadro branco com *post-its* ou um sistema de cartões eletrônicos, para acompanhar o fluxo de trabalho em suas diferentes etapas. As atividades representam o trabalho a ser realizado e as características podem ser atribuídas a uma atividade, transitando em um dos estados abaixo:

- *To Do* (A Fazer): cartões recém-puxados para serem realizados, mas ainda não atribuídos a um desenvolvedor;
- *Doing* (Em Processo): cartões em andamento;
- *Done* (Finalizado): cartões concluídos.

Outros estados possíveis, além dos itens já citados, são *Backlog* e *Released* que denotam o estado inicial e final, respectivamente. Elas podem ser livremente configuradas, conforme representado na Figura 5. Esse método possui quatro práticas, que são: começar com o que você tem, concordar em buscar mudanças incrementais e evolutivas, respeitar os

papéis, processos e responsabilidades existentes, e incentivar atos de liderança durante todo o processo de desenvolvimento (ALAIDAROS et al., 2021) (ANDERSON et al., 2011).

De acordo com Anderson et al. (2011), essaa metodologia proporciona transparência, expõe gargalos, filas, variabilidade e desperdício, permitindo ter visibilidade dos impactos das ações, além de estimular a colaboração e o aprimoramento contínuo dos processos. Com o *Kanban*, equipes que o utilizam têm maior visibilidade sobre suas atividades, facilitando a priorização e a tomada de decisões, possibilitando um planejamento mais eficiente, com entregas mais confiáveis e previsíveis, reduzindo o retrabalho, gerenciando melhor as expectativas e aumentando a confiabilidade das entregas.

2.2.2 Unified Modeling Language (UML)

A UML é composta por diagramas esquemáticos utilizados na especificação, documentação, visualização e desenvolvimento de sistemas complexos, criada por Jim Rumbaugh e Grady Booch. Essa linguagem vai além de símbolos gráficos, possuindo uma semântica bem definida que facilita sua compreensão pelos profissionais da área, sendo amplamente utilizada para criar modelos que descrevem as características e o comportamento de um *software*, auxiliando na identificação de suas funcionalidades e no planejamento de sua construção, o que torna esse modelo uma representação de uma simplificação da realidade (GUEDES, 2014).

2.2.2.1 Diagramas de Casos de Uso

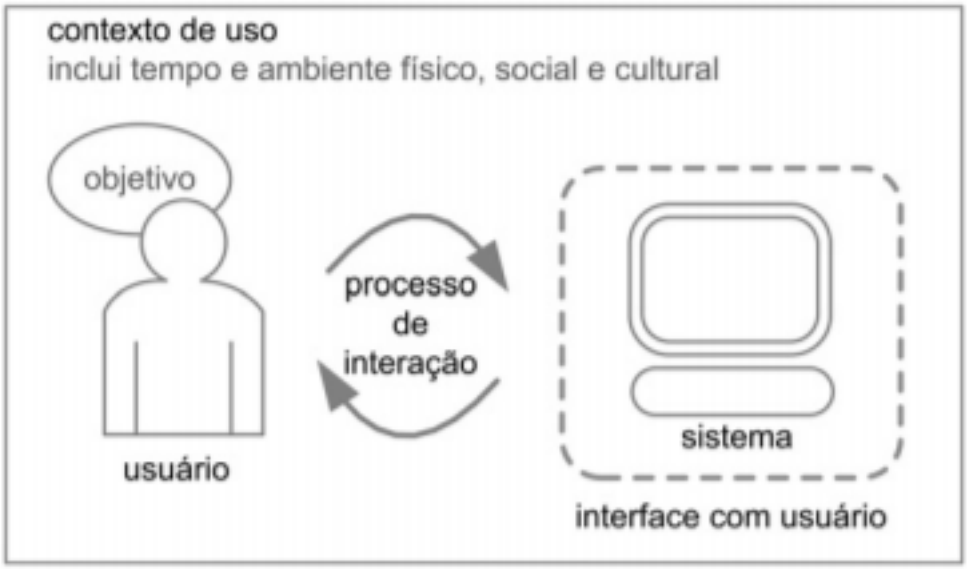
Segundo Cano et al. (2015), os Diagramas de Casos de Uso da UML são utilizados para descrever as funcionalidades de um sistema do ponto de vista das interações com ele, sem detalhar a implementação e determinar os requisitos funcionais do sistema. Os elementos que compõem esses diagramas são os seguintes:

- Ator: usuário que interage com o sistema, executando casos de uso. Um mesmo ator pode realizar vários casos de uso.
- Caso de Uso: descreve uma tarefa realizada pelo sistema e representa como o sistema colabora com o ator para executar pelo menos uma tarefa.
- Relações entre Casos de Uso: Existem três tipos de relações entre os casos de uso: extensão, inclusão e generalização. A extensão ocorre quando um caso de uso tem comportamentos diferentes dependendo das circunstâncias. A inclusão ocorre quando um caso de uso incorpora explicitamente o comportamento de outro caso em sua sequência. A generalização envolve herdar o comportamento e significado de outro caso de uso.

2.3 Interface Humano-Computador (IHC)

A Interface Humano-Computador (IHC) é uma disciplina que se preocupa com o projeto, avaliação e implementação de sistemas interativos de computação para uso humano, além do estudo dos principais fenômenos que os envolvem. A interação entre pessoas e sistemas por meio de suas interfaces de usuário é fundamental para determinar seu estado de aceitação ou rejeição. Essa área desempenha um papel interdisciplinar e interprofissional ao contribuir nos projetos, avaliar e compreender o uso de sistemas computacionais interativos, além de explorar os fenômenos associados a essa utilização.

Figura 6 – Elementos envolvidos no processo de iteração em IHC



Fonte: (BARBOSA; SILVA, 2010)

A Figura 6 apresenta um usuário envolvido em uma interação com a interface de um sistema interativo, buscando alcançar um objetivo específico em um contexto de uso definido. O contexto de uso abrange todos os aspectos relevantes para a interação do usuário com o sistema, incluindo o momento em que o sistema é utilizado e o ambiente físico, social e cultural em que a interação ocorre (BARBOSA; SILVA, 2010).

Diante dessas considerações, identificou-se a possibilidade de promover a interdisciplinaridade entre a Interface Humano-Computador e a Engenharia de *Software*.

2.3.1 Interface de Usuário

A interface, tanto de *software* quanto de *hardware*, é um elemento essencial no processo de interação entre o usuário e o sistema. Ela é responsável por viabilizar e facilitar a comunicação entre o usuário e a aplicação, e abrange a parte de uma aplicação que os usuários podem perceber, interagir e compreender. Seu objetivo é melhorar a interação entre

pessoas e computadores, promovendo uma relação mais eficiente e satisfatória entre ambos. A interação entre o usuário e o computador envolve dois componentes essenciais: entrada (*input*) é a forma como o usuário se comunica ou direciona o computador, utilizando dispositivos como mouse e touch; a saída (*output*) é a maneira como o computador apresenta as informações e respostas aos comandos do usuário (PREECE et al., 1994); (GALITZ, 2007).

Galitz (2007) diz que para desenvolver uma interface eficiente, é crucial ter um profundo entendimento do usuário e dos princípios de *design* de interface, que se baseiam em características de interação, como familiaridade e diversidade. Além disso, um aspecto fundamental das interfaces é a interação do usuário, que envolve a troca de comandos e dados com o sistema.

A análise realizada por Shneiderman e Plaisant (2009) resultou na definição de heurísticas para projetos envolvendo interfaces gráficas, conhecidas como "As Oito Regras de Ouro". Essas regras são as seguintes:

- Consistência: Procedimentos semelhantes devem seguir uma sequência de ações consistente, com padrões visuais e terminologia uniforme.
- Atalhos para usuários assíduos: Teclas de atalho, macros e navegação simplificada são recursos que agilizam a interação dos usuários experientes.
- *Feedback* informativo: O sistema deve fornecer respostas adequadas às ações do usuário, com níveis de detalhes apropriados.
- Diálogos que indiquem término da ação: O usuário deve ser informado sobre o progresso e conclusão das ações executadas.
- Prevenção e tratamento de erros: A interface deve evitar erros graves e oferecer mecanismos para tratá-los ou fornecer soluções alternativas.
- Reversão de ações: Quando possível, as ações devem ser reversíveis para proporcionar tranquilidade ao usuário.
- Controle: Os usuários experientes devem sentir que têm controle sobre o sistema e que este responde adequadamente às suas ações.
- Baixa carga de memorização: A interface deve ser fácil de memorizar, com uma estrutura organizada que facilite a memorização das telas pelo usuário, sem exigir esforço excessivo.

Assim, uma interface sendo projetada levando em consideração as regras elegidas por Shneiderman e Plaisant (2009) e a capacidade de adaptação às habilidades e limitações

do usuário, se torna adequada, além disso, ela deve atender objetivamente às tarefas para as quais foi desenvolvida. Portanto, a interface do usuário se torna um elemento essencial no desenvolvimento de um aplicativo ou *website* (SILVA et al., 2016).

2.3.2 Integração das Atividades de Interação Humano-Computador com Engenharia de *Software*

A interface de um sistema permite ao engenheiro de *software* definir a interação com o mundo externo. Ao construir o sistema, é comum esperar que a comunicação siga a interface estabelecida. No entanto, a interação entre pessoas e sistemas computacionais apresenta características diferentes da interação entre sistemas. É necessário considerar as características humanas e o contexto em que ocorre a interação (BARBOSA; SILVA, 2010).

De acordo com Barbosa e Silva (2010), a integração entre as áreas de IHC e ES pode ocorrer por meio de três principais abordagens. A primeira consiste na definição de características que um processo de desenvolvimento deve ter para fomentar mais qualidade de uso. A segunda abordagem envolve a incorporação de processos de IHC paralelos aos processos propostos pela ES. Por fim, a terceira abordagem é a identificação de pontos nos processos da ES em que atividades e métodos de IHC podem ser inseridos. Essas estratégias permitem uma maior sinergia entre as disciplinas, visando a criação de sistemas que atendam às necessidades dos usuários e proporcionem uma melhor experiência de uso.

2.4 *Single Page Application* (SPA)

Os Aplicativos de Página Única (SPAs) são aplicações *web* que possuem uma única página, carregando seus elementos quando solicitado pela primeira vez e atualizando sua interface, seja de forma parcial ou completa dos seus componentes, sem a necessidade de recarregar a página a cada ação do usuário e para que isso ocorra de forma automática a comunicação entre a SPA e o servidor deve acontecer de maneira assíncrona (SCOTTI, 2019).

Anteriormente, a cada interação do usuário, era necessário recarregar toda a página da *web* para refletir o resultado da interação. Isso levava a tempos de resposta mais lentos, maior uso de largura de banda e insatisfação entre os usuários. A solução para esse problema é a aplicação de página única (SPA), na qual a página inicial é carregada apenas uma vez, juntamente com todos os seus recursos, como CSS, imagens, *scripts*, etc. Em seguida, apenas os componentes relevantes são atualizados dinamicamente com base nas interações do usuário. Esse método reduz significativamente o tempo necessário para cada solicitação subsequente após o carregamento inicial, pois apenas uma parte da página é atualizada em vez de recarregar a página inteira (KOMPERLA et al., 2022).

2.5 Programação Orientada em Fluxos de Dados Assíncronos

Trata-se uma forma de programar que foi concebido em torno de um fluxo de dados baseado em eventos, o qual responde às alterações nas entradas. Esses fluxos podem ser desencadeados por eventos de entrada, como um clique de mouse, uma requisição de Protocolo de Transferência de Hipertexto (HTTP) ou uma ação do servidor. Esses valores podem ser entendidos como abstrações assíncronas de fluxos contínuos de dados em constante evolução, provenientes de eventos recorrentes. Tal abordagem resulta na capacidade de executar tarefas complexas em múltiplos fluxos simultaneamente, garantindo uma resposta reativa a todas as entidades que estão ouvindo esse evento (CHI, 2018).

Essa orientação incorpora mecanismos para lidar com erros que possibilita a repetição de um evento caso o resultado obtido não seja o esperado, além disso, viabiliza a execução de operações de forma assíncrona, permitindo o processamento de múltiplas tarefas e a realização de chamadas síncronas de forma paralela. Esses recursos contribuem para a eficiência e a escalabilidade do sistema, entregando um bom desempenho em situações que envolvem a manipulação de grandes volumes de dados ou o processamento simultâneo de múltiplas operações. (MORETTI, 2020).

2.5.1 *React*

Criado em 2011 pela equipe de desenvolvedores do *Facebook*, o *React* é uma biblioteca *JavaScript* de código aberto que possui recursos que facilitam a estruturação de uma aplicação utilizada para construir interfaces de usuário interativas, principalmente se tratando de SPA e através de componentes modulares reativos, traz mais acessibilidade para quem desenvolve, pois promove uma escrita de código mais limpa. Possui também eficácia nas mudanças visuais que as aplicações apresentam, além de ser consistente quando se trata de um programa *web*, integrando-se a várias bibliotecas de apoio. Seu objetivo é construir aplicações simples, rápidas e escaláveis (PEREIRA; PETRUCELLI, 2019).

O *React* se tornou popular devido à sua natureza declarativa e baseada em componentes, que permite a criação de componentes independentes que gerenciam seu próprio estado, facilitando a construção de interfaces de usuário complexas. Esses componentes podem ser reutilizados em diferentes partes da aplicação, proporcionando maior eficiência no desenvolvimento. Essa biblioteca suporta o *ECMAScript6* (ES6) e pode ser usado tanto no lado do cliente quanto no lado do servidor com o *NodeJS* (KOMPERLA et al., 2022).

Também se destaca pela sua simplicidade e suporte ao JSX, tem como característica a utilização de Modelo de Documento por Objetos (DOM) virtual para otimizar o desempenho da interface do usuário e oferece recursos de roteamento para facilitar a navegação.(REACT, 2023)

Além disso, os componentes construídos a partir de objetos *JavaScript* no *React* são

fundamentais para permitir uma composição simplificada, o que facilita o entendimento do código e oferece maior flexibilidade na estruturação e exibição de conteúdo, adotando um fluxo unidirecional de dados, simplificando o gerenciamento e a atualização do estado da aplicação (STAFF, 2016).

2.6 Trabalhos relacionados

Serão apresentados alguns trabalhos que se assemelham e contribuem com o projeto a ser realizado, possuindo temas e tecnologias relacionadas, como organização de tarefas e divisão de custos.

- **Tarefaça**: É um aplicativo móvel gamificado desenvolvido por Rodrigues (2021). Tem como abordagem o tema de gestão de tarefas para moradia compartilhada e como resultado, tem-se a avaliação de interface SUS (*System Usability Scale*), avaliando eficiência, efetividade e satisfação do usuário ao utilizar o aplicativo (RODRIGUES, 2021). Apesar da similaridade temática, este estudo focaliza exclusivamente na organização de tarefas e emprega uma abordagem metodológica voltada para a análise de usabilidade após a implementação da aplicação móvel.

- ***Trello***: é uma aplicação que pode atuar de forma colaborativa para gerenciar projetos, assim como tarefas que promovem o alinhamento, a organização e a produtividade nas equipes que o utilizam. Essa colaboração que possui também permite controlar remotamente as atividades, mostrar seu andamento e direcioná-las para os membros do grupo (MULLER, 2020). Assim como o Tarefaça, esta aplicação implementa exclusivamente uma lista de tarefas para fins de organização, não alcançando um dos objetivos propostos neste projeto. No entanto, serviu como referência para as ideias de imersão e inspiração durante o desenvolvimento do projeto.

- ***Splitwise***: é uma ferramenta de gestão de despesas com finalidade de organizar as despesas de um grupo de pessoas, que permite adicionar todos os gastos, dividi-los entre os participantes de forma igual ou fracionada, direcionar a quem estão devendo, emite alertas de cobrança e solicitações de pagamento (DUBARD, 2019). Semelhante ao *Trello*, foi utilizado de referência para a construção do projeto, especificamente na área de gestão de despesas, onde serviu como objeto de estudo para a elaboração da interface de usuário.

3 Materiais e Métodos

Nesse capítulo serão abordados os materiais e metodologias necessários para o desenvolvimento do projeto proposto.

3.1 Materiais

Esta seção consiste em oferecer uma visão abrangente dos materiais em análise. os sistemas de *hardware* e *software* que foram fundamentais para o desenvolvimento do presente trabalho. A tabela e os itens apresentados abaixo tem como finalidade demonstrar detalhes sobre os equipamentos físicos, tecnologias, linguagens de programação e suas versões correspondentes que foram utilizadas para a realização deste trabalho.

Tabela 1 – Materiais utilizados

Item	**Descrição**
Hardware	Notebook Acer A316-53-52ZZ
	Processador Intel Core i7-7500U 2.5GHz com Turbo Boost para 3.1GHz
	Memória RAM 8GB DDR4
	HDD 1TB
Sistema Operacional	Windows 10 Pro, 64 bits.
IDE	Visual Studio Code 1.78.2
Linguagem de programação	Typescript / HTML / CSS

- HTML[1](Hypertext Markup Language) é o bloco de construção mais básico da *web*, uma linguagem usada para criar páginas por meio de marcadores e atributos, definindo o conteúdo que deve ser apresentado em um navegador.
- CSS[2](Cascading Style Sheets) é uma linguagem de estilo amplamente utilizada na *web* para descrever a apresentação visual e o *layout* de documentos HTML e XML(*Extensible Markup Language*), sendo possível especificar propriedades como cores, fontes, tamanhos e espaçamentos dos elementos, permitindo controlar sua exibição em diferentes meios.
- *TypeScript*[3] é uma linguagem de programação de código aberto desenvolvida pela *Microsoft*, semelhante ao *JavaScript* (uma linguagem de *script* simples para navegadores, com utilização de trechos de código em uma página *web*) que adiciona em sua sintaxe, uma tipagem estática à linguagem, fornecendo melhor detecção de erros.

[1] <https://developer.mozilla.org/pt-BR/docs/Web/HTML>. Acessado em mar/23
[2] <https://developer.mozilla.org/pt-BR/docs/Web/CSS>. Acessado em mar/23
[3] <https://typescriptlang.org/>. Acessado em mar/23

- *Node.JS*[4] é um ambiente de execução *JavaScript* assíncrono orientado a eventos, projetado para desenvolvimento de aplicações *web* escaláveis e de rede, realizando tratamento de conexões de forma simultânea.

- NPM[5](Node Package Manager) é o gerenciador de pacotes para o *Node.JS*, criado para facilitar o compartilhamento de módulos de código *JavaScript*. Ele inclui o registro do NPM, uma coleção de pacotes de código aberto para várias finalidades, sendo utilizado como um cliente de linha de comando para instalar e publicar pacotes (bibliotecas).

- *Visual Studio Code*[6] é um editor de código-fonte, com suporte integrado para *Node.JS*, *JavaScript* e *TypeScript*, que possui um depurador de código, integração com *Git*, um ecossistema de extensões para outras linguagens e tempos de execução, entre outras integrações e funcionalidades.

- *React*, conforme apresentado na 2.5.1

- *Redux*[7] é uma biblioteca para armazenamento e gerenciamento de estados de aplicações *JavaScript*, passando-os para um objeto global de fácil acesso, podendo ser acessado sem precisar que os componentes tenham ligações entre si.

- *Redux-saga*[8] é uma biblioteca que é usada como um *middleware* para o *Redux*, ajuda a gerenciar os efeitos colaterais das aplicações, permitindo tratar erros de maneira mais assertiva.

- *Express*[9] é um *framework* de aplicação que oferece recursos robustos para criar Interface de Programação de Aplicativos (APIs) de forma rápida e fácil. Ele fornece uma camada leve de funcionalidades essenciais para aplicações *web*, sem comprometer os recursos do *Node.JS*.

- *MongoDB*[10] é um banco de dados de código aberto orientado a documentos, projetado para armazenar uma grande escala de dados como objetos semelhantes a JSON, para facilitar o desenvolvimento e o dimensionamento de aplicativos.

- *MongoDB Atlas*[11] é um hospedeiro integrado ao *MongoDB* que fornece uma maneira fácil de gerenciar os dados na nuvem, podendo ser acessado de acordo com o *cluster* que está alocado junto ao banco de dados.

[4] <https://nodejs.org/pt-br/about>. Acessado em mar/23
[5] <https://www.npmjs.com/about>Acessado em jun/23
[6] <https://code.visualstudio.com/docs>. Acessado em mar/23
[7] <https://redux.js.org/>. Acessado em mar/23
[8] <https://redux-saga.js.org/>. Acessado em mar/23
[9] <https://redux-saga.js.org/>. Acessado em mar/23
[10] <https://mongodb.com/>. Acessado em mar/23
[11] <https://mongodb.com/docs/atlas/getting-started/>. Acessado em mar/23

- *Git*[12] é um sistema de controle de versão e de código aberto projetado para permitir ter várias ramificações locais que podem ser totalmente independentes umas das outras, mas também permite fundir e excluir essas linhas de desenvolvimento.

- *GitLab*[13] é uma plataforma de hospedagem de projetos que integra Desenvolvimento, Segurança e Operações (DevSecOps) e acompanha todo o ciclo de vida do projeto através de automação, promovendo segurança e conformidade.

- *Render*[14] é um hospedeiro na nuvem que para criar e executar aplicativos e sites com certificados, possuindo integração com o *Git*.

- *Miro*[15] uma plataforma colaborativa com uma ferramenta de design gráfico digital que possui colaboração em tempo real e assíncrona, além de integrar o fluxo de trabalho com outras tecnologias.

- *Trello*[16] é uma ferramenta de gestão de trabalho que permite criar quadros e listas de tarefas podendo ser ou não colaborativa, o que ajuda a simplificar, padronizar o processo de trabalho de uma forma intuitiva, produtiva e organizada.

- *Figma*[17] é uma plataforma de edição gráfica que cria prototipagem de projetos de *design* baseado em navegadores *web*. Possui configurações que simplificam as transcrição da interface para o código do desenvolvedor.

- *Lucidchart*[18] é uma ferramenta de diagramação que permite aos usuários colaborarem visualmente na interface, revisão e compartilhamento de gráficos e diagramas, aprimorando processos, sistemas e estruturas organizacionais.

- *Material UI*[19] é uma biblioteca de interface do usuário que fornece um conjunto abrangente de ferramentas para acelerar o desenvolvimento de novos recursos.

3.2 Métodos

Nessa seção serão abordados todos os aspectos metodológicos da pesquisa realizada, descrevendo-se os procedimentos necessários e úteis para a criação do protótipo de organização de despesas e tarefas. Esse estudo tem por finalidade realizar uma pesquisa para alcançar os objetivos propostos e melhor apreciação deste trabalho.

[12] <https://git-scm.com/>. Acessado em mai/23
[13] <https://about.gitlab.com/>. Acessado em mai/23
[14] <https://render.com/>. Acessado em mai/23
[15] <https://miro.com/pt/>. Acessado em mar/23
[16] <https://trello.com/pt-BR>. Acessado em mar/23
[17] <https://www.figma.com/design/>. Acessado em mai/23
[18] <https://www.lucidchart.com/pages>. Acessado em mai/23
[19] <https://mui.com/>. Acessado em mai/23

Com intuito de conhecer a problemática sobre a área de estudo, foi feita uma pesquisa em cima de trabalhos já realizados, como o *Trello* e *Splitwise*, que são dois parâmetros para a construção do protótipo no geral, observando suas estruturas e *design*, além da forma com que eles poderiam se integrar em uma única aplicação. Também foi estudado o trabalho de Rodrigues (2021) em que elege alguns aplicativos semelhantes ao proposto, disponíveis em lojas *on-line*, promovendo um comparativo de funções dos aplicativos analisados, para entender melhor quais funcionalidades são interessantes para se criar o protótipo final.

Após essa pesquisa foi utilizado a metodologia OKR para definição de objetivos e resultados esperados na fase de desenvolvimento, outra metodologia utilizada foi o *design thinking* para auxiliar nos processos e estratégias, identificar e tratar possíveis problemas relacionados a falhas de projeto. Outros métodos da ES também participaram da modelagem, como diagrama de caso de uso UML e BDD para documentar e visualizar as funcionalidades, além de melhorar a visão sobre requisitos e cenários para catalogar as ações do usuário, tornando-se um guia tanto para desenvolver, quanto para testar, promovendo um entendimento das tarefas e objetivos. Também foi utilizado o sistema de gestão visual *Kanban*, para gerenciar o desenvolvimento, com o intuito de proporcionar rapidez e flexibilidade diante as demandas levantadas, se orientando por um ciclo de vida e visando o sucesso da aplicação.

Logo que definido a parte estrutural que guiará o projeto, foi desenvolvido um *design* das telas, visando acessibilidade e facilidade na navegação entre suas funcionalidades, promovendo um ambiente fluido e intuitivo. Após a estruturação, foi configurado o ambiente com *Node.JS*, para assim começar a implementação da aplicação utilizando *React*, criando um SPA, com *MongoDB* como base de dados, integrando bibliotecas como *Redux* e *Express* para mais acessibilidade ao desenvolvedor e por fim disponibilizar a aplicação em uma plataforma de hospedagem na nuvem.

4 Modelagem e Implementação do Protótipo

Este capítulo apresenta a evolução da proposta desenvolvida por meio de metas estrategicamente definidas, as quais, em sua totalidade, foram essenciais para alcançar o êxito do trabalho proposto.

4.1 Definição de OKRs

O projeto foi iniciado utilizando a metodologia OKR desenhado através da plataforma *Miro*[1], com a definição de objetivos para o desenvolvimento prático, dividido entre estruturação do projeto, que visa entender como será o projeto diante ao problema, levantando requisitos, cenários de uso e *design* do protótipo e o desenvolvimento em si, onde houve o levantamento da parte de codificação, desde a configuração de ambiente, definição de *back-end* e *front-end*, banco de dados e hospedagem na nuvem.

Nessa fase foi determinado a estrutura que o projeto teria para atingir seu objetivo, assim foi definido, tendo quatro fases de pré-desenvolvimento:

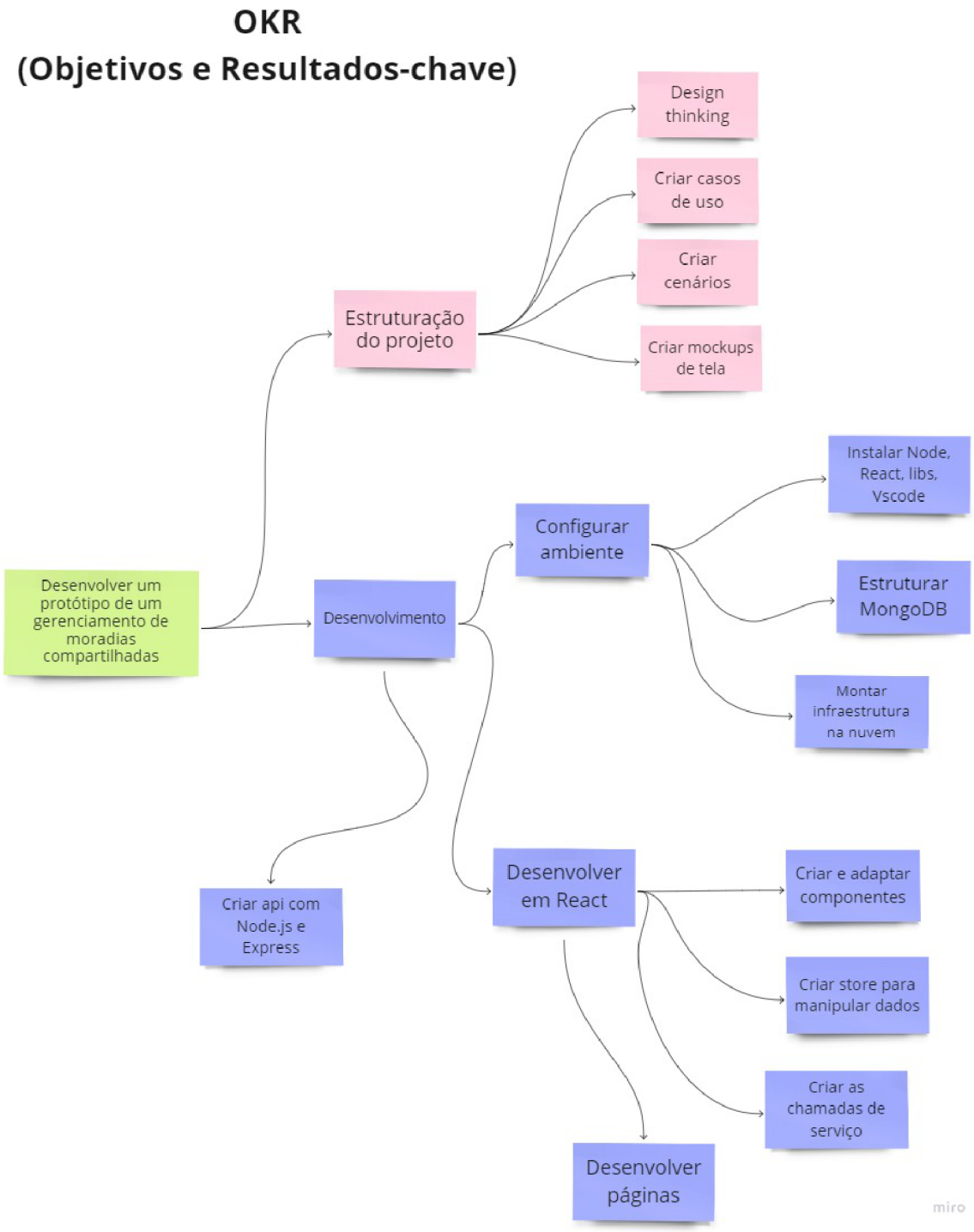
- o DT;
- diagrama de caso de uso;
- criação de cenários;
- criação da interface de usuário;

outra determinação foi em relação ao desenvolvimento, dividindo em:

- configuração de ambiente:
  - definição da a instalação do ambiente de desenvolvimento integrado, instalação de pacotes e bibliotecas auxiliares para executar o *React*;
  - definição do banco de dados e sua estrutura;
  - definição da estrutura de infraestrutura de hospedagem em nuvem;
- desenvolvimento do *back-end*:
  - criação da interface de programação de aplicação (API);
  - utilização da biblioteca *Express* para comunicação entre *React* e *Node.Js*;

[1] Disponível em: <https://miro.com/app/board/uXjVPfujPeE=/?share_link_id=294915323165>

Figura 7 – Recorte representativo do OKR



- desenvolvimento do *front-end*:
  - utilização do *Material UI* e criação ou adaptação de componentes;
  - criação da parte de *store*[2] que contém toda a árvore de estado da aplicação;
  - criação da camada de serviço, que se comunica com o *back-end* através de rotas;

[2] <https://redux.js.org/api/store>. Acessado em ago/22

– desenvolver as páginas da aplicação.

Dessa forma, ao realizar todas as etapas, chega-se ao objetivo final de desenvolver o protótipo.

4.2 Imersão ao problema através do *Design Thinking*

Através do *Miro*[3] foi ilustrado o *design thinking*, utilizando *Double Diamond* exemplificado na Figura 3, com as fases de:

- Imersão: para organizar as ideias, descobrindo sobre o problema de divisão de tarefas e custos de pessoas que moram juntas para gerar um *brainstorm* de ideias;
- Definição: para entender e definir os pontos principais, lapidado na fase de imersão;
- Ideação: para elencar as possíveis soluções para mapeamento do que seria ou não factível para se criar o projeto;
- Prototipação: para estipular os itens para implementação, divididos entre as funcionalidades definidas para o protótipo;

conforme ilustrado no Apêndice A. A Figura 8 permite ver uma breve representação do item citado, em que suas cores em sua ultima etapa diferenciam as funcionalidades elencadas para a implementação.

Figura 8 – *Double Diamond* aplicado com *Design Thinking*

[3] Disponível em: <https://miro.com/app/board/uXjVPfujPeE=/?share_link_id=294915323165>

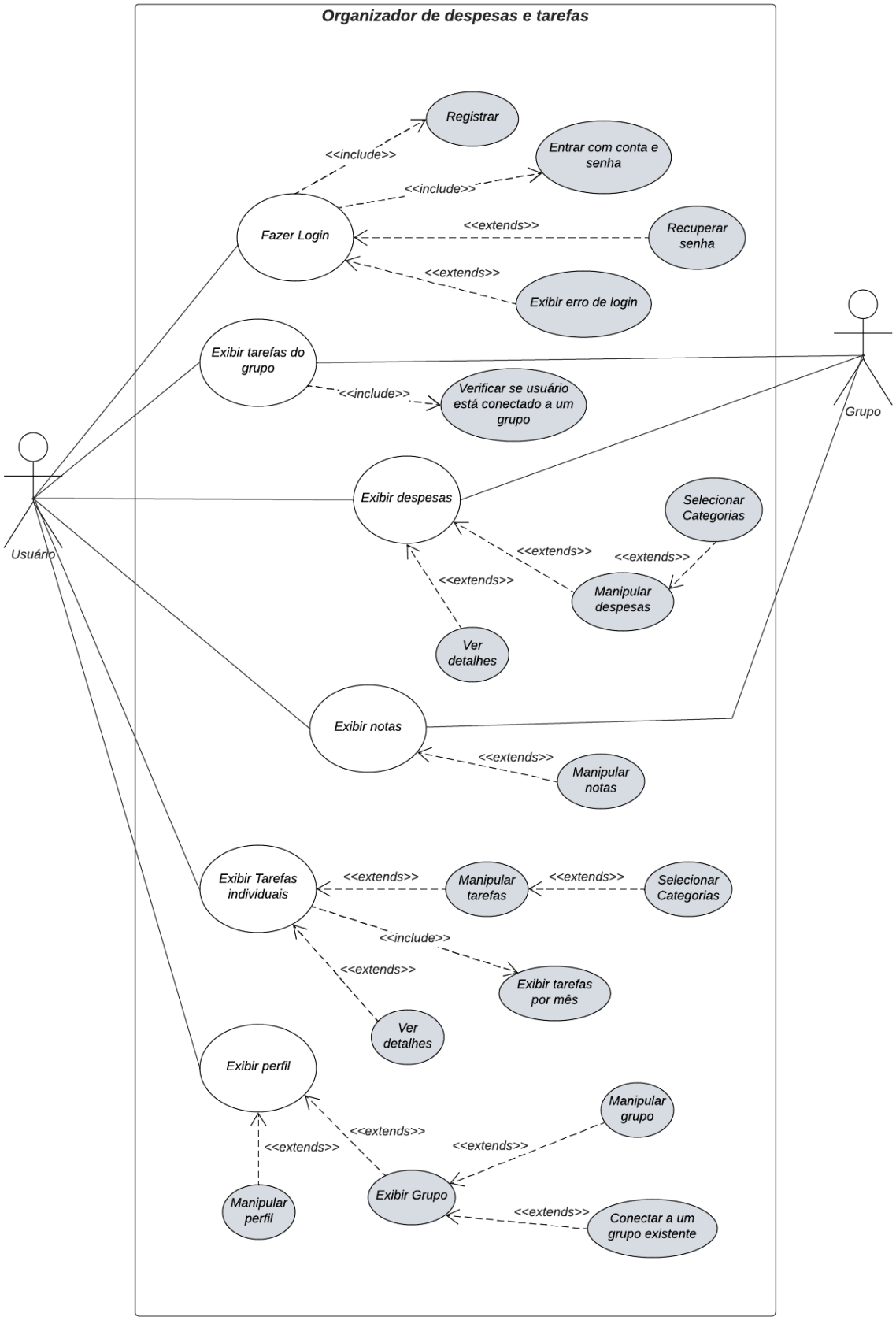
4.3 Diagrama de caso de uso UML

Na fase de definição do diagrama de casos de uso, desenhada no *Lucidchart*[4], é importante ter uma visão geral das funcionalidades que serão acessíveis aos usuários e ao grupo na aplicação. Além disso, é necessário modelar o fluxo em que os eventos podem ocorrer, sendo definido dois atores: usuário e grupo e seis funcionalidades: realizar *login*, acessar tarefas do grupo, acessar despesas, acessar tarefas individuais, acessar notas e acessar perfil. Conforme a Figura 9, cada funcionalidade possui suas derivações e conexões, podendo citar:

- realizar *login*: recurso em que leva o usuário para se registrar, entrar na aplicação com *e-mail* e senha e tendo a possibilidade de recuperar senha e se os dados estiverem errados, exibe erro nos campos de inserção, onde apenas o usuário acessa;
- acessar tarefas do grupo: atributo que exibe ao usuário as tarefas cadastradas por outras pessoas de um grupo que ele pertence, podendo ser visitado pelo usuário e pelo grupo;
- acessar despesas: função que exibe ao usuário as despesas cadastradas por ele e qualquer pessoa de um grupo que ele pertence, além disso realiza edição e deleção dos mesmos, se extendendo para permitir ver detalhes gerais dos itens cadastrados e também se extende para realizar cadastro e leitura de categorias relacionadas a despesas, podendo ser explorado pelo usuário e pelo grupo.
- acessar tarefas individuais: recurso que exibe ao usuário as tarefas cadastradas por ele, mostrando apenas informações privadas, além disso realiza edição e deleção dos mesmos, se extendendo para permitir ver detalhes gerais dos itens cadastrados e também se extende para realizar cadastro e leitura de categorias relacionadas a tarefas individuais, sendo utilizada apenas pelo usuário;
- acessar notas: função que exibe ao usuário as notas cadastradas por ele e qualquer pessoa de um grupo que ele pertence, além disso realiza edição e deleção dos mesmos, sendo interagido pelo usuário e pelo grupo;
- acessar perfil: atributo em que permite o usuário realizar Criar, Consultar, Atualizar, Deletar (CRUD) de seus dados, extendendo a possibilidade de realizar CRUD em um grupo, além de conectar a um grupo já existente, navegado apenas pelo usuário.

[4] Disponível em: <https://lucid.app/lucidchart/54644bbf-d885-4e81-ac2a-a9b301f2df42/edit?viewport_loc=-4214%2C-4884%2C5991%2C2801%2C0_0&invitationId=inv_8841e2a4-e656-4b9d-8dda-534ccd12ca7d>

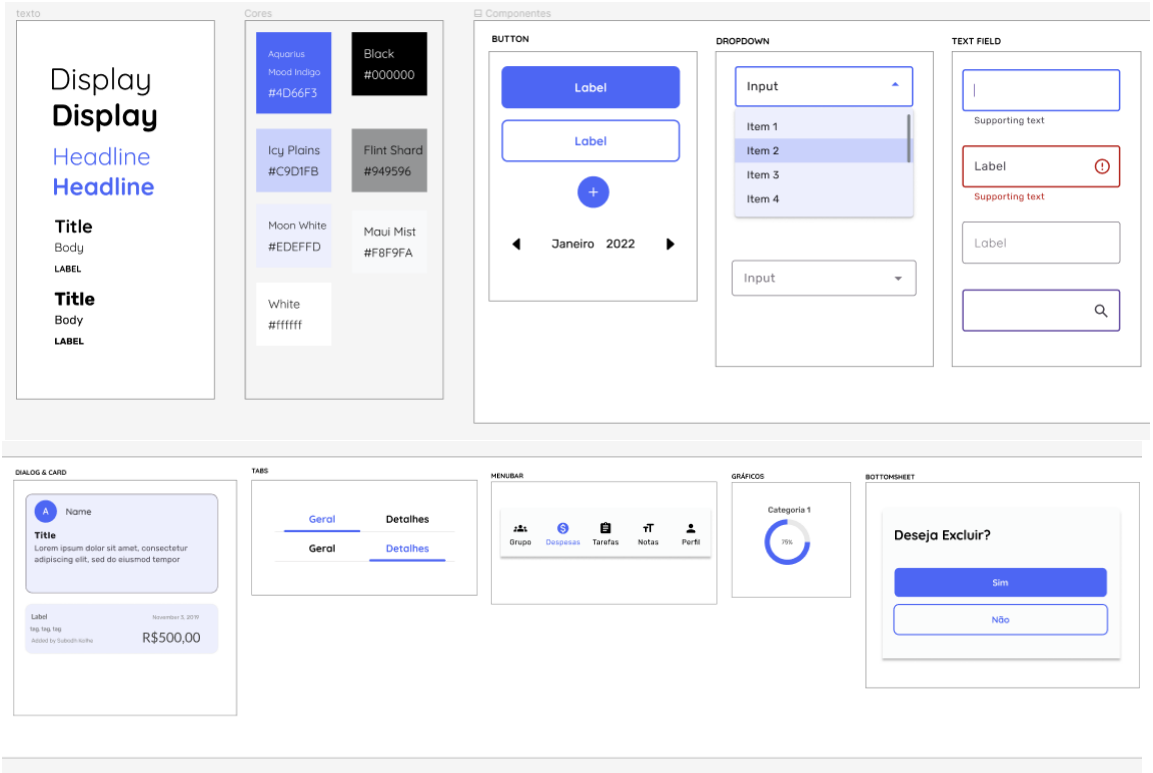
Figura 9 – Diagrama de casos de uso UML



4.4 Interface de usuário

Com essas definições feitas, seguiu-se para a parte de implementação da interface, através da plataforma de *design Figma*[5], tendo um cuidado para criar um ambiente que seja amigável, optando por cores mais neutras, um *design* interativo, onde consigam encontrar informações com facilidade, bem como acessar sua navegação, tentando manter uma consistência entre os elementos comuns da interface. Inicialmente foi criado um guia de componentes, onde foi desenhado botões, campos de texto, cartões, barra de navegação e todas as suas variações, bem como um guia de cores e fontes.

Figura 10 – Guia da interface

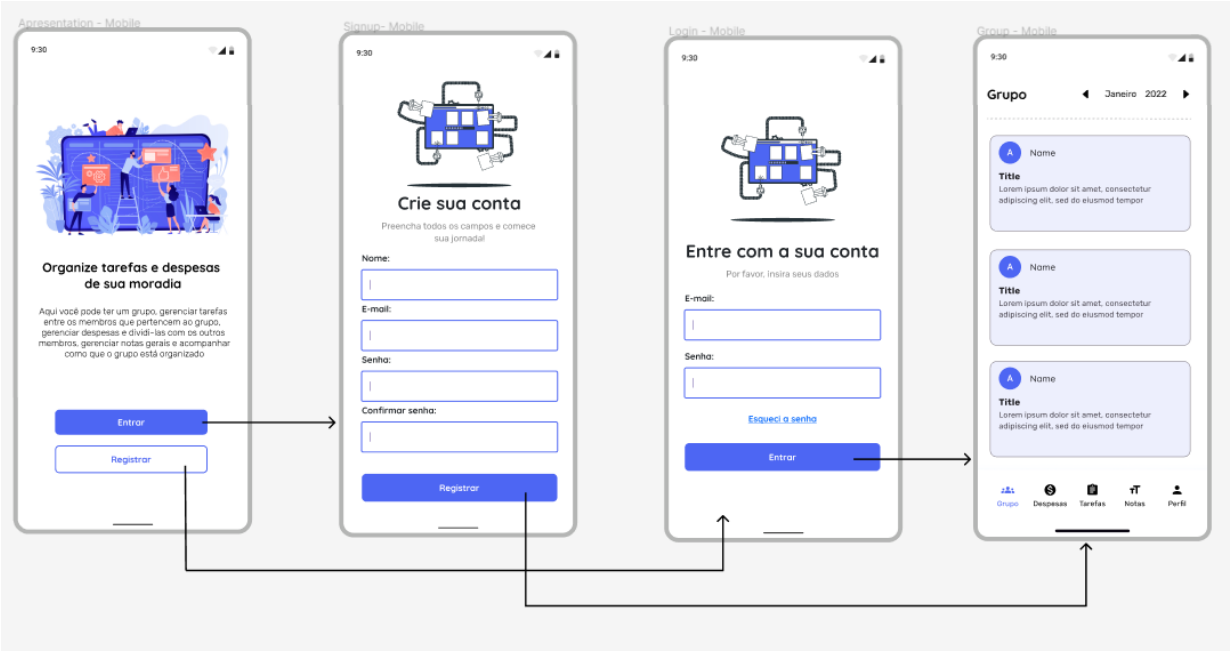


O *template* foi construído baseado em uma visualização de aplicativo móvel e toda ação está mapeada pelas setas que indicam para onde ir quando aquele componente for selecionado. As figuras nesta subseção são uma breve representação do mapeamento.

- Tela de apresentação: tela introdutória da aplicação, que possibilita o usuário a se registrar e entrar na aplicação para poder acessar e conseguir interagir com o protótipo.
- Tela de registro: tela onde o usuário se cadastra na aplicação, inserindo dados de nome, *e-mail*, senha e confirmação de senha.

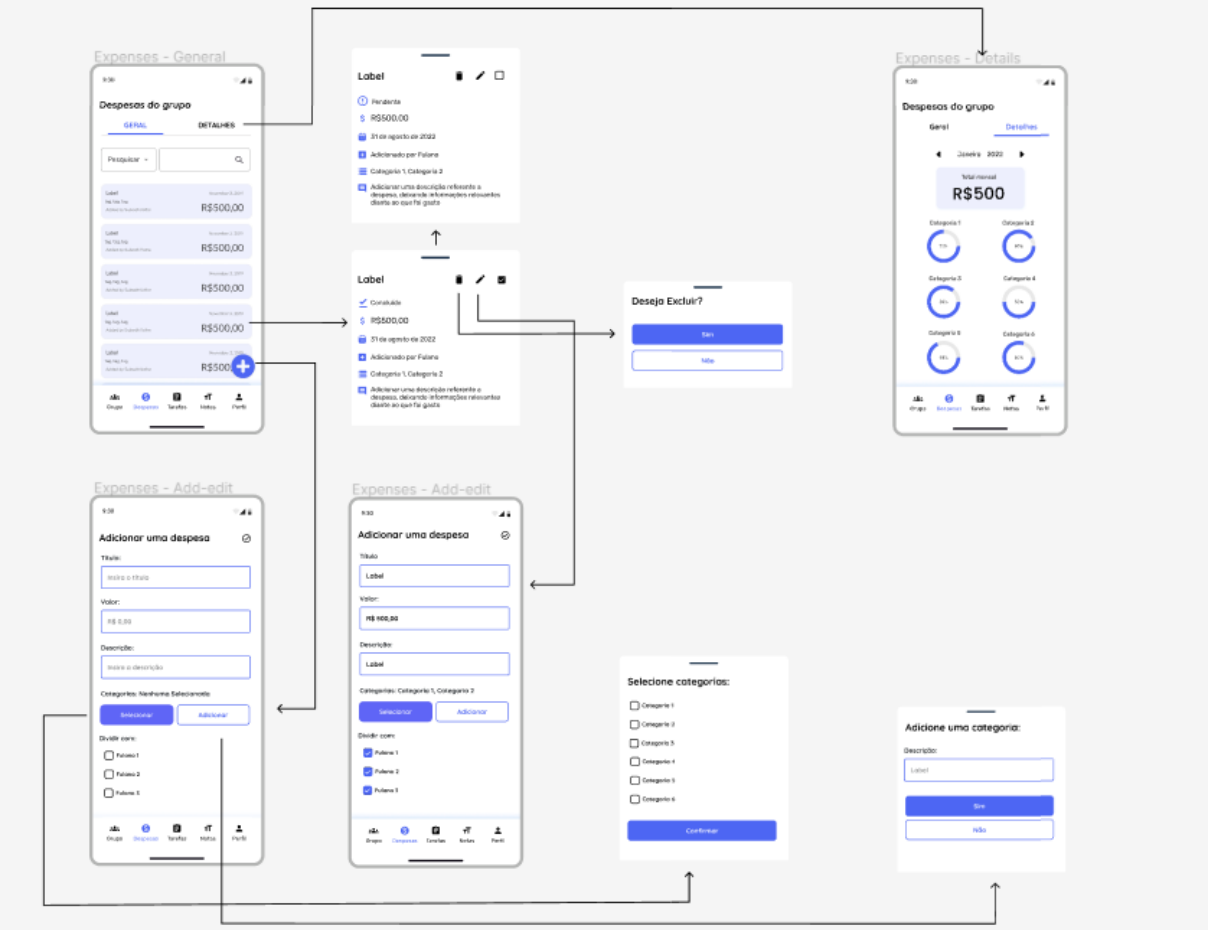
[5] Disponível em: <https://www.figma.com/file/STEbjZxEyVgK1LA8VGyuFk/Organizador-de-tarefas-e-despesas?type=design&node-id=0%3A1&mode=design&t=MhtlbsWLAecDlKtP-1>

Figura 11 – *Mockup* das telas de Apresentação, *Login*, Registro e Grupo



- Tela de entrada: tela em que o usuário possa se conectar informando seu *e-mail* e senha ou que possa recuperar seu acesso.
- Tela de grupo: tela de listagem de tarefas referente ao grupo, podendo ser visualizadas através do mês selecionado.
- Tela de despesas: essa tela possui uma aba de listagem de despesas criadas pelo usuário e despesas conectadas ao grupo que o usuário pertence e um campo de pesquisa com filtragem para realizar pesquisar por título, descrição, categoria e usuário, além de uma aba de visualização do total gasto e detalhes de categorias de despesas de acordo com o mês selecionado. Ao selecionar uma despesa, aparece um *bottomsheet* que exibe as informações e teclas de ações rápidas como deletar, editar e concluir. Ao selecionar o botão flutuante, direciona para a tela de adicionar uma nova despesa, podendo criar e selecionar uma categoria.
- Tela de tarefas: semelhante a tela de despesas, possui uma aba que lista tarefas porém nessa funcionalidade a listagem é apenas de tarefas exclusivamente criadas pelo usuário com a visualização de acordo com o mês selecionado, diferente da tela de Grupo que reune todas as tarefas do grupo pertencente. Ao selecionar um item, aparece um *bottomsheet* que exibe as informações e teclas de ações rápidas como deletar, editar e concluir e o botão flutuante direciona para adicionar uma nova tarefa, podendo criar e selecionar uma categoria. Possui outra aba de detalhamento de categorias criadas pelo usuário.

Figura 12 – *Mockup* da tela de Despesas



- Tela de notas: uma tela que lista todas as notas criadas pelo grupo, que contém um campo de pesquisa que realiza uma filtragem por todas as notas, seja pelo título ou descrição. Ao selecionar no botão flutuante, direciona para a criação de uma nova nota e ao selecionar um item da lista, abre a edição da mesma nota selecionada, contendo botões de ações rápidas como salvar e deletar.

- Tela de perfil: uma tela que contém as informações pessoais do usuário, permitindo editar os campos, assim como criar um grupo, adicionar ou excluir um amigo no grupo, além dos botões de ações rápidas que permite excluir, salvar ou sair da conta e finalizar a sessão.

4.5 Escrita dos cenários de comportamento

Assim que definiu-se os eventos, seguiu-se para a metodologia BDD, em que foram separados os cenários de cada funcionalidade elencada, especificando seus detalhes de pré-condição, condição e pós-condição, para facilitar o desenvolvimento através de comportamentos qualitativos, em que pode-se observar no Apêndice B.

Figura 13 – *Mockup* da tela de Tarefas

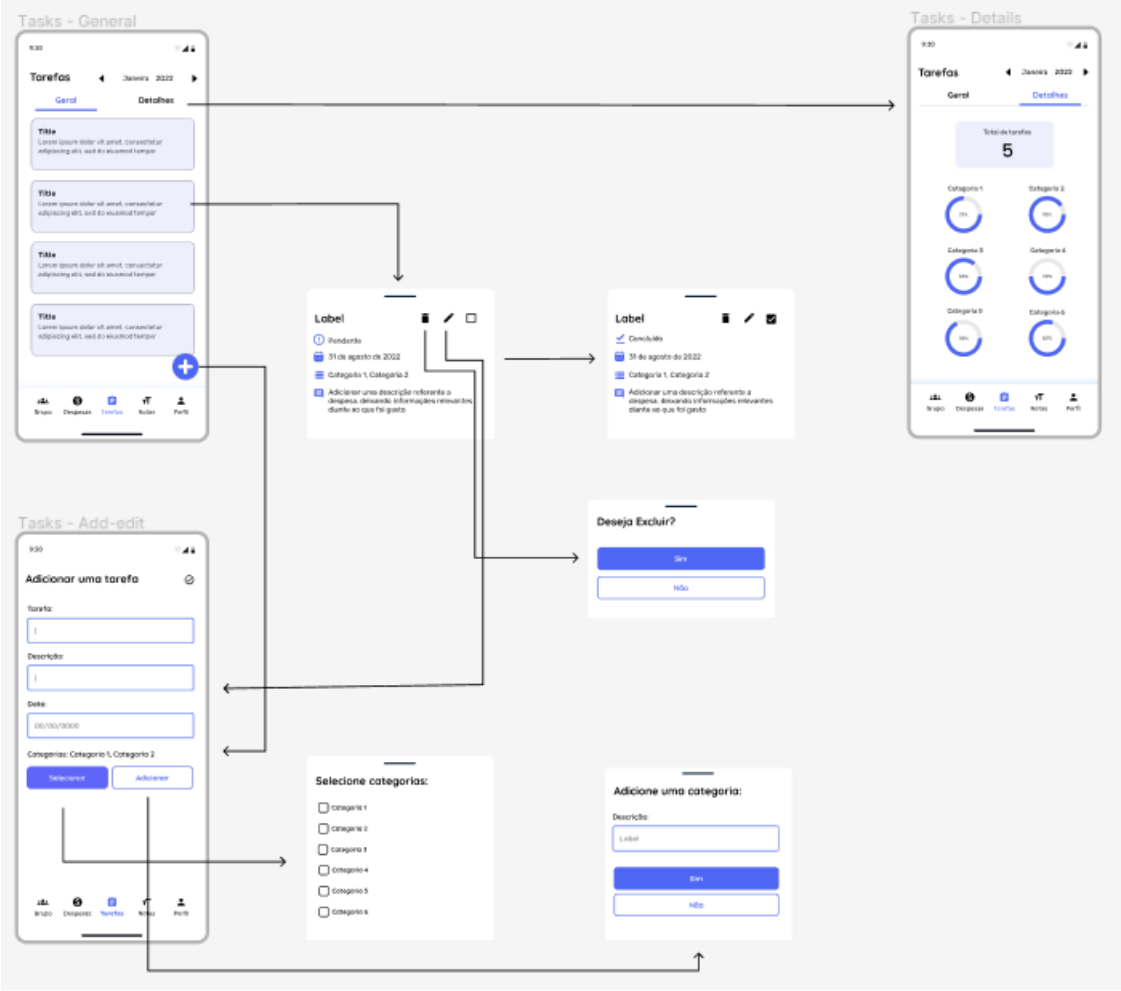


Figura 14 – *Mockup* da tela de Notas

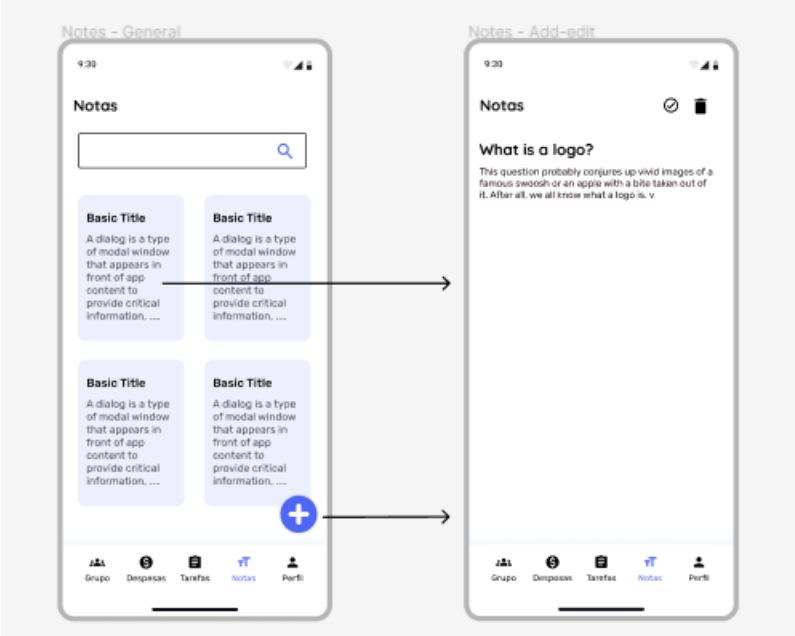


Figura 15 – *Mockup* da tela de Perfil

4.6 *Kanban*

Finalizado a parte de definição dos detalhes principais do projeto, criou-se um quadro no *Trello*, baseado na Figura 5 com quatro fases que aconteciam suceessivamente, são elas:

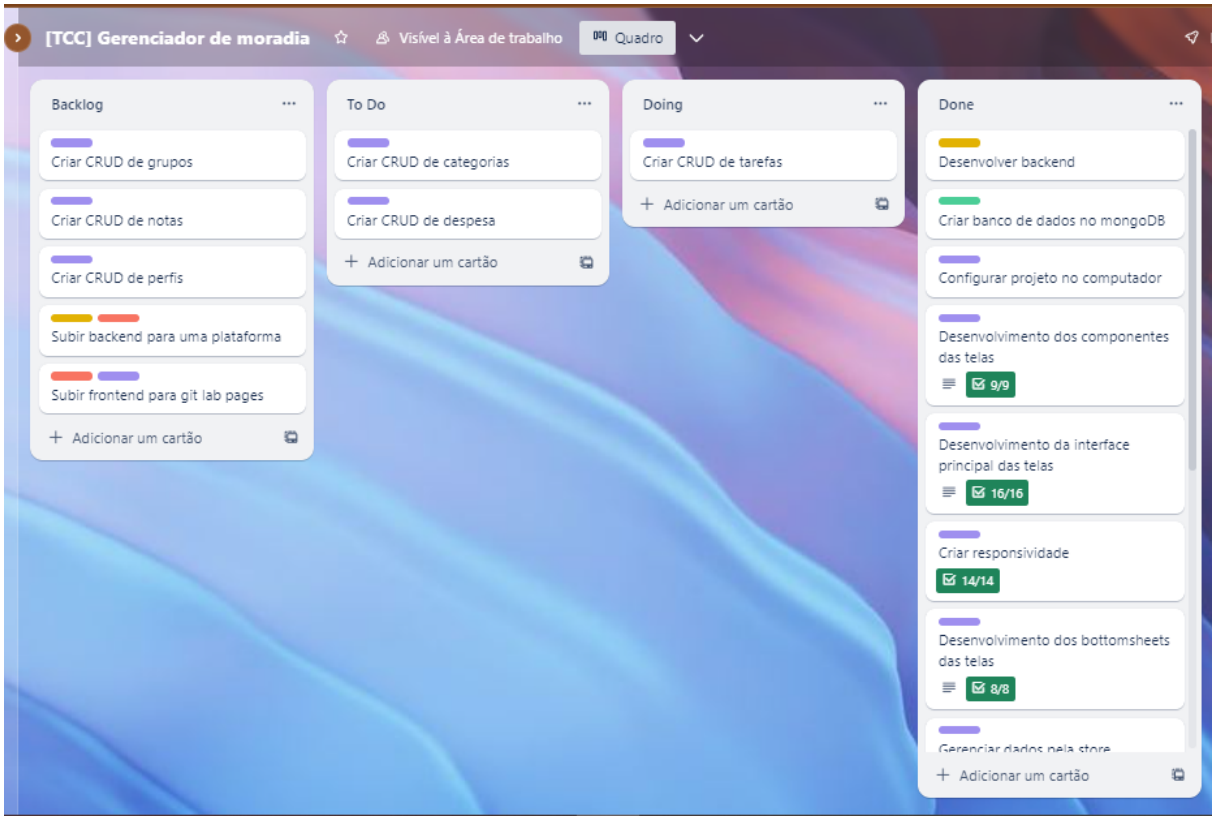
- *Backlog*: onde foi criado todas as tarefas a serem realizadas no projeto;
- *To Do*: em que se elencou as tarefas a serem desenvolvidas, observando sua prioridade;
- *Doing*: onde se dispunha de tarefas selecionadas para serem executadas;
- *Done*: lista de itens já realizados, definido pela conclusão de uma tarefa.

Uma nova demanda começa pelo *Backlog*, passando pela *To Do*, *Doing* e encerrando seu ciclo em *Done*, assim, fazendo mapeando contínuo para elencar melhorias que possam ser somadas ao desenvolvimento.

4.7 Configurando o ambiente

Por ser o mais prioritário, o desenvolvimento começou realizando as configurações necessárias para que o ambiente do *React* execute no computador. Foi instalado o *VSCode, Node.JS* e NPM como interpretadores, criado o "*create-react-app*", adicionado as bibliotecas principais de *react-redux* e *react-redux-saga* que realizam o mapeamento árvore de estados da aplicação, *Typescript* para utilização de uma linguagem tipada, instalado o *Git* para controle de versão e conectado a uma conta no *GitLab* para armazenar o projeto e então configurado um banco de dados não relacional, o *MongoDB*, hospedado na nuvem através do *MongoDB Atlas*.

Figura 16 – Quadro *Kanban* na ferramenta *Trello*



4.8 Programando o protótipo

O desenvolvimento começou criando a aplicação *back-end*, implementada em *TypeScript* utilizando a bilbioteca *Express* que auxilia na que efetua a comunicação entre o banco de dados e a aplicação *front-end* através da biblioteca *Express*. Toda a parte de regra de negócio está dentro do *front-end*, também desenvolvido em *TypeScript* em conjunto com o *React*.

O "*create-react-app*" disponibiliza um *template* inicial no formado para uma SPA, sendo assim elaboração do *front-end* se iniciou com a adição da biblioteca *Material UI* que possui vários componentes em sua base de dados, que foram analisados e adaptados para se assemelharem aos componentes do projeto visual originado na interface de usuário e após os componentes prontos, iniciou-se a codificação das telas.

Foram feitas adaptações na interface para melhorar o desempenho, como a remoção de componentes desnecessários, a simplificação de campos e a adição de botões de navegação, fazendo reduzir o número de renderizações e melhorando a experiência do usuário.

Foi configurado a *store* de cada funcionalidade, para gerenciar os estados dos componentes, realizando um mapeamento global na aplicação, também foi desenvolvido rotas para navegação entre os elementos de página. Os serviços foram criados para se

comunicarem com o *back-end* e realizarem a persistência, junto com o *redux-saga* que realiza a manipulação de dados antes do envio para seus respectivos endereços.

Dessa forma, foi utilizado a biblioteca *Express* para realizar a comunicação entre o aplicativo *web* e a API. Após toda a persistência configurada, foi viável disponibilizar a aplicação em uma plataforma de hospedagem em nuvem, que foram o *Render* e o *GitLab Pages*, em que possibilitou o acesso remoto da aplicação por meio da *internet*.

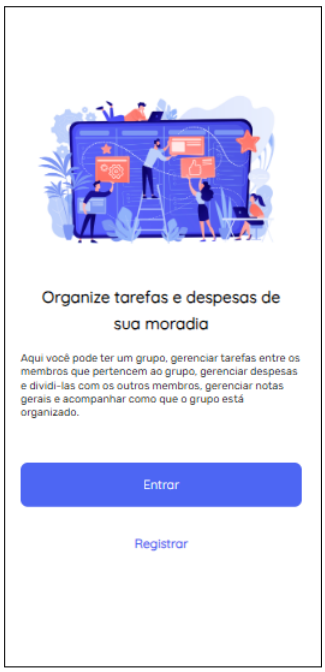
Por fim, foi realizado um refinamento para correção e melhorias sobre o comportamento dos componentes criados, aprimorando mais a interação entre as ações e as respostas que os elementos visuais. E dessa forma, é possível disponibilizar a aplicação em uma plataforma de hospedagem em nuvem.

5 Resultados

Neste capítulo, será realizada a exposição dos resultados obtidos na utilização das metodologias ágeis em conjunto com a interface de usuário, considerando todos os artefatos resultantes do processo de modelagem e planejamento da solução. O objetivo é demonstrar de forma clara e sistemática os frutos alcançados a partir da aplicação dessas abordagens, evidenciando sua contribuição para o desenvolvimento da solução proposta. A análise e interpretação dos resultados permitirão compreender o impacto das metodologias ágeis em conjunto com a interface de usuário e avaliar de forma precisa a eficácia e eficiência do trabalho realizado.

O protótipo[1] foi projetado com seu *layout* responsivo, se adaptando tanto para dimensões referentes a um dispositivo *mobile*, quanto se adequando a dimensões referentes que um visor horizontal possui. Os exemplos apresentados nessa seção contemplam uma visão vertical e no Apêndice C é possivel observar como as principais telas se adaptam a uma visão horizontal, diferente do apresentado.

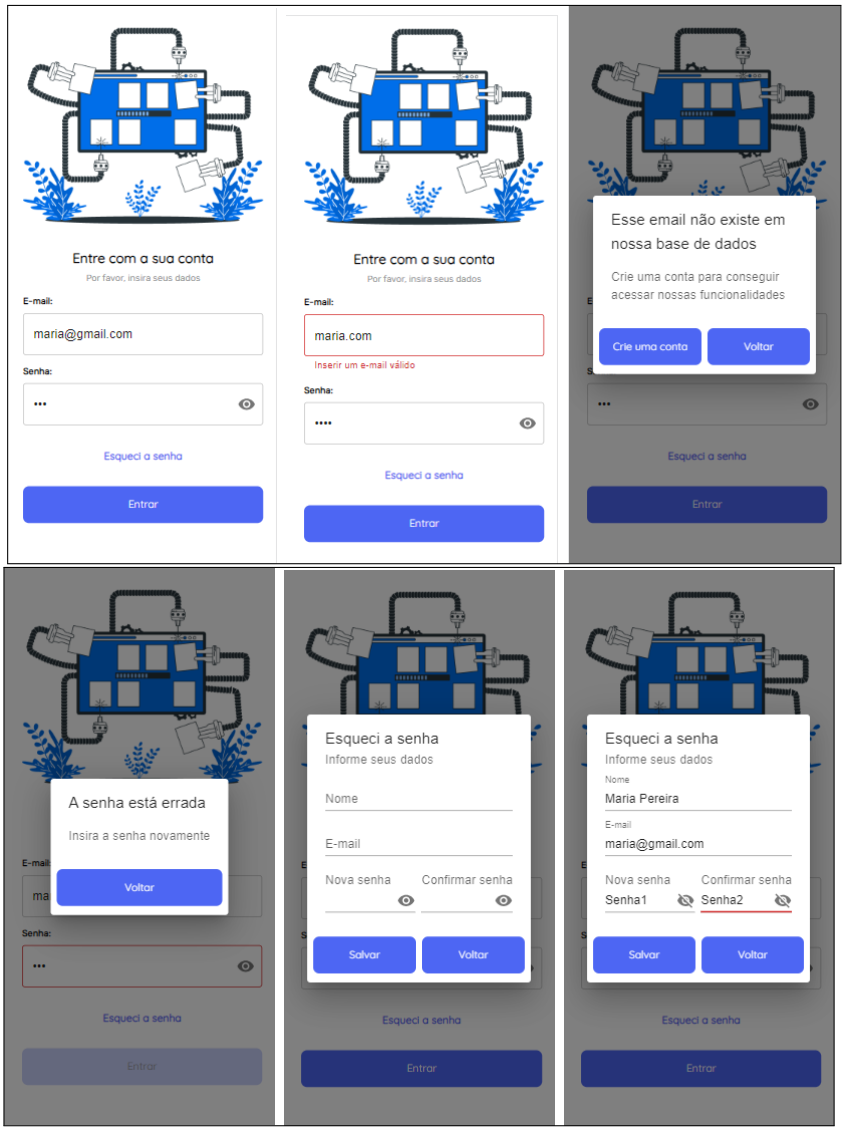
Figura 17 – Tela de Apresentação



Como rota principal da aplicação está a tela de introdução, em que é introduzido brevemente uma descrição da sua função, além de dois botões de navegação, o botão "Entrar" que direciona para a tela de *login*, caso o usuário já possua um cadastro e o botão "Registro" que direciona para a tela de registro para o usuário se cadastrar.

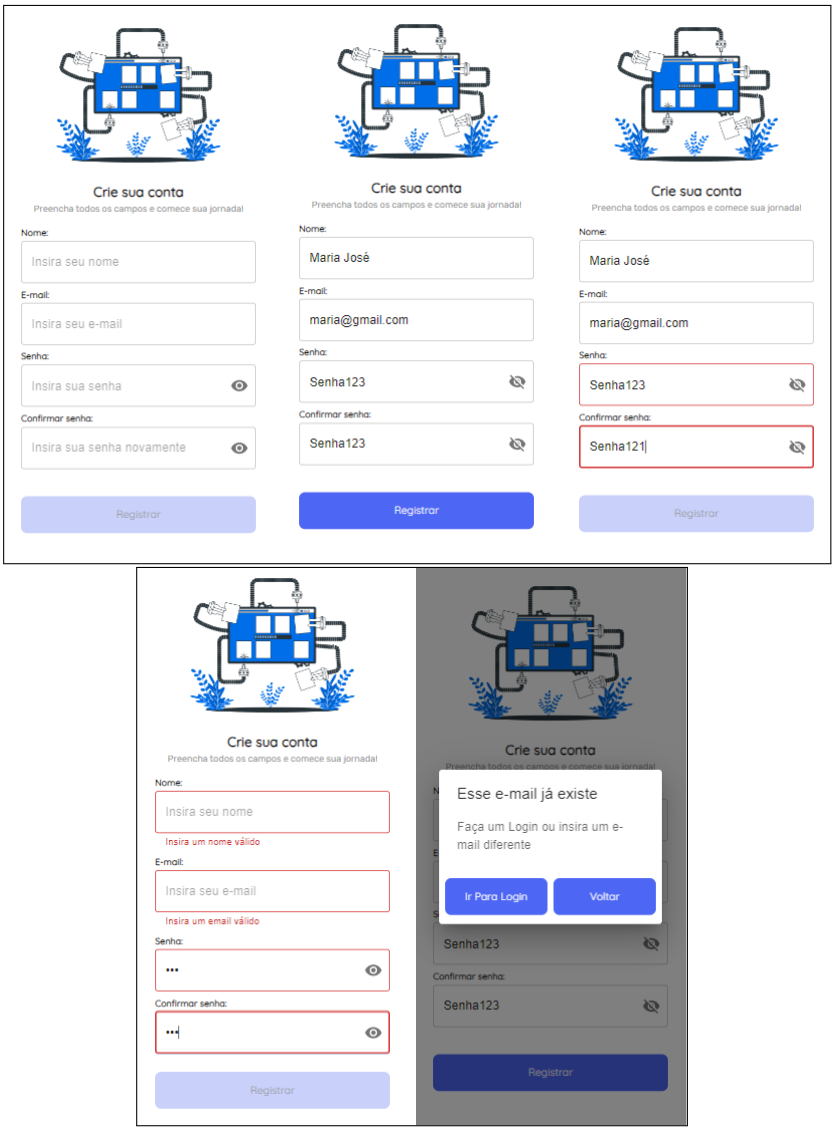
[1] Disponível em: <https://kimlamounier.gitlab.io/organizador-tarefas-despesas>

Figura 18 – Tela de *Login*, erros e *bottomsheets*



Na rota de *login*, é possivel preencher um *e-mail* e uma senha para realizar a conexão com a implementação, bem como recuperar a senha. No primeiro campo foi feito uma verificação com *regex* para validar as informações inseridas e caso não atenda aos requisitos, um erro acontece, exibindo uma mensagem indicativa. Caso esse campo esteja preenchido corretamente e tente se conectar, é verificado se aquele conteúdo existe na base de dados e se não existir, um *bottomsheet* aparece para poder encaminhar o usuário para a tela de Registro para assim, ter uma conta e poder acessar. Já no segundo campo é possível ver ou ocultar as informações digitadas e caso a senha não esteja em conformidade com a base de dados, uma mensagem é exibida em um *bottomsheet* informando que o dado está incorreto. Na recuperação de senha, um *bottomsheet* é exibido, pedindo os campos de nome, *e-mail*, nova senha e confirmação de nova senha, onde é verificado os dois primeiros itens e se eles estiverem coesos, atualiza a informação em seu repositório, como apresentado na Figura 18.

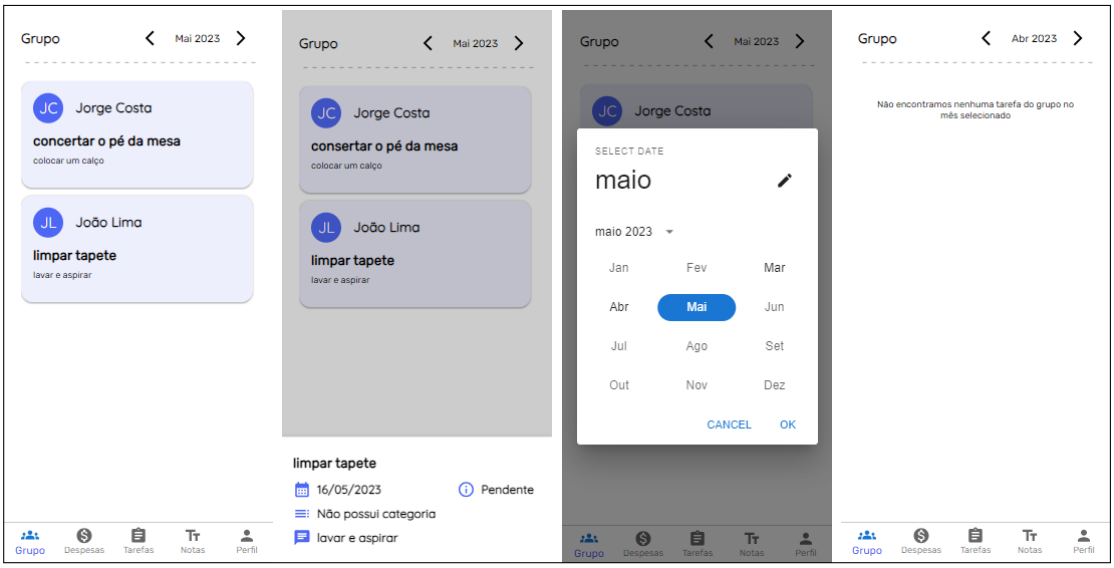
Figura 19 – Tela de Registro, erros e *bottomsheets*



Na rota de registro, podem-se inserir dados para realizar um cadastro, o botão "Registrar" fica desabilitado até todos os campos serem preenchidos corretamente. Os campos de verificação de senha podem ser abertos ou ocultos, revelando seu valor e a realização de checagem de compatibilidade entre os dados inseridos e os campos de inserção de texto, quando perdem o foco, realizam a verificação de dados, como caractéres especias, numeros e *regex*. Quando o botão é selecionado, é conferido se o *e-mail*, que é a chave mais importante do registro, já foi anteriormente adicionado e caso for falso, se conecta com sucesso e é direcionado para a tela "Grupo", caso verdadeiro, exibe um *bottomsheet* que dá a possibilidade do usuário de se conectar pelo *login*, como podemos acompanhar na Figura 19.

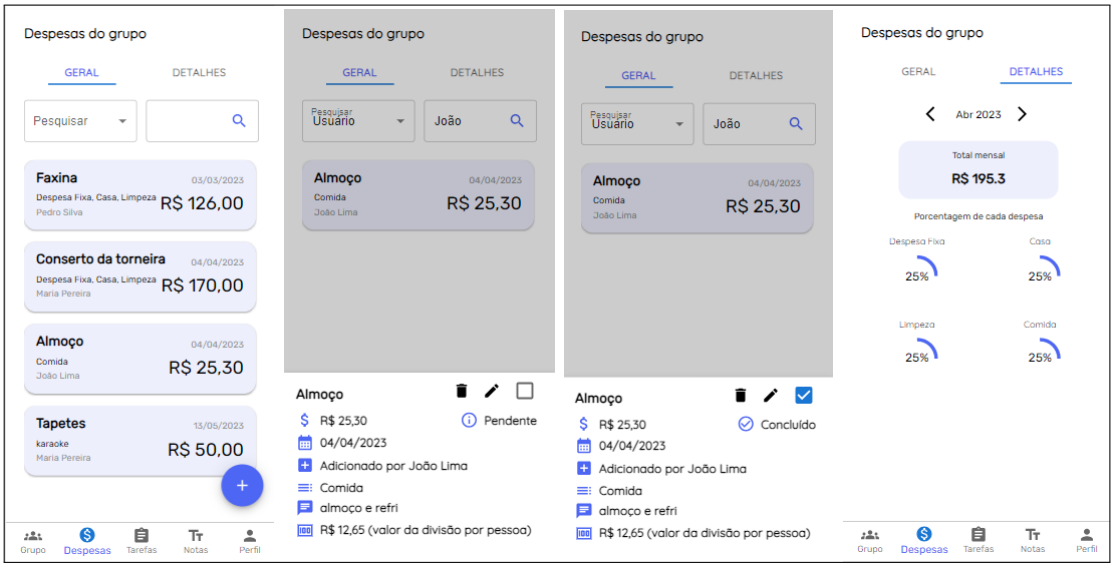
Já na rota de grupo, tem-se a alternativa de visualizar tarefas de outros membros do grupo, se já esteja conectado em um e exista tarefas no mês selecionado, mas caso for falso, exibirá uma mensagem de que não existem informações para serem exibidas.

Figura 20 – Tela de Grupo, erros e *bottomsheets*



Quando um item é exibido na lista, é possível selecionar e visualizar informações referentes à outras tarefas criadas por pessoas do grupo, de acordo com a Figura 20.

Figura 21 – Tela de Despesas, *bottomsheet* de informação e detalhes



Na rota de despesas, é apresentado duas abas: geral e despesas. Na aba “Geral” pode-se observar uma lista de todas as despesas criadas pelo usuário e pelo grupo pertencente, em que é exibido informações sobre o respectivo item quando selecionado e um botão flutuante que encaminha para adição de uma nova despesa, direcionamento apresentado na Figura 23. Além dos elementos informativos, é possível interagir com os ícones de: deleção, este que pede confirmação para poder excluir o item correspondente; edição, que encaminha para uma rota em que o usuário possa editar os dados; e um caixa de seleção que marca se uma despesa está concluída. Na aba “Detalhes” apresenta o valor total gasto no mês selecionado e o progresso circular de cada categoria utilizada nas despesas em

questão, como observado na Figura 21.

Figura 22 – Tela de Despesas e realização de pesquisa

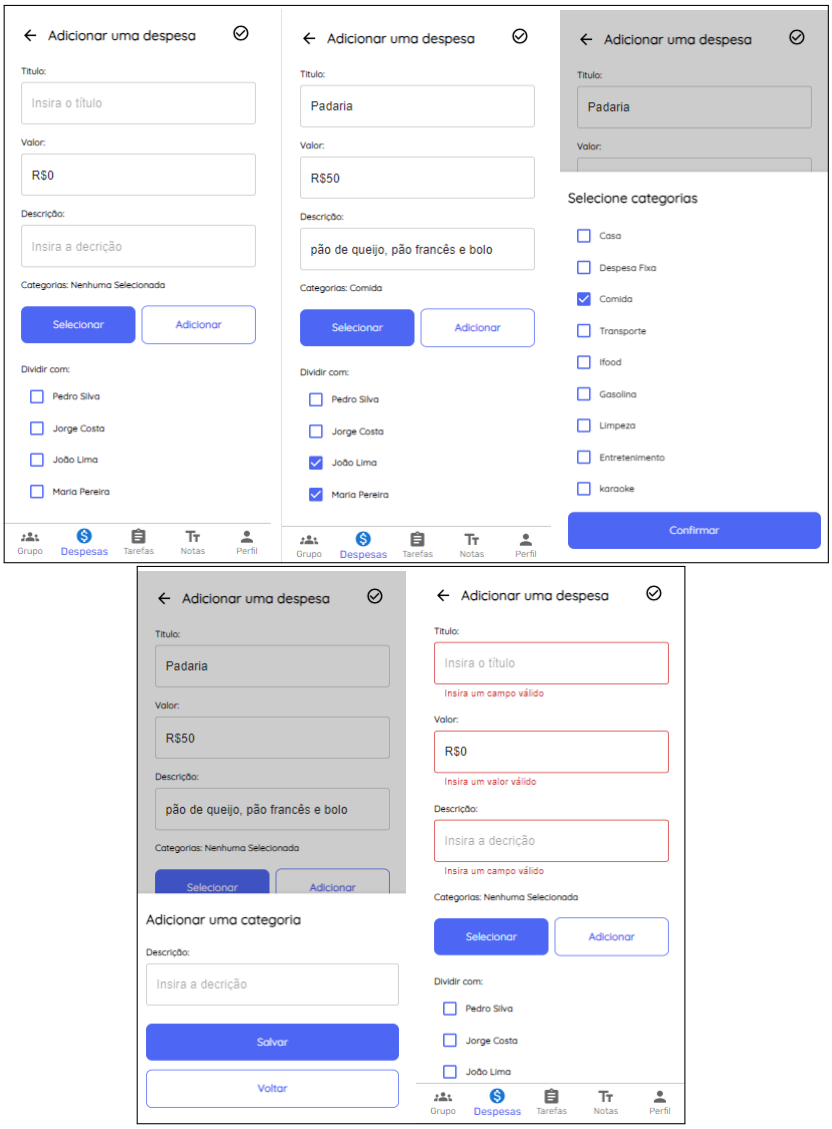
As despesas podem ser filtradas por valores referentes ao título, descrição, categoria e usuário, mostrando seu resultado quando um item tem compatibilidade entre seus dados e o dado de pesquisa ou entregando uma mensagem que informa a ausência de dados compatíveis, apresentado na Figura 22.

Já a Figura 23 mostra os estados possíveis para a criação ou edição de uma despesa, contendo três campos de inserção de valores, dois botões chamados “Selecionar” e “Adicionar” referentes a categoria e caixas de seleção em que se pode selecionar outros membros do grupo. Falando um pouco mais dos botões, na seleção de categoria é apresentado todas as despesas já criadas para que assim possa escolher os itens desejados e na adição de categoria pede-se sua descrição e uma confirmação para adicionar na base de dados. Se algum dos espaços de digitação estiverem vazios e houver a tentativa de salvar, é levantado um erro nos campos pedindo para que insira um valor válido para realizar a ação de salvar, caso os espaços estejam preenchidos corretamente, a despesa é salva e uma notificação aparece em tela informando seu sucesso.

Como visto na rota de grupo, tarefas individuais precisam ser criadas e é nessa rota que isso é realizado, como pode-se observar na Figura 25. Possui duas abas, semelhante a tela de “Despesas” no qual a primeira aba é listado tarefas, separadas pelo mês selecionado, bem como seu *bottomsheet* informativo que se porta semelhante. e quando não encontrado, uma mensagem é exibida e a segunda aba são de detalhes das categorias.

A parte de criação e edição de uma tarefa possui três campos de inserção, sendo um deles oferece a opção de selecionar uma data de maneira visual, em que apresenta um

Figura 23 – Tela de criação de Despesas

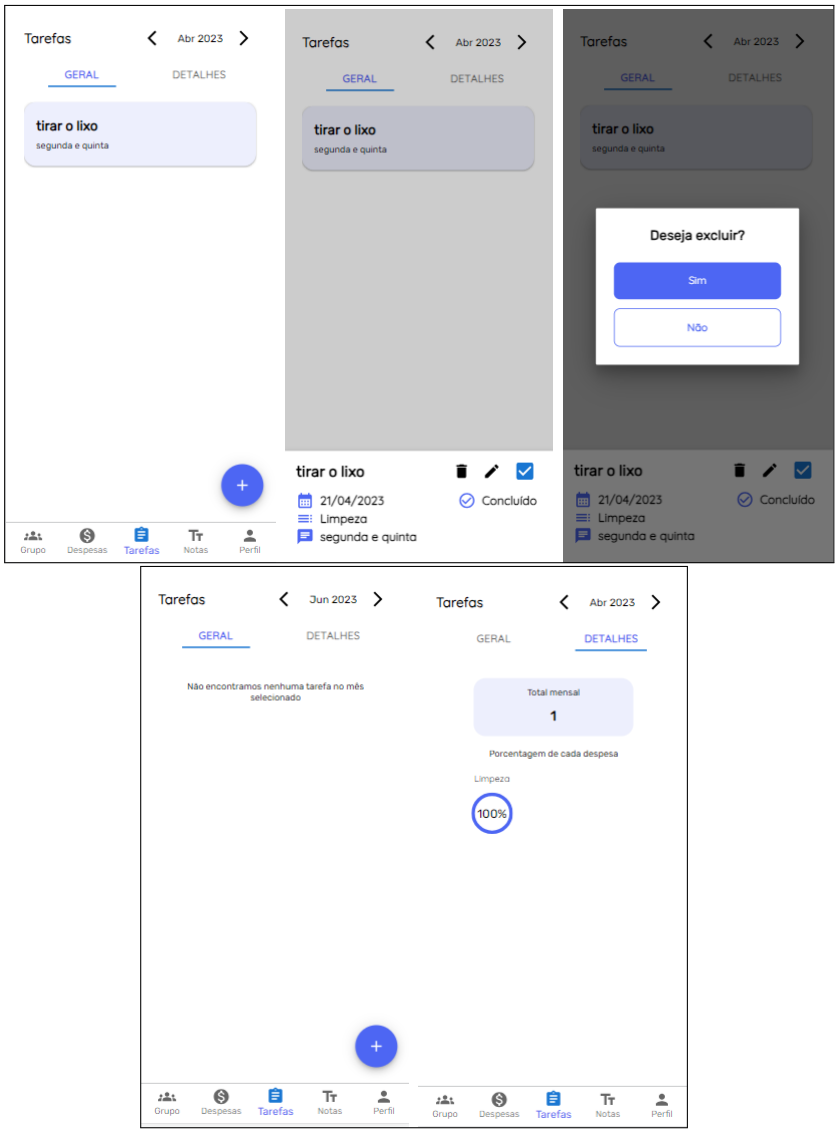


calendário mensal completo. Também existe dois botões de seleção e adição de categorias, semelhante ao de "Despesas" porém diferentes quando se trata de qual funcionalidade a classificação é criada. O processo de exibição de erros e armazenamento de dados também são assemelham a tela descrita anteriormente.

Na rota de notas é listado todas as notas criadas pelo usuário e pelo grupo, permitindo realizar uma pesquisa tanto pelo título, quanto pela descrição, além de um botão flutuante que encaminha para a criação de uma nova nota. Ao selecionar um elemento da lista, sua exibição ocorre no mesmo ambiente em de criação e, se modificado com qualquer valor, pede confirmação para eliminar esses dados, observando a Figura 26.

Por último, na rota de perfil, são apresentados os dados pessoais do usuário, acompanhados de um botão para excluir a conta e outro para realizar o logout da aplicação. Além disso, são disponibilizadas as opções de editar o perfil ou criar um grupo, caso o usuário não esteja atualmente associado a nenhum. Caso o usuário esteja vinculado

Figura 24 – Tela de Tarefas, *bottomsheet* de informação e detalhes



a um grupo, é possível editar as informações desse grupo e se desvincular dele, conforme Figura 27.

Ao criar um grupo, é gerado automaticamente um código que possibilita a adição de membros, além disso, ao realizar a edição do grupo, é possível incluir novos membros para participar e remover membros já existentes. Os botões em forma de ícone permitem: salvar as informações atualizadas do grupo no armazenamento persistente e realizar a desconexão, respectivamente, como visto na Figura 28.

Figura 25 – Tela de criação de Tarefas

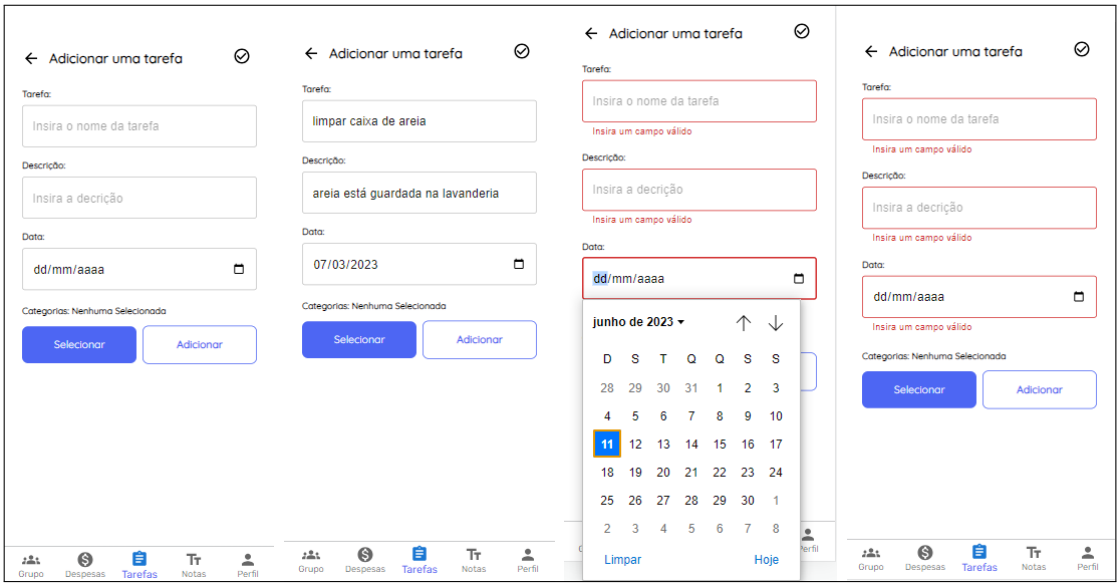


Figura 26 – Tela de Notas e pesquisa

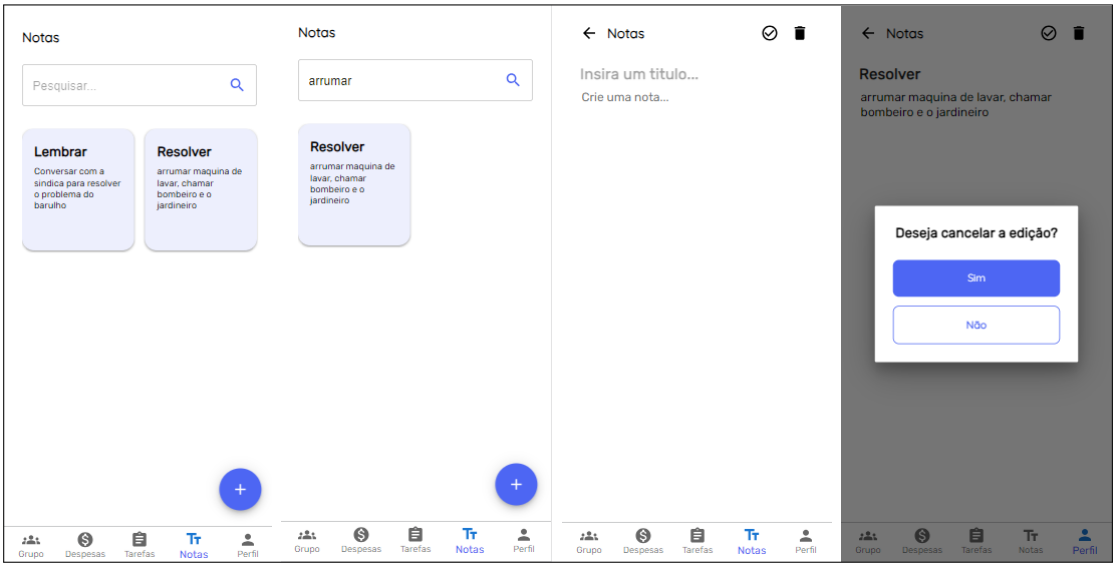


Figura 27 – Tela de Perfil e Perfil de Grupo

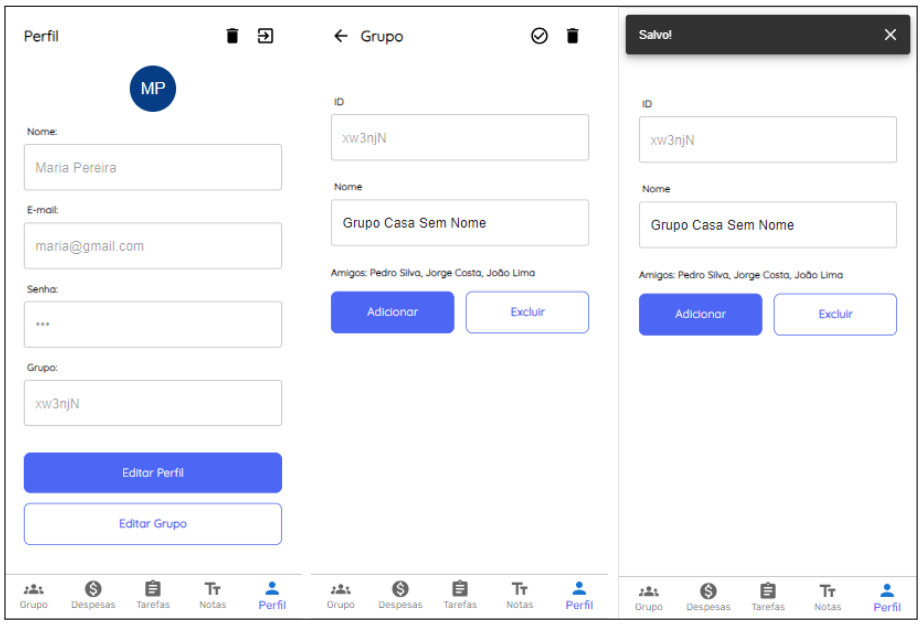
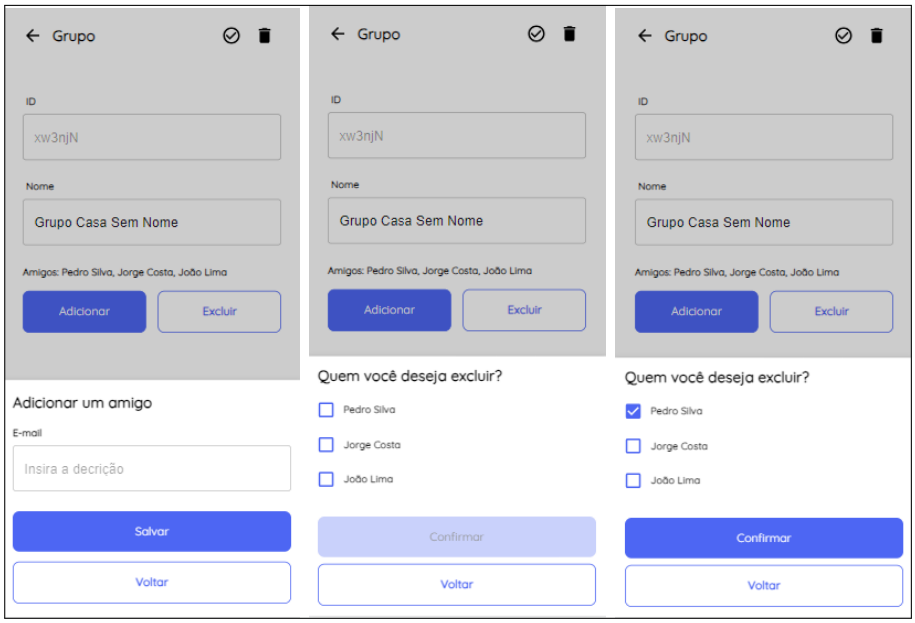


Figura 28 – Tela de Perfil de Grupo



6 Considerações Finais

No decorrer deste trabalho, foi possível constatar que a aplicação das práticas ágeis desempenhou um papel fundamental no enfrentamento das situações apresentadas, demonstrando sua eficiência e eficácia. A adoção dessas práticas proporcionou benefícios significativos, como flexibilidade, adaptabilidade e capacidade de resposta rápida diante das demandas do projeto, permitindo uma abordagem proativa diante de mudanças e imprevistos e resultando em um processo de desenvolvimento mais dinâmico, iterativo e colaborativo.

No entanto, é importante ressaltar que a exclusão da etapa de modelagem e planejamento pode comprometer significativamente o resultado do desenvolvimento do sistema. Essas fases são fundamentais para garantir a compreensão clara dos requisitos, a identificação de possíveis problemas e a definição de uma arquitetura sólida. A falta de um planejamento adequado pode levar a erros de implementação, falta de escalabilidade e dificuldades na manutenção do sistema, tornando-o menos confiável e custoso a longo prazo.

Portanto, recomenda-se que as práticas ágeis sejam aplicadas em conjunto com uma abordagem de modelagem da interface e planejamento adequados. Essa combinação permitirá aproveitar os benefícios da agilidade e da resposta rápida às mudanças, ao mesmo tempo promove um desenvolvimento com processos bem definidos, robustez e manutenção eficiente do *software*. É essencial encontrar um equilíbrio entre a flexibilidade das práticas ágeis e a necessidade de um planejamento adequado para garantir o sucesso do projeto.

Diante disso, é possível concluir que as práticas ágeis se mostraram indispensáveis para lidar com os desafios e as complexidades do projeto, permitindo um desenvolvimento mais eficiente e eficaz. Recomenda-se a continuidade da aplicação dessas práticas em futuros projetos, visando obter resultados semelhantes e promover uma cultura ágil de desenvolvimento de *software*.

6.1 Trabalhos Futuros

No decorrer deste estudo, foram identificados algumas oportunidades de aprimoramento que podem ser exploradas como trabalhos futuros. Essas melhorias têm como objetivo transformar o protótipo desenvolvido em uma ferramenta completa e abrangente. A seguir, será detalhado cada uma dessas áreas de aprimoramento:

- **Realização de testes de usabilidade através de heuristicas**: A usabilidade do sistema desempenha um papel fundamental na obtenção de uma experiência

satisfatória para os usuários. Recomenda-se realizar testes de usabilidade empregando heurísticas reconhecidas, que são diretrizes estabelecidas para avaliar a qualidade da interface do usuário e a interação com o sistema.

- **Utilização dos cenários para realização de testes automatizados**: Além dos testes de usabilidade, é válido explorar a utilização do *framework Cucumber* para a realização de testes automatizados. A execução desses testes automatizados ajudará a identificar possíveis falhas e garantir a estabilidade do sistema em diferentes cenários de uso. O *Cucumber* permite escrever cenários de teste de forma mais legível e compreensível, melhorando a confiabilidade e a robustez do sistema.

- **Desenvolvimento *mobile* do protótipo**: Desenvolvimento de uma versão mobile do protótipo, levando em consideração as características e particularidades de *smartphones* e *tablets*, a fim de proporcionar aos usuários maior comodidade e acessibilidade no acesso e uso do sistema.

- **Aprofundamento sobre experiência do usuário**: Recomenda-se, como direcionamento futuro, aprofundar os estudos sobre a experiência do usuário, utilizando técnicas como a análise de jornada do usuário, a criação de personas e a realização de pesquisas qualitativas. Essas abordagens ajudarão a compreender melhor as necessidades, expectativas e preferências dos usuários, permitindo o desenvolvimento de um sistema mais personalizado e alinhado a essas demandas.

- **Estudo de caso de aplicação do protótipo em usuários reais**: A realização de um estudo de caso que explorasse a aplicação prática do protótipo em um contexto real. Isso permitiria avaliar sua eficácia e identificar possíveis melhorias com base no *feedback* e nas experiências de usuários reais, para obter informações valiosas sobre a usabilidade, desempenho e aceitação da aplicação. Além disso, esse estudo poderia abordar diferentes cenários de uso, considerando variáveis como o perfil dos usuários, a diversidade de dispositivos e as demandas específicas de cada contexto.

Em resumo, os trabalhos futuros mencionados acima abrangem diferentes aspectos do projeto, desde a melhoria da usabilidade e realização de testes automatizados até a exploração de novas plataformas, aprofundamento na compreensão da experiência do usuário e estudo de caso em usuários reais. Cada uma dessas áreas oferece oportunidades valiosas para aprimorar o sistema e proporcionar uma experiência mais satisfatória e eficiente. Ao direcionar esforços e recursos para esses trabalhos futuros, será possível consolidar o sistema como uma solução completa no contexto em que se insere.

Referências

ALAIDAROS, H.; OMAR, M.; ROMLI, R. The state of the art of agile kanban method: Challenges and opportunities. *Independent Journal of Management & Production*, v. 12, n. 8, 2021. ISSN 2236-269X. Citado 2 vezes nas páginas 19 e 20.

ANDERSON, D. et al. Studying lean-kanban approach using software process simulation. In: *Agile Processes in Software Engineering and Extreme Programming*. Berlin: Springer Berlin Heidelberg, 2011. p. 12–26. ISBN 978-3-642-20677-1. Citado na página 20.

BARBOSA, S.; SILVA, B. *Interação Humano-Computador*. Elsevier Brasil, 2010. ISBN 9788535211207. Disponível em: <https://books.google.com.br/books?id=qk0skwr_cewC>. Citado 2 vezes nas páginas 21 e 23.

BROWN, T.; KATZ, B. *Design Thinking: Uma metodologia poderosa para decretar o fim das velhas ideias*. Rio de Janeiro: Elsevier, 2010. ISBN 978-85352-3862-4. Citado na página 15.

CANO, I. M. et al. Mecanismo de transformación de diagramas uml de casos de uso a código wscdl. *Research in Computing Science*, v. 93, p. 151–162, 12 2015. Citado na página 20.

CASTRO, F. *Guia do Iniciante para OKR*. 2020. 4-10 p. Citado na página 17.

CAVADAS, M. *Coliving no Brasil: como anda o mercado de moradia compartilhada?* 2022. Coworking Brasil. Disponível em: <https://coworkingbrasil.org/news/coliving-no-brasil-como-anda-o-mercado-de-moradia-compartilhada/>. Acesso em: 04 de maio de 2022. Citado na página 14.

CHI, N. H. N. *Functional Reactive Programming in React Application*. 2018. Theseus. Disponível em: <https://www.theseus.fi/bitstream/handle/10024/144018/ChiNguyen_1304976_revised.pdf?sequence=1&isAllowed=y>. Acesso em: 3 de junho de 2022. Citado na página 24.

DUBARD, C. *O seu organizador financeiro: as melhores ferramentas de controle financeiro para você se organizar!* 2019. Magnetis. Disponível em: <https://blog.magnetis.com.br/melhores-ferramentas-de-controle-financeiro/>. Acesso em: 24 de março de 2022. Citado na página 25.

ESTUDIO MARTE. *Design Thinking*. 2022. Estúdio Marte. Disponível em: <https://www.marte.design/knowledge/design-thinking#info>. Acesso em: 06 de junho de 2023. Citado 3 vezes nas páginas 15, 16 e 17.

FARO. *Sem estresse! 9 dicas para morar em república e conviver bem.* 2019. Faculdade de Rondonia. Disponível em: <https://faro.edu.br/blog/sem-estresse-7-dicas-para-morar-em-republica-e-conviver-bem/>. Acesso em: 20 de março de 2022. Citado na página 12.

GALITZ, W. O. *The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques*. New Jersey: John Wiley & Sons, Inc., 2007. ISBN 0470053429. Citado na página 22.

GODINHO, T. *Como organizar: Vida em república*. 2013. Vida Organizada. Disponível em: <https://vidaorganizada.com/2013/05/16/vivendo-em-republica-como-se-organizar/>. Acesso em: 22 de março de 2022. Citado na página 12.

GROZDANIC, L. *How Coworking and Coliving are Redefining Space as a Service*. 2016. ArchDaily Articles. Disponível em: <https://www.archdaily.com/785550/how-coworking-and-coliving-are-redefining-space-as-a-service>. Acesso em: 04 de maio de 2022. Citado na página 14.

GUEDES, G. T. A. *UML 2: Guia Prático*. São Paulo: Novatec, 2014. Citado na página 20.

HPI ACADEMY. *What is Design Thinking?* 2023. HPI Academy. Disponível em: <https://hpi-academy.de/en/design-thinking/what-is-design-thinking/>. Acesso em: 06 de junho de 2023. Citado na página 17.

KOMPERLA, V. et al. React: A detailed survey. *Indonesian Journal of Electrical Engineering and Computer Science*, v. 26, p. 1710, 06 2022. Citado 2 vezes nas páginas 23 e 24.

LI, P.; LIU, Z. Research on MR navigation design based on double diamond model: a case of South China Botanical Garden. In: LU, Y.; CHENG, C. (Ed.). *Third International Conference on Computer Science and Communication Technology (ICCSCT 2022)*. SPIE, 2022. v. 12506, p. 125061W. Disponível em: <https://doi.org/10.1117/12.2662602>. Citado na página 16.

MORETTI. *Programação Reativa: O que é?* 2020. Medium Article. Disponível em: <https://moretti-dev.medium.com/programa%C3%A7%C3%A3o-reativa-o-que-%C3%A9-63e0e93db941>. Acesso em: 25 de março de 2022. Citado na página 24.

MULLER, E. *Trello: Saiba como usar e aumente sua produtividade*. 2020. B2B Stack. Disponível em: <https://blog.b2bstack.com.br/trello-como-usar/>. Acesso em: 23 de março de 2022. Citado na página 25.

NIVEN, P.; LAMORTE, B. *Objectives and Key Results: Driving Focus, Alignment, and Engagement with OKRs*. New Jersey: John Wiley & Sons, 2016. Citado na página 17.

NORTH, D. The evolution of behavior-driven development. *Better Software*, v. 2006, n. 03, 2006. Citado na página 18.

PEREIRA, A. S.; PETRUCELLI, E. E. Análise das semelhanças e diferenças entre utilizar jsx ou javascript puro ao construir interfaces com react. *Revista Interface Tecnológica*, v. 16, p. 136–145, 2019. Citado na página 24.

PINHEIRO, T.; FERREIRA, L. *Design Thinking Brasil: Empatia, Colaboraração e Experimentação para Pessoas, Negócios e Sociedade*. São Paulo: Elsevier Brasil, 2017. ISBN 9788535232578. Citado na página 16.

PM3. *OKR: o que é a metodologia e como definir seus objetivos*. 2022. PM3 Cursos. Disponível em: <https://www.cursospm3.com.br/blog/okr-o-que-e-como-definir-seus-objetivos/>. Citado na página 18.

POSCH, S. *Kanban*. 2019. Integrated Consulting Group. Disponível em: <https://www.integratedconsulting.eu/insights/kanban/>. Acesso em: 06 de junho de 2023. Citado na página 19.

PREECE, J. et al. *Human-Computer Interaction*. United Kingdom: Addison-Wesley Longman Ltd., 1994. Citado na página 22.

REACT. *React: The library for web and native user interfaces*. 2023. Disponível em: <https://react.dev/>. Acesso em: 06 de maio de 2023. Citado na página 24.

REZENDE, D. A. *Engenharia de Software e Sistemas de Informação*. Rio de Janeiro: Brasport, 2006. v. 3. 2-3 p. Citado na página 14.

RODRIGUES, R. M. D. *Tarefaça: Ferramenta Gamificada Para Gestão De Moradias Compartilhadas*. 2021. Universidade Federal De Ouro Preto, Ouro Preto - MG - Brasil. 53 p. Disponível em: <https://monografias.ufop.br/handle/35400000/3018>. Acesso em: 05 de maio de 2022. Citado 2 vezes nas páginas 25 e 29.

SCOTTI, G. A. d. C. *Análise Comparativa de Front-Ends de Frameworks Baseados Em Javascript*. 2019. Universidade Federal de Minas Gerais, Belo Horizonte - MG - Brasil. Disponível em: <https://hdl.handle.net/1843/32038>. Acesso em: 8 de junho de 2022. Citado na página 23.

SHNEIDERMAN, B.; PLAISANT, C. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Fifth edition. United Kingdom: Addison-Wesley Publ. Co., 2009. Citado na página 22.

SILVA, A. G. da. *A importância dos métodos ágeis na engenharia de software*. 2016. Universidade Federal Fluminense, Niterói - RJ - Brasil. Disponível em: <https://app.uff.br/riuff/handle/1/5488>. Acesso em: 28 de março de 2022. Citado na página 15.

SILVA, V.; ADAMATTI, D.; BARBOSA, R. Princípios de usabilidade e a importância do usuário no projeto de interfaces. *Revista Jr de Iniciação Científica em Ciências Exatas e Engenharia*, v. 1, p. 29–37, 06 2016. ISSN 2236-0093. Citado na página 23.

SOMMERVILLE, I. *Engenharia de software*. [S.l.]: Pearson Prentice Hall, 2011. ISBN 9788579361081. Citado na página 15.

STAFF, C. React: Facebook's functional turn on writing javascript. *Commun. ACM*, Association for Computing Machinery, New York, v. 59, n. 12, p. 56–62, dec 2016. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/2980991>. Citado na página 25.

TAVARES, H. et al. A tool stack for implementing behaviour-driven development in python language. 07 2010. Citado 2 vezes nas páginas 18 e 19.

UNIFOA. *Quais os principais desafios e benefícios de morar em república?* 2017. Centro Universitario de Volta Redonda. Disponível em: <https://blog.unifoa.edu.br/quais-os-principais-desafios-e-beneficios-de-morar-em-republica/>. Acesso em: 21 de março de 2022. Citado na página 12.

VIANNA, M. et al. *Design Thinking: Inovação em negócios*. Rio de Janeiro: MJV Press, 2012. 161 p. Citado na página 16.

VINTEROVA, J. *Repúblicas de Coimbra*. 2008. Masarykova univerzita, Filozofická fakulta, Brno. Disponível em: <https://is.muni.cz/th/wzrog/>. Acesso em: 20 de março de 2022. Citado na página 12.

APÊNDICE A – Design Thinking

APÊNDICE B – Cenários BDD

- **Cenários de Registro**
  - **Cenário 1**: Criar uma conta

    Dado que o usuário não tenha cadastro no sistema
    E deseja se cadastrar
    Quando o usuário solicitar criar um registro
    E preencher os campos de nome, e-mail, senha e confirmação de senha
    E o sistema habilitar o botão de registro
    Então registra uma conta
    E exibe uma mensagem de sucesso
    E redireciona para a tela "Grupo"
  - **Cenário 2**: Manter o botão de se registrar desabilitado quando todos os campos não são preenchidos

    Dado que o usuário não tenha cadastro no sistema
    E deseja se cadastrar
    Quando o usuário solicitar criar um registro
    E não preencher todos os campos
    Então o botão de registro fica desabilitado
  - **Cenário 3**: Manter o botão de se registrar desabilitado quando os campos de senha e confirmar senha não são iguais

    Dado que o usuário não tenha cadastro no sistema
    E deseja se cadastrar
    Quando o usuário solicitar criar um registro
    E preencher todos os campos, porém o campo de senha e confirmar senha não forem idênticos
    Então o botão de registro fica desabilitado
  - **Cenário 4**: Exibir mensagem de erro ao tentar registrar um e-mail registrado no sistema

    Dado que o usuário tenha um cadastro no sistema
    E tente se cadastrar novamente
    Quando o usuário tentar registrar um e-mail existente na criação de uma conta
    Então entra o sistema exibe uma mensagem indicando que o e-mail existe
  - **Cenário 5**: Exibir mensagem de erro ao inserir um e-mail inválido

Dado que o usuário queira se registra
Quando o usuário tentar registrar um e-mail inválido
Então entra o sistema exibe uma mensagem indicando que o e-mail não está no padrão correto

- **Cenário 6**: Exibir mensagem de erro ao deixar os campos de nome e e-mail vazios

  Dado que o usuário queira se registrar
  Quando o usuário inserir valores nos campos de nome e e-mail
  E remover esses valores, deixando os campos vazios
  Então entra o sistema exibe uma mensagem indicando que os valores não são válidos

- **Cenário 7**: : Exibir erro quando os campos de senha e confirmação de senha não estejam idênticos

  Dado que o usuário queira se registrar
  Quando o usuário inserir valores nos campos de senha e confirmação de senha
  E esses valores não correspondem um ao outro
  Então entra o sistema exibe erro nos campos

- **Cenário 8**: Acessar páginas pela navegação inferior

  Dado que o usuário se cadastrou
  E deseja navegar pelo sistema
  Quando o usuário selecionar uma opção na barra de navegação
  Então o sistema navega para a página selecionada

- **Cenários de Login**

  - **Cenário 1**: Entrar em uma conta

    Dado que o usuário tenha um cadastro e deseja entrar no sistema
    Quando o usuário solicitar entrar no sistema
    E preencher todos os campos corretamente
    Então exibe uma mensagem de sucesso
    E entra no sistema
    E direciona para a tela de "Grupo"

  - **Cenário 2**: Exibir mensagem de erro ao tentar entrar com um e-mail não existente nos registros

    Dado que o usuário não tenha um cadastro no sistema
    Quando o usuário solicitar entrar no sistema
    E preencher o campo com um e-mail não existente nos registros

Então o sistema exibe uma mensagem indicando que o e-mail não existe
E permite o usuário ir para a tela de registro

- **Cenário 3**: Exibir mensagem de erro ao tentar entrar com um e-mail correto e uma senha errada

  Dado que o usuário tenha um cadastro no sistema
  Quando o usuário solicitar entrar no sistema
  E preencher o campo senha que não existe nos registros
  Então o sistema exibe uma mensagem indicando que a senha está errada

- **Cenário 4**: : Acessar páginas pela navegação inferior

  Dado que o usuário esteja autenticado
  E deseja navegar pelo sistema
  Quando o o usuário selecionar uma opção na barra de navegação
  Então o sistema navega para a página selecionada

- **Cenário 5**: Recuperar uma senha

  Dado que o usuário tenha esquecido sua senha e já possua uma conta
  Quando o usuário solicitar uma nova
  E preencher os campos de nome e e-mail conforme seu registro no sistema
  E preencher uma nova senha e confirmar nova senha
  E salvar
  Então o sistema exibe uma mensagem indicando que a recuperação foi bem-sucedida

- **Cenário 6**: Exibir mensagem de erro caso tente recuperar uma senha quando o campo de nome não está conforme registrado

  Dado que o usuário tenha esquecido sua senha
  Quando o usuário solicita uma nova
  E não preencher o campo de nome com um dado já registrado
  Então apresenta uma mensagem de erro informando que não possui aquele nome no sistema

- **Cenário 7**: : Exibir mensagem de erro caso tente recuperar uma senha quando o campo de e-mail não está conforme registrado

  Dado que o usuário tenha esquecido sua senha
  Quando o usuário solicitar uma nova
  E não preencher o campo de e-mail com um dado já registrado
  Então apresenta uma mensagem de erro informando que não possui aquele e-mail no sistema

- **Cenário 8**: Exibir mensagem de erro caso tente recuperar uma senha quando os campos de senha e confirmar senha não estão idênticos

Dado que o usuário tenha esquecido sua senha
E o usuário solicitar uma nova
E não preencher os campos de senha e confirmar senha corretamente
Então apresenta uma mensagem de erro informando que esses campos não são correspondentes

- **Cenários de Perfil do Usuário**
  - **Cenário 1**: Editar o perfil
    Dado que o usuário solicita a edição de seu perfil
    Quando modificar algum campo
    E salvar
    Então o sistema realiza atualização do perfil
  - **Cenário 2**: Excluir o perfil
    Dado que o usuário esteja na tela de perfil
    E solicitar a exclusão de seu perfil
    Quando confirmar a exclusão
    Então apaga o perfil no sistema
  - **Cenário 3**: Não permite salvar a edição de um perfil contendo um campo vazio
    Dado que o usuário esteja na tela de perfil
    E deseja editar os dados
    Quando o usuário deixar um campo vazio
    E tirar o foco dele
    Então exibe uma mensagem informando que o item vazio precisa ser preenchido
    E não permite salvar

- **Cenários de Perfil do Grupo**
  - **Cenário 1**: Criar um grupo
    Dado que o usuário seleciona criar um grupo
    Quando o sistema gerar um código automático
    E o usuário inserir o campo de nome
    E salvar
    Então cria um novo grupo
    E conecta o grupo ao usuário
  - **Cenário 2**: Exibir mensagem de erro ao tentar salvar um nome de grupo vazio
    Dado que o usuário seleciona criar ou editar um grupo
    Quando o o sistema gerar um código automático

E o usuário não inserir o campo de nome
E tentar salvar
Então exibe a mensagem de erro informando que o campo está vazio

- **Cenário 3**: Editar um grupo
  Dado que o usuário seleciona editar um grupo
  Quando o sistema exibir código e nome
  E o usuário deseja trocar o campo nome do grupo
  E salvar
  Então salva o novo nome do grupo
- **Cenário 4**: : Editar um grupo adicionando um amigo
  Dado que o usuário deseja criar ou editar um grupo
  Quando selecionar para adicionar um amigo
  E o usuário inserir um e-mail já existente no sistema
  E o cadastro adicionado não pertencer a nenhum outro grupo
  E salvar
  Então salva o novo amigo ao grupo
- **Cenário 5**: Editar um grupo excluindo um amigo
  Dado que o usuário deseja editar um grupo
  Quando selecionar para excluir um amigo
  E o usuário selecionar um dos componentes do grupo
  E confirmar
  Então exclui aquele amigo do grupo

- **Cenários de Exibição de Tarefas do Grupo**
  - **Cenário 1**: Listar tarefas de amigos do grupo do mês atual
    Dado que o usuário deseja visualizar tarefas do grupo
    E esteja conectado a um grupo
    E exista uma tarefa cadastrada por um membro do grupo no mês atual
    Quando selecionar na navegação o item “Grupo”
    Então lista as tarefas conforme solicitado
  - **Cenário 2**: Listar tarefas de amigos do grupo de meses anteriores
    Dado que o usuário esteja na listagem de tarefas do grupo
    E esteja conectado a um grupo
    E deseja visualizar tarefas de outros meses
    E exista uma tarefa cadastrada por um membro do grupo
    Quando selecionar o mês desejado
    Então lista as tarefas conforme solicitado

  - **Cenário 3**: Ver detalhes de uma tarefa selecionada
    Dado que o usuário deseja ver os detalhes da tarefa do grupo
    Quando o usuário selecionar o item desejado
    Então apresenta na tela os respectivos dados da tarefa

- **Cenários de Tarefas**

  - **Cenário 1**: Listar tarefas pessoais do mês atual
    Dado que o usuário deseja visualizar tarefas que cadastrou
    E exista uma tarefa cadastrada por ele no mês atual
    Quando selecionar na navegação o item "Tarefas"
    Então lista as tarefas conforme solicitado
  - **Cenário 2**: Listar tarefas pessoais de meses anteriores
    Dado que o usuário deseja visualizar tarefas que cadastrou
    E esteja na página de tarefas
    E exista uma tarefa cadastrada por ele no mês desejado
    Quando selecionar o mês em questão
    Então lista as tarefas conforme solicitado
  - **Cenário 3**: Ver detalhes de uma tarefa selecionada
    Dado que o usuário esteja na listagem de tarefas
    E exista uma tarefa cadastrada
    Quando o usuário selecionar um item
    Então apresenta os detalhes da tarefa desejada
  - **Cenário 4**: : Concluir uma tarefa
    Dado que o usuário seleciona uma tarefa
    E visualiza suas informações
    E deseja mudar seu estado
    Quando o usuário selecionar a caixa de seleção
    E solicitar a modificação
    Então modifica o estado da tarefa para concluído
  - **Cenário 5**: Excluir uma tarefa
    Dado que o usuário seleciona uma tarefa
    E visualiza suas informações
    E deseja excluir o item selecionado
    Quando o usuário selecionar o respectivo botão
    E confirmar
    Então exclui tarefa
    E atualiza a lista

- **Cenário 6**: Navegar pelas abas

  Dado que o usuário esteja na página “Tarefas”

  E deseja navegar pelas abas

  Quando o usuário selecionar uma opção diferente da que ele está

  Então o sistema exibe as informações referentes ao item selecionado

- **Cenário 7**: : Ver métricas de categoria

  Dado que o usuário esteja na listagem de tarefas

  E deseja ver quantas de tarefas foram cadastradas no mês selecionado

  E detalhes sobre categorias de tarefas cadastradas do respectivo mês

  Quando o usuário solicitar ir para a aba de “Detalhes”

  Então apresenta o total mensal e a porcentagem de cada categoria no mês solicitado

- **Cenário 8**: Criar uma tarefa

  Dado que o usuário deseja criar uma tarefa

  Quando o usuário selecionar a criação de uma tarefa

  E preencher os dados de tarefa, descrição, data e categoria

  E salvar

  Então o sistema registra uma tarefa

  E direciona para a listagem de tarefas pessoais

- **Cenário 9**: Criar uma categoria de tarefas

  Dado que o usuário esteja na criação ou edição de uma tarefa

  E deseja criar uma categoria

  Quando o usuário selecionar a opção de adicionar

  E preencher a descrição

  E salvar

  Então o sistema salva uma nova categoria

- **Cenário 10**: Selecionar categorias de tarefas já criadas

  Dado que o usuário esteja na criação ou edição de uma tarefa

  E deseja selecionar uma categoria

  Quando o usuário selecionar a respectiva opção

  E selecionar uma categoria através da caixa de seleção

  E confirmar

  Então o sistema mostra a categoria selecionada

- **Cenário 11**: Não permite criar uma categoria com o campo nome vazio

  Dado que o usuário queira criar uma categoria

  Quando o usuário selecionar a opção de adicionar

  E não preencher o campo de descrição

  Então apresenta uma mensagem de erro informando que é necessário inserir

um valor

- **Cenário 12**: Exibe mensagem de erro ao tentar salvar uma tarefa com um campo vazio

  Dado que o usuário queira criar ou editar uma tarefa
  Quando o usuário não inserir um campo de tarefa ou descrição ou data
  E tentar salvar
  Então exibe a mensagem de erro informando que o campo está vazio

- **Cenários de Notas**

  - **Cenário 1**:Listar notas

    Dado o que o usuário deseja visualizar notas
    E exista uma nota cadastrada, criada por ele ou outro membro no grupo
    Quando selecionar na navegação o item "Notas"
    Então lista as notas registradas no sistema

  - **Cenário 2**: Pesquisar uma nota

    Dado que o usuário esteja na listagem de notas
    E deseja realizar uma pesquisa
    Quando o usuário inserir a palavra-chave para a pesquisa
    Então lista as notas relacionadas para aquela palavra-chave

  - **Cenário 3**: Ver detalhes de uma nota selecionada

    Dado que o usuário deseja ver os detalhes de uma nota
    Quando o o usuário selecionar o item desejado
    Então apresenta na tela os respectivos dados da nota

  - **Cenário 4**: Criar uma nota

    Dado que o usuário deseja criar uma nota
    E seleciona a opção de criação
    Quando o usuário inserir o título e o texto
    E salvar
    Então o sistema cria uma nota

  - **Cenário 5**: Excluir uma nota

    Dado que o usuário deseja excluir uma nota
    E abre as informações do item desejado
    Quando o usuário selecionar a exclusão
    E confirmar
    Então o sistema exclui a nota

  - **Cenário 6**: Editar uma nota

Dado que o usuário deseja editar uma nota
E abre as informações do item desejado
Quando o usuário modificar o título ou texto
E salvar
Então o sistema edita a nota

- **Cenário 7**: Sair da página sem editar
  Dado que o usuário esteja na tela de criação ou edição de uma nota
  Quando o usuário solicita voltar antes de salvar
  Então apresenta uma mensagem de erro informando que não foi salvo e permite o usuário descarte os itens modificados
- **Cenário 8**: Exibe mensagem de erro ao tentar salvar uma nota com um campo vazio
  Dado que o usuário deseja criar ou editar uma nota
  E esteja na página correspondente
  Quando o usuário não preencher algum campo
  E tentar salvar
  Então o sistema exibe uma mensagem informando que não é possível salvar um item vazio
- **Cenário 9**: : Exibe mensagem de erro ao tentar excluir uma nota com um campo vazio
  Dado que o usuário deseja criar ou editar uma nota
  E esteja na página correspondente
  Quando o usuário não preencher algum campo
  E tentar excluir
  Então o sistema exibe uma mensagem informando que não é possível excluir um item vazio

- **Cenários de Despesas**
  - **Cenário 1**: Listar despesas do grupo
    Dado que o usuário deseja visualizar despesas
    E exista uma despesa cadastrada, criada por ele ou outro membro no grupo
    Quando selecionar na navegação o item “Despesas”
    Então lista as despesas registradas no sistema
  - **Cenário 2**: Ver detalhes de uma despesa selecionada
    Dado que o usuário esteja na listagem de despesas
    E exista uma despesa cadastrada
    Quando o usuário selecionar um item
    Então apresenta as informações do item desejado

- **Cenário 3**: Concluir uma despesa
  Dado que o usuário seleciona uma despesa
  E visualiza suas informações
  E deseja mudar seu estado
  Quando o usuário selecionar a caixa de seleção
  E solicitar a modificação
  Então modifica o estado da despesa para concluído
- **Cenário 4**: Excluir uma despesa
  Dado que o usuário seleciona uma despesa
  E visualiza suas informações
  E deseja excluir o item selecionado
  Quando o usuário selecionar o respectivo botão
  E confirmar
  Então exclui despesa
  E atualiza a lista
- **Cenário 5**: Pesquisar uma despesa por título
  Dado que o usuário esteja na listagem de despesas
  E deseja realizar uma pesquisa
  E selecione pesquisar por título
  Quando o o usuário inserir a palavra-chave para a pesquisa
  Então lista as despesas relacionadas para aquela palavra-chave e tipo de pesquisa
- **Cenário 6**: Pesquisar uma despesa por categoria
  Dado que o usuário esteja na listagem de despesas
  E deseja realizar uma pesquisa
  E selecione pesquisar por categoria
  Quando o usuário inserir a palavra-chave para a pesquisa
  Então lista as despesas relacionadas para aquela palavra-chave e tipo de pesquisa
- **Cenário 7**: Pesquisar uma despesa por usuário
  Dado que o usuário esteja na listagem de despesas
  E deseja realizar uma pesquisa
  E selecione pesquisar por usuário
  Quando o usuário inserir a palavra-chave para a pesquisa
  Então o lista as despesas relacionadas para aquela palavra-chave e tipo de pesquisa
- **Cenário 8**: Pesquisar uma despesa por descrição
  Dado que o usuário esteja na listagem de despesas

E deseja realizar uma pesquisa
E selecione pesquisar por descrição
Quando o usuário inserir a palavra-chave para a pesquisa
Então lista as despesas relacionadas para aquela palavra-chave e tipo de pesquisa

- **Cenário 9**: Não permite pesquisar sem antes escolher um tipo de pesquisa
  Dado que o usuário esteja na listagem de despesas
  E deseja realizar uma pesquisa
  E não selecione pesquisar por usuário
  Quando o usuário tentar inserir a palavra-chave para a pesquisa
  Então campo de pesquisa se manterá bloqueado até a seleção de um tipo de pesquisa
- **Cenário 10**: Navegar pelas abas
  Dado o que o usuário esteja na página "Despesas"
  E deseja navegar pelas abas
  Quando o usuário selecionar uma opção diferente da que ele está
  Então o sistema exibe as informações referentes ao item selecionado
- **Cenário 11**: Ver métricas de categoria
  Dado que o usuário esteja na listagem de despesas
  E deseja ver o valor total mensal das despesas foram cadastradas no mês selecionado
  E detalhes sobre categorias de despesas cadastradas do respectivo mês
  Quando o usuário solicitar ir para a aba de "Detalhes"
  Então apresenta o valor total mensal e a porcentagem de cada categoria no mês solicitado
- **Cenário 12**: Criar uma despesa
  Dado que o usuário deseja criar uma despesa
  Quando o usuário selecionar a criação de uma despesa
  E preencher os dados obrigatórios de título, valor e descrição, podendo ou não preencher categoria e com quem dividir a despesa
  E salvar
  Então o sistema registra uma despesa
  E direciona para a listagem de despesas
- **Cenário 13**:: Criar uma categoria de despesas
  Dado que o usuário esteja na criação ou edição de uma despesa
  E deseja criar uma categoria
  Quando o usuário selecionar a opção de adicionar
  E preencher a descrição

E salvar

Então o sistema salva uma nova categoria

- **Cenário 14**: Selecionar categorias de despesas já criadas

  Dado que o usuário esteja na criação ou edição de uma despesa

  E deseja selecionar uma categoria

  Quando o usuário selecionar a respectiva opção

  E selecionar uma categoria através da caixa de seleção

  E confirmar

  Então o sistema mostra a categoria selecionada

- **Cenário 15**: Não permite criar uma categoria com o campo nome vazio

  Dado que o usuário queira criar uma categoria

  Quando o usuário selecionar a opção de adicionar

  E não preencher o campo de descrição

  Então apresenta uma mensagem de erro informando que é necessário inserir um valor

- **Cenário 16**: Exibe mensagem de erro ao tentar salvar uma despesa com um campo vazio

  Dado que o usuário queira criar ou editar uma despesa

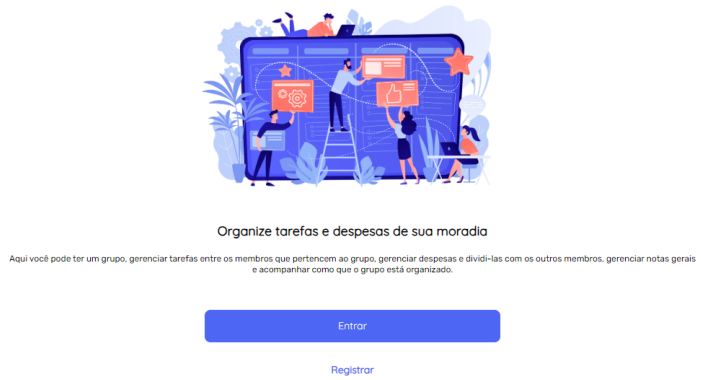
  Quando o usuário não inserir um campo de título ou descrição ou valor

  E tentar salvar

  Então exibe a mensagem de erro informando que o campo está vazio

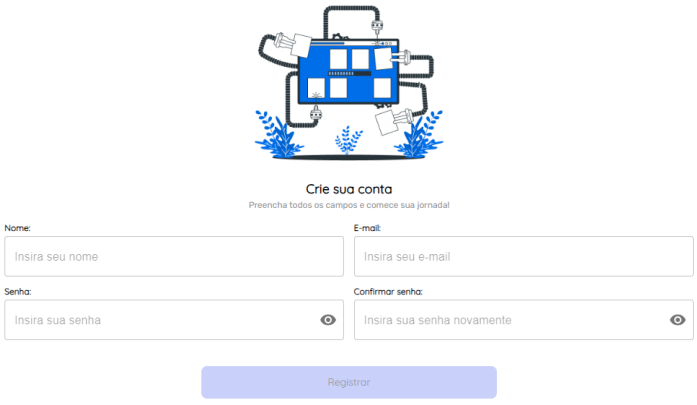
APÊNDICE C – *Layout Web*

Tela de apresentação

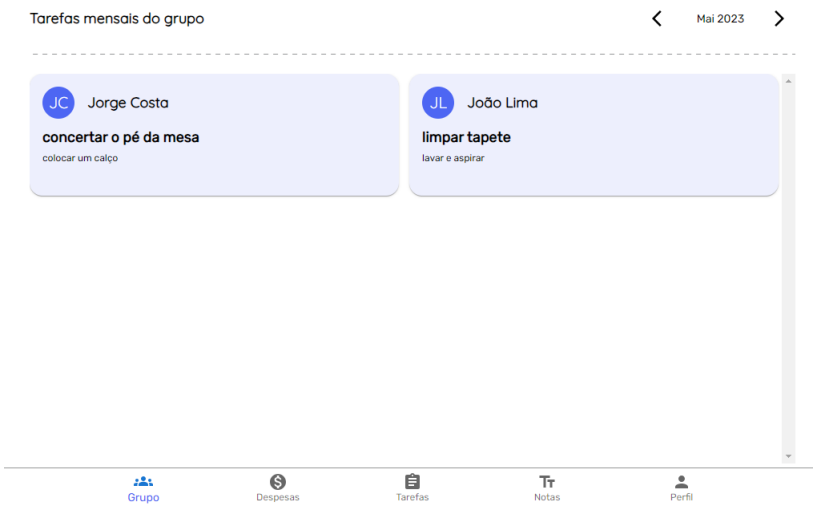


Tela de *login*

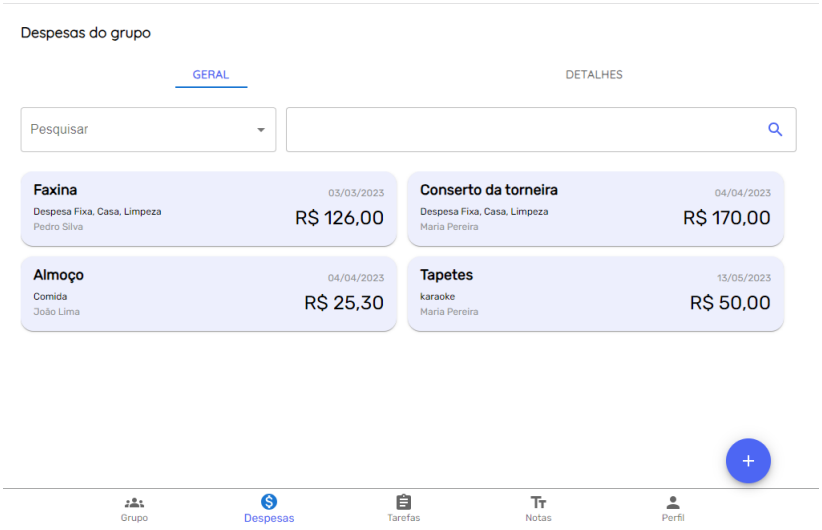
Tela de registro



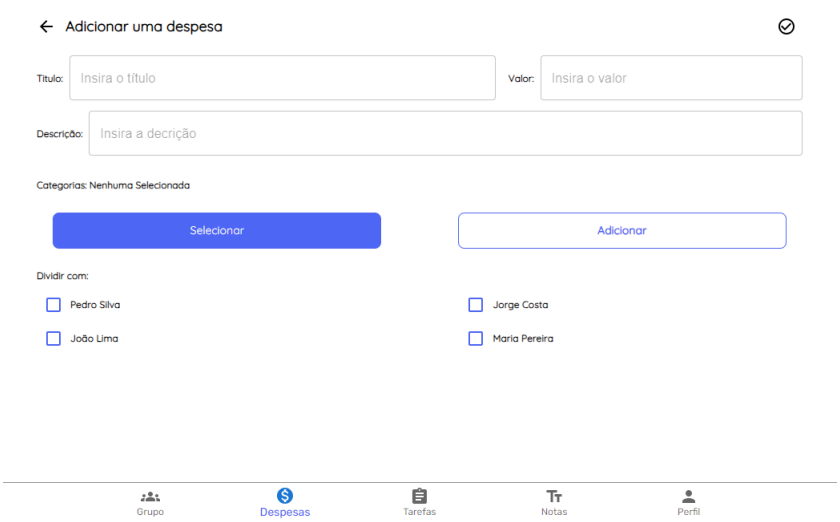
Tela de grupo



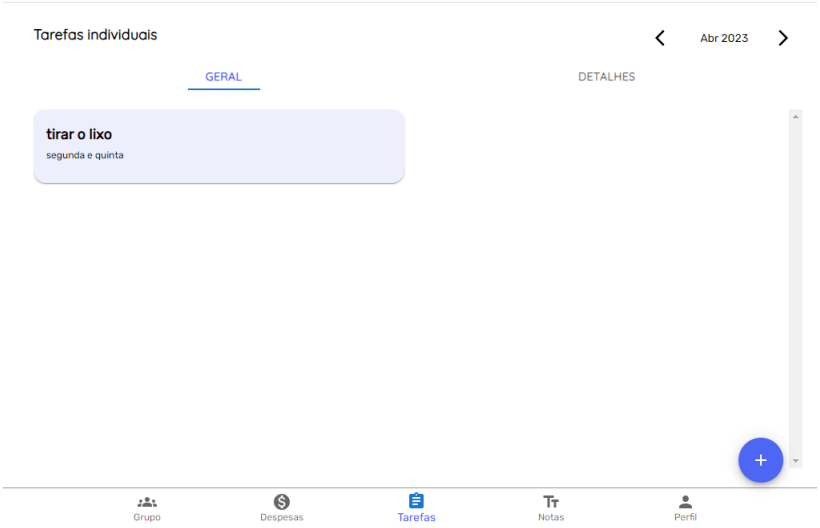
Tela de despesa



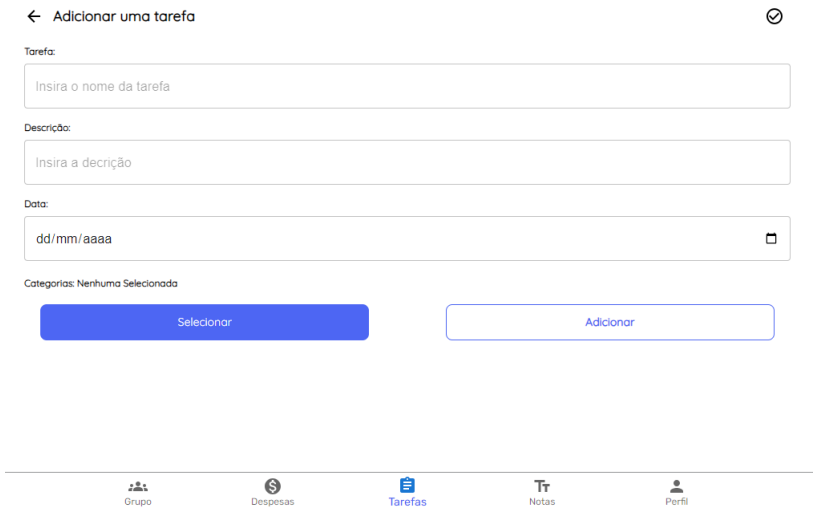
Tela de adicionar despesa



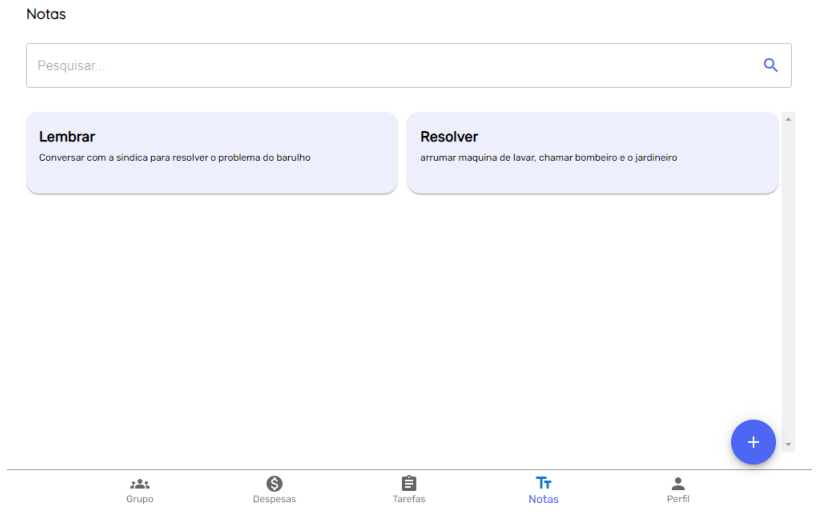
Tela de tarefa



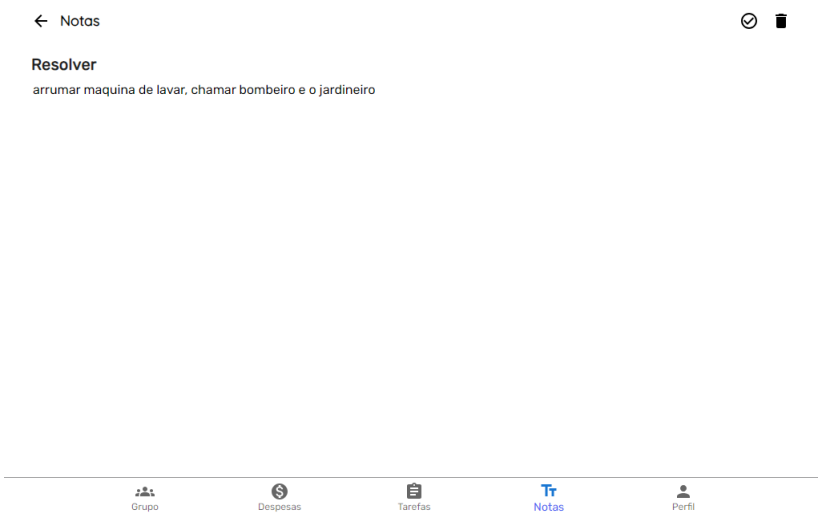
Tela de adicionar tarefa



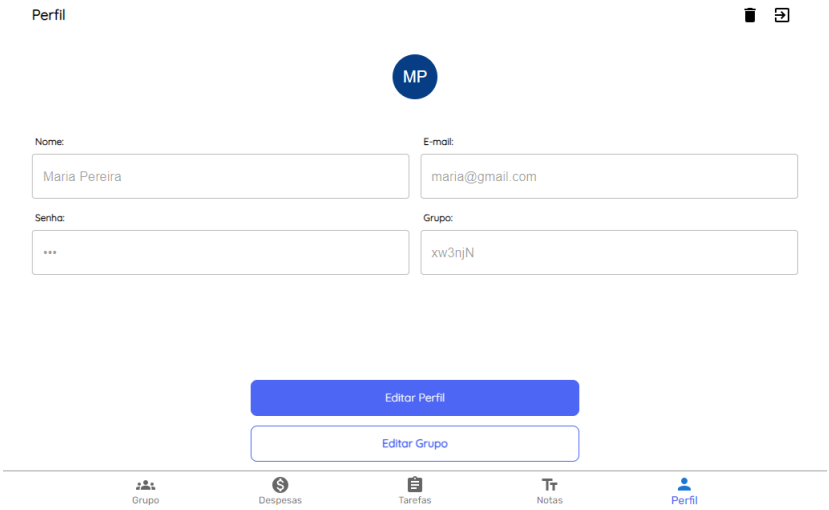
Tela de nota



Tela de adicionar nota



Tela de perfil



Tela de perfil do grupo