CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ – UNIDAVI

WELITON LISBOA

PROTÓTIPO DE APLICAÇÃO GERENCIADORA DE METAS (MI.GO)

RIO DO SUL
2022

CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ - UNIDAVI

WELITON LISBOA

PROTÓTIPO DE APLICAÇÃO GERENCIADORA DE METAS (MI.GO)

Trabalho de Conclusão de Curso a ser apresentado ao curso de Sistemas da Informação, da Área das Ciências Naturais, da Computação e das Engenharias, do Centro Universitário para o Desenvolvimento do Alto Vale do Itajaí, como condição parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

Prof. Orientador: M.e Marcondes Maçaneiro

RIO DO SUL
2022

CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ - UNIDAVI

WELITON LISBOA

PROTÓTIPO DE APLICAÇÃO GERENCIADORA DE METAS (MI.GO)

Trabalho de Conclusão de Curso a ser apresentado ao curso de Sistemas da Informação, da Área das Ciências Naturais, da Computação e das Engenharias, do Centro Universitário para o Desenvolvimento do Alto Vale do Itajaí- UNIDAVI, a ser apreciado pela Banca Examinadora, formada por:

_____

Professor Orientador: M.e Marcondes Maçaneiro

Banca Examinadora:

_____

Prof. M.e Fernando Andrade Bastos

_____

Prof. M.e Jeancarlo Visentainer

Rio do Sul, 28 de novembro de 2022.

A simplicidade é o último degrau da sabedoria (Khalil Gibran).

Dedico este trabalho a todas as pessoas que acreditaram nos meus esforços e sonhos, aos meus pais que nunca fraquejaram em dar o melhor para mim e meu futuro, também a um grande amigo que infelizmente perdeu a batalha da vida e não está mais entre nós.

AGRADECIMENTOS

Gostaria de agradecer os meus pais que diante de qualquer adversidade sempre acreditaram em mim e nunca mediram esforços para que pudesse alcançar todos os meus sonhos, sem a educação que foi me dada, os valores, ensinamentos e amor, tenho certeza absoluta que não estaria onde estou e muito menos seria o que sou.

Gostaria de agradecer todas as pessoas e amigos que me ajudaram e acreditarem em mim nesta etapa tão importante da minha vida, onde cresci, amadureci e encontrei minha aspiração. Também gostaria de agradecer os meus colegas e professores de Sistemas de Informação da UNIDAVI que fizeram parte de toda essa jornada, trocando experiencias, criando amizades e formando boas memorias que guardarei com muito carinho.

Por fim e não menos importante, gostaria de agradecer a mim mesmo, por ter lutado tanto desde que me recordo, por ter tido uma fé inabalável na conquista dos meus sonhos, por ter abdicado de muitos momentos na busca de uma completude maior dos meus objetivos e por nunca ter desistido.

RESUMO

Avanço tecnológico, pandemia, mudanças radicais nas atividades e cotidiano das pessoas, este é o cenário dos últimos anos. Quando forçados ao isolamento, que acarretou o surgimento e aumento de diversos problemas, as pessoas buscaram na tecnologia uma forma de continuar realizando suas atividades e necessidades diárias. Um dos problemas que emergiram na pandemia e perpetuaram na pós-pandemia é a procrastinação, que acarreta a perda de produtividade, adiamento e cancelamento de tarefas rotineiras e até atividades importantes do dia a dia. O presente trabalho levou em consideração as oportunidades e tecnologias dos dias atuais, como tecnologias que facilitam o acesso a recursos importantes e que possibilitam que novas soluções para o mercado, e a crescente de problemas ligados à produtividade e o bem-estar diário das pessoas para realizar uma pesquisa aplicada e descritiva. A pesquisa consistiu em desenvolver um protótipo de aplicação, apelidado como Mi.GO, que buscou ajudar os usuários a organizarem seu cotidiano e a alcançarem suas metas com a tecnologia atual. O protótipo é capaz de gerenciar as tarefas rotineiras dos usuários, também ajudar os usuários e dividirem de forma interativa suas metas em pequenas tarefas que possam ser realizadas de maneira mais fácil. Com o desenvolvimento do protótipo foi possível perceber que os recursos tecnológicos atuais possuem grande capacidade para o desenvolvimento de diferentes tipos de projetos e soluções, assim em conjunto com a pesquisa de aderência é possível identificar que a solução aplicada possui espaço para ser explorado no mercado.

Palavras-chave: solução, procrastinação, metas.

ABSTRACT

Technological advancement, pandemic, radical changes in people's activities and daily lives, this is the scenario of recent years. When forced into isolation, which led to the emergence and increase of various problems, people sought technology as a way to continue carrying out their daily activities and needs. One of the problems that arose in the pandemic and continued in the post-pandemic is procrastination, which leads to loss of productivity, postponement and cancellation of routine tasks and even important day-to-day activities. This work took into account current opportunities and technologies, such as technologies that facilitate access to important resources and that enable new solutions for the market, and the growing number of problems related to productivity and the daily well-being of people to carry out a applied and descriptive research. The research consisted of the development of an application prototype, called Mi.GO, which sought to help users organize their daily lives and achieve their goals with current technology. The prototype is capable of managing users' routine tasks, also helping users and interactively dividing their goals into small tasks that can be performed more easily. With the development of the prototype it can be seen that the current technological resources have great capacity for different types of projects, and with the adherence research it can be analyzed that the applied solution has space to be explored in the market.

Keywords: solution, procrastination, goals.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
DOM	Document Object Model
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
SVG	Scalable Vector Graphics
UI	Interface de usuário
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language

SUMÁRIO

1. INTRODUÇÃO

Nos últimos anos houve uma imensa mudança no cotidiano das pessoas, efeito causado principalmente pela evolução tecnológica que vem ocorrendo de forma evidente todos os dias, por exemplo, a internet das coisas, diversas ferramentas na nuvem que eliminam a dependência de hardwares físicos e em destaque, smartphones cada vez mais potentes e avançados que possibilitam a realização de diferentes tipos de atividades diretamente da palma da mão.

Este avanço tecnológico foi impactado pela pandemia que forçou as pessoas ao isolamento e consequentemente na busca de novos soluções para continuar realizando suas atividades e necessidades diárias. Por exemplo, com o isolamento muitos profissionais necessitaram continuar realizando suas atividades laborais de forma remota, modelo de trabalho este que mudou como os profissionais gerenciam seu tempo de trabalho, tendo que dividir e gerenciar melhor suas atividades laborais com deveres de casa, família e outros fatores, diante disso, soluções de comunicação, gerenciamento de tempo e compartilhamento de dados se tornaram opções importantíssimas para continuar trabalhando (GUIMARÃES, 2021).

Diante deste cenário de pós-pandemia e do advento tecnológico, as pessoas passaram a acumular tarefas que antes normalmente não faziam parte da sua rotina, a receberem novos estímulos e novas distrações, fatores que podem ajudar no aumento da procrastinação, que por sua vez, acaba gerando perca de produtividade, adiamento de tarefas e implicando em prejuízos funcionais (MORAES, 2020).

E é levando em consideração o cenário atual de surgimento de novas soluções e fatores que podem aumentar a procrastinação que nasce este projeto, o Mi.GO, consistindo no protótipo de uma aplicação que busca simplificar a forma como é organizado os afazeres do cotidiano e principalmente, a nortear o usuário na conquista das suas metas de aquisição, e para tal, propondo a quebra das metas em pequenas tarefas que facilitam a realização da meta.

1.1 PROBLEMA DE PESQUISA

Como ajudar os usuários a organizarem seu cotidiano e a alcançarem suas metas com a tecnologia atual?

1.2 OBJETIVOS

Esta seção trata dos objetivos e metas a serem alcançados a través da pesquisa e execução do trabalho de conclusão de curso.

1.2.1 Geral

Elaborar um protótipo de uma aplicação para gerenciar e simplificar metas e tarefas pessoais.

1.2.2 Específicos

- Definir caso de uso para o desenvolvimento do protótipo;
- Projetar um protótipo conceitual da aplicação para apresentar ao público;
- Realizar pesquisa de aderência de uso de uma aplicação de gerenciamento de metas;
- Definir soluções tecnológicas para o desenvolvimento da aplicação protótipo;
- Definir requisitos de desenvolvimento da aplicação protótipo;
- Desenvolver aplicação protótipo de acordo com os requisitos e soluções definidas.

1.3 JUSTIFICATIVA

Com o advento tecnológico nos últimos anos, evidenciou-se que há um número grande de possibilidades usando a tecnologia como meio para solucionar problemas de todas as esferas sociais. Isso ficou mais evidente com a adoção da tecnologia para solucionar diversos problemas que surgiram durante o momento pandêmico passado entre os anos de 2020 e 2021, que forçou repentinamente o isolamento social como forma de evitar o agravamento da pandemia, estas soluções ajudaram as pessoas a realizarem suas atividades, que antes feitas de forma convencional e até presencial, agora de forma virtual, utilizando como grande aliada a internet.

Em contraponto às soluções que o advento tecnológico evidenciou neste cenário incomum, a pandemia gerou uma crise global de saúde mental, que fez com que mais indivíduos sofressem com estresse, ansiedade e depressão, e consequentemente procrastinassem mais, já que procrastinar está intimamente ligada a questões emocionais (JOHNSON, 2021).

Na busca de uma solução para este grande problema nos tempos atuais, que é a procrastinação, pode-se encontrar em grande escala recomendações de práticas como a criação de lista de tarefas para se ter maior clareza daquilo que deve ser feito, ter uma rotina e dividir grandes tarefas e metas em pequenos passos palpáveis e claros (ORUI, 2021).

Diante disso, é possível visualizar um impacto positivo ao utilizar uma aplicação que se propõe a ajudar indivíduos a se organizarem com suas tarefas diárias, criando uma rotina e principalmente a dividir grandes metas em pequenas tarefas, visando evitar uma sobrecarga com coisas mais complexas, já que para o cérebro é mais fácil realizar pequenos passos (JOHNSON, 2021).

2. REFERENCIAL TEÓRICO

Este capítulo apresenta os conceitos das tecnologias, bibliotecas e outras ferramentas que serão utilizadas para o desenvolvimento da aplicação.

2.1. DESENVOLVIMENTO FRONT-END

Segundo o artigo da TOTVS (2021), "O Front-end está muito relacionado com a interface gráfica do projeto. Ou seja, é onde se desenvolve a aplicação com a qual o usuário irá interagir diretamente, seja em softwares, sites, aplicativos etc."

Já Souto (2019, n.p.), diz:

> Podemos classificar como a parte visual de um site, aquilo que conseguimos interagir. Quem trabalha com front-end é responsável por desenvolver por meio do código uma interface gráfica e, normalmente, com as tecnologias base da Web: HTML, CSS e JavaScript. Algumas pessoas podem confundir um pouco esse trabalho com o que um designer faz, pois no passado existia uma entidade chamada Webmaster que fazia tudo isso e mais um pouco, mas a diferença aqui é que o designer vai utilizar alguma ferramenta visual para desenhar a interface, do Photoshop ao Sketch e, quem faz front-end, estará mais próxima do código em si, que irá rodar num navegador Web como o Chrome, Firefox ou Safari.

Ainda no artigo da TOTVS (2021), ele diz que o desenvolvimento front-end tem o papel de facilitar a usabilidade do projeto para o usuário, como por exemplo, tendo responsividade e garantindo que as ferramentas do projeto funcionem como o esperado, assim tendo maior confiança do usuário e evitando a perca de tráfego orgânico por conta de erros.

2.2. DESENVOLVIMENTO BACK-END

De acordo com Souto (2019, n.p.), explica sobre desenvolvimento back-end da seguinte forma:

> Como o próprio nome sugere, vem da ideia daquilo que tem por trás de uma aplicação. Pode ficar meio abstrato num primeiro momento, mas pense que para conseguir usar o Facebook no dia a dia, os dados (as informações) do seu perfil, amigos e publicações precisam estar salvos em algum lugar e serem processados a partir dele, sendo este lugar um banco de dados. O Back-End trabalha em boa parte dos casos fazendo a ponte entre os dados que vem do navegador rumo ao banco de dados e vice-versa, sempre aplicando as devidas regras de negócio, validações e garantias num ambiente restrito ao usuário final (ou seja, onde ele não consegue acessar ou manipular algo).

No mesmo artigo sobre front-end a TOTVS (2021) diz que o back-end pode ser classificado como tudo que envolve o funcionamento do projeto assim como o bom

funcionamento e cumprimento da sua proposta, sendo o projeto um site ou mesmo um sistema. Sendo aquilo que o usuário não pode ver, como um banco de dados e os servidores do projeto, além de também ser a parte de segurança do projeto, estrutura e gerenciamento de conteúdo. Por exemplo, se o site é dinâmico, tendo suas informações sendo atualizadas em tempo real, é necessário que o banco de dados funcione corretamente para que não apresente problemas técnicos.

2.3. HTML

Segundo a documentação da MDN (2021, n.p.), “HTML é a linguagem de marcação que nós usamos para estruturar e dar significado para o nosso conteúdo web. Por exemplo, definindo parágrafos, cabeçalhos, tabelas de conteúdo, ou inserindo imagens e vídeos na página.”

A documentação da MDN (2021, n.p.), também detalha o que é HTML num âmbito mais técnico:

> HTML (Linguagem de Marcação de Hipertexto) é o bloco de construção mais básico da web. Define o significado e a estrutura do conteúdo da web. Outras tecnologias além do HTML geralmente são usadas para descrever a aparência/apresentação (CSS) ou a funcionalidade/comportamento (JavaScript) de uma página da web. "Hipertexto" refere-se aos links que conectam páginas da Web entre si, seja dentro de um único site ou entre sites. Links são um aspecto fundamental da web. Ao carregar conteúdo na Internet e vinculá-lo a páginas criadas por outras pessoas, você se torna um participante ativo na *world wide web*. O HTML usa "Marcação" para anotar texto, imagem e outros conteúdos para exibição em um navegador da Web. A marcação HTML inclui "elementos" especiais, como *<head>, <title>, <body>, <header>, <footer>, <article>, <section>, <p>, , , <img>, <aside>, <audio>, <canvas>, <datalist>, <details>, <embed>, <nav>, <output>, <progress>, <video>, <ul>, <ol>, * e muitos outros. Um elemento HTML é separado de outro texto em um documento por "*tags*", que consistem no nome do elemento entre "<" e ">". O nome de um elemento dentro de uma *tag* é insensível a maiúsculas e minúsculas. Isto é, pode ser escrito em maiúsculas, minúsculas ou uma mistura. Por exemplo, a tag *<title>* pode ser escrita como *<Title>, <TITLE>* ou de qualquer outra forma.

Complementando o entendimento do HTML, de acordo com o artigo da Digital House (2019), o HTML seria a base de toda aplicação, sendo utilizado para criar a estrutura fundamental do conteúdo do site e seus principais elementos, por exemplo, se a aplicação fosse uma casa, o HTML seria o responsável por subir as paredes e a estrutura dela, como por exemplo os pilares e vigas.

2.4. CSS

Segundo a MDN (2021, n.p.):

> CSS (*Cascading Style Sheets* ou Folhas de Estilo em Cascata) é uma linguagem de estilo usada para descrever a apresentação de um documento escrito em HTML ou em XML (incluindo várias linguagens em XML como SVG, MathML ou XHTML). O CSS descreve como elementos são mostrados na tela, no papel, na fala ou em outras mídias. O CSS pode ser usado para estilizar um documento muito básico de texto — por exemplo, alterando a cor e tamanho dos títulos e links. Pode ser usado para criar layout — por exemplo, transformando uma simples coluna de texto em um layout com uma área de conteúdo principal e um *sidebar* (barra lateral) para as informações relacionadas. Pode até ser usado para efeitos tais como animação.

A MDN (2021, n.p.) ainda complementa sobre a linguagem CSS e sua sintaxe:

> CSS é uma linguagem baseada em regras. — Você define regras especificando grupos de estilo que devem ser aplicados para elementos particulares ou grupos de elementos na sua página web. Por exemplo, "Quero que o título principal, na minha página, seja mostrado como um texto grande e de cor vermelha.". O código seguinte mostra uma regra CSS muito simples, que chegaria perto do estilo descrito acima:
>
> *h1 {*
>
> *color: red;*
>
> *font-size: 5em;*
>
> *}*
>
> A regra é aberta com um *selector*. Isso seleciona o elemento HTML que vamos estilizar. Neste caso, estamos estilizando títulos de nível um *(<h1>).* Temos, então, um conjunto de chaves { }. Dentro deles, haverá uma ou mais declarações, que tomam a forma de pares propriedade e valor. Cada par especifica uma propriedade do(s) elemento(s) que estamos selecionando e, em seguida, então um valor que gostaríamos de atribuir à propriedade. Antes dos dois pontos, temos a propriedade, e, depois, o valor. CSS *properties* possui diferentes valores permitidos, dependendo de qual propriedade está sendo especificado. Em nosso exemplo, temos a propriedade color, que pode tomar vários valores para cor. Também temos a propriedade *font-size*. Essa propriedade pode ter várias unidades de tamanho como um valor. Uma folha de estilo CSS conterá muitas regras tais como essa, escrita uma após a outra.

Assim como para um melhor entendimento do HTML, para entendermos melhor ainda o CSS, de acordo com a Digital House (2019), o CSS, pensando como se a aplicação fosse uma casa, onde o HTML seria sua estrutura, como paredes, vigas e pilares, o CSS seria a pintura e decoração dessa estrutura.

2.5. JAVASCRIPT

De acordo com Flanagan (2013, p.18):

> JavaScript é a linguagem de programação da Web. A ampla maioria dos sites modernos usa JavaScript e todos os navegadores modernos – em computadores de mesa, consoles de jogos, tablets e smartphones – incluem interpretadores JavaScript, tornando-a a linguagem de programação mais onipresente da história. JavaScript faz parte da tríade de tecnologias que todos os desenvolvedores Web devem conhecer: HTML, para especificar o conteúdo de páginas Web; CSS, para especificar a apresentação dessas páginas; e JavaScript, para especificar o comportamento delas.

Já Oliveira e Zanetti (2020, p.47), definem o Javascript em termos mais técnicos:

> O JavaScript é uma linguagem de programação orientada a objetos, sendo interpretada e executada pelo navegador web (*server-side script*). Apresenta uma sintaxe similar à linguagem Java e tem como objetivo principal oferecer melhor interatividade às páginas. Uma característica importante da linguagem JavaScript é que ela não apresenta tipos de dados, isto é, qualquer variável definida é do tipo variante. Isso quer dizer que o tipo de dados é definido de acordo com a informação armazenada naquele momento. O tipo de dados de determinada variável também pode ser modificado ao longo da execução da aplicação, conforme seu conteúdo é alterado.

Se aprofundando mais ainda no Javascript, pode-se citar os tipos que existem no JavaScript, que segundo Flanagan (2022, p.44):

> Os tipos de JavaScript podem ser divididos em duas categorias: tipos primitivos e tipos de objeto. Os tipos primitivos de JavaScript incluem números, sequências de texto (conhecidas como strings) e valores de verdade (conhecidos como booleanos). Os valores especiais null e undefined de JavaScript são valores primitivos, mas não são números, nem strings e nem booleanos. Cada valor normalmente é considerado como membro único de seu próprio tipo especial. Qualquer valor em JavaScript que não seja número, string, booleano, null ou undefined é um objeto. Um objeto (isto é, um membro do tipo objeto) é um conjunto de propriedades, em que cada propriedade tem um nome e um valor (ou um valor primitivo, como um número ou string, ou um objeto). Um objeto normal em JavaScript é um conjunto não ordenado de valores nomeados. A linguagem também define um tipo especial de objeto, conhecido como array, que representa um conjunto ordenado de valores numerados. A linguagem JavaScript contêm sintaxe especial para trabalhar com arrays, sendo que os arrays têm um comportamento especial que os diferencia dos objetos normais.

Para uma melhor compreensão da importância dos tipos, de acordo com Roveda (2021), "As variáveis são parte essencial do código desenvolvido em Javascript. Elas armazenam valores que são manipulados por nossos programas, permitindo que os algoritmos sejam implementados para dar a resposta esperada para o usuário."

Em complemento à apresentação dos tipos de JavaScript, o Quadro 01 explica cada uma delas de forma mais detalhada.

Quadro 01 - Os tipos do Javascript

Números	Ao contrário de muitas linguagens, JavaScript não faz distinção entre valores inteiros e valores em ponto flutuante. Todos os números em JavaScript são representados como valores em ponto flutuante. JavaScript representa números usando o formato de ponto flutuante de 64 bits definido pelo padrão IEEE 7541, isso significa que pode representar números tão grandes quanto ±1,7976931348623157 × 10308 e tão pequenos quanto ±5 × 10−324.
Texto (ou String)	Uma string é uma sequência ordenada imutável de valores de 16 bits, cada um dos quais normalmente representa um caractere Unicode – as strings são um tipo de JavaScript usado para representar texto. O comprimento de uma string é o número de valores de 16 bits que ela contém. As strings (e seus arrays) de JavaScript utilizam indexação com base em zero: o primeiro valor de 16 bits está na posição 0, o segundo na posição 1 e assim por diante. A string vazia é a string de comprimento 0. JavaScript não tem um tipo especial que represente um único elemento de uma string. Para representar um único valor de 16 bits, basta usar uma string que tenha comprimento 1.
Boolean	Um valor booleano representa verdadeiro ou falso, ligado ou desligado, sim ou não. Só existem dois valores possíveis desse tipo. As palavras reservadas true e false são avaliadas nesses dois valores.
null e undefined	null é uma palavra-chave da linguagem avaliada com um valor especial, normalmente utilizado para indicar a ausência de um valor. Usar o operador typeof em null retorna a string "object", indicando que null pode ser considerado um valor de objeto especial que significa "nenhum objeto". Na prática, contudo, null normalmente é considerado como o único membro de seu próprio tipo e pode ser usado para indicar "nenhum valor" para números e strings, assim como para objetos. A maioria das linguagens de programação tem um equivalente para o null de JavaScript: talvez você já o conheça como null ou nil. JavaScript também tem um segundo valor que indica ausência de valor. O valor indefinido representa uma ausência mais profunda. É o valor de variáveis que não foram inicializadas e o valor obtido quando se consulta o valor de uma propriedade de objeto ou elemento de array que não existe. O valor indefinido também é retornado por funções que não têm valor de retorno e o valor de parâmetros de função quando os quais nenhum argumento é fornecido. undefined é uma variável global predefinida (e não uma palavra-chave da linguagem, como null) que é inicializada com o valor indefinido
Objetos	O tipo fundamental de dados de JavaScript é o objeto. Um objeto é um valor composto: ele agrega diversos valores (valores primitivos ou outros objetos) e permite armazenar e recuperar esses valores pelo nome. Um objeto é um conjunto não ordenado de propriedades, cada uma das quais tendo um nome e um valor. Os nomes de propriedade são strings; portanto, podemos dizer que os objetos mapeiam strings em valores. Contudo, um objeto é mais do que um simples mapeamento de strings para valores. Além de manter seu próprio conjunto de propriedades, um objeto JavaScript também herda as propriedades de outro objeto, conhecido como seu "protótipo". Os métodos de um objeto normalmente são propriedades herdadas e essa "herança de protótipos" é um recurso importante de JavaScript.
Arrays	Um array é um conjunto ordenado de valores. Cada valor é chamado de elemento e cada elemento tem uma posição numérica no array, conhecida como índice. Os arrays em JavaScript são não tipados: um elemento do array pode ser de qualquer tipo e

	diferentes elementos do mesmo array podem ser de tipos diferentes. Os elementos podem ser até objetos ou outros arrays, o que permite a criação de estruturas de dados complexas, como arrays de objetos e arrays de arrays. Os arrays em JavaScript são baseados em zero e usam índices de 32 bits: o índice do primeiro elemento é 0 e o índice mais alto possível é 4294967294 (232−2), para um tamanho de array máximo de 4.294.967.295 elementos. Os arrays em JavaScript são dinâmicos: eles crescem ou diminuem conforme o necessário e não há necessidade de declarar um tamanho fixo para o array ao cria-lo ou realocá-lo quando o tamanho muda.
Funções	Uma função é um bloco de código JavaScript definido uma vez, mas que pode ser executado (ou chamado) qualquer número de vezes. As funções em JavaScript são parametrizadas: uma definição de função pode incluir uma lista de identificadores, conhecidos como parâmetros, que funcionam como variáveis locais para o corpo da função. As chamadas de função fornecem valores (ou argumentos) para os parâmetros da função. Se uma função é atribuída à propriedade de um objeto, ela é conhecida como método desse objeto. Em JavaScript, as funções são objetos e podem ser manipuladas pelos programas. JavaScript pode atribuir funções a variáveis e passá-las para outras funções, por exemplo. Como as funções são objetos, é possível definir propriedades e até mesmo chamar métodos a partir delas. As definições de função JavaScript podem ser aninhadas dentro de outras funções e têm acesso a qualquer variável que esteja no escopo onde são definidas.

Fonte: Elaborado a partir de Flanagan (2022).

2.5.1. TypeScript

De acordo com Cavalcante (2020), o TypeScript é uma linguagem de código aberto desenvolvida pela Microsoft, ela foi construída em cima do JavaScript, sendo considerada um "*superset*" do JavaScript por adicionar recursos de tipagem estática à linguagem original. Assim, no TypeScript estão presentes todas as funcionalidades do JavaScript, com adição de novas funcionalidades, porém, na hora de sua compilação, todo o código do TypeScript é convertido em JavaScript.

2.6. REACT

Na documentação descrita pelo React (2022, n.p.), "O React é uma biblioteca JavaScript declarativa, eficiente e flexível para criar interfaces com o usuário. Ele permite compor UIs complexas a partir de pequenos e isolados códigos chamados "componentes"."

Para Rascia (2018), o React é um projeto de código aberto criado pela Facebook (agora conhecido como Meta), sendo utilizado como a camada de visualização de uma aplicação MVC (*Model View Controller*). Um dos aspectos mais importantes dele, é a utilização de componentes, que são como elementos HTML personalizados e reutilizáveis, usados para construir as UIs de uma forma rápida e eficiente, além de claro, adicionar dinamicidade a esses elementos.

2.6.1. JSX

De acordo com Marchiori (2022) a sintaxe de React utiliza o conceito de JSX, que é basicamente uma mistura de JavaScript com HTML, e tem uma estrutura próxima ao XML. O JSX significa *JavaScript Syntax Extension*, e sua mesclagem de JavaScript e HTML oferece uma sintaxe de fácil aprendizado, desenvolvimento e entendimento do código. Para que o JSX possa ser interpretado, ele passa por um transpilador de código chamado Babel, que transforma o arquivo "jsx" em um arquivo de JavaScript.

Para entender melhor o funcionamento do JSX, Marchiori (2022) afirma, "O React trabalha com o tipo de programação declarativa para interagir com o DOM. Na prática, basta dizer o que é preciso fazer e o React se encarrega de transformar a instrução em código JavaScript e executá-la da melhor maneira. Ou seja, não é feita a manipulação direta do DOM."

2.6.2. Componentes

Em relação aos componentes, Marchiori (2022) explica que componentes são uma das principais características do React e que eles podem ser reutilizados em diversas páginas, os componentes são blocos de códigos que agregam funcionalidades e que retornam ao código HTML após a renderização.

2.6.3. Hooks

De acordo com a documentação do React (s.d.), os hooks são funções que permitem a ligação a partir de componentes com os recursos de estado e ciclo de vida do React, que nada mais são que o estado ou forma que um componente se apresenta em tela e execução de certos códigos em momentos específicos do processo. O React possui dois hooks, o State Hook e o Effect Hook, que estão detalhados no Quadro 02, além disso, também é possível desenvolver hooks.

Quadro 02 – Hooks do React

State Hook	O State Hook é chamado dentro de um componente funcional par adicionar *states* (estados) locais a ele. Fazendo com que o React preserve o valor de uma variável, ou várias, dependendo quais passamos o *state* inicial, dentro do componente entre chamadas de funções. Isso pois normalmente os valores das variáveis de uma função desaparecem depois que a função sai, mas estas variáveis *states* são preservadas. Para utilizar o *State Hook*, é chamado a função *useState()* dentro do componente.

Effect Hook	O *Effect Hook* serve para que seja possível executar códigos, funções e componentes posteriormente a renderização e atualização do DOM. Para utilizar utilizar o *Effect Hook*, é chamado a função *useEffect()* dentro do componente, assim pode-se acessar qualquer propriedade do React e até um State Hook.

Fonte: Elaborado a partir de React (s.d.).

2.7. REACT NATIVE

De acordo com Cunha (2022), "React Native (também conhecido como RN) é uma estrutura de aplicativo móvel popular, baseada na linguagem JavaScript, que permite criar aplicativos móveis renderizados nativamente para iOS e Android. A estrutura permite criar um aplicativo para várias plataformas usando a mesma base de código."

Cunha (2022) também explica que o React Native foi construído com base no React e foi lançado em 2015, assim como o React, pelo Facebook, que com apenas alguns anos se tornou uma das principais soluções adotadas no desenvolvimento móvel, pela sua vantagem de criar aplicativo nativo para IOS e Android com apenas um código.

De acordo com Junior (2020), mesmo que o React Native tenha sido criado com base no React, ele não utiliza *tags* HTML para criar seus componentes como o React faz, mas sim tags nativas.

2.7.1. Diferença de Sintaxe do React para o React Native

Becker (2021) explica que diferente da sintaxe do React, os elementos do React Native não são escritos com as *tags* HTML, como por exemplo "<*div*>", "<*span*>" e "<*ul*>" e sim com elementos que são convertidos diretamente para a linguagem nativa.

2.7.2. Componentes

Como visto acima e de acordo com a documentação do React Native (s.d.), os componentes do React Native possuem a mesma fundamentação do React, ou seja, pacotes de códigos reutilizáveis que descrevem a aparência e o comportamento da UI que podem ser visualizados na aplicação.

2.7.2.1. View e desenvolvimento mobile

Para que seja possível entender melhor os componentes documentação do React Native (s.d.), exemplifica a *View*, ou visualização, é um bloco de código de UI, ela pode exibir texto, imagens e até ações do usuário. Independentemente do tamanho do elemento visual de um aplicativo, todos eles são tipos de visualização, assim como esses elementos visuais, podem conter outras visualizações.

2.7.2.2. Componentes Nativos

A React Native (s.d.) explica que para o desenvolvimento Android, escrevemos views (Blocos de código de UI) em Kotlin e Java, já para IOS, escrevemos em Swift ou Objective-C, mas para o React Native, é possível utilizar estas views com JavaScript usando os componentes React. Para isso, o React Native cria componentes que correspondem as views do Android e IOS, estes componentes correspondentes as views nativas, são chamados de "Core components" e são essenciais para o desenvolvimento das aplicações. No quadro 03 é possível verificar a correspondência destes "Core components" com as views do Android e iOS, assim como com os elementos HTML da web.

Quadro 03 – Comparativo "core componentes"

Componente RN	**View do Android**	**View do iOS**	**HTML**	**Descrição**
<View>	<ViewGroup>	<UIView>	sem scroll	Um container que suporta layout com *flexbox*, estilo, eventos de clique e controle de acessibilidade
<Text>	<TextView>	<UITextView>	<p>	Exibi texto, suporta estilização, manipulação e eventos de clique
<Image>	<ImageView>	<UIImageView>	<img>	Exibi imagens
<ScrollView>	<ScrollView>	<UIScrollView>		Um container genérico que suporta rolagem e outros componentes
<TextInput>	<EditText>	<UITextField>	<input type="text">	Possibilita a inserção de texto pelo usuário.

Fonte: Elaborado a partir de React Native (s.d.).

2.7.3. Estilos de interface dos aplicativos

Com relação a estilização dos aplicativos em React Native, Becker (2021) explica que normalmente se é utilizado arquivos de estilo separados, que podem ser escritos em CSS, SASS ou LESS, mas o React Native possui uma abordagem diferente, os estilos são em linha e escritos utilizando JavaScript. Essa forma de abordagem é muito semelhante ao CSS, mas neste caso, são criados objetos no modelo de "cascata", também podendo ser reutilizáveis.

2.7.4. Arquitetura do React Native

Para que seja possível entender melhor o funcionamento do React Native, Cunha (2022) explica que a arquitetura do React Native pode ser dividida em 4 seções principais. Uma breve explicação sobre as 4 seções no Quadro 04.

Quadro 04 – As 4 seções principais da Arquitetura do React Native

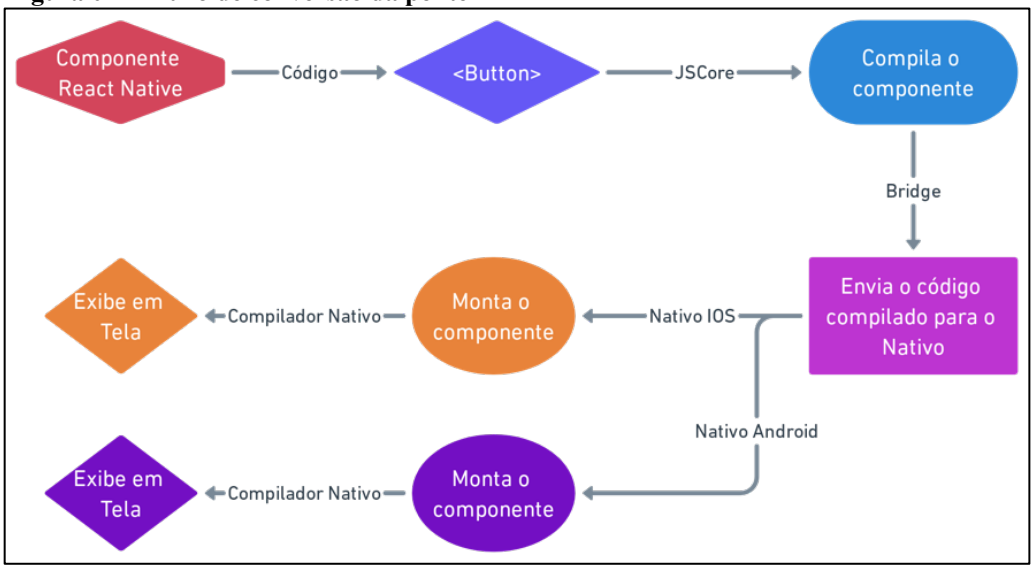
Código React	É o código do React Native, aquele desenvolvido com os componentes e funcionalidades nativas do React Native, assim como dependências e plugins.
JavaScript	É o código executado, para isso o React Native usa score, um mecanismo JavaScript de código aberto para *WebKit*. Esse mecanismo é executado dentro do aplicativo em um thread.
Parte Nativa	O código nativo para iOS é desenvolvido em Object Swift, e o nativo para Android é desenvolvido com Java/Kotlin, que são linguagens de programação.
Ponte	Conhecida como "*The Bridge*", a ponte funciona se comunicando o código JavaScript com a parte nativa, Object Swift para iOS e Java/Kotlin para Android,

Fonte: Elaborado a partir de Cunha (2022).

2.7.4.1. The Bridge

Para que se possa entender melhor uma característica importante do React Native, Cunha (2022) diz que The Bridge é como se fosse o coração do React Native, pois ela se comunica com os dois lados, que no caso são as partes nativas de iOS e Android, de forma assíncrona. Para um melhor entendimento do funcionamento, pode-se verificar o fluxo presente na Figura 01.

Figura 01 – Fluxo de conversão da ponte



Fonte: Elaborado a partir de Cunha (2022).

2.7.5. Expo

Uma ferramenta muito interessante para se usar com o React Native é o Expo, que de acordo com Cunha (2022), o Expo é uma ferramenta open-source, que foi construída em torno do React Native e ajuda os desenvolvedores a criar aplicativos móveis multiplataformas, basicamente ele facilita a criação dos aplicativos usando o React Native, abstraindo toda a configuração necessária para iniciar o desenvolvimento, poupando tempo nesse processo, para que o desenvolvedor possa focar apenas no código, além disso, o Expo também facilita a emulação da aplicação, pois disponibiliza a ferramenta Expo Go, que torna possível executar o projeto diretamente do dispositivo móvel físico.

Para complementar, Fernandes (2018) explica que o Expo facilita e simplifica o acesso às API's e recursos nativos dos dispositivos pela aplicação, como por exemplo câmera, microfone, player de música e outros.

2.7.6. Babel

Citado anteriormente, o Babel é a ferramenta utilizada para transpilar a sintaxe JSX em JavaScript, de acordo com a documentação do Babel (s.d.), o Babel é uma cadeia de ferramentas usada principalmente para converter o código ECMAScript 2015+ em uma versão compatível com as versões anteriores do JavaScript em navegadores ou ambientes atuais e antigos. Além disso ele converte também JSX e TypeScript.

2.7.7. NativeBase

De acordo com a documentação da NativeBase (s.d.), o NativeBase é uma biblioteca de componentes que permite criar sistemas com design universal, sendo construída em cima do React Native e permitindo que seja desenvolvido aplicativos para Android, iOS e Web.

2.8. NODE.JS

Segundo a documentação do Node.js (s.d.), o Node.js é um ambiente de tempo de execução JavaScript, sendo de código aberto e multiplataforma, ele executa o mecanismo JavaScript V8, o núcleo do Google Chrome, assim permitindo que o código JavaScript seja executado fora do navegador, com isso, o JavaScript, que antes escrito e executado no navegador e exclusivo para o desenvolvimento front-end, agora pode ser usado para o desenvolvimento back-end, possibilitando o uso de uma única linguagem (sendo o JavaScript) para desenvolver toda a aplicação. Além disso, o Node.js conta com o npm, que é o gerenciador de pacotes padrão para o Node.js, que hospeda uma quantidade massiva de bibliotecas de código aberto, que podem ser usados para o desenvolvimento de aplicações no ambiente Node.js.

2.8.1. Mecanismo JavaScript V8

Continuando na documentação do Node.js (s.d.), o V8 é o que analisa o código JavaScript e o executa enquanto navega com o Google Chrome, por exemplo. O mecanismo V8 é independente do navegador, e esse recurso que acabou permitindo o surgimento do Node.js.

2.9. FRAMEWORK

A Digital House (2020), diz que Framework é um conjunto de códigos para serem usados no desenvolvimento de aplicações. Frameworks tem como objetivo aplicar funcionalidades, comandos e até estruturas já prontas que garantem qualidade, padronização e ganho de produtividade no desenvolvimento de um projeto.

2.10. JSON

De acordo com a documentação da JSON (s.d.), "JSON (JavaScript Object Notation - Notação de Objetos JavaScript) é uma formatação leve de troca de dados. Para seres humanos, é fácil de ler e escrever. Para máquinas, é fácil de interpretar e gerar. Está baseado em um subconjunto da linguagem de programação JavaScript, Standard ECMA-262 3a Edição - Dezembro - 1999. JSON é em formato texto e completamente independente de linguagem, pois usa convenções que são familiares às linguagens C e familiares, incluindo C++, C#, Java, JavaScript, Perl, Python e muitas outras. Estas propriedades fazem com que JSON seja um formato ideal de troca de dados."

Além disso, a documentação do JSON (s.d.), explica que o JSON está constituído em duas estruturas, que estão presentes no Quadro 05.

Quadro 05 – Estruturas de um JSON

Uma coleção de pares nome/valor	Em várias linguagens, isto é caracterizado como um *object*, *record, struct, dicionário, hash table, keyed list,* ou arrays associativas.
Uma lista ordenada de valores	Na maioria das linguagens, isto é caracterizado como uma *array*, vetor, lista ou sequência.

Fonte: Elaborado a partir de JSON (s.d.).

Para complementar, ainda de acordo documentação do JSON(s.d.), as estruturas são de dados universais, fazendo com que todas as linguagens de programação modernas as suportem.

3. METODOLOGIA DA PESQUISA

O presente trabalho, para conclusão do curso, caracteriza-se como pesquisa aplicada e descritiva, seu intuito é desenvolver uma aplicação protótipo para mobile que realize o gerenciamento de metas e tarefas. E busca responder o seguinte problema: Como ajudar os usuários a organizarem seu cotidiano e a alcançarem suas metas com a tecnologia atual?

Para tal, foi desenvolvido um desenho conceitual da aplicação e com ele realizada uma pesquisa descritiva que executada por meio de um questionário anônimo e compartilhada em redes sociais e aplicativos mensageiros, tal questionário foi criado utilizando a ferramenta *Google Formulários* e seu objetivo é traçar um perfil de usuário para aplicação e entender se há aderência e utilidade do aplicativo no seu cotidiano e também entender quais as aplicações que são utilizadas pelos indivíduos que responderam a pesquisa.

Com base nos dados levantados pela pesquisa descritiva foi efetuada a pesquisa de estado da arte, onde ocorreu a análise de layout, utilidade e as diferenças das aplicações utilizadas pelos indivíduos e outras aplicações em alta no mercado de aplicativos, posteriormente a esta análise, foi efetuado o levantamento de requisitos para o desenvolvimento da aplicação, para tal, além de ser levado em consideração estes dados, também será utilizado o referencial teórico, cujo qual buscou responder a utilidade de cada tecnologia para a concepção do projeto.

Para o desenvolvimento do Mi.GO foi utilizado o ambiente de desenvolvimento Node.js, utilizando a linguagem JavaScript complementada por TypeScript e utilizando a biblioteca React Native para a criação de uma aplicação que possa ser utilizada nos principais sistemas operacionais de dispositivos móveis do mercado (Android e iOS).

3.2 PESQUISA DE ADERÊNCIA

A pesquisa de aderência do projeto foi realizada utilizando a ferramenta do Google Formulários e seu objetivo foi traçar um perfil dos indivíduos pesquisados e entender o quão aderente e útil o Mi.GO seria no seu cotidiano.

Para obter as respostas necessárias e concluir os objetivos o formulário gerado foi compartilhado pelas ferramentas Whatsapp e Instagram, a pesquisa se constituiu em duas partes, a primeira eram questões básicas sobre o perfil do entrevistado, conforme pode-se visualizar no Quadro 06.

Quadro 06 – Perguntas para traçar o perfil do entrevistado

Número	Pergunta	Respostas
1	Você possui metas, sonhos ou grandes objetivos?	Múltipla escolha: • Sim • Não
2	Em relação a metas, sonhos e objetivos, você costuma procrastinar?	Múltipla escolha: • Sim • Não
3	Você já sentiu dificuldade em definir quais os passos que deveria seguir para conquistar suas metas?	Múltipla escolha: • Sim • Não • Nunca tentei
4	Levando em consideração que você tenha noção do que fazer para atingir seus objetivos, você conseguiu ou consegue seguir um cronograma ou se organizar?	Múltipla escolha: • Sim • Não • Nunca tentei
5	Você usa alguma aplicação/software para se organizar?	Múltipla escolha: • Sim • Não
6	Se sim, poderia dizer qual software?	Texto de resposta curta

Fonte: Elaborado a partir de JSON (s.d.).

Posteriormente ao questionário onde o perfil do entrevistado é traçado, o mesmo era apresentado a uma breve explicação do Mi.GO, assim como seu desenho conceitual. Com base nisso, a segunda sessão do questionário buscou entender se o entrevistado utilizaria o Mi.GO em seu cotidiano, e se acredita na sua utilidade. É possível verificar as perguntas da segunda sessão no Quadro 07.

Quadro 07 – Perguntas de aderência do Mi.GO

Número	Pergunta	Respostas
1	Quanto a ideia do Projeto, você usaria no seu cotidiano?	Múltipla escolha: • Sim • Não
2	Você acredita que este projeto poderia ajudá-lo com a definição dos passos para atingir suas metas, produtividade ou evitar a procrastinação?	Múltipla escolha: • Sim • Não • Talvez
3	Se você acredita que o projeto poderia ajudá-lo, em uma escala de 1 a 5, o quanto ele seria útil?	Escala linear, de 1 a 5, sendo: • 1 muito pouco

		• 5 muito útil.
4	Gostaria de acrescentar algo? Pode ser uma crítica, sugestões, etc.	Texto de resposta longa.

Fonte: Elaborado a partir de JSON (s.d.).

3.2. ESTADO DA ARTE

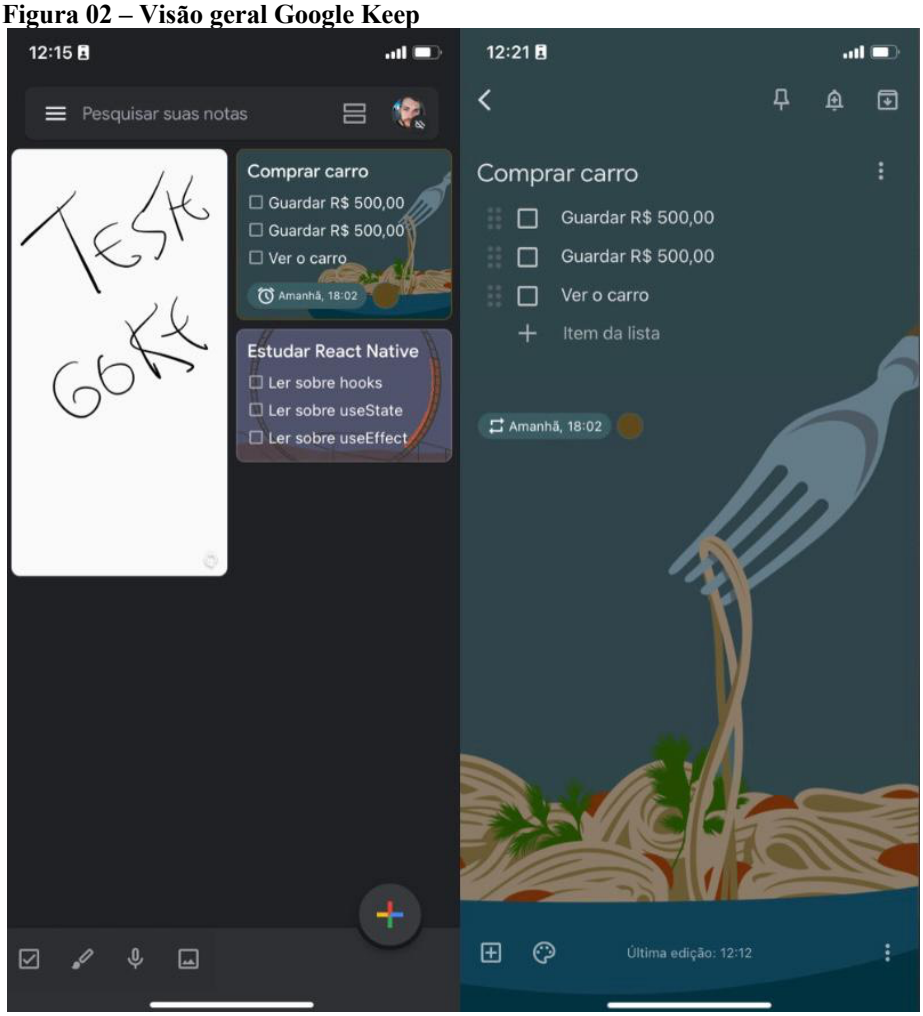
A proposta do projeto é muito comum, tendo uma grande quantidade de aplicações que realizam o gerenciamento de tarefas diárias e se propõem em ajudar o usuário a seguir uma rotina, então para um estudo mais aprofundado e valoroso, foi analisada ferramentas populares com a proposta semelhante ao projeto com finalidade de entender o diferencial do Mi.GO em comparação a elas.

3.1.1 Google Keep: notas e listas

O Google Keep é um aplicativo disponível para web, Android e iOS, que tem como objetivo criar notas e listas. Nele é possível criar uma nota com uma lista de afazeres ou lembretes dentro dela, ou notas separadas que podem ser lembres, afazeres e outras coisas, além disso, é possível criar notas com imagens, alterar sua cor e plano de fundo, compartilhar, adicionar colaboradores, fazer uma cópia, adicionar marcador e até desenhar e capturar áudio (este último apenas com conexão a internet). Esta série de recursos o torna uma ferramenta muito completa e ao mesmo tempo muito simples de usar.

A sua tela inicial apresenta as notas criadas, sendo possível ordená-las visualmente, conforme mostra a Figura 02 onde temos a sua tela inicial e a tela de criação de notas.

Figura 02 – Visão geral Google Keep



Fonte: Google Keep (2022).

Pode-se observar que as telas são visualmente simples e de fácil uso e se faz possível criar notas por exemplo, que correspondem a grandes metas. Porém o Google Keep não tem possui um recurso para criar lembretes de cada tarefa inserida dentro de uma nota, sendo possível inserir apenas um por nota (este pode ser configurado para repetir de acordo com as configurações do usuário), assim como a criação de uma grande quantidade de tarefa é moroso.

3.1.1.1 Outras ferramentas semelhantes ao Google Keep

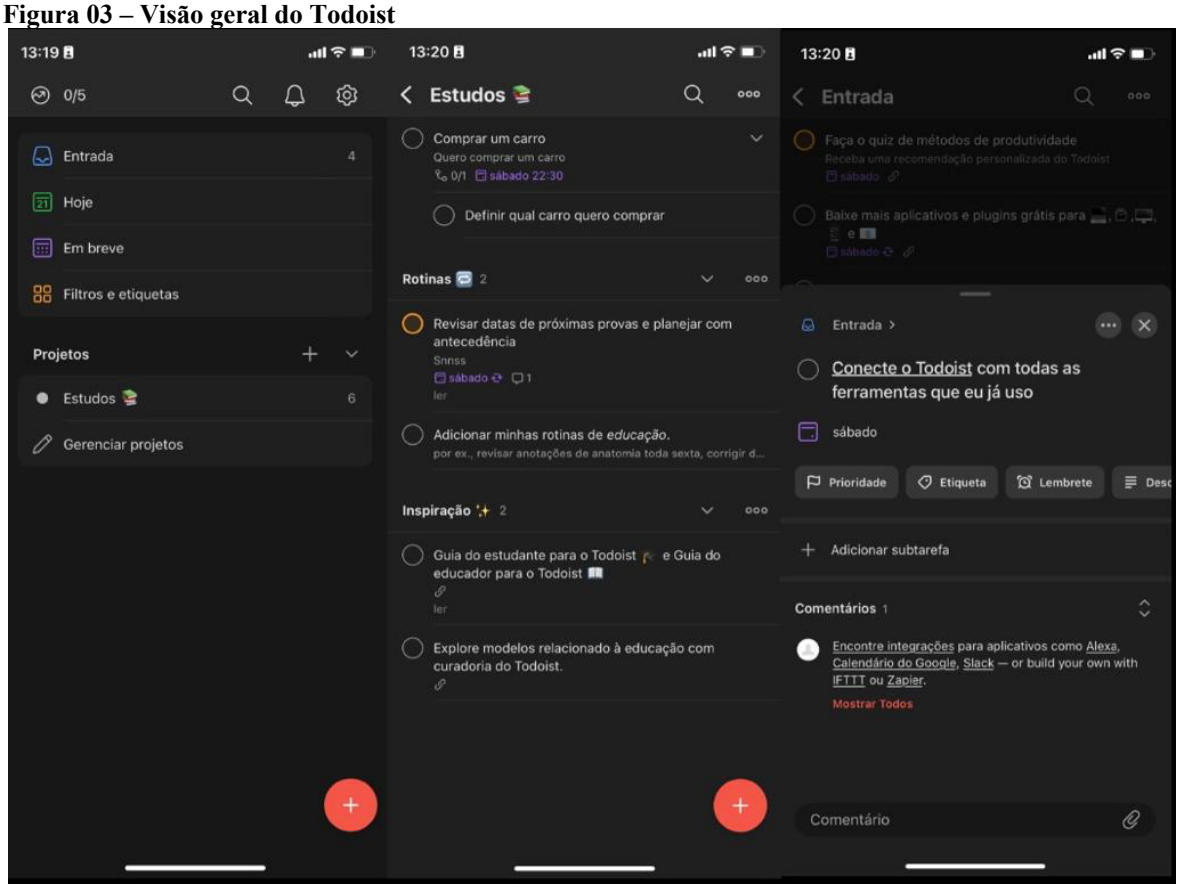
Levando em consideração as propostas e recursos do Google Keep, é possível encontrar outras ferramentas muito parecidas, que possuem praticamente todos os recursos do Google Keep e são visualmente semelhantes, como o Notas, ferramenta exclusiva do iOS, o Notion, uma ferramenta muito usada no meio de TI, sendo possível nele também criar notas e listas de afazeres.

3.1.2 Todoist: planner e to-do list

O Todoist é um aplicativo disponível para dispositivos móveis (Android e iOS), e desktop sendo possível instalá-lo ou acessá-lo pela web. Sua proposta é específica para tarefas e seus recursos vão de criar tarefas com lembretes (necessária versão paga), data, hora, nível de prioridade, inserção de link, compartilhamento, criação de subtarefas e até também inseri-los em agrupadores, chamados de Projetos, estes por sua vez, possibilitam a organização das tarefas em seções (são como temas) e visualização das tarefas em forma de lista ou painel.

A quantidade de recursos do Todoist é muito vasta e posiciona ele na lista de melhores aplicativos com esta proposta, em sua tela inicial são mostradas listas de agrupadores das tarefas, sendo possível acessar unicamente um agrupador específico para visualizar suas tarefas e outros recursos como pesquisa, notificações, configuração e criação de notas. Conforme a Figura 03 é possível verificar as suas principais telas.

Figura 03 – Visão geral do Todoist



Fonte: Todoist (2022).

As telas do Todoist são relativamente simples e organizadas, além disso, o acesso aos seus recursos básicos é fácil. Em contraponto, o Todoist possui recursos que só podem ser

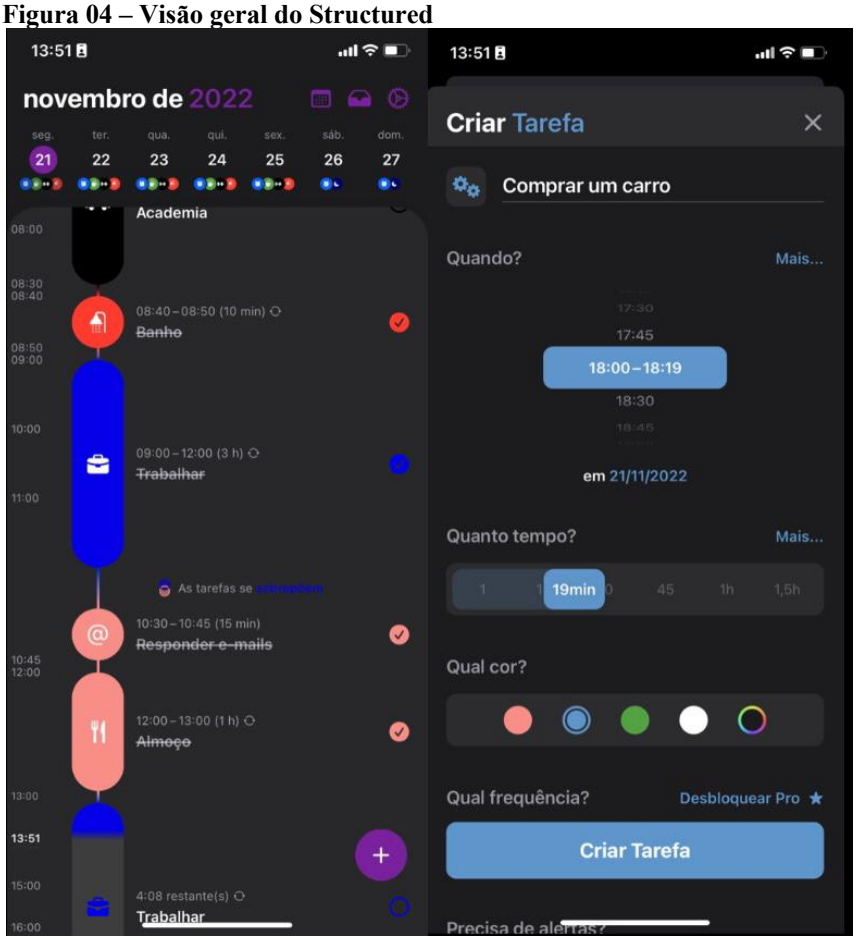
utilizados caso seja adquirido sua versão paga, como as notificações, isso faz com que o usuário tenha que sempre entrar no aplicativo, ou utilizar um widget para ser lembrado de suas tarefas, e também para configurá-lo de forma que se tenha algo parecido com Metas é bastante moroso e requer um bom tempo para criar todas as tarefas manualmente, também durante a criação de uma tarefa, caso o usuário queira adicionar recursividade, é necessário que ele tenha um conhecimento mais aprofundando da ferramenta, já só é possível adicionar recursividade por meio de comandos de texto.

3.1.1.1 Outras ferramentas semelhantes ao Todoist

O Todoist possui em sua proposta o gerenciamento de tarefas, recurso muito comum e encontrado em diversos aplicativos do mercado. Destes aplicativos, em suma, a grande maioria é extremamente semelhante ao Todoist, tanto em recursos, quanto visualmente. Em destaque, pode-se citar o “Structured – Planejador Diário” e “ToDo List: Lembretes e Tarefas” que são populares e muito bem ranqueados na loja de aplicativos dos dispositivos móveis.

O Structured possui em sua tela inicial a listagem de tarefas de acordo com o horário, organizando como uma espécie de linha do tempo, também o calendário logo acima da lista. Seus recursos são bem específicos para gerenciamento de tarefas diárias, que diferente do Todoist, não é possível agrupá-las em listas, projetos e outras coisas. Além disso, a criação de uma tarefa é simples e muito bonita, mas para adicionar recursividade em uma tarefa, é necessário a versão paga. Pode-se analisar o Structured pela figura 04.

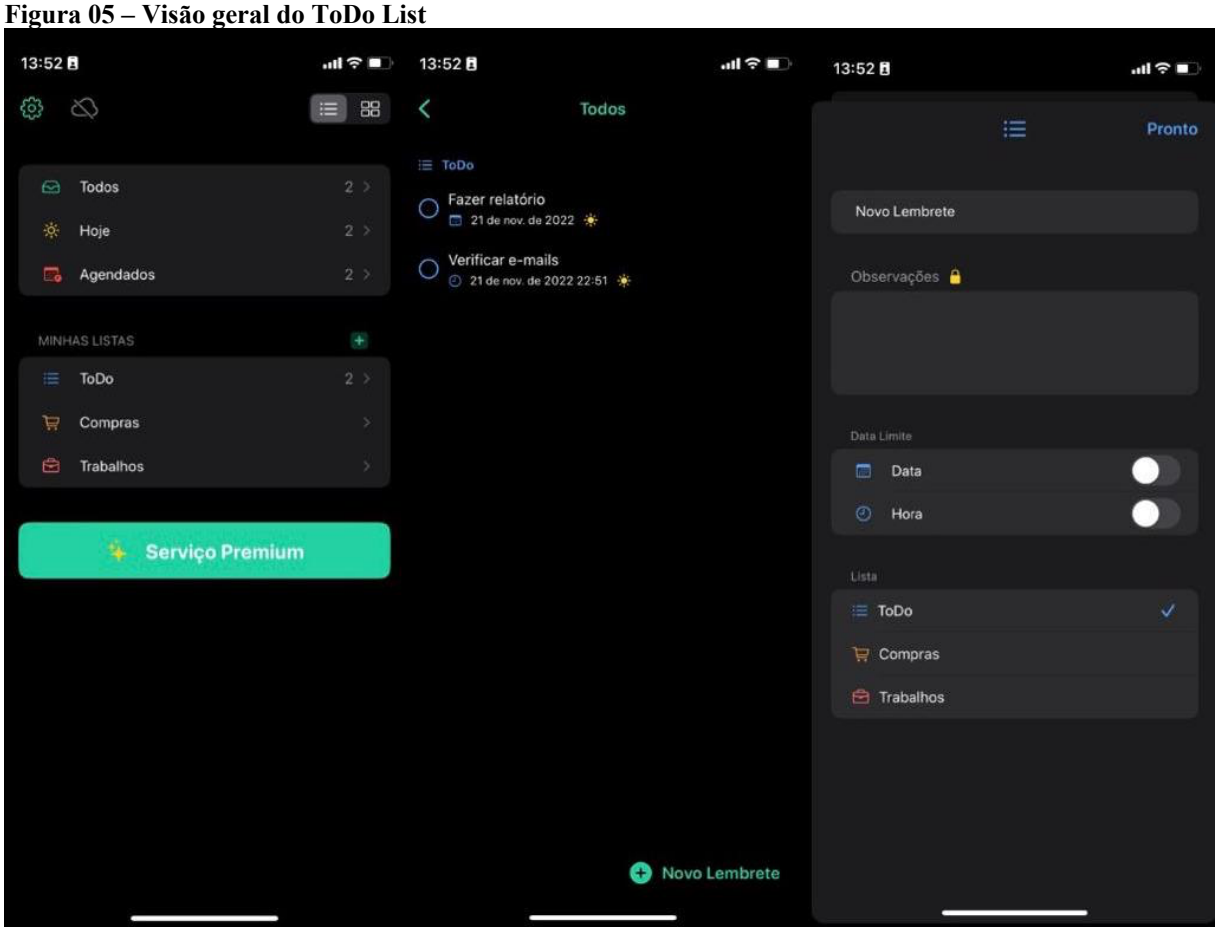
Figura 04 – Visão geral do Structured



Fonte: Structured (2022).

Por último, o ToDo List, que possui em sua tela inicial o mesmo layout do Todoist, onde é possível observar os agrupadores de tarefas, porém diferente do Todoist, no ToDo List não é possível criar uma tarefa diretamente da tela inicial, para isso o usuário tem que acessar um dos agrupadores. A criação de tarefas é simples, mas um pouco diferente do comum, deve-se criar a tarefa primeiro, para que depois seja possível acessá-la e adicionar uma data e hora, além disso, os lembretes são definidos automaticamente quando configurada a hora, mas não é possível adicionar um tempo para a tarefa e nem recursividade. Na figura 05 há uma visão geral do ToDo List.

Figura 05 – Visão geral do ToDo List



Fonte: ToDo List. (2022).

4. PROTÓTIPO DE APLICAÇÃO GERENCIADORA DE METAS INTERATIVA

Este capítulo aborda todos os processos que foram executados para a concepção do protótipo Mi.GO.

4.1. SOBRE O PROJETO

Esta seção tem como intuito apresentar o projeto, explicando seus objetivos, funcionalidades, assim como o comparativo de estado da Arte e os resultados obtidos pela pesquisa descritiva quando a aderência do Mi.GO com seu público.

4.1.1. Visão geral do projeto

O Mi.GO consiste em um protótipo de aplicação para dispositivos móveis que tem como objetivo ajudar os usuários a realizarem suas metas, que no âmbito do protótipo será com foco em metas aquisitivas, propondo a divisão destas metas em pequenas tarefas mais palpáveis e possíveis de serem alcançadas, em conjunto a isso, entrega em seu escopo a possibilidade de gerenciamento das suas tarefas diárias, que podem ser as tarefas de suas metas e outras tarefas rotineiras.

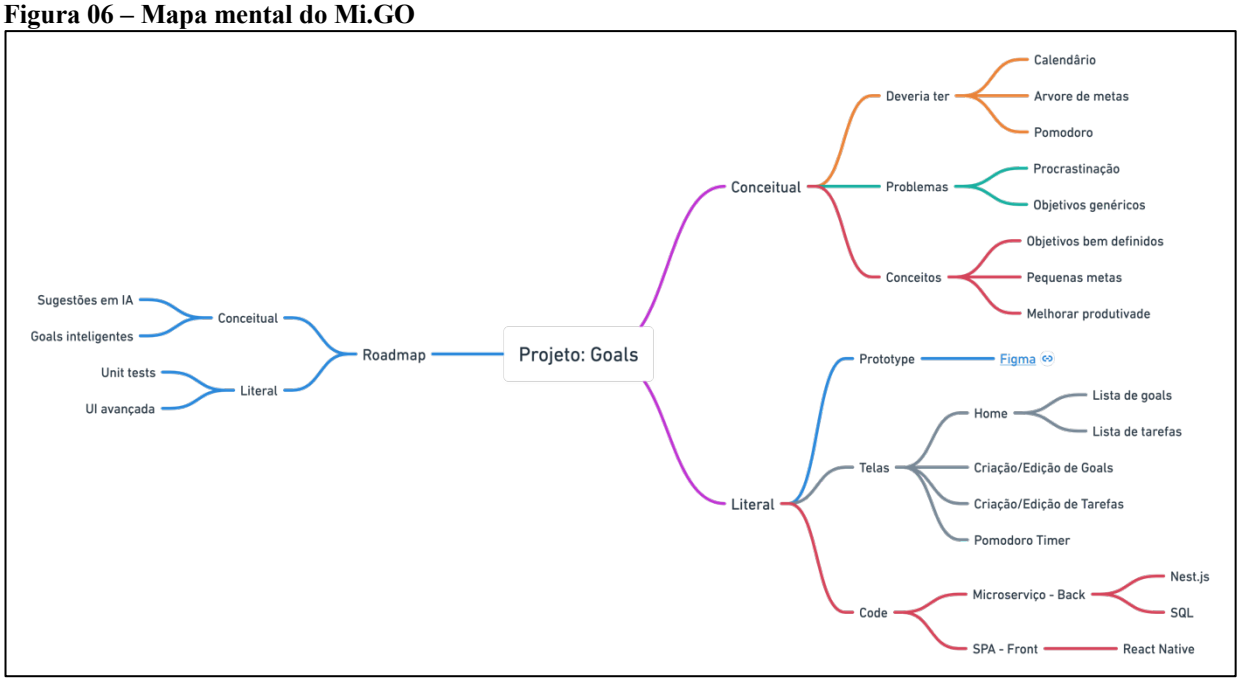
A proposta do projeto crê que ao dividir grandes metas em pequenas tarefas, os passos a serem dados para alcançá-las se tornam menos morosos, desafiadores e menos prováveis de serem motivadores de procrastinação, já que os tornam mais “reais” e claros. Além disso, também combina esse processo de divisão com a lista de afazeres (tarefas) do cotidiano do usuário, fazendo com que se tenha uma rotina de afazeres.

O Mi.GO irá possuir uma tela inicial, onde em seu corpo terá duas listas, a listagem de metas, para que o usuário possa visualizar o seu progresso seguida pela listagem de tarefas do dia, já em seu rodapé, estará presente as ações para que seja possível navegar entre a criação de metas, tarefas e a tela inicial (home). O fluxo principal da proposta é a criação da meta, onde o usuário preenche campos interativos e simples, providos por predefinições da aplicação, desta forma, a aplicação compreende melhor as ações que devem ser tomadas para que seja possível a conclusão da meta e cria automaticamente tarefas com base nestas ações também evitando que o usuário tenha que dispor de muito tempo para criar inúmeras tarefas. Este fluxo ajuda não só a aplicação a compreender as tarefas a serem criadas, mas também o usuário a compreender melhor o que deseja.

4.1.2. Conceito Mi.GO

Antes que o projeto Mi.GO tomasse um visual e se tornasse um protótipo desenvolvido pelas tecnologias citadas no item 3., seu conceito nasceu de um mapa mental criado na ferramenta *whimiscal,* neste mapa mental, foram levantados os primeiros requisitos, ideias de telas, recursos da parte literal, além disso, também foram levantados os motivadores do nascimento do Mi.GO, e os problemas que ele buscava solucionar. Na Figura 6 pode-se verificar melhor todo esse processo.

Figura 06 – Mapa mental do Mi.GO



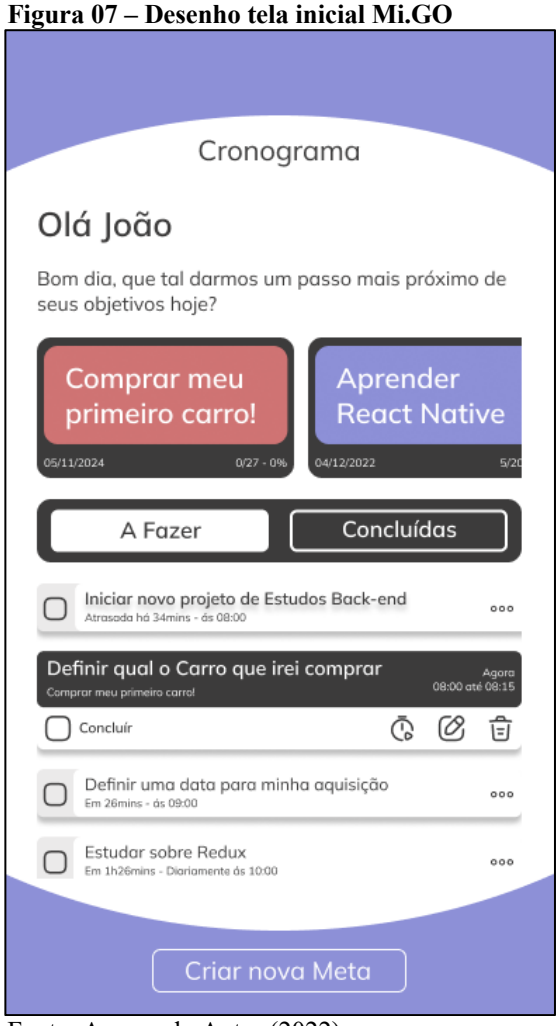
Fonte: Acervo do Autor (2022).

4.1.3. Desenho inicial do Protótipo

Nesta seção, serão apresentados os desenhos inicias do protótipo Mi.GO, estes desenhos foram desenvolvidos pela ferramenta Figma e serviram como base para o desenvolvimento do protótipo e amostra conceitual do protótipo na pesquisa de aderência do Mi.GO no mercado.

A Figura 7 representa a tela inicial do Mi.GO, onde pode-se observar a interação com o usuário, a lista de metas, a lista de tarefas e também o rodapé de navegação.

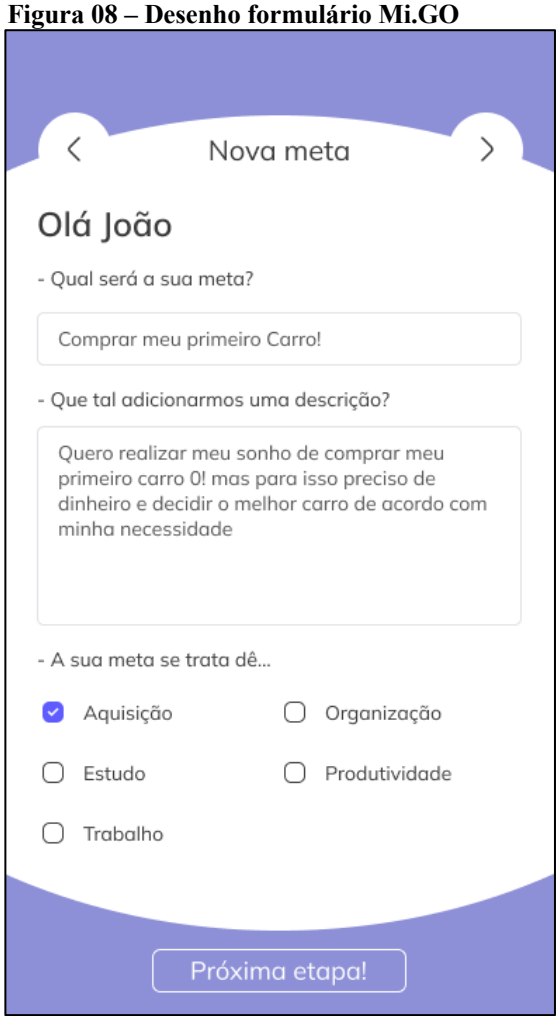
Figura 07 – Desenho tela inicial Mi.GO



Fonte: Acervo do Autor (2022).

Na Figura 8 é possível observar o desenho do formulário de criação de uma meta, nele estão presentes os campos para preenchimento que irão nortear a criação de uma meta.

Figura 08 – Desenho formulário Mi.GO



Fonte: Acervo do Autor (2022).

4.1.4 Análise da pesquisa de aderência

Durante o processo de pesquisa, foram obtidas 46 respostas anônimas, com intuito de uma análise mais assertiva, inicialmente foram analisadas as respostas da seção de perfil, para que seja possível entender de fato se os entrevistados possuíam como necessidade uma ferramenta que os ajudassem na conquista de suas metas e organização das tarefas.

A primeira questão do formulário se tratava de entender se os entrevistados possuaim metas, sonhos ou grandes, onde 100% dos 46 entrevistados responderam que possuem.

A próxima pergunta se tratava de entender se eles possuíam problemas com procrastinação, o questionamento foi o seguinte “Em relação a metas, sonhos e objetivos, você costuma procrastinar?”, e 89,1% dos entrevistados responderam que sim e apenas 10,9% que não.

A terceira questão foi a seguinte “Você já sentiu dificuldade em definir quais os passos que deveria seguir para conquistar suas metas?”, e nesta havia três opções de respostas, sendo “Sim”, “Não” e “Nunca tentei”, onde 89,1% afirmaram que sentiram dificuldades em definir os passos para conquistar suas metas, 8,7% responderam que não e 2,2% nunca tentaram definir os passos.

Além disso, mais da metade dos entrevistados não consegue ou nunca tentou seguir um cronograma para se organizar, conclusão este obtida pela quarta questão, tendo o seguinte questionamento “Levando em consideração que você tenha noção do que fazer para atingir seus objetivos, você conseguiu ou consegue seguir um cronograma ou se organizar?” e as respostas obtidas foram 43,5% para “Sim”, 32,6% para “Não” e 23,9% nunca tentaram seguir um cronograma ou se organizar.

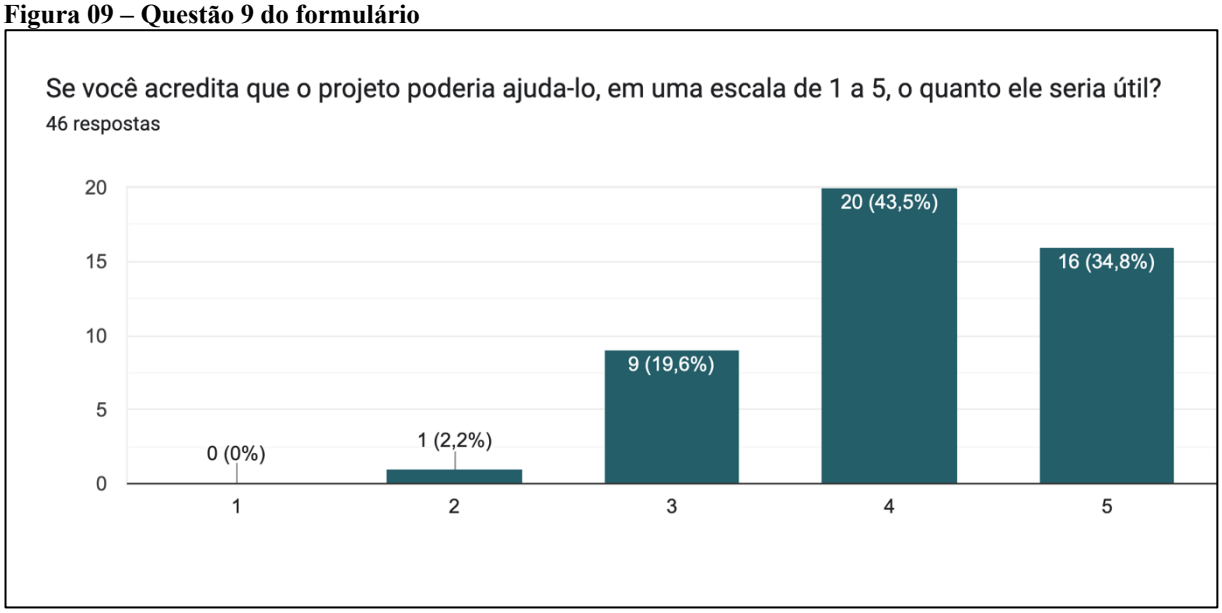
Por último nas questões de perfil, na quinta questão os entrevistados são indagados se utilizam alguma aplicação ou software para se organizar, tal questão tinha como objetivo compreender melhor se os entrevistados já estão habituados a utilizar aplicações parecidas, assim como também se estariam abertos a utilização, onde 80,4% não utilizam nenhuma ferramenta e apenas 19,6% utilizam.

Diante disso, é possível identificar que os valores são positivos para a proposta do Mi.GO, já que seu objetivo é atuar ajudando as pessoas que possuem dificuldade com se organizar, definir passos a serem dados para a conquista de suas metas e diminuir a procrastinação.

Com base nos resultados das questões de perfil dos entrevistados, torna-se possível analisar os resultados quanto a aderência do Mi.GO em seu cotidiano, onde inicialmente os mesmos são questionados se usariam o projeto em seu cotidiano, e 91,3% responderam que usariam e apenas 8,7% não usariam.

Posterior ao questionamento de uso, os entrevistados foram questionados se acreditavam que o projeto poderia ajudá-los com a definição dos passos para atingir suas metas, ajudar na produtividade ou evitar que procrastinassem, nesta 67,4% afirmaram que acreditavam e 32,6% que talvez o projeto ajudaria. Um ponto importante sobre este questionamento é a existência da resposta “Não”, possibilitando os entrevistados afirmarem que não acreditavam na proposta do projeto, porém nenhum deles respondeu que não.

Figura 09 – Questão 9 do formulário



Fonte: Acervo do Autor (2022).

Por fim, mais de 90% dos entrevistados acredita que o Mi.GO pode ser de alguma forma útil como pode ser visto na Figura 9, onde em uma escala de 1 a 5, onde 1 significa pouco útil e 5 muito útil, e 34,8% dos entrevistados disseram que o projeto poderia ser muito útil.

4.1.3. Requisitos

Nesta seção são elencados os requisitos necessários para que o protótipo cumpra seu proposito com sucesso. No Quadro 8, estão os requisitos funcionais do Mi.GO.

Quadro 08 – Requisitos Funcionais

Número	**Nome**	**Descrição**
RF01	Cadastro de Metas	O aplicativo deve dispor de uma rotina que permita o usuário a criar metas de aquisição, tendo como principais campos: nome, descrição, valor da aquisição, escolha do período para juntar o montante, escolha da forma para juntar o montante no período definido, data para realização de cada passo definido e criação e edição de cada tarefa gerada.
RF02	Edição de Metas	O aplicativo deve dispor de uma rotina que permita o usuário a editar as metas de aquisição já criadas, tendo como campos editáveis o valor da aquisição, escolha do período para juntar o montante, escolha da forma para juntar o montante e criação e edição de cada tarefa gerada.

RF03	Criação, edição e exclusão de Tarefas via criação e edição de Metas	O aplicativo deve ser capaz de criar, editar e deletar tarefas automaticamente a partir da criação e edição de metas, entendendo cada passo definido durante a criação.
RF04	Cadastro de Tarefas	O aplicativo deve dispor de uma rotina que permita o usuário criar tarefas, tendo como campos o nome, data e hora para realização da tarefa.
RF05	Edição de Tarefas genérico	O aplicativo deve dispor de uma rotina que permita o usuário a editar as tarefas tendo como campos editáveis o nome, data e hora para realização da tarefa.
RF06	Listagem de Metas	O aplicativo deve possuir uma listagem de metas ordenada pela quantidade de tarefas concluídas da meta e filtradas pelas metas ainda não concluídas.
RF07	Item da lista de metas	O aplicativo deve apresentar cada item da listagem de metas com seu nome, quantidade de tarefas total, quantidade de tarefas concluídas e data a ser concluída.
RF08	Listagem de Tarefas	O aplicativo deve possuir uma listagem de tarefas ordenadas por data e hora de realização, iniciando-se da mais próxima. Além disso, deve ser possível filtrar por data e conclusão.
RF09	Item não selecionado da lista de Tarefas	O aplicativo deve apresentar cada item da listagem de tarefas com uma caixa de seleção para concluir a tarefa, seu nome, tempo até a realização da tarefa e botão de ações, que possibilitem a alteração e exclusão da tarefa.
RF10	Item selecionado da lista de Tarefas	O aplicativo deve apresentar o item selecionado da listagem de tarefas com uma caixa de seleção para concluir a tarefa, seu nome, meta que ele pertence (caso pertença) tempo até a realização da tarefa e botão de ações, que possibilitem a alteração e exclusão da tarefa.
RF11	Tela inicial	O aplicativo deve possuir uma tela inicial que acople as listagens de meta e tarefas, assim com um rodapé para navegação entre os recursos.
RF12	Rodapé da Tela inicial	O aplicativo deve possuir no rodapé da tela inicial os botões criação de metas e tarefas.
RF13	Cancelamento de um Cadastro ou edição	O aplicativo deve possuir nas rotinas de cadastro um botão que permita cancelar as ações tomadas pelo usuário até o momento.
RF14	Rodapé de um cadastro e edição	O aplicativo deve possuir em seu rodapé nas telas de cadastro um botão que permita o usuário salvar as ações tomadas até então e/ou avançar para os próximos passos.

RF15	Tela de primeiro acesso	O aplicativo deve dispor de uma tela de primeiro acesso a aplicação, e nela solicitar apenas o nome que o usuário gostaria de ser chamado e em seguida redirecionar o usuário a Tela inicial.

Fonte: Acervo do Autor (2022).

Já no Quadro 9, estão presentes os requisitos não funcionais, aquelas que estão relacionados às características do protótipo.

Quadro 09 – Requisitos não funcionais

Número	**Descrição**
RNF01	O aplicativo deve ser desenvolvido com a linguagem de programação Javascript e complementado com Typescript.
RNF02	O aplicativo deve ser desenvolvido para os principais sistemas operacionais de dispositivos móveis do mercado, sendo eles, Android e IOS.
RNF03	O aplicativo deve ser desenvolvido utilizando a biblioteca React Native, cujo qual possibilita a criação de uma aplicação nativa para os sistemas operacionais.
RNF04	O aplicativo deve ser desenvolvido com o framework Expo, que irá ajudar e otimizar o tempo de preparação para o desenvolvimento com React Native.
RNF05	O aplicativo deve armazenar todas as informações na memoria do dispositivo móvel.
RNF06	O aplicativo deve funcionar de maneira offline, ou seja, sem depender de uma conexão com a internet.
RNF07	O aplicativo deve conter uma paleta de cores definida, sendo apenas 4 cores e utilizadas em todo o seu escopo.
RNF08	O aplicativo deve possuir em todo escopo elementos familiares, não fugindo de elementos já utilizados em outras telas.
RNF09	O aplicativo deve possuir uma listagem de tarefas ordenadas por data e hora de realização, iniciando-se da mais próxima. Além disso, deve ser possível filtrar por data e conclusão.

Fonte: Acervo do Autor (2022).

No Quadro 10 estão as regras de negócios, que garantem que a usabilidade do protótipo cumpra determinados regras para um bom funcionamento.

Quadro 10 – Regras de negócio

Número	**Descrição**
RN01	O sistema não deve permitir a alteração do nome e tipo de uma meta.

RN02	O sistema não deve permitir a exclusão de tarefas vinculadas a uma meta.
RN03	O sistema deve excluir todos os itens vinculados a uma meta que foi excluída.
RN04	O sistema não deve permitir que seja criada tarefas com data e hora iguais a uma tarefa já existente

Fonte: Acervo do Autor (2022).

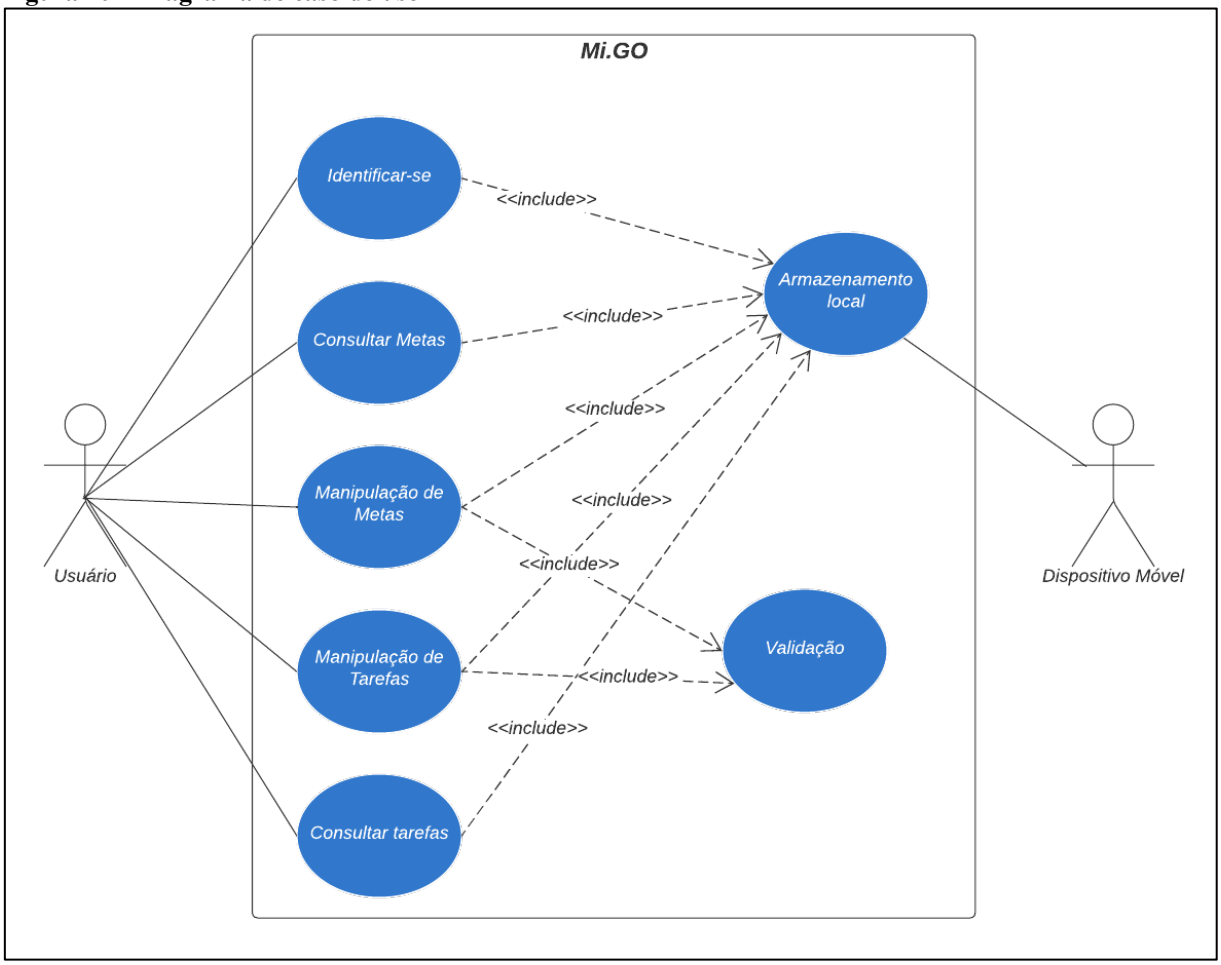
4.1.3. Diagramas

Nesta seção serão listados os diagramas de casos de uso, atividade e de entidade relacionamento para que se tenha um melhor entendimento nos fluxos e funcionamento do Mi.GO.

4.1.3.1. Diagrama de caso de uso

O diagrama de caso de uso tem o intuito de mostrar as funcionalidades que cada ator poderá executar dentro do protótipo. No Mi.GO existem dois atores, o usuário e o dispositivo móvel, conforme pode-se verificar na Figura 10.

Figura 10 – Diagrama de caso de uso

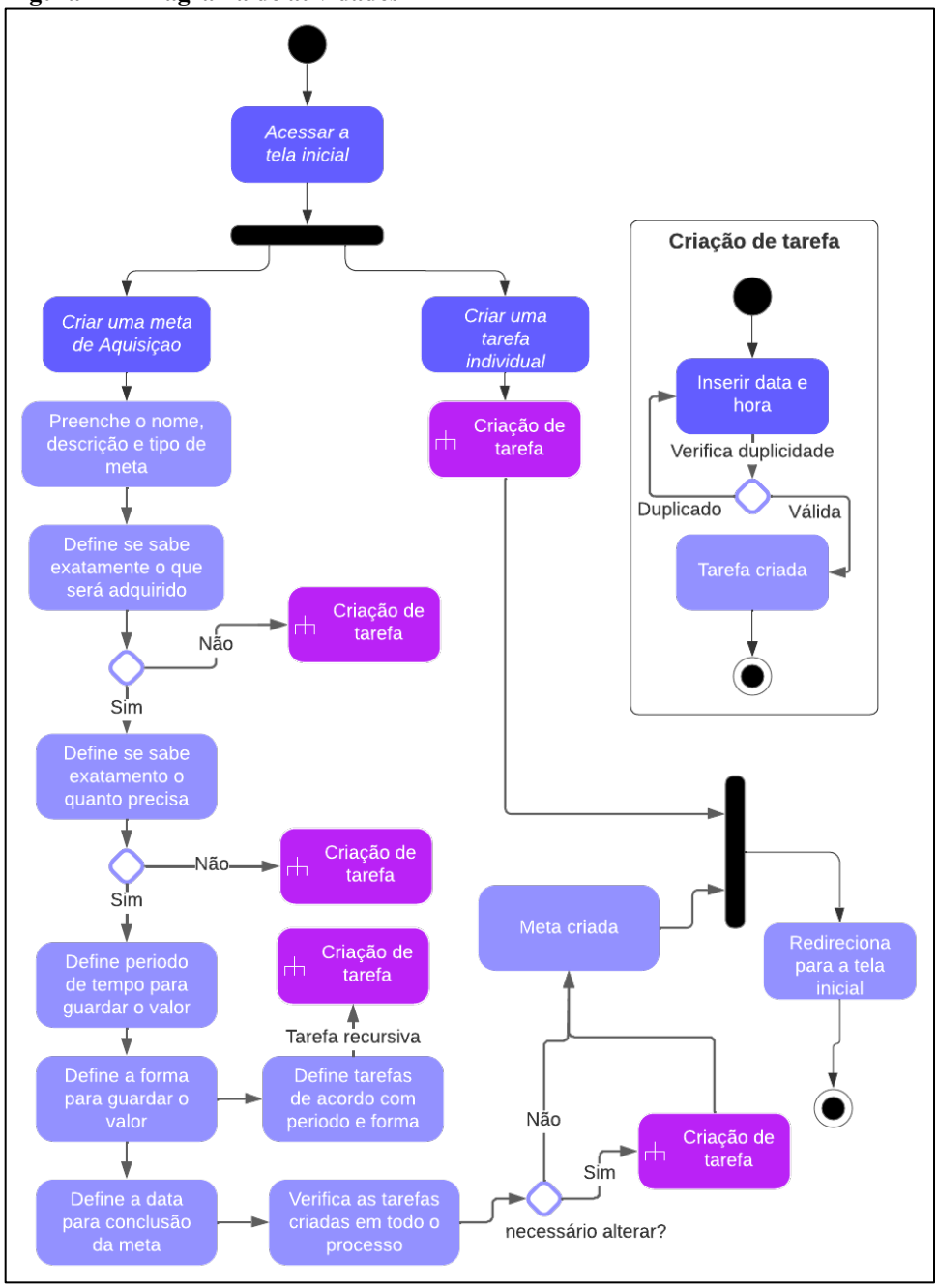


Fonte: Acervo do Autor (2022).

4.1.3.2. Diagrama de atividades

O diagrama de atividades demonstra as principais atividades que devem ser executadas e seu fluxo dentro do Mi.GO. O projeto possui dos fluxos principais que devem ser executados para que sua proposta seja concluída com êxito. A Figura 11 representa o diagrama de atividades de Mi.GO.

Figura 11 – Diagrama de atividades



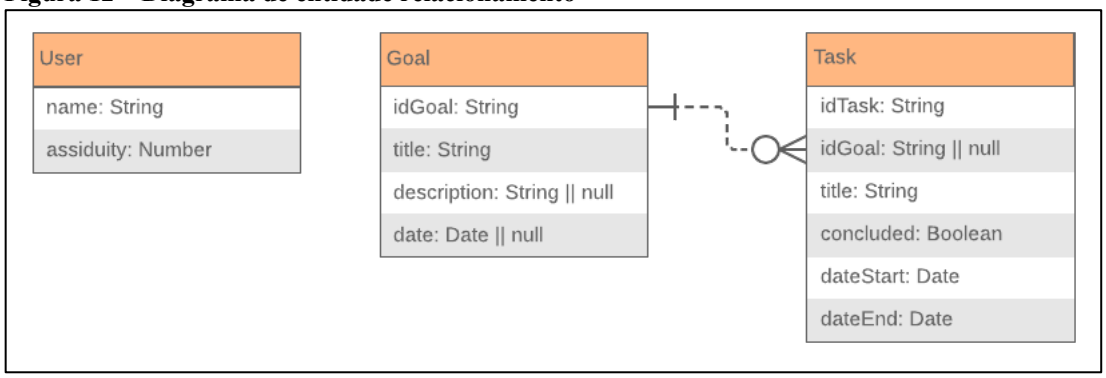
Fonte: Acervo do Autor (2022).

4.1.3.2. Diagrama de entidade relacionamento

O diagrama de entidade relacionamento representa as entidades, que podem ser pessoas, conceitos ou objetos e como elas se relacionam entre si. No Mi.GO as entidades não estão diretamente ligadas a um banco de dados, mas sim ao armazenamento local do dispositivo móvel, elas existem diretamente como interfaces. Estas interfaces garantem que os dados salvos

e manipulados serão fidedignos para que o Mi.GO funcione de forma correta. A Figura 12 representa as entendidas e relacionamento do Mi.GO.

Figura 12 – Diagrama de entidade relacionamento



Fonte: Acervo do Autor (2022).

4.2. IMPLEMENTAÇÃO

Nesta seção são detalhadas as ferramentas e tecnologias utilizadas para o desenvolvimento do protótipo Mi.GO, bem como as rotinas implementadas.

4.2.1 Técnicas e Ferramentas utilizadas

O Mi.GO foi iniciado a partir de uma ideia abstrata, apenas um conceito, para a construção do conceito, foi utilizado a ferramenta *whimiscal*, uma ferramenta de uso simples que possibilita a criação de mapas mentais, layouts simples, gráficos, fluxos e outros desenhos simples.

Após a construção do conceito, foi possível desenhar na ferramenta *Figma* um protótipo conceitual do projeto, neste já foram elaborados a disposição dos elementos na tela inicial e no formulário de criação da meta, assim como um caso de uso de criação de meta aquisitiva, caso este que consiste no protótipo desenvolvido em código. Com o desenho conceitual, foi possível obter resultados mais expressivos nos próximos passos da construção do projeto, já que o desenho foi utilizado como forma de apresentar o projeto aos usuários que seriam entrevistados na pesquisa de aderência e uso do protótipo, tal pesquisa utilizou a ferramenta *Google Formulários* que por sua vez, possibilitou obter dados concisos.

Validado a ideia do projeto, foi possível realizar o levantamento de requisitos, que corroborou com o desenho dos diagramas de caso de uso, estes por sua vez, foram desenhados na ferramenta *Lucid.app.*

Todo esse processo tornou possível a construção do protótipo em código. Para o desenvolvimento do protótipo foi utilizado a biblioteca *React Native*, que é uma biblioteca *JavaScript*, dentro do ambiente *node.js,* com ela foi possível desenvolver em um só código as aplicações nativas para iOS e Android. Para que fosse possível desenvolver com maior facilidade no *React Native,* já que cada aplicação nativa possui suas particularidades, foi utilizado o conjunto de ferramentas *Expo,* esse conjunto não só facilitou a configuração do ambiente para desenvolvimento, tornou possível executar a aplicação nos emuladores iOS e Android, como também trouxe bibliotecas com recursos interessantes para o desenvolvimento da aplicação, dessas, pode-se destacar o *react-navigation,* biblioteca que já trás configurado os navegadores do *React Native.*

O desenvolvimento do protótipo consiste apenas no front-end da aplicação, que como já citado, foi desenvolvido com *React Native*, *JavaScript* e *TypeScript,* já que não houve necessidade da criação de um *back-end* para seu funcionamento, bem como o armazenamento dos seus dados. Isso se deu devido a biblioteca do *React Native*, chamada *React Native Async Storage,* é possível verificar na Figura 13, um exemplo de código da busca de Metas no armazenamento local utilizando *Async Storage*, seu funcionamento se assemelha com a busca de dados em um banco, utilizando bibliotecas no *node.js,* também utiliza contratos, criando objetos com estrutura definida, podendo no caso serem considerados entidades.

Figura 13 – Busca de dados em armazenamento local

```
export async function goalsGetAll() {
  try {
    const storage = await AsyncStorage.getItem(GOAL_COLLECTION);

    const goals: string[] = storage ? JSON.parse(storage) : [];

    return goals;
  } catch (error) {
    throw error;
  }
}
```

Fonte: Acervo do Autor (2022).

Outro ponto importante a se destacar é a utilização da biblioteca *NativeBase,* que trás consigo componentes prontos que facilitam a criação da visualização da aplicação, bem como a possibilidade de estilização dos componentes mais semelhante aos das tags *HTML*. A Figura

14 mostra o código fonte do header da tela inicial da aplicação, bem como a utilização dos componentes do *NativeBase* sendo utilizados em conjunto com os *core componentes* do *React Native.*

Figura 14 – Componente HomeHeader com Nativebase

```
export function HomeHeader({ name }: Props) {
  return (
    <HStack bg="dark.100" pt={'10%'} pb={4} px={4} alignItems="center" roundedBottom="56">
      <VStack flex={1} alignItems="center">
        <Heading color="gray.100" fontSize="md" fontFamily="heading">
          Olá, {name}
        </Heading>
        <Text color="gray.100" fontSize="sm">
          O que deseja fazer hoje?
        </Text>
      </VStack>
    </HStack>
  );
}
```

Fonte: Acervo do Autor (2022).

Para finalizar, a biblioteca react-navigation auxiliou no desenvolvimento da navegação entre os recursos do Mi.GO, pois ela funciona como um Hook que salva o estado da navegação, assim como possui componentes próprios para a criação de um Header ou footer de navegação, na Figura 15 é possível analisar o fonte de uma chamada do seu Hook em uma tela e na Figura 16 está o componente AppRoutes, que utiliza o react-navigation para criar a navegação entra as telas do App

Figura 15 – useNavigation dentro de um componente

```
const navigation = useNavigation<AppNavigatorRoutesProps>();

function handleBackToHome() {
  navigation.navigate('home');
}
```

Fonte: Acervo do Autor (2022).

Figura 16 – AppRoutes como navegação do aplicativo

```tsx
export function AppRoutes() {

  const { sizes, colors } = useTheme();

  const iconSize = sizes[6];

  return (
    <Navigator screenOptions={{
      headerShown: false,
      tabBarShowLabel: false,
      tabBarActiveTintColor: colors.light[300],
      tabBarInactiveTintColor: colors.light[800],
      tabBarStyle: {
        backgroundColor: colors.dark[100],
        borderTopWidth: 0,
        height: Platform.OS === "android" ? 'auto' : '8%',
        paddingBottom: sizes[10],
        paddingTop: sizes[6],
        borderTopStartRadius: sizes['96'],
        borderTopEndRadius: sizes['96'],
      },
    }}>
      <Screen
        name='home'
        component={Home}
        options={{
          tabBarIcon: ({ color,  }) => (
            <HomeSvg fill={color} width={iconSize} height={iconSize} />
          )
        }}
      />

      <Screen
        name='create'
        component={Create}
        options={{
          tabBarIcon: ({ color }) => (
            <CreateSvg fill={color} width={iconSize} height={iconSize} />
          )
        }}
      />
    </Navigator>
  );
}
```

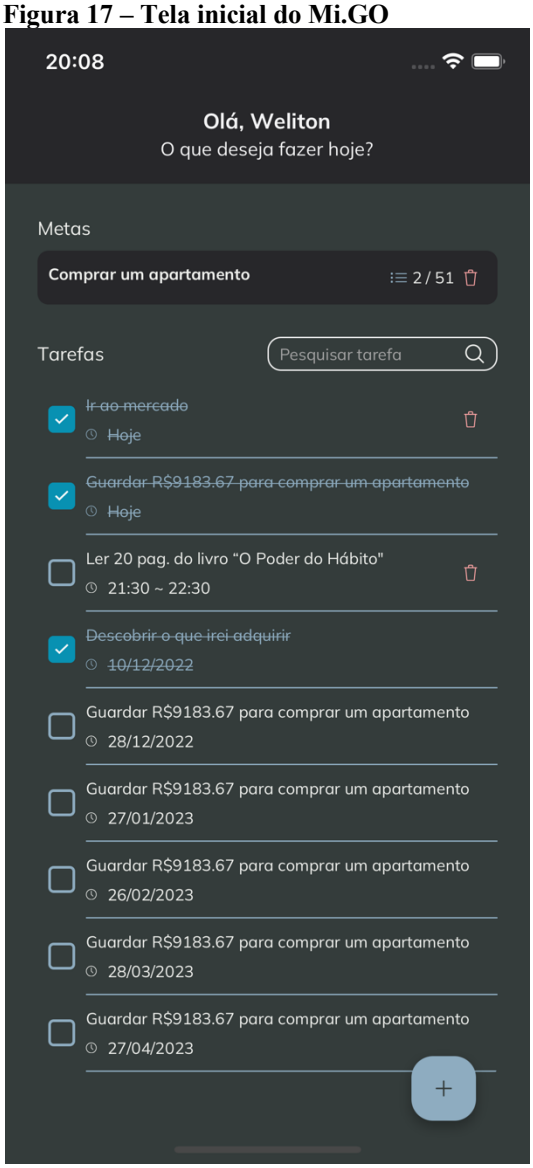
Fonte: Acervo do Autor (2022).

4.2.2 O Protótipo

O protótipo da aplicação engloba todo o diagrama de atividades (citado no item 4.1.3.2.), dito isso, nesta seção serão apresentadas as figuras que representam a criação de uma meta, bem como a tela inicial. Também é interessante apontar que, desde a concepção do projeto e do seu desenho, o protótipo recebeu melhorias em seu layout, para que a experiência do usuário fosse aprimorada.

Diante disso, na figura 17 pode-se verificar a tela inicial da aplicação, nela há um cabeçalho que interage com o usuário, no seu corpo a lista de metas, também a listagem de tarefas e por fim, em seu rodapé a navegação do aplicativo, onde é possível acessar os recursos de criação de metas e tarefas.

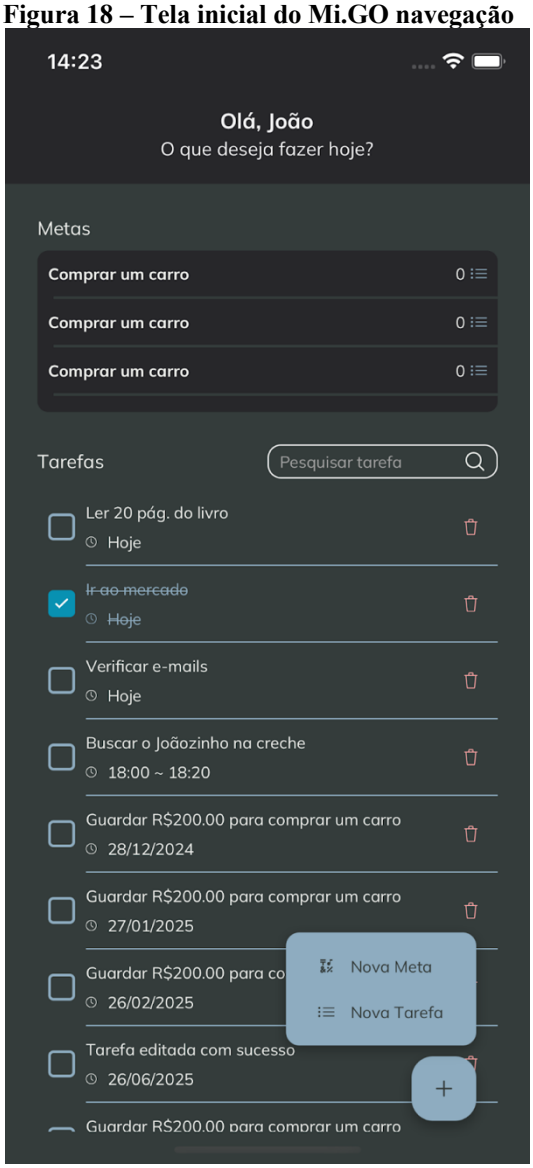
Figura 17 – Tela inicial do Mi.GO



Fonte: Acervo do Autor (2022).

Um melhor entendimento da navegação pela tela inicial está presente na Figura 18, onde pode-se observar que ao clicar no botão inferior direito do rodapé, abre uma pequena caixa flutuante que provê as opções de Nova meta e nova tarefa.

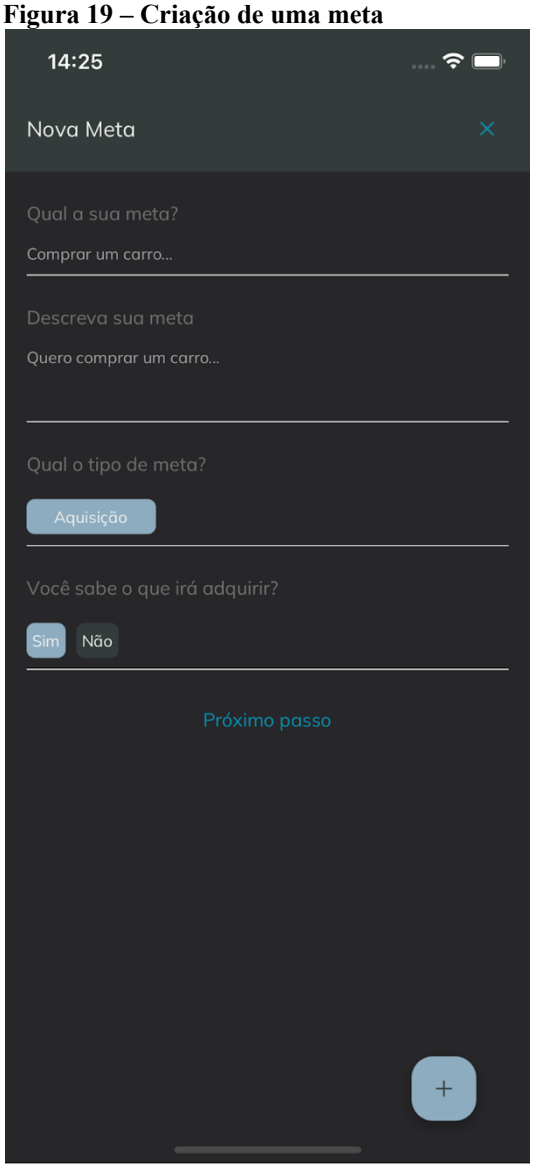
Figura 18 – Tela inicial do Mi.GO navegação



Fonte: Acervo do Autor (2022).

A criação de uma meta envolve várias interações com o usuário, onde o usuário passa por diversos formulários, visualmente parecidos, é possível entender melhor esta tela verificando a figura 19.

Figura 19 – Criação de uma meta



Fonte: Acervo do Autor (2022)

5. CONCLUSÃO

Analisando o cenário atual, é possível concluir que há grandes oportunidades para serem exploradas no mercado tecnológico, e com esse crescente de oportunidades, onde todo dia aparecem novas tecnologias, há possibilidade e facilidade em aplicar e desenvolver novas soluções e melhorar soluções já existentes. Além disso, a pandemia causou um impacto muito grande no dia a dia das pessoas, o que fez com que problemas antes deixados em segundo plano ficassem mais evidentes e novas necessidades surgissem, dando um espaço maior ainda para o desenvolvimento de novas soluções tecnológicas.

Foi levando em consideração este cenário então, mais especificamente o cenário de problemas relacionados com procrastinação, gerenciamento de tempo e produtividade que foi desenvolvido o Mi.GO, um protótipo que gerencia tarefas e simplifica metas pessoais, e essa simplificação partiu do principio que é muito mais fácil realizar pequenas tarefas, assim ele divide de forma inteligente essas metas em pequenas tarefas para que possam ser concluídas mais facilmente, o que fez ser possível concluir o objetivo principal do presente trabalho com êxito.

Para fazer com que o Mi.GO pudesse cumprir seu proposito inicial, foi necessário definir um caso de uso. O motivador deste objetivo específico era que seu escopo envolvia um extenso estudo e não seria possível desenvolver múltiplos tipos de metas em tempo hábil, já que toda sua inteligência de simplificação de metas parte de conceitos pré-definidos, portanto foi definido o caso de uso de metas do tipo aquisição, onde o foco é ajudar os usuários na conquista de bens materiais.

O Mi.GO, como proposito de ajudar as pessoas não teria sentido se a solução que propõe não fosse uma necessidade real para seus usuários, assim foi realizado a pesquisa de aderência onde foi apresentada a sua proposta e conceito em conjunto a um desenho conceitual desenvolvido com o *Figma,* que por sua vez apresentou a interação do usuário com a aplicação e a criação de uma meta para os entrevistados. Com o resultado da pesquisa, foi possivel concluir que a proposta do Mi.GO gerou resultados bastante positivos, tendo mais de 90% de aderência da aplicação entre os entrevistados, também que eles realmente possuíam problemas com a realização de suas metas e procrastinação. Diante disso, foi possível alcançar com grande êxito o segundo e terceiro objetivo específico.

Já pensando no quarto objetivo específico, o protótipo foi desenvolvido utilizando *React Native* em conjunto com *JavaScript* e *TypeScript*, uma das melhores e mais atuais tecnologias para o desenvolvimento de aplicações para dispositivos móveis. Além disso, para que fosse

possível seu desenvolvimento, foram levantados todos os requisitos, que nortearam a boa funcionalidade e entrega real de valor do Mi.GO, também foram desenhados os diagramas de atividade, de uso e entidade relacionamento, que simplificaram a visualização do funcionamento do protótipo, fazendo com que os dois últimos objetivos fossem alcançados.

O Mi.GO partiu de uma ideia simples e dor real, e em passos largos conseguiu atingir com êxito aquilo que estava proposta, se tornando uma aplicação real, palpável e utilizável.

5.1 TRABALHOS FUTUROS

O próximo passo do Mi.GO é ser capaz de ajudar o usuário com as mais diversas metas, levando em consideração não só metas do tipo aquisitivo, mas também financeiras, estudo, trabalho, viagens, entre outros. Posteriormente a isso, o Mi.GO receberá um novo recurso chamado Pomodoro, que se baseia na técnica de Pomodoro para gerenciamento de tempo e foco, que irá funcionar integrado com as tarefas do usuário.

Outro recurso importante, é uma IA integrada com o cadastro de metas, o Mi.GO será capaz de compreender muito melhor as necessidades do usuário, podendo ajuda-lo na melhor divisão das suas metas, e construção mais clara de suas tarefas.

Por fim, e não menos importante, o Mi.GO é um projeto que se tornará de fato uma aplicação, e será lançado nas principais lojas de aplicativos do mercado.

REFERÊNCIAS

BABEL. ***Wha tis Babel?.*** Disponível em: < https://babeljs.io/docs/en/index.html> Acesso em: 12 de outubro de 2022.

BECKER, Lauro. **O que é React Native?.** Disponível em: <https://www.organicadigital.com/blog/o-que-e-react-native/> Acesso em: 12 de outubro de 2022.

CAVALCANTE, Pablo Henrique Aguiar. **Introdução a Typescript: o que é e como começar?.** Disponível em: < https://blog.geekhunter.com.br/introducao-a-typescript/> Acesso em: 12 de outubro de 2022.

CUNHA, André. **React Native: o que é e tudo sobre o Framework.** Disponível em: <https://www.alura.com.br/artigos/react-native> Acesso em: 12 de outubro de 2022.

DIGITAL HOUSE. **15 frameworks mais usados em programação que você precisa conhecer.** Disponível em: <https://www.digitalhouse.com/br/blog/frameworks-mais-usados-em-programacao/> Acesso em: 29 de março de 2022.

DIGITAL HOUSE. **Front-end: o que é, para que serve e como aprender?.** Disponível em: <https://www.digitalhouse.com/br/blog/front-end-o-que-e-para-que-serve-e-como-aprender/> Acesso em: 11 de março de 2022.

EXPO. ***FileSystem.*** Disponível em: <https://www.docker.com/> Acesso em: 29 de junho de 2022.

FERNANDES, Diego. **Expo: o que é, para que serve e quando utilizar?.** Disponível em: <https://blog.rocketseat.com.br/expo-react-native/> Acesso em: 20 de outubro de 2022.

FLANAGAN, David. **JavaScript: o guia definitivo**. Grupo A, 2013.

GUIMARÃES, Bruna. **Home office produtivo: 20 dicas e 7 ferramentas para te auxiliar.** Disponível em: <https://www.gupy.io/blog/home-office-dicas-e-ferramentas> Acesso em: 5 de dezembro de 2022.

JSON. **Introdução ao JSON.** Disponível em: < https://www.json.org/json-pt.html> Acesso em: 21 de outubro de 2022.

JOHNSON, Nicole. **A procrastinação aumentou? A culpa é da pandemia.** Disponível em: <https://www.nationalgeographicbrasil.com/ciencia/2021/04/a-procrastinacao-aumentou-a-culpa-e-da-pandemia> Acesso em: 10 de outubro de 2022.

CALOMENO JUNIOR, Adil. **React Native: o que é, suas funcionalidades e suas vantagens.** Disponível em: < https://ateliware.com/blog/react-native> Acesso em: 10 de outubro de 2022.

MARCHIORI, Lucas. **React: o que é e como funciona essa biblioteca Javascript?.** Disponível em: < https://blog.betrybe.com/react/#2> Acesso em: 12 de outubro de 2022.

MDN. **HTML: Linguagem de Marcação de Hipertexto.** Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTML> Acesso em: 24 de março de 2022.

MDN. **CSS.** Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/CSS> Acesso em: 24 de março de 2022.

MDN. **O que é CSS?.** Disponível em: <https://developer.mozilla.org/pt-BR/docs/Learn/CSS/First_steps/What_is_CSS> Acesso em: 24 de março de 2022.

MORAES, Rodrigo. **Quarentena e home office facilitam procrastinação: como sair desse ciclo.** Disponível em: <https://www.uol.com.br/vivabem/noticias/redacao/2020/05/01/quarentena-e-home-office-facilitam-procrastinacao-como-sair-desse-ciclo.htm> Acesso em: 12 de outubro de 2022.

NATIVEBASE. ***Getting Started.*** Disponível em: <https://docs.nativebase.io/> Acesso em: 20 de novembro de 2022.

NODEJS. ***The V8 JavaScript Engine.*** Disponível em: <https://nodejs.dev/learn/the-v8-javascript-engine> Acesso em: 24 de junho de 2022.

NODEJS. ***Introduction to Node.js.*** Disponível em: <https://nodejs.dev/learn/introduction-to-nodejs> Acesso em: 12 de junho de 2022.

OLIVEIRA, Cláudio Luís V.; ZANETTI, Humberto Augusto P. **Javascript descomplicado - programação para web, IOT e dispositivos móveis**. Editora Saraiva, 2020.

ORUI, Heidy. **7 dicas para evitar a procrastinação.** Disponível em: <https://www.eql.com.br/usufruir/2021/12/7-dicas-para-evitar-a-procrastinacao/> Acesso em: 12 de outubro de 2022.

RASCIA, Tania. ***React Tutorial: An Overview and Walkthrough***. Disponível em: <https://www.taniarascia.com/getting-started-with-react/> Acesso em: 29 de junho de 2022.

REACT. **Estado e Ciclo de Vida.** Disponível em: <https://pt-br.reactjs.org/docs/state-and-lifecycle.html> Acesso em: 20 de novembro de 2022.

REACT. **Hooks de Forma resumida.** Disponível em: <https://pt-br.reactjs.org/docs/hooks-overview.html> Acesso em: 20 de novembro de 2022.

REACT. **Introdução.** Disponível em: <https://pt-br.reactjs.org/docs/getting-started.html> Acesso em: 12 de junho de 2022.

REACT. **React – Uma biblioteca JavaScript para criar interfaces de usuário.** Disponível em: <https://pt-br.reactjs.org/> Acesso em: 12 de junho de 2022.

REACT NATIVE. ***Core Components and Native Components***. Disponível em: <https://reactnative.dev/docs/next/intro-react-native-components> Acesso em: 20 de novembro de 2022.

ROVEDA, Ugo. **O que são variáveis JavaScript, para que servem e como criar?**. Kenzie. Disponível em: <https://kenzie.com.br/blog/variaveis-javascript/> Acesso em: 21 de novembro de 2022.

SOUTO, Mario. **O que é front-end e back-end?**. Alura. Disponível em: <https://www.alura.com.br/artigos/o-que-e-front-end-e-back-end> Acesso em: 11 de março de 2022.

TOTVS. **Front end: O que é, como funciona e qual a importância.** Disponível em: <https://www.totvs.com/blog/developers/front-end/> Acesso em: 12 de março de 2022.