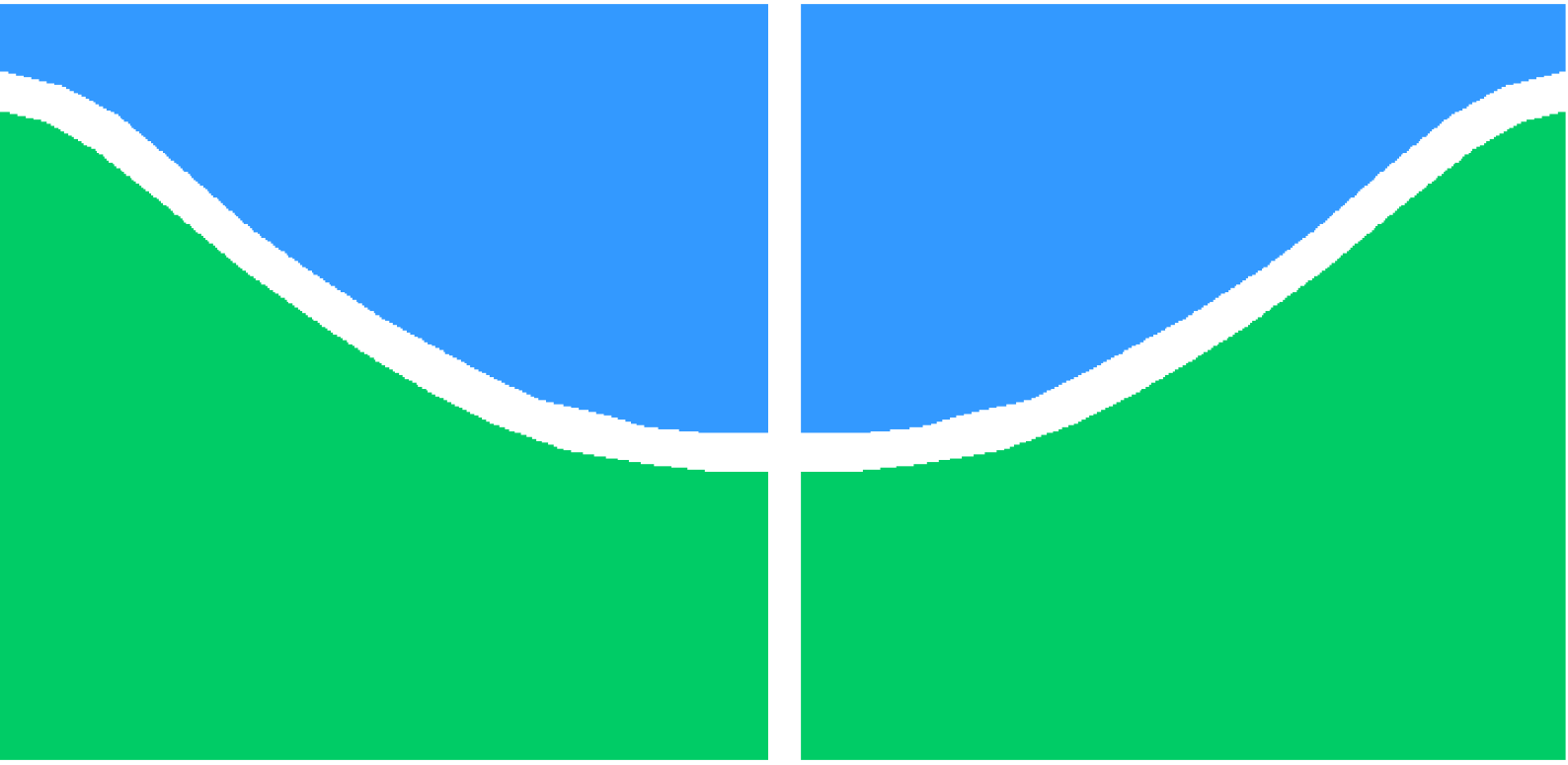


Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Bacharelado em Engenharia de Software

HelloWorld: Um Aplicativo de Ensino de Programação para Iniciantes

Autora: Gabriela Medeiros da Silva
Orientador: Prof. MSc. Giovanni Almeida Santos

Brasília, DF
2022

Gabriela Medeiros da Silva

HelloWorld: Um Aplicativo de Ensino de Programação para Iniciantes

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. MSc. Giovanni Almeida Santos

Brasília, DF

2022

Gabriela Medeiros da Silva

HelloWorld: Um Aplicativo de Ensino de Programação para Iniciantes/ Gabriela Medeiros da Silva. – Brasília, DF, 2022-

140 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. MSc. Giovanni Almeida Santos

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2022.

1. helloworld. 2. programação. I. Prof. MSc. Giovanni Almeida Santos. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. HelloWorld: Um Aplicativo de Ensino de Programação para Iniciantes

CDU 02:141:005.6

Gabriela Medeiros da Silva

HelloWorld: Um Aplicativo de Ensino de Programação para Iniciantes

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 19 de abril de 2022:

Prof. MSc. Giovanni Almeida Santos
Orientador

Profa. Dra. Carla Silva Rocha Aguiar
Convidado 1

Profa. Dra. Milene Serrano
Convidado 2

Brasília, DF
2022

Agradecimentos

Agradeço à Deus por ter me colocado nesse caminho, me mantido firme durante essa jornada e me ajudado em todos os momentos que pensei em desistir.

Agradeço aos meus pais por terem me mostrado que o melhor caminho é através da educação e por me ensinarem o quão privilegiada sou por poder escolhê-lo.

Agradeço ao meu marido, Herbert, por ser meu escape nos momentos difíceis, por nunca duvidar da minha capacidade e por ser o responsável por me fazer cogitar esse curso.

Agradeço às minhas irmãs por me suportarem e me alegrarem nos momentos de maior estresse e desânimo.

Agradeço a todos os meus amigos, principalmente à Geovana Ramos, Daniel Maike, Guilherme Guy e Joberth Rogers, que fizeram os momentos mais cansativos dessa graduação se tornarem histórias que levarei comigo para onde eu for.

Agradeço aos professores de Engenharia de Software da FGA, que dedicam o seu tempo para trazer conteúdos de qualidade incomparável, por serem exemplos de educadores e por demonstrarem uma paixão pelo ensino capaz de extravasar os portões da Universidade.

Agradeço ao meu orientador, Prof. MSc. Giovanni Santos, que me conduziu em todos os aspectos desse trabalho e por cada um que me ajudou, direta ou indiretamente.

"[Eu] Pensava que nós seguíamos caminhos já feitos, mas parece que não os há. O nosso ir faz o caminho.
(C. S. Lewis)

Resumo

A disseminação da tecnologia e a crescente demanda do mercado por desenvolvedores estão fazendo muitas pessoas se atraírem pela área. O interesse de pessoas com diferentes níveis de estudo e dificuldades de aprendizado evidencia um problema que muitos brasileiros enfrentam ao tentar aprender programação: o idioma. Apesar de existirem muitos aplicativos móveis que permitam aos usuários aprender programação de forma simplificada, os mais conhecidos na loja do Android, a Play Store, possuem seu conteúdo em inglês. Esse trabalho propõe o desenvolvimento de um aplicativo móvel para Android, nomeado "HelloWorld", que ensina lógica de programação através de explicações e atividades completamente em português. Para isso, será utilizado o Calango, um pseudo código em português, em todas as sessões de conteúdos que envolvam código programado.

Palavras-chaves: introdução à programação. algoritmo. Calango. aplicativo móvel.

Abstract

The spread of technology and the growing market demand for developers are making many people attracted to the area. The interest of people with different levels of study and learning difficulties highlights a problem that many Brazilians face when trying to learn programming: the language. Although there are many mobile applications that allow users to learn programming in a simplified way, the best known on the Android store, the Play Store, have their content in English. This work proposes the development of a mobile application for Android, named "HelloWorld", which teaches programming logic through explanations and activities completely in Portuguese. For this, Calango, a pseudo code in Portuguese, will be used in all content sessions that involve programmed code.

Key-words: programming introduction. algorithms. Calango. mobile application.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

MVP	*Minimum Viable Product*
PO	*Product Owner*
SO	Sistema Operacional
API	*Application Programming Interface*
DER	Diagrama Entidade-Relacionamento
HTTP	*Hypertext Transfer Protocol*
OMR	*Object-Relational Mapper*
MVC	*Model-View-Controller*
JSON	*JavaScript Object Notation*
CI	*Continuous Integration*
CRA	*Create React App*
RNFs	*Requisitos Não Funcionais*

Sumário

1 Introdução

Este capítulo apresenta uma breve justificativa desse trabalho, seus objetivos e, por fim, a estrutura do documento.

1.1 Justificativa

Possuir o conhecimento de programação torna-se mais atraente a cada dia, devido à crescente demanda do mercado por novos desenvolvedores. Para que as expectativas dos interessados na área sejam cumpridas, é necessário que essas pessoas passem pelo desafio de aprender o básico: a lógica de programação.

São muitas as dificuldades que novos aprendizes de programação têm que enfrentar, incluindo a forma de comunicação completamente diferente do que estão acostumados, e em um idioma não nativo. Esse trabalho busca oferecer uma alternativa de aprendizado de lógica de programação a partir de uma aplicação interativa, móvel e em português, reaproveitando a experiência de usuário e alguns elementos da interface de outros aplicativos amplamente disseminados no contexto das aplicações *mobile*.

Considerando o alto dinamismo dos conteúdos na área de tecnologia, esse trabalho também propõe uma aplicação Web, que permite o gerenciamento e a análise de conteúdos do aplicativo.

1.2 Objetivo Geral

Desenvolver um sistema integrado, constituído por um MVP (*Minimum Viable Product*) de um aplicativo móvel, que forneça um meio para o aprendizado de lógica de programação, e uma aplicação Web que permita a manutenção do conteúdo que será disponibilizado.

1.3 Objetivos Específicos

- Desenvolver um protótipo de aplicativo móvel que guie o desenvolvimento da interface da aplicação para dispositivos Android;
- Modular uma Base de Dados que permita o acesso aos dados necessários para execução da aplicação e em futuras análises;
- Popular o Banco de Dados de modo a visualizar o fluxo de ensino do usuário;
- Fornecer um aplicativo estável para dispositivos móveis, pronto para o uso de qualquer usuário interessado, baseado no protótipo de alta fidelidade, e
- Fornecer uma aplicação Web estável, pronta para ser utilizada por usuários cadastrados como administradores do sistema.

1.4 Estrutura do Documento

Esse documento foi dividido em 7 (sete) capítulos, incluindo a introdução, listados e descritos a seguir:

- Capítulo 2 - Metodologia de Pesquisa: Caracterização da metodologia de pesquisa utilizada nesse trabalho e levantamento da questão de pesquisa;
- Capítulo 3 - Referencial Teórico: Levantamento teórico e bibliográfico de alguns conhecimentos relevantes para o embasamento do trabalho;
- Capítulo 4 - Engenharia de Software: Capítulo dedicado à exposição de todos os itens relacionados a requisitos e metodologia de desenvolvimento;
- Capítulo 5 - Tecnologias e Arquitetura: Descrição de ferramentas utilizadas no desenvolvimento, da arquitetura de cada módulo do sistema e do *Deploy* e integração contínua;
- Capítulo 6 - Resultados: Indicadores de resultados gerados pelo uso e teste das aplicações, e

- Capítulo 7 - Conclusão: Abstração dos tópicos e resultado desse trabalho, o que foi desenvolvido e o que pode ser otimizado.

2 Metodologia de Pesquisa

Esse capítulo irá decorrer sobre como foram desenvolvidas as questões de conteúdo e pesquisa desse trabalho.

2.1 Questão de Pesquisa

Para iniciar o processo de pesquisa, a seguinte questão foi definida:

Quais requisitos e metodologias devem ser aplicadas a um sistema para que o mesmo seja capaz de facilitar o aprendizado inicial de um estudante de programação?

Essa questão foi definida considerando que a satisfação do usuário durante o uso de um aplicativo pode ser tão relevante quanto o conteúdo de ensino exposto por ele. Tendo isso em vista, o foco de pesquisa foi destacado pela elicitação e análise de requisitos.

2.2 Classificação da Pesquisa

Resumidamente, o objetivo de uma Pesquisa Científica é explorar o contexto de um determinado tema e gerar um conhecimento relevante a partir disso. Considerando os diversos temas e contextos a que uma pesquisa pode ser aplicada, existem numerosas variações da mesma, que podem ser classificadas a partir de quatro pontos de vista (FREITAS, 2013):

- da sua natureza;
- de seus objetivos;
- dos procedimentos técnicos, e
- da forma de abordagem do problema.

Do ponto de vista de sua natureza, essa pesquisa é classificada como uma **pesquisa aplicada**. Em uma pesquisa aplicada, a finalidade é gerar conhecimento para uma aplicação prática. Já em uma pesquisa básica, não há previsão de aplicação do conhecimento gerado (FREITAS, 2013).

Do ponto de vista dos objetivos, ela é classificada como uma **pesquisa exploratória**, caracterizada por possuir o objetivo de desenvolver mais proximidade com o assunto investigado ou formular hipóteses. A pesquisa exploratória, geralmente envolve levantamento bibliográfico, entrevistas com pessoas que conviveram com o contexto pesquisado e estudo de exemplos práticos para estimular a compreensão (FREITAS, 2013).

Do ponto de vista dos procedimentos técnicos, que diz respeito a como os dados da pesquisa são adquiridos, essa pesquisa se classifica como um **Estudo de Caso**. Um Estudo de caso é caracterizado pela busca de uma aplicação prática do aprendizado como solução de um problema social (FREITAS, 2013).

Do ponto de vista da forma de abordagem do problema, essa pesquisa é classificada como **qualitativa**. A pesquisa qualitativa considera haver uma relação dinâmica entre a realidade e o sujeito, que não pode ser expressa de forma numérica, gerando resultados que devem ser analisados de maneira indutiva. A coleta de dados de uma pesquisa qualitativa possui como fonte o ambiente natural (FREITAS, 2013).

3 Referencial Teórico

Esse capítulo possui o objetivo de fornecer um embasamento teórico e informações sobre assuntos citados ao longo desse trabalho.

3.1 Kanban

O *Kanban* é um método de controle de fluxo de tarefas, que propõe uma limitação das atividades em desenvolvimento (*Work in Progress*) e uma ampla visualização dos seus estados. Para que o *Kanban* seja aplicado da forma correta, é necessária a modularização de funcionalidades de maneira que cada item seja independente dos demais e sempre agregue valor ao usuário (AHMAD; MARKKULA; OIVO, 2013).

Uma forma simples de implementação do Kanban é a separação das tarefas em 3 (três) colunas: "todo (para fazer)", "doing (fazendo)"e "done (feito)". O formato ou tecnologia utilizada para realizar o processo não é ditada pela metodologia. Isso pode ser feito utilizando notas adesivas, um quadro branco, ou até mesmo um aplicativo que permita a categorização de tarefas e a visualização do quadro geral.

3.2 Metodologia *SCRUM*

O SCRUM é uma metodologia ágil com foco em entregas simplificadas de qualidade, e com valor para o usuário. Neste tópico, serão abordadas as definições de papéis, ciclos e eventos que a metodologia propõe.

Os papéis do SCRUM são geralmente definidos em *Product Owner* (PO), *Scrum Master*, e time de desenvolvimento.

O *Scrum Master* é responsável pelo processo e pela aplicação da metodologia. Ele é encarregado de garantir que os ritos sejam cumpridos corretamente, e

uma das suas principais atribuições é lidar e eliminar os impedimentos do time, sempre que isso for possível (SRIVASTAVA; BHARDWAJ; SARASWAT, 2017).

O PO é responsável por tratar de assuntos relacionados ao produto. Isso inclui prazos, tarefas da *sprint*, definição das funcionalidades, histórias de usuário e priorização de requisitos. Ele também é responsável por expor o valor agregado ao usuário à cada *sprint* (SRIVASTAVA; BHARDWAJ; SARASWAT, 2017).

O time de desenvolvimento inclui especialistas de qualidade, testadores de *software*, desenvolvedores, *designers*, e responsáveis pela produção do *software* (SRIVASTAVA; BHARDWAJ; SARASWAT, 2017).

Apesar desses três papéis serem os mais comuns, o SCRUM permite a flexibilização do processo e a adição de outras funções que podem ser importantes, dependendo do contexto do projeto.

A *sprint* é uma janela de entrega com intervalo de dias definidos, que se repete durante o desenvolvimento do *software*, em que o PO trabalha em colaboração com o time de desenvolvimento para uma evolução e entrega contínua do *software*. Isso significa que, ao final de cada *sprint*, deve haver um incremento de funcionalidades ao *software* (SRIVASTAVA; BHARDWAJ; SARASWAT, 2017).

A Tabela 1 descreve os principais ritos do SCRUM, o momento que devem ser realizados, sua frequência, e sua duração.

3.3 Aquitetura *Model-View-Controller*

A arquitetura *Model-View-Controller* (MVC) consiste em um padrão de projeto que divide a aplicação em três camadas: *Model*, *View* e *Controller*. A camada de modelo empacota objetos e fornece funções que lidam diretamente com ele, além de cooperar com a camada de controle para fornecer aos usuários regras de negócio mais complexas. Nesse caso, o modelo lida com métodos de alteração, criação e obtenção de informações relacionadas às suas instâncias, e o controlador utiliza desses dados e funções para aplicar regras específicas e mais complexas. Por último, tem-se a camada de visualização, responsável pela interface e interação com o usuário/cliente (NING et al., 2008).

Tabela 1 – Ritos do SCRUM

Evento	**Frequência**	**Duração**	**Descrição**
Daily Meeting	Diariamente durante todo o ciclo de desenvolvimento	Máximo de 15 minutos	Cada membro fala de forma resumida o que fez desde a última *daily*, o que está fazendo e se tem algum impedimento para realização daquela tarefa.
Sprint Review	Ao final de cada sprint	Não definido	Nesse rito, o time de desenvolvimento deve apresentar a evolução do produto ao PO e *stakeholders* do projeto.
Sprint Retrospective	Ao final de cada sprint	Não definido	O time se reúne, discute o que foi feito na *sprint*, os pontos positivos, negativos e o que pode ser melhorado.
Sprint Planning	No início de cada sprint	Não definido	O PO apresenta quais as tarefas foram priorizadas para a *sprint*, o time de desenvolvimento discute as questões técnicas, estima o tempo de cada uma e se as estimativas preencherem a *sprint*, o planejamento é finalizado.

Para que a arquitetura seja seguida e suas vantagens sejam válidas, algumas regras devem ser aplicadas:

- A camada de modelo não pode ser dependente de nenhuma outra camada. Porém, é permitido que uma entidade possua diferentes tipos de relacionamentos com as outras;

- Não é permitido que um item de controle possua dependência com um item de visualização. Unidades de controle podem ter acesso a outros controladores e à camada de modelo apenas, e

- Não deve haver regras de negócio na camada de visualização. Essa camada deve fazer uso dos controladores para esse fim, e sua responsabilidade deve ser retornar a interface para o usuário.

3.4 Arquitetura de Componentes

A arquitetura baseada em componentes possui como característica a composição de um *software* em pequenos pedaços, chamados componentes. A definição de um componente pode ser abstraída a uma unidade que possui contratos de utilização e responsabilidade conhecida (CRNKOVIC, 2001).

Os componentes podem ser compostos por outros componentes, porém sua implementação deve ser desconhecida para o componente consumidor. Outra característica importante é que um componente só deve ser atualizado se valores diretamente ligados a ele forem modificados. Isso possibilita que a aplicação não seja inteiramente atualizada sempre que há uma alteração de valor em um componente (CRNKOVIC, 2001).

De forma resumida, uma aplicação orientada a componentes é um sistema formado por pedaços menores e independentes, responsáveis por sua própria alteração, regras e interface.

3.5 Calango

O Calango é uma aplicação educacional que permite o desenvolvimento de algoritmos em português através de pseudocódigos, facilitando o primeiro contato com programação (SILVA; SANTOS; RISSOLI, 2020). A versão 1.0 da aplicação pode ser baixada nesse site (último acesso em abril de 2022), e não possui requisito relacionado ao Sistema Operacional (SO), bastando ter o JAVA instalado na máquina para conseguir utilizá-lo.

O pseudocódigo utilizado pelo Calango é uma abstração da linguagem C, e foi desenvolvido para facilitar o reconhecimento de lógica de programação a partir de uma linguagem mais simples e natural. Na Figura 1, pode ser visualizada a tela inicial da aplicação.

De acordo com Silva, Santos e Rissoli (2020), os benefícios do uso do Calango como apoio ao aprendizado de programação incluem:

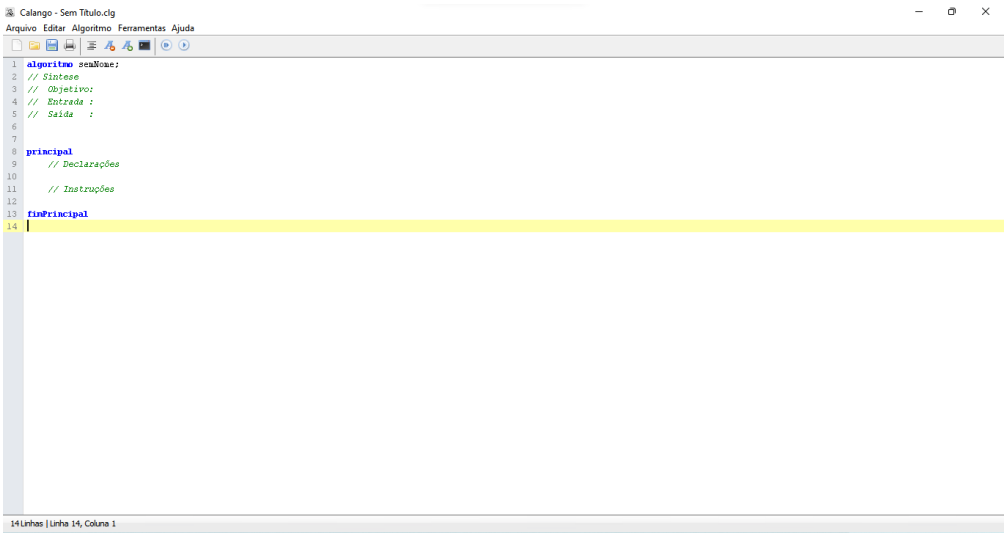
- Facilidade no entendimento do código para nativos da língua portuguesa;
- Depuração simplificada, que possibilita a visualização do passo a passo do código e das alterações nos valores das variáveis em tempo real;
- Oportunidade de autonomia de aprendizado ao estudante iniciante, e
- Assistência na resolução e amostragem de erros no código.

3.6 Introspecção

A Introspecção é uma técnica de levantamento de requisitos, em que o desenvolvedor se coloca no lugar de um usuário específico, e descreve quais são as suas necessidades dentro de um contexto determinado (GOGUEN; LINDE, 1993).

De acordo com Aurum e Wohlin (2006), só é recomendado o uso da Introspecção para elicitação de requisitos, quando o desenvolvedor possuir um alto grau de familiaridade com os contextos aplicados e com os possíveis perfis de usuário.

Figura 1 – Tela Inicial do Ambiente de Desenvolvimento do Calango



Fonte: a autora

3.7 Entrevista de Questionário

Uma entrevista de Questionário é uma técnica de elicitação de requisitos, onde algumas pessoas (preferencialmente potenciais usuários) são questionadas sobre pontos relevantes do produto. Nessa categoria de entrevista, as perguntas geralmente possuem respostas fechadas e objetivas (GOGUEN; LINDE, 1993).

Entre as vantagens de se utilizar a entrevista como um instrumento, está a alta chance de respostas relevantes, se as perguntas forem assertivas, além de evitar mais de uma interpretação durante a sua leitura. A dificuldade em garantir que isso aconteça está entre os pontos complexos em aplicá-la (GOGUEN; LINDE, 1993).

3.8 Requisitos Não Funcionais

Um *software* é um produto composto por duas naturezas de requisitos. Uma dessas naturezas, a de Requisito Funcionais, é capaz de documentar e descrever funcionalidades concretas, e a outra, de Requisitos Não Funcionais (RNFs), nos desafia no cumprimento das necessidades do usuário final, que nem sempre podem

ser descobertas ou sequer notadas por ele (CYSNEIROS; LEITE, 2001).

Os RNFs são demandas gerais, que não descrevem uma funcionalidade de sistema (CYSNEIROS; LEITE, 2001). Esses requisitos podem ser divididos em muitas categorias como desempenho, manutenibilidade, portabilidade, usabilidade, entre outras.

Além das categorizações já descritas, os RNFs podem ser tipificados como qualitativos, aqueles que não podem ser medidos através de métricas numéricas, ou como quantitativos, no qual o seu critério de satisfação pode ser dado por um número exato (CYSNEIROS; LEITE, 2001).

A dificuldade de percepção dos usuários em relação aos requisitos não funcionais é responsável pela invalidade de elicitação por meio de técnicas que necessitem das opiniões diretas do usuário, como as de entrevista e introspecção.

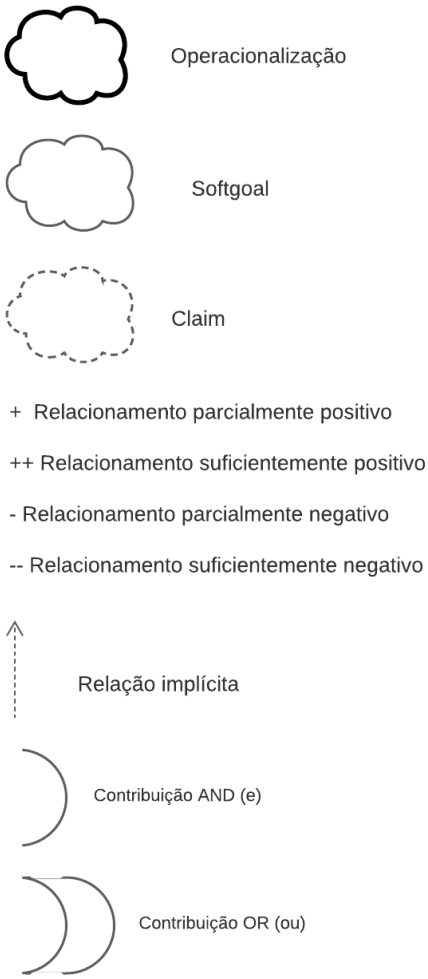
3.9 NFR *framework*

O *framework* NFR é uma ferramenta utilizada para explorar e reproduzir RNFs, além de auxiliar no desenvolvimento de soluções específicas para cada caso de uso (CHUNG et al., 2000).

No NFR *framework*, requisitos não funcionais são objetivos a serem cumpridos e fragmentados em objetivos menores, gerando relacionamentos entre eles. Esses relacionamentos podem ser positivos, quando um objetivo contribui para implementação de outro, ou negativo, quando a realização de um objetivo atrapalha o cumprimento de outro. Essa quebra de objetivos deve ser feita até que se chegue aos que são denominados "operacionalizações", objetivos que não podem mais ser quebrados e que permitem implementações (CYSNEIROS; LEITE, 2001).

Para representar visualmente essa cadeia de objetivos, são usadas as notações presentes na Figura 2.

Figura 2 – Representação do Framework NFR



Fonte: Promobit, 2020

3.10 Aplicações Educacionais

Nesse tópico, serão apresentadas as aplicações que possuem similaridades com a aplicação "HelloWorld". Nem todas as aplicações possuem o objetivo de ensinar programação, mas possuem outras similidades que são interessantes de serem comparadas. Na Tabela 2, pode-se ver algumas das características citadas de forma sintetizada.

3.10.1 Duolingo

Aplicativo educacional de ensino de línguas estrangeiras. Permite a navegação através de módulos. Seu incentivo de uso baseia-se na gamificação, de forma que o usuário é recompensado com prêmios a cada nova atividade realizada.

- Similaridades:
  - Trabalha com sessões de explicações e atividades, e
  - Possui fases e módulos.
- Diferenciais:
  - Não é relacionado com o ensino de programação.
- Acesso: link.

3.10.2 Mimo

Um aplicativo que proporciona ao usuário a opção de escolher entre o aprendizado da linguagem de programação *Pyhton* e a *stack* de desenvolvimento Web.

- Similaridades:
  - Possui questões interativas, e
  - Possui uma jornada dividida em sessões.
- Diferenciais:
  - O aplicativo e seu conteúdo são em inglês;
  - Não possui suporte a pseudocódigo;
  - Possui um período gratuito de 7 (sete) dias. Após esse tempo, é cobrado R$199.99/ano, e
  - Não possui a opção de aprender lógica de programação.
- Acesso: link.

3.10.3 Programming Hub

Um aplicativo com vários cursos relacionados à programação e assuntos similares.

- Similaridades:
  - Possui uma jornada dividida em sessões;
  - Possui questões interativas, e
  - Possui a opção de aprender lógica de programação.
- Diferenciais:
  - O aplicativo e seu conteúdo são em inglês;
  - Não possui suporte a pseudocódigo, e
  - Para ter acesso a todos os recursos, é necessário pagar a versão PRO.
- Acesso: link.

3.10.4 Grasshopper

Uma aplicação, com opção de acesso pela WEB ou pelo celular, que ensina programação a iniciantes através da linguagem *JavaScript*.

- Similaridades:
  - Possui uma jornada dividida em sessões, e
  - Possui questões interativas.
- Diferenciais:
  - Não possui suporte a pseudocódigo (os códigos são em inglês).
- Acesso: link.

Tabela 2 – Características das Aplicações Similares

Aplicativo	**Pseudocódigo**	**Português**	**Gratuito**	**Gamificado**
Duolingo	Não	Sim	Sim	Sim
Programming Hub	Não	Não	Sim	Não
Grasshopper	Não	Sim	Sim	Sim
Mimo	Não	Não	Não	Sim

4 Engenharia de Software

Esse capítulo irá decorrer sobre os métodos de gerenciamento e planejamento desse trabalho, as técnicas de levantamento de requisitos utilizadas e os requisitos elicitados.

4.1 Gestão do projeto

Buscando uma evolução contínua e incremental desse documento e do sistema proposto por ele, a metodologia utilizada inclui um desenvolvimento em paralelo das aplicações e dessa monografia.

Os ciclos de desenvolvimento possuem a duração de 7 (sete) dias, e as tarefas planejadas para cada um deles representam funcionalidades completas, com *Back-End* e *Front-End*, para garantir entrega de valor para o usuário final.

Apesar dos ciclos de entrega contínua e a busca por um desenvolvimento ágil ter relação com a metodologia SCRUM (Tópico 3.2), não se pode dizer que ela foi implementada em sua completude nesse processo, pois, para isso, teriam que ter ocorrido os eventos do SCRUM, o desenvolvimento dos artefatos e a representação dos seus papéis.

Os artefatos mutáveis durante o desenvolvimento do sistema possuem uma tabela de rastreabilidade, que indica as modificações feitas e a data realizada. Essas tabelas podem ser encontradas no Apêndice E.

4.1.1 Cronograma

O cronograma geral dessa proposta foi dividido em duas partes. Na primeira, há apenas o planejamento das entregas relacionadas a esse documento e à aplicação móvel, já incluindo os *endpoints* e recursos requisitados por suas funcionalidades na *Application Programming Interface* (API). Na segunda, há foco no planejamento das funcionalidades da aplicação WEB.

Tabela 3 – Primeira Parte do Cronograma: Desenvolvimento da Aplicação *Mobile* e da Documentação

Semana	**Requisitos**	**Documentação**	**Funcionalidade**
De: 04/07 Até: 10/07	-	- Levantamento de Requisitos - Cronograma	- Iniciar repositórios no GIT
De: 11/07 Até: 17/07	-	- Protótipo de alta fidelidade [APP]	- Conteinerização [API]
De: 18/07 Até: 24/07	MB16, MB17	- Diagrama Entidade-Relacionamento (DER) - Adicionar tarefas ao Trello	- Cadastro - Autenticação (Login)
De: 25/07 Até: 31/07	-	- Capítulo 4: Requisitos	- *Onboarding*
De: 01/08 Até: 07/08	MB11, MB5	- Capítulo 3: Metodologia	- História (Sessões e Capítulos) - Tela principal
De: 08/08 Até: 14/08	-	-	-
De: 09/08 Até: 21/08	MB3, MB9	- Diagrama de Pacotes [API] - Diagrama de Pacotes [APP] - Diagrama de Pacotes [WEB]	- Sessão de explicação (Markdown)
De: 22/08 Até: 28/08	MB12	- Diagrama de integração dos sistemas	- Telas de pesquisa de satisfação
De: 29/08 Até: 04/09	MB19, MB13	- Capítulo 5: Tecnologia e Arquitetura	- Editar perfil - Ver Perfil
De: 05/09 Até: 11/09	-	- Capítulo 2: Referencial teórico	- Integração e *(*deploy) contínuos da API
De: 12/09 Até: 18/09	PS1, PS2, PS3	- Capítulo 6: Arquitetura da Informação	- Integração e *(*deploy) contínuos do aplicativo móvel
De: 19/09 Até: 25/09	MB4, MB1, MB2	- Capítulo 7: Resultados	- Sessão de atividades
De: 26/09 Até: 02/10	MB23	- Capítulo 8: Conclusão - Capítulo 1: Introdução	
De: 03/10 Até: 09/10	MB6	- Programar a apresentação	- Recuperar Senha
De: 10/10 Até: 16/10	MB10		- Questões Avulsas - Filtro de Questões
De: 17/10 Até: 23/10	MB20, MB21	APRESENTAÇÃO	- Tela de Resumo (Estatísticas)

Na Tabela 3, pode-se ver a primeira parte do planejamento semanal das entregas no âmbito de funcionalidade e de documentação. A primeira coluna da tabela mostra o intervalo de duração daquele ciclo, começando aos domingos e finalizando aos sábados. A segunda coluna relaciona os requisitos funcionais às funcionalidades que foram planejadas para serem desenvolvidas naquela semana. Na quarta coluna, podem ser vistas as funcionalidades baseadas nos requisitos desse sistema, presentes na Tabela 6, além de algumas tarefas necessárias para garantir a integração e o *deploy* contínuo.

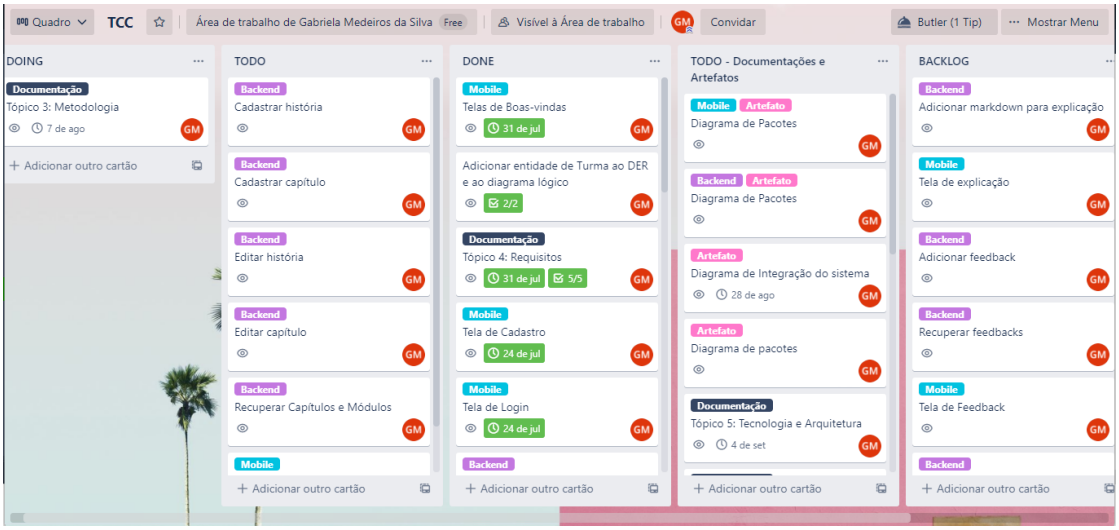
A segunda parte do cronograma é similar à tabela anterior, porém há uma coluna a mais para funcionalidades da interface *WEB*. Ela pode ser vista na Tabela

4.

4.1.2 Kanban

A Figura 3 apresenta a aplicação do Kanban, feita através da plataforma *Trello*, com o uso de um quadro composto por 5 (cinco) listas, que serão mais bem explicadas nos tópicos abaixo. Foram feitas algumas adaptações no uso comum da metodologia, como a adição de uma lista com as tarefas de desenvolvimento desse documento, e a criação de uma lista de *backlog*, contendo as funcionalidades relacionadas às aplicações do sistema.

Figura 3 – Captura do Quadro "TCC"no Trello do Dia 3 de Agosto de 2021



Fonte: a autora

- ***DOING***: tarefas que estão em andamento e devem ser finalizadas no ciclo de desenvolvimento corrente;
- ***TODO***: tarefas que devem ser concluídas até o fim da *sprint*, mas ainda não foram iniciadas;
- ***DONE***: tarefas já concluídas;
- ***BACKLOG***: tarefas relacionadas às funcionalidades já mapeadas, mas que não foram priorizadas para o ciclo atual e

Tabela 4 – Segunda Parte do Cronograma

Semana	**Requisitos**	**Documentação**	**Móvel**	**Web**
De: 31/10 Até: 06/11	WEB11	-	-	- Login
De: 07/11 Até: 13/11	WEB18	-	-	- Cadastro de turmas
De: 14/11 Até: 20/11	WEB5	-	-	- Lista de Turmas - Lista de Usuários - Detalhe de uma turma - Detalhe de um usuário
De: 21/11 Até: 27/11	WEB13	-	-	- Lista de Módulos - Detalhes de um Módulo
De: 28/11 Até: 04/12	WEB14, WEB15	-	-	- Lista de Capítulos - Detalhe de um capítulo
De: 05/12 Até: 11/12	WEB15, WEB16	-	-	- Lista de questões - Detalhe de uma questão
De: 12/12 Até: 18/12	WEB6	-	-	- Criar questão
De: 19/12 Até: 25/12	WEB7	-	-	- Editar questões - Editar opções de respostas - Excluir opção de resposta
De: 26/12 Até: 01/01	WEB3, WEB4	-	-	- Cadastrar capítulo - Editar capítulo
De: 02/01 Até: 08/01	-	-	-	- CI/CD WEB
De: 09/01 Até: 15/01	WEB10, WEB20	-	-	- Cadastro de usuário
De: 16/01 Até: 22/01	WEB1, WEB2	-	-	- Criar Módulo - Editar Módulo
De: 23/01 Até: 29/01	-	-	-	- Popular banco de dados de produção - Diagrama de Pacotes
De: 30/01 Até: 05/02	WEB12 MB23	- Requisitos não funcionais	- Recuperar Senha	- Configuração de E-mail
De: 06/02 Até: 12/02		- Metodologia de Pesquisa - Formulário de validação de requisitos	- Adicionar opção de participação em pesquisa	-
De: 13/02 Até: 19/02	MB15, WEB19	- Requisitos base (Duolingo)	- Completar cadastro do usuário cadastrado pelo site	- Importação de usuários
De: 20/02 Até: 26/02	MB18	-	- Excluir conta	- Remover Turma
De: 27/02 Até: 05/03	WEB17	-	-	- Tela de resultados
De: 06/03 Até: 12/03		- Páginas do Pré texto	-	-
De: 13/03 Até: 19/03	-	- Atualizar Cronograma		-
De: 20/03 Até: 26/03	-	- Resultados	- Melhorias e Correções de *Bugs*	-
De: 27/03 Até: 02/04	-			-
De: 03/04 Até: 09/04	-	- Programar a apresentação	-	-
De: 10/04 Até: 16/04	-		-	-
De: 17/04 Até: 23/04	-	-	-	-
De: 24/04 Até: 30/04	-	Apresentação	-	-

- ***TODO* - Documentação e Artefatos**: tarefas de documentação, que não são relacionadas ao desenvolvimento da aplicação final, e que não agregam valor ao usuário.

4.2 Requisitos Funcionais

Nessa sessão, serão abordados os métodos utilizados para a elicitação dos requisitos funcionais.

A análise exploratória das funcionalidades do *Duolingo* foi o principal método de elicitação de requisitos, sendo que a introspecção (Sessão 3.6) e o questionário (Sessão 3.7) também foram realizados para esse fim.

Em busca de uma maior similaridade com a aplicação proposta, foram feitas 3 (três) introspecções, nas quais cada uma delas expõe um potencial usuário do sistema.

4.2.1 Duolingo

Como exposto anteriormente, as técnicas de elicitação de requisitos foram guiadas por uma exploração de funcionalidades presentes no aplicativo móvel *Duolingo* para Sistemas Operacionais *Android*.

Por que o *Duolingo* foi escolhido para esse embasamento? Apesar do objetivo desse trabalho ser relacionado com ensino de programação, o público alvo da aplicação móvel são usuários em um contato inicial com esse novo contexto. O *Duolingo*, sendo um aplicativo para os que querem começar a aprender um novo idioma, demonstrou com sucesso como cativar pessoas, mesmo que ainda não possuam um conhecimento amplo sobre o assunto.

O sucesso do Duolingo pode ser visto através das plataformas mais conhecidas de aquisição de aplicativos. Apenas na *Play Store*, loja de aplicativos para dispositivos *Android*, o aplicativo obteve mais de 100 milhões de *downloads*, contando com uma nota de 4.7, média de quase 12 milhões de avaliações fornecidas por usuários da Store (2022).

Na Tabela 5, estão listadas as funcionalidades notadas durante a exploração do aplicativo. Visando manter o foco na experiência de aprendizado do usuário, serão listadas apenas as funcionalidades relacionadas com a jornada de ensino, e que podem ser reutilizadas para ensino de programação.

Tabela 5 – Funcionalidades da Aplicação Duolingo

Funcionalidade
Jornada separada em sessões
Possibilidade de aumento de nível em um mesmo capítulo
Opção de ler a explicação de um capítulo ou fazer apenas questões
Botão para verificar a resposta de uma questão após selecioná-la
Ganho de "conquistas"por diferentes marcos
Uso de imagens em todas as telas
Questões interativas
Atividades de diferentes níveis
Feedbacks sonoros ao errar ou acertar uma questão
Histórias que aplicam o conhecimento aprendido a um contexto
Quantidade de erros limitados por dia
Fórum de perguntas para cada questão
Sugestão de questões para praticar o assunto de uma resposta errada

4.2.2 Primeira Introspecção

Contexto: Uma jovem de 20 anos que ainda não fez a escolha de qual curso deseja fazer na faculdade. Ela está pesquisando diversas áreas possíveis de ingresso, inclusive desenvolvimento de *software*, mas tudo parece ser muito difícil e cansativo. Ela tem interesse em saber mais do assunto, pois sabe que o mercado está aquecido.

Necessidades do Contexto:

- Disponibilizar o aplicativo para ser baixado pelo celular;
- Possuir uma página de *download*;
- Explicar de forma resumida o objetivo do aplicativo;

- Possuir imagens que demonstram a facilidade e a simplicidade em utilizar a aplicação;
- Possuir uma interface simples e intuitiva;
- Possuir técnicas que tornem a navegação e a experiência simples;
- Fazer *login* para acessar a aplicação;
- Permitir gerenciamento da conta (criar, editar e excluir);
- Possuir um fluxo de aprendizado em fases, para que o usuário seja habituado, gradualmente, ao conteúdo;
- Possuir questões que levem a um melhor entendimento do conteúdo;
- Salvar o progresso do usuário, e
- Conseguir fazer questões de um nível, mesmo quando ele já foi concluído.

4.2.3 Segunda Introspecção

Contexto: Um estudante do primeiro semestre do seu curso de graduação está fazendo sua primeira matéria de programação. Seu principal meio de transporte é o ônibus. Ele não entende alguns detalhes do conteúdo, e sente que isso está se acumulando a cada novo assunto da matéria. Sua principal necessidade é um meio de estudo que ele possa utilizar em seu tempo em intervalos do dia, e durante seu trajeto entre casa e faculdade.

Necessidades do Contexto:

- Possuir questões para maior entendimento do conteúdo, e que possam ser acessadas rapidamente;
- Utilizar textos curtos e explicativos que possam ser lidos em pequenos intervalos de tempo;
- Mostrar quantas questões de cada assunto foram respondidas corretamente ou erroneamente, para que o usuário saiba quais os tópicos ele precisa estudar mais, e

- As questões devem utilizar conhecimentos já ensinados em níveis anteriores, para que o usuário adquira familiaridade com todos os conteúdos.

4.2.4 Terceira Introspecção

Essa introspecção refere-se a um perfil de administrador do sistema, por isso algumas necessidades são sobre a aplicação *Web*, e outras sobre a aplicação *mobile*. Para diferenciar a aplicação que cada requisito se refere, ela será exposta no início de cada item entre parênteses.

Contexto: Um professor de uma universidade, que irá ministrar uma matéria de programação para alunos de diferentes cursos da Universidade e utiliza o *Moodle* como principal forma de compartilhamento do conteúdo entre os alunos, mas percebe um desinteresse generalizado, e uma dificuldade crescente a cada novo assunto ministrado.

Necessidades do Contexto:

- (WEB) Acessar o site através de um *login*;
- (WEB) Criar, editar e excluir turmas;
- (WEB) Cadastrar vários alunos em uma turma específica;
- (WEB) Importar informações da turma de uma planilha;
- (WEB) O usuário deve receber uma senha provisória quando for cadastrado pelo professor;
- (MOBILE) O professor deve saber se os alunos já tiveram algum contato com programação antes;
- (MOBILE) O aluno cadastrado pelo administrador deve conseguir editar seus dados pessoais e senha no primeiro acesso;
- (WEB) Visualizar informações sobre o desempenho dos alunos;
- (WEB) Gerenciar módulos (cadastrar e editar);

- (WEB) Gerenciar capítulos (cadastrar e editar);
- (WEB) Gerenciar questões (cadastrar e editar), e
- (WEB) Cadastrar outros professores e/ou monitores.

4.2.5 Entrevista de Questionário

Visando validar requisitos expostos pela introspecção e pela análise de aplicações similares, foi feito um questionário, no formato de um formulário com questões objetivas e direcionadas. Nas Figuras 4 e 5, podem ser vistos o fluxo de questões do usuário e suas alternativas de resposta.

Em alguns momentos, o fluxo do usuário dependerá de qual foi a resposta em uma pergunta anterior. As alternativas de fluxo podem ser vistas acompanhando as setas que conectam perguntas e respostas no diagrama.

Para entender melhor o diagrama, deve ser considerado o significado de cada um dos seus elementos:

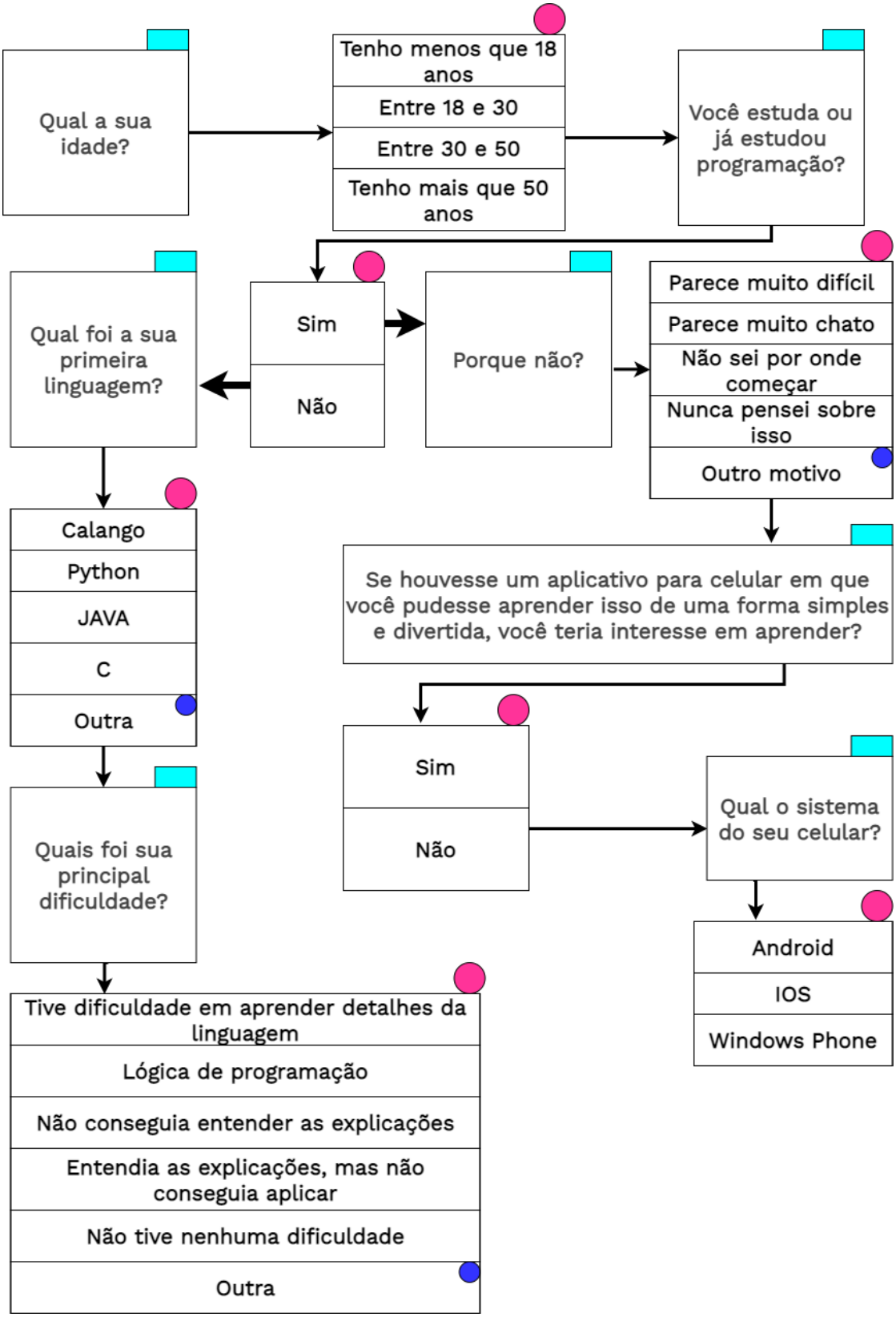
- Círculo rosa: resposta fechada, onde a questão permite apenas uma opção;
- Retângulo rosa: resposta fechada de uma questão, onde o usuário pode escolher mais de uma opção;
- Círculo azul: opção de texto aberto;
- Retângulo azul ciano: questões, e
- Retângulo amarelo: texto complementar para transmitir uma informação importante para aquela questão.

Pelo aplicativo proposto possuir muitas aplicações similares e amplamente disseminadas, todos os requisitos funcionais que poderiam ser obtidos pelo questionário já haviam sido elicitados nas etapas anteriores. Porém, houve uma coleta relevante de *feedbacks* com relação às expectativas dos usuários sobre o aplicativo. A seguir, podem ser vistas algumas informações relevantes, retiradas das respostas desse formulário:

- A Figura 6 mostra que as dificuldades mais comuns entre as 37 pessoas que disseram já ter estudado programação foram aprender detalhes da linguagem (51,4%) e lógica de programação (40,5%);
- Das 27 (vinte e sete) pessoas que disseram nunca ter estudado programação, 25 (vinte e cinco) possuem interesse em utilizar a aplicação (Figura 7);
- Das 64 (sessenta e quatro) pessoas que responderam, apenas 10 (dez) possuem um celular com sistema operacional diferente de *Android* (Figura 8);
- As funcionalidades vistas como sendo mais úteis para os usuários foram (Figura 9):
  - uma jornada com fases, em que o usuário pudesse avançar gradualmente no aprendizado (67,7%);
  - uma sessão que mostre o desempenho do usuário em cada tópico já estudado (58,1%);
  - o uso de uma linguagem simples e em português no ensino de lógica de programação (40,3%), e
  - a possibilidade de fazer questões avulsas sobre qualquer assunto já estudado (37,1%).
- As vantagens que causaram mais interesse nos usuários foram (Figura 10):
  - a possibilidade de utilizá-lo em qualquer lugar (48,4%);
  - a possibilidade de utilizá-lo durante pequenos intervalos no dia a dia (53,2%), e
  - um formato de aprendizado menos "chato"ou "entediante"(48,4%).
- 50 (cinquenta) pessoas compartilharam seus *e-mails*, e se dispuseram a ser usuários de teste da aplicação quando a mesma estiver nessa fase de desenvolvimento (Figura 11).

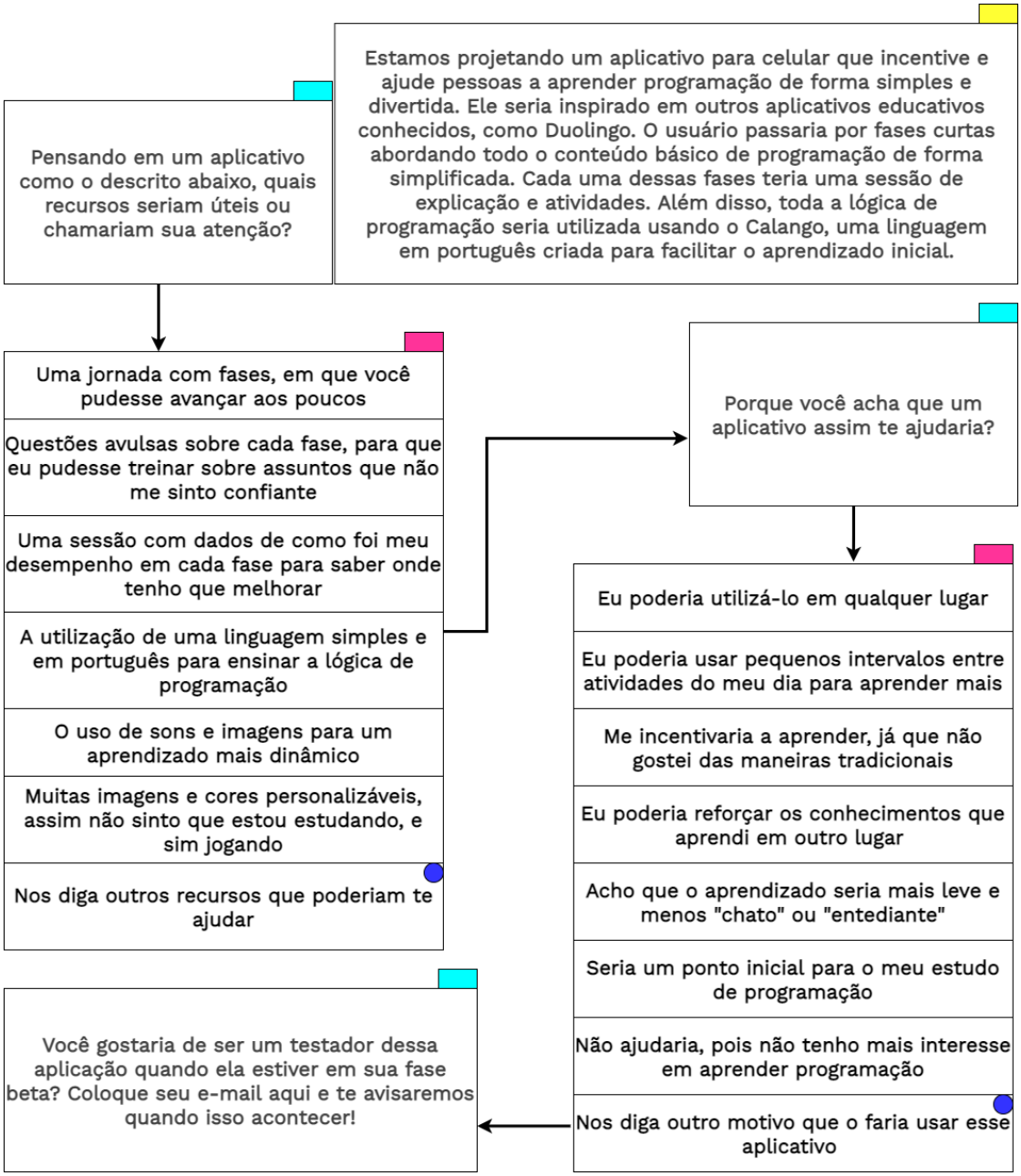
Os gráficos de resumo dessas e de outras repostas podem ser vistos no Apêndice B.2 desse documento.

Figura 4 – Diagrama do Questionário de Elicitação de Requisitos - Parte 1



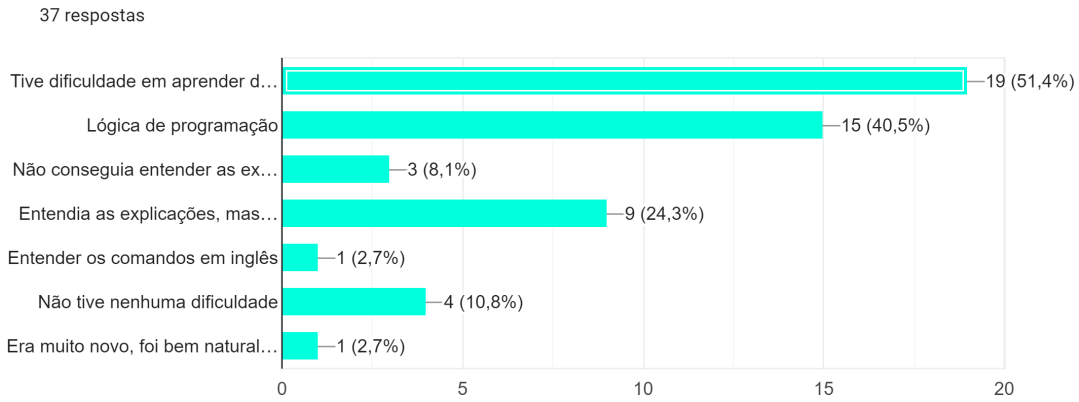
Fonte: a autora

Figura 5 – Diagrama do Questionário de Elicitação de Requisitos - Parte 2



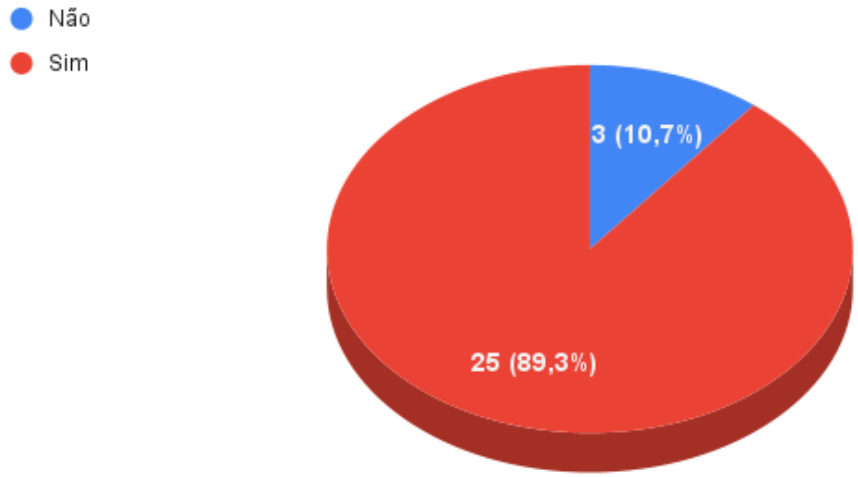
Fonte: a autora

Figura 6 – Resultado da Pergunta 2.2.2: Principais Dificuldades



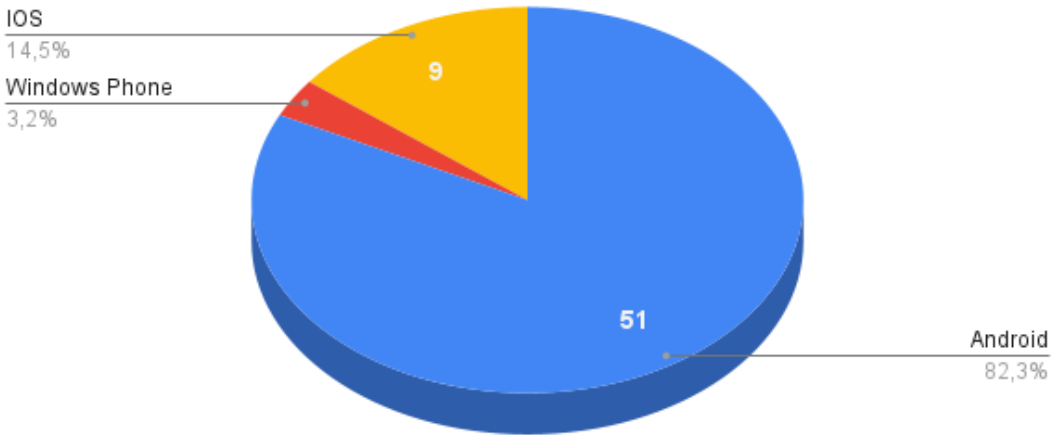
Fonte: a autora

Figura 7 – Resultado da Pergunta 2.1.2: Interessados no Aplicativo



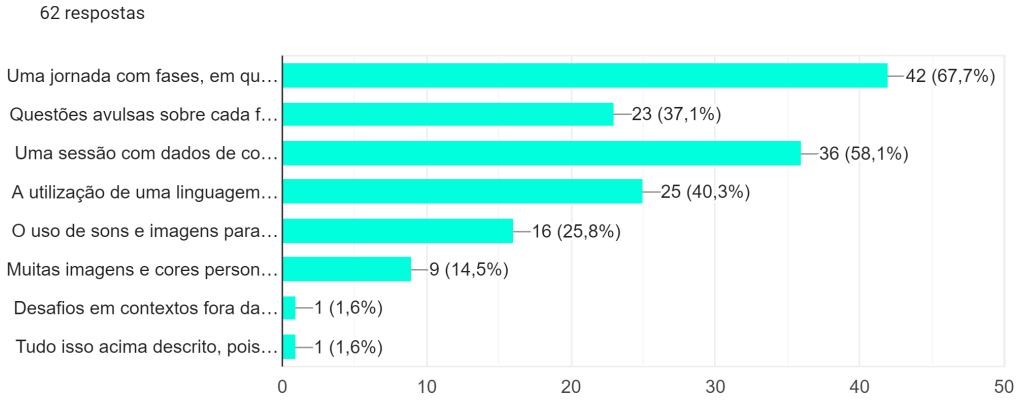
Fonte: a autora

Figura 8 – Resultado da Pergunta 3: Sistema Operacional



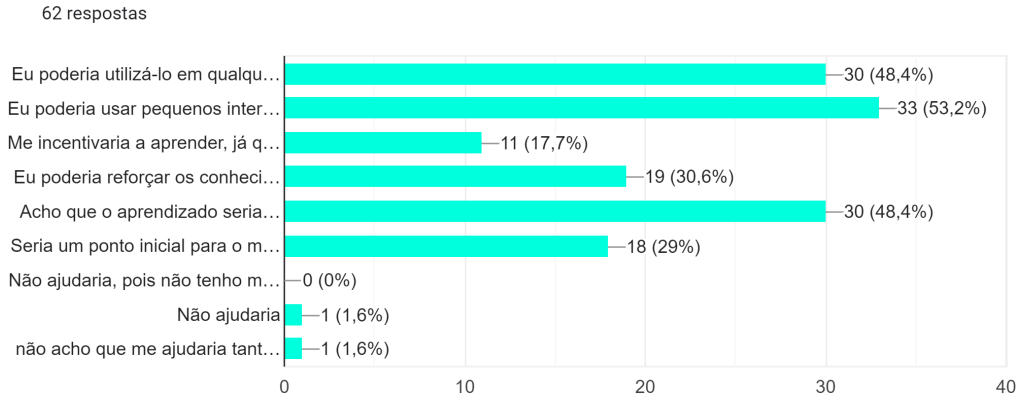
Fonte: a autora

Figura 9 – Resultado da Pergunta 4: Recursos de Interesse



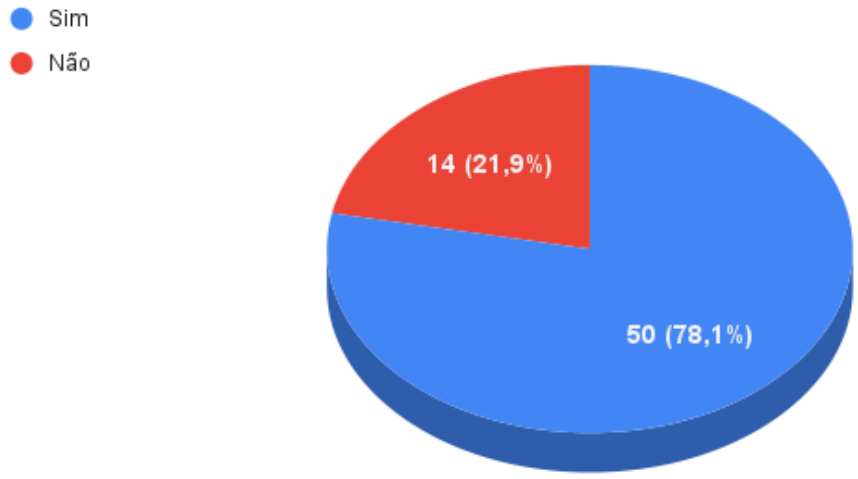
Fonte: a autora

Figura 10 – Resultado da Pergunta 5: Vantagens do Aplicativo



Fonte: a autora

Figura 11 – Resultado: Interessados em Testar o Aplicativo em Fase Beta



Fonte: a autora

4.2.6 Requisitos Funcionais Elicitados

Na Tabela 6, podemos ver todos os requisitos funcionais elicitados pelas técnicas citadas anteriormente. Para facilitar a identificação e a rastreabilidade, o código de cada requisito segue o padrão a seguir:

- Requisitos da aplicação *mobile* começam com "MB", e
- Requisitos da aplicação *web* começam com "WEB".

Com os itens já ordenados por prioridade, os critérios usados para a priorização dos requisitos foram:

- a relevância de cada item no lançamento de uma versão, e
- se ele é ou não um pré-requisito para uma funcionalidade considerada relevante no critério anterior.

Tabela 6 – Requisitos Funcionais

CÓDIGO	**DESCRIÇÃO**
MB16	Fazer cadastro pelo aplicativo
MB17	Fazer *login* pelo aplicativo com os meus dados cadastrados
MB11	Uma área que mostre os módulos e capítulos que o usuário irá percorrer
MB5	O usuário possuir um nível que indique em qual etapa ele está
MB3	Possuir uma sessão de explicação do conteúdo antes das questões
MB9	Cadastrar conteúdos com imagens entre os textos
MB12	Fazer pesquisas de satisfação com o usuário periodicamente
MB19	Editar dados do perfil
MB13	Mostrar perfil cadastrado
MB14	Fornecer aplicativo para o usuário baixar e instalar em seu celular
MB4	Possuir diferentes tipos de questões

MB23	Recuperar senha utilizando o e-mail cadastrado
MB6	Possuir questões avulsas que englobam todo o conteúdo dos capítulos
MB10	Possuir uma área que o usuário consiga ver seu desempenho em cada sessão e nível
WEB11	O administrador do sistema deve conseguir acessar o site através de um e-mail e uma senha
WEB18	O administrador deve conseguir cadastrar uma turma
WEB5	O administrador deve conseguir cadastrar novos alunos apenas com o e-mail e vinculá-los a uma turma
WEB20	O administrador deve visualizar as turmas e seus respectivos alunos
WEB13	Administrador deve visualizar a lista de módulos
WEB14	Administrador deve ter acesso à lista dos capítulos de cada módulo
WEB15	Administrador deve ter acesso à lista de questões de cada capítulo, seu título e texto de descrição
WEB16	Administrador deve ter acesso à lista de todas as questões
MB18	Excluir sua própria conta de usuário
WEB10	O administrador deve conseguir cadastrar outro administrador
MB15	Completar perfil ao fazer *login* pela primeira vez (quando cadastrado pelo site)
WEB1	O administrador deve conseguir cadastrar módulos
WEB2	O administrador deve conseguir editar módulos
WEB3	O administrador deve conseguir cadastrar capítulos
WEB4	O administrador deve conseguir editar capítulos
WEB6	O administrador deve conseguir cadastrar questões
WEB7	O administrador deve conseguir editar questões
MB22	As questões devem possuir exemplos práticos para que o usuário entenda como isso poderia ser aplicado

WEB9	O administrador deve conseguir fazer *upload* de uma planilha para adicionar vários alunos de uma vez
WEB19	O administrador deve conseguir remover uma turma sem excluir os usuários que estão nela

4.3 Requisitos Não Funcionais

Para elicitação dos requisitos não funcionais (RNFs), foram utilizadas as estruturas básicas do *Framework NFR*. O *framework* foi utilizado como base para um modelo mais generalista e simplificado, com apenas relações neutras, positivas, ou negativas entre os itens.

Os RNFs elicitados estão representados nas imagens pelas operacionalizações (itens com borda).

4.3.1 Usabilidade

A Figura 12 mostra a expansão de objetivos de usabilidade até a chegada dos requisitos. Os requisitos elicitados, possuem relações com experiência de usuário e itens de interface.

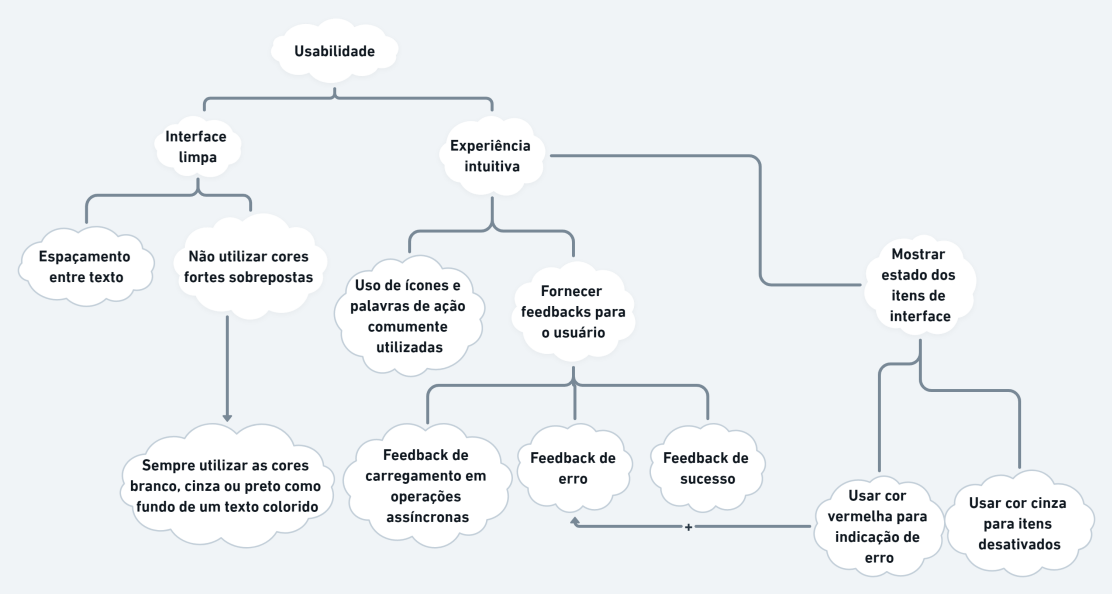
4.3.2 Manutenibilidade

A Figura 13 mostra a expansão de objetivos de manutenibilidade, sendo essa característica enquadrada como a facilidade de se fazer a manutenção em um *software*. Os requisitos elicitados possuem relações com facilidade de testes, de modificação e de leitura do código.

4.3.3 Segurança

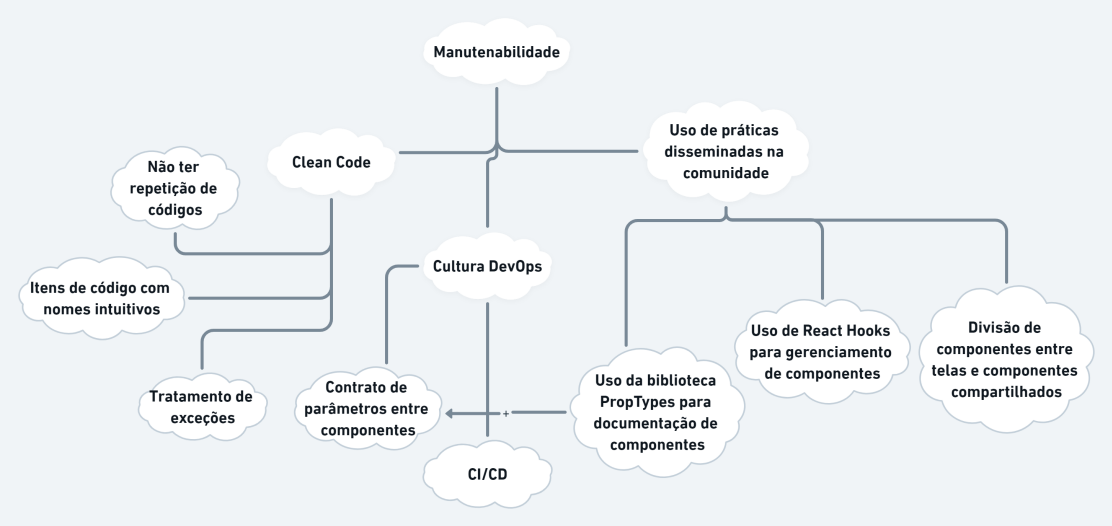
A Figura 14 mostra a expansão de objetivos de segurança. Esses requisitos possuem relação com autenticação, autorização e segurança de dados.

Figura 12 – Usabilidade



Fonte: a autora

Figura 13 – Manutenibilidade

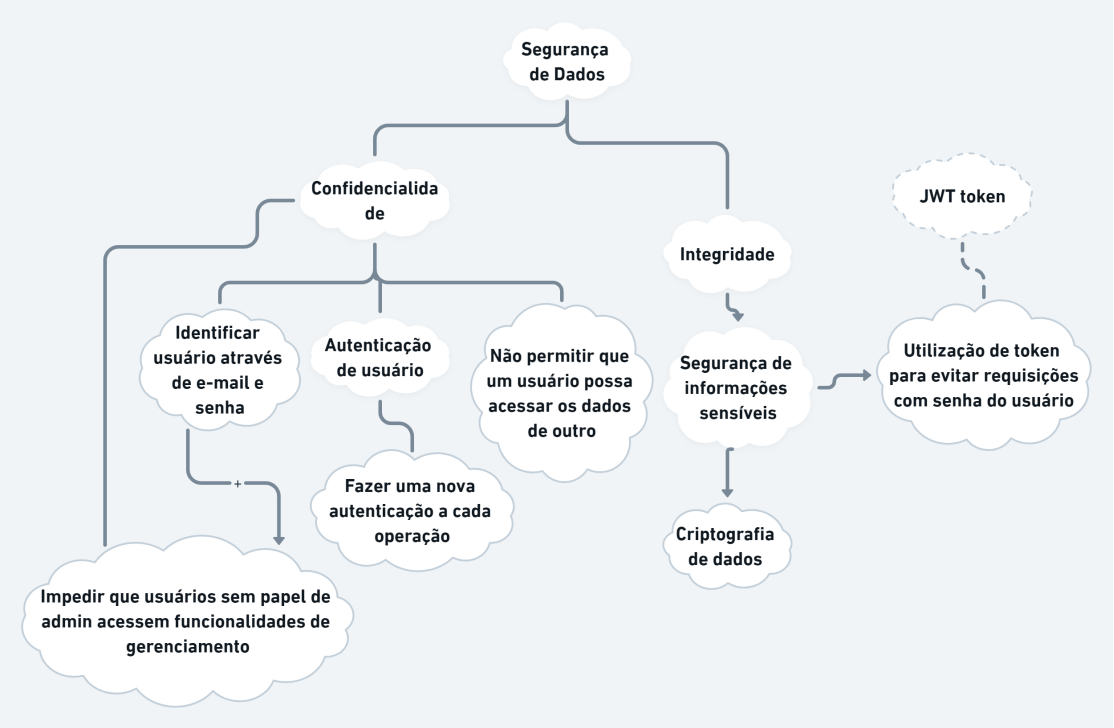


Fonte: a autora

4.3.4 Portabilidade

As Figuras 15 e 16 mostram a expansão de objetivos de portabilidade das aplicações para dispositivos móveis e navegadores *Web*.

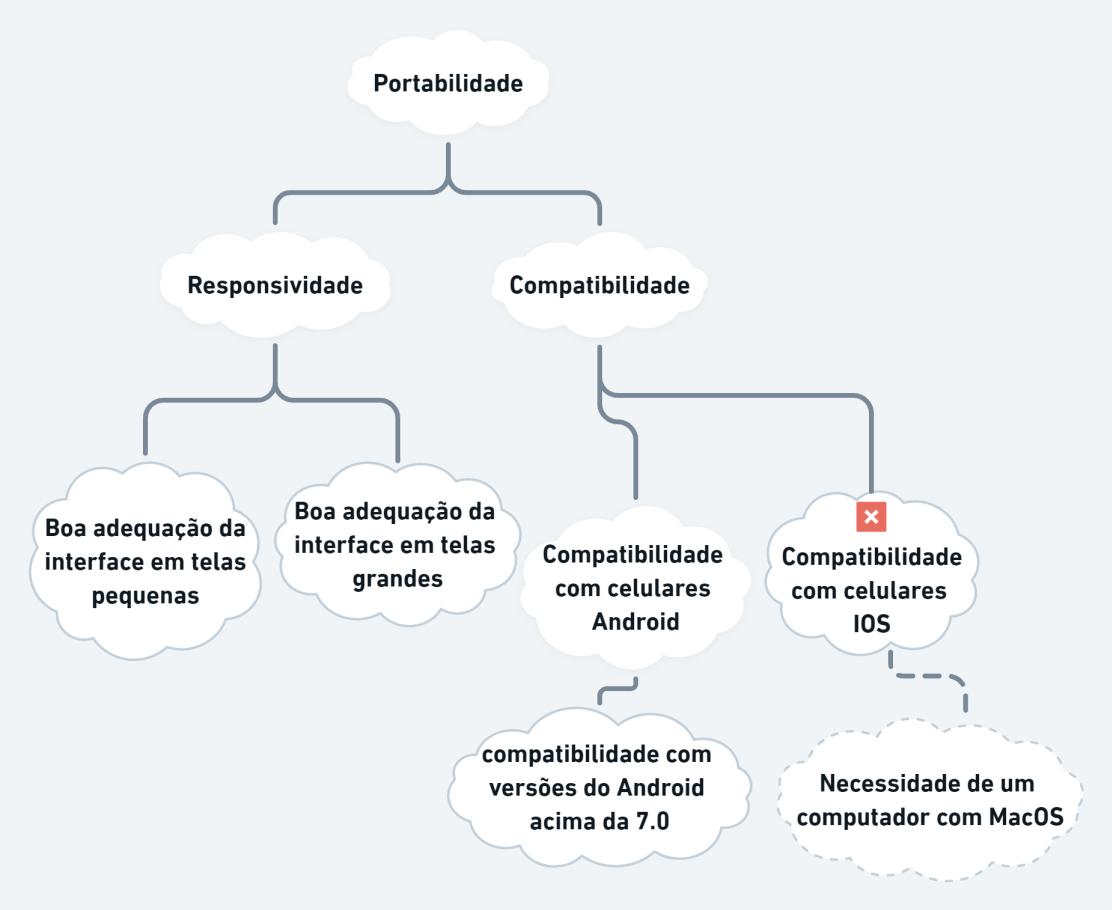
Figura 14 – Segurança



Fonte: a autora

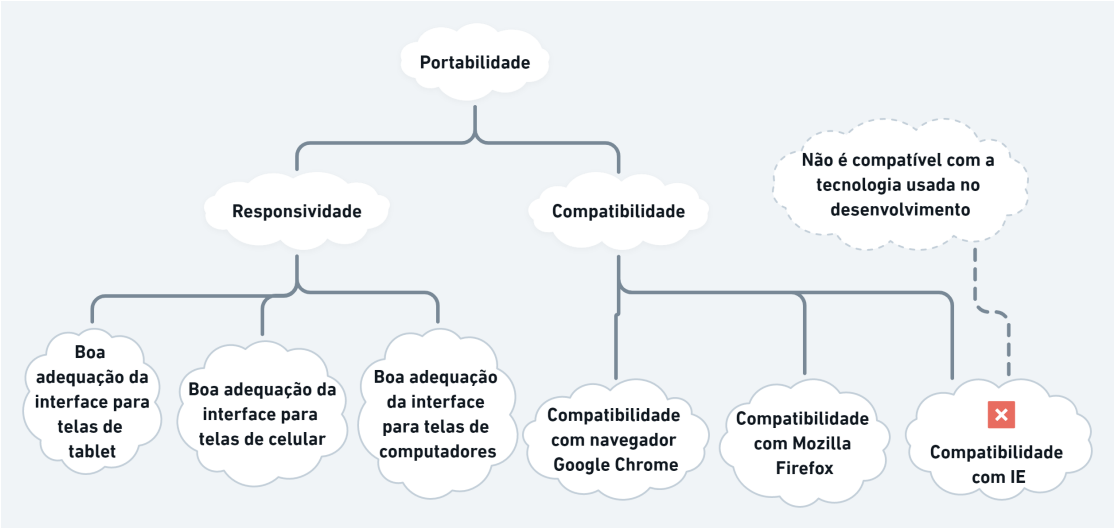
Os requisitos de portabilidade são relacionados com flexibilidade do uso da aplicação. Pode-se ver que não será possível utilizar a aplicação *mobile* em sistemas operacionais da *Apple*, sendo algo justificado pela exigência de equipamentos de desenvolvimento que não estão disponíveis no contexto desse trabalho.

Figura 15 – Portabilidade da Aplicação Mobile



Fonte: a autora

Figura 16 – Portabilidade da Aplicação Web



Fonte: a autora

5 Tecnologias e Arquitetura

Nesse capítulo, são apresentadas as tecnologias que possuem relação com o desenvolvimento, manutenção e o *deploy* da aplicação.

5.1 Ferramentas de Apoio ao Desenvolvimento

Na Tabela 7, são mostradas as ferramentas usadas como apoio para desenvolvimento e gerenciamento de artefatos da aplicação.

Tabela 7 – Ferramentas de Apoio

Tecnologia	**Descrição**	**Versão**
GIT	Ferramenta de controle de versão utilizada para gerenciar artefatos de arquivo e suas alterações.	2.33.1
VSCode	Editor de código com extensões úteis para o desenvolvimento com JavaScript e React.	1.64.2
Postman	Ferramemta que permite documentar e executar requisições HTTP (*Hypertext Transfer Protocol*)	8.12.4

5.2 Aplicação Móvel

5.2.1 Tecnologias

A Tabela 8 descreve as principais tecnologias utilizadas no desenvolvimento da aplicação *mobile*, uma breve descrição, e a versão utilizada.

Tabela 8 – Tecnologias e Ferramentas para Desenvolvimento da Aplicação *Mobile*

Tecnologia	**Descrição**	**Versão**
React	Biblioteca em JavaScript utilizada para desenvolver aplicações em componentes	16.13.1
ReactNative	Framework de desenvolvimento *mobile* híbrido (IOS e Android)	0.63
Expo	Plataforma *open source* que facilita a construção do *software*, oferecendo várias ferramentas para diferentes momentos do desenvolvimento	42.0.0
Axios	Cliente HTTP utilizado para fazer requisições à API	0.21.1
Yup	Biblioteca de validação de dados. Foi usada para validação de regras dos formulários da aplicação	0.32.11
Formik	Biblioteca de gerenciamento de formulários que possui integração com a MUI	2.2.9

Uma tecnologia de extrema importância para o desenvolvimento do aplicativo móvel foi o *Expo*.

O *Expo* é um *framework* para aplicações *React*, que foi usado em todas as etapas do desenvolvimento móvel. A seguir estão listadas as vantagens que o *Expo* fornece, e que foram exploradas no desenvolvimento do aplicativo:

- Permite o teste local da aplicação sem a necessidade de conexão por cabo;
- Gera um QRCode global que pode ser usado para obter acesso ao ambiente de homologação;
- Gera o executável com extensão ".apk"para instalação nos celulares Android (usada para o ambiente de produção), e
- Facilita a instalação de bibliotecas nativas, dispensando o *link* manual.

A criação de uma aplicação com *Expo* também altera a forma padrão da arquitetura de pastas de uma aplicação *React*. A alteração pertinente ao nosso aplicativo é a remoção das pastas "android"e "ios"de dentro do projeto. Isso acontece porque o próprio *Expo* gera os projetos nativos das plataformas, diferente de

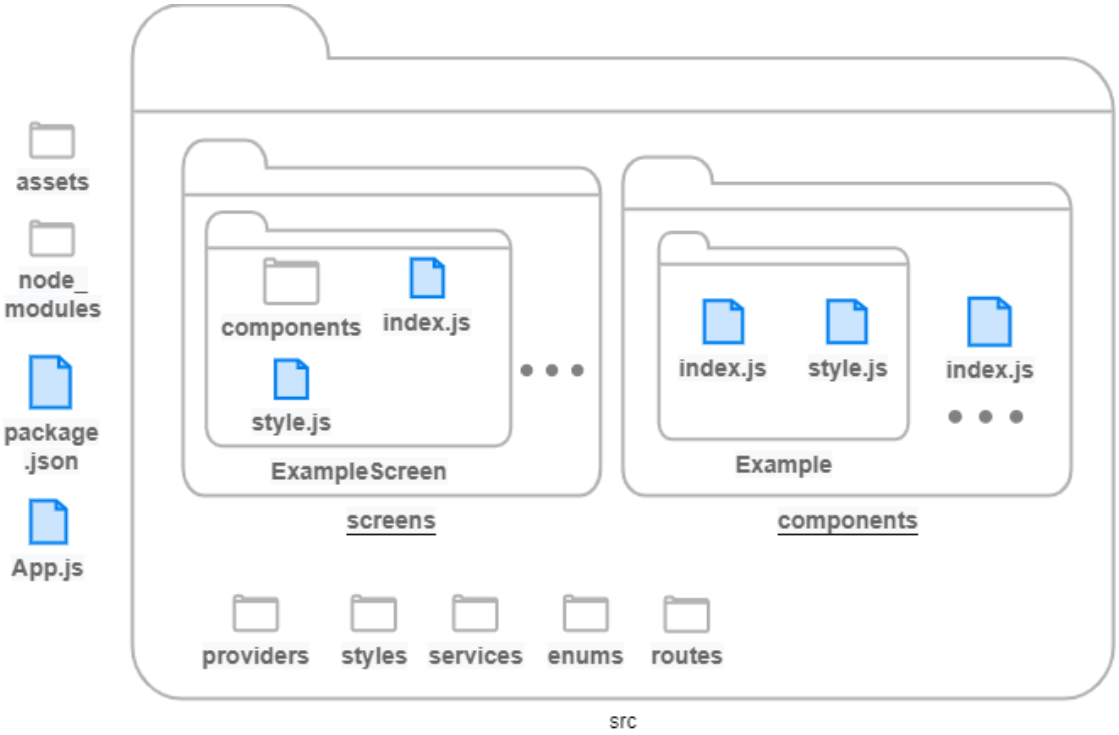
uma aplicação *React* comum, em que os arquivos dessas pastas são atualizados a cada alteração no código em JavaScript.

5.2.2 Diagrama de Pacotes

A seguir será descrito o que cada pasta e arquivo relevante para o projeto representa. Na Figura 17, pode-se visualizar a organização das pastas do projeto de desenvolvimento da aplicação *mobile*.

- *assets*: pasta que armazena arquivos estáticos, como imagens e ícones;
- *node-modules*: diretório das dependências do projeto;
- *package.json*: lista as dependências que serão baixadas no node-modules e configurações do projeto;
- App.js: arquivo inicial do projeto, no qual estará o que deve ser carregado na aplicação, além de ser o local mais propício para incluir *providers* globais;
- *screens*: pasta de componentes que representam telas;
- *components*: pasta de componentes compartilhados, que podem ser reutilizados;
- *providers*: serviços que incluem estruturas globais da aplicação, como contexto de estado e tema;
- *styles*: arquivos que exportam qualquer configuração de estilo;
- *services*: essa pasta possui os arquivos responsáveis por consumir os recursos do servidor;
- *routes*: arquivo de roteamento da aplicação, e
- enums: pasta com os enumeradores utilizados no projeto.

Figura 17 – Diagrama de Pacotes da Aplicação Mobile



Fonte: a autora

5.2.3 Arquitetura

A aplicação *mobile* foi desenvolvida utilizando a arquitetura em componentes. Essa arquitetura propõe um formato de desenvolvimento em "peças", em que vários blocos independentes e com responsabilidades definidas formam a aplicação. Os componentes podem ser reutilizados e transformados em parte integrante de outro componente, simplificando a reutilização de código e facilitando a manutenção (CRNKOVIC, 2001).

O projeto possui três principais categorias de componentes: telas, componentes compartilhados e componentes de telas.

As telas são responsáveis por ocupar a visão completa do dispositivo do usuário, ou pelo menos sua sessão principal (parte dinâmica do aplicativo). Elas são encontradas no pacote *src/screens*. Os componentes compartilhados, como o próprio nome já informa, são componentes utilizados em mais de 1 (um) com-

ponente ou tela. Em caso de necessidade de variações, eles recebem parâmetros enviados pelos componentes pais, informando qual a variação deve ser utilizada em cada caso. Eles estão agrupados na pasta *src/components*.

Os componentes de tela ficam alocados nas pastas *components* de cada tela. Eles só podem ser usados pelo componente em que estão localizados. São criados para facilitar a leitura e a manutenção do código, diminuindo a quantidade de linhas de um mesmo componente, e modularizando as responsabilidades de componentes mais complexos.

5.3 Aplicação *Web*

5.3.1 Tecnologias

A Tabela 9 descreve as principais tecnologias utilizadas no desenvolvimento da aplicação *WEB*, uma breve descrição e a versão utilizada.

Tabela 9 – Tecnologias e Ferramentas para Desenvolvimento da aplicação *WEB*

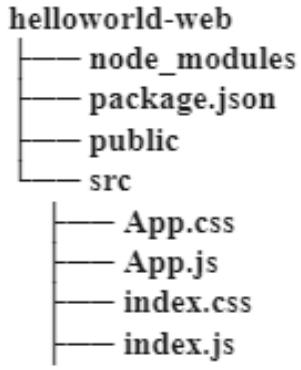
Tecnologia	**Descrição**	**Versão**
React	Biblioteca em JavaScript utilizada para desenvolver aplicações em componentes	17.0.2
Axios	Cliente HTTP utilizado para fazer requisições à API	0.24.0
MUI	Uma biblioteca de componentes que permite o desenvolvimento de um *Design System* para aplicações React	5.1.0
Yup	Biblioteca de validação de dados, usada para validação de regras dos formulários da aplicação	0.32.11
Formik	Biblioteca de gerenciamento de formulários que possui integração com a MUI	2.2.9

5.3.2 Diagrama de Pacotes

A estrutura inicial da aplicação foi gerada pela biblioteca *Create React App* (CRA), uma biblioteca que proporciona a geração de uma aplicação inicial em

ReactJS com aspectos de desenvolvimento, como testes, pacotes *NPM* e *scripts* já configurados. A estrutura inicial, gerada pela CRA, está descrita na Figura 18.

Figura 18 – Estrutura Inicial da Aplicação Web

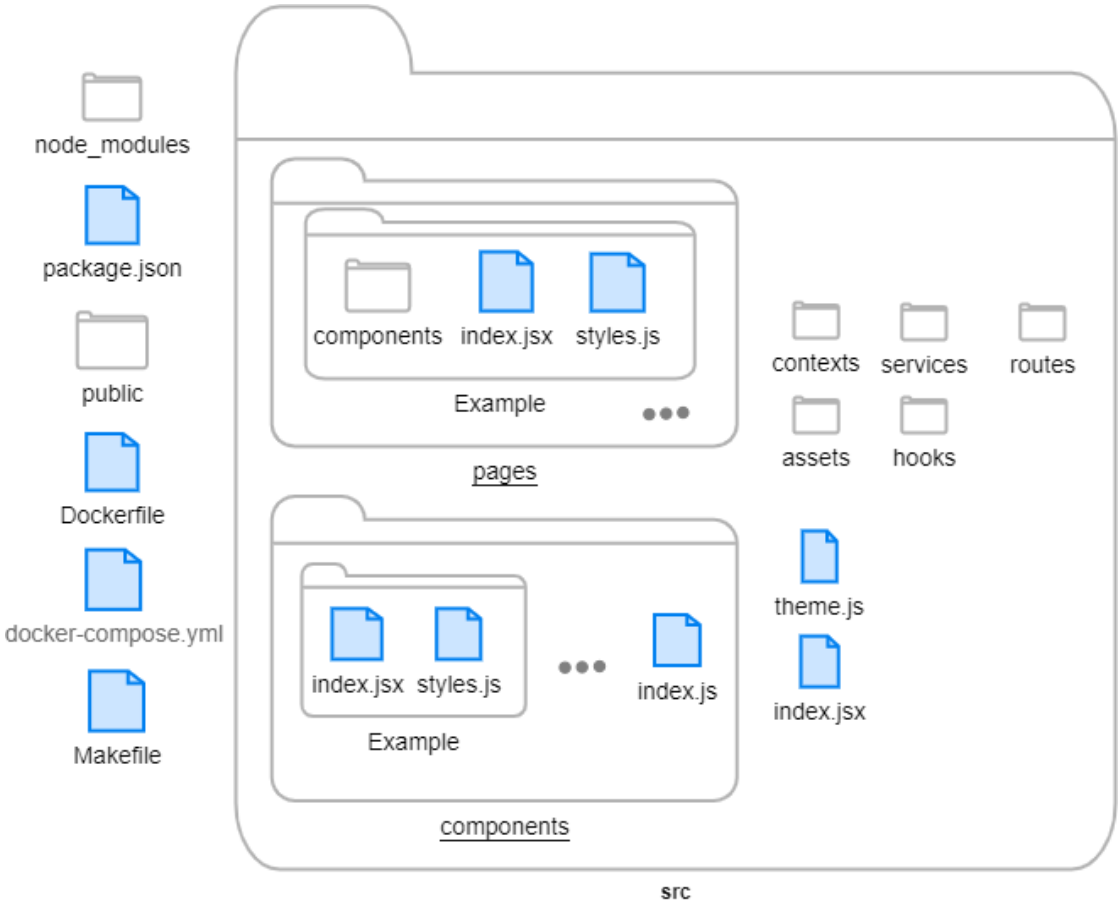


Fonte: a autora

Para uma estrutura modularizada, algumas alterações e adições na árvore de arquivos e pastas foram realizadas. A estrutura final da aplicação pode ser vista na Figura 19.

- *node-modules*: diretório das dependências do projeto;
- *package.json*: lista as dependências que serão baixadas no node-modules e possui algumas configurações do projeto;
- *public*: diretório de arquivos que podem ter uma visibilidade maior, como imagens públicas;
- *Dockerfile* e *docker-compose*: responsáveis pela conteinerização da aplicação em desenvolvimento;
- *Makefile*: usado para mapear e facilitar a execução de comandos muito utilizados, atribuindo-lhes um atalho acessível pelo terminal através do comando "make <nome-do-atalho>";
- *pages*: diretório de todos os componentes que representam páginas;
- *components*: pasta de componentes compartilhados;

Figura 19 – Estrutura Definitiva da Aplicação Web



- *contexts*: diretório de contextos, gerenciadores de estado da aplicação;
- *services*: pasta que armazena os arquivos responsáveis por consumir os recursos do servidor, e
- *routes*: diretório de arquivos de roteamento da aplicação.

5.3.3 Arquitetura

A arquitetura da aplicação *WEB* é muito similar à utilizada na aplicação *mobile*, possuindo como base a arquitetura em componentes.

O projeto possui três principais categorias de componentes: páginas, componentes compartilhados e componentes exclusivos de uma página.

As páginas são componentes que ocupam toda a janela do navegador do usuário, ou pelo menos o espaço abaixo da *navbar* (área logada). Elas estão localizadas no diretório "*src/pages*", vide Figura 19.

Os componentes compartilhados, que podem ser reutilizados em várias áreas da aplicação, estão agrupados na pasta *src/components*.

Os componentes específicos de uma página, estão alocados nas pastas *components*, dentro do diretório de cada página.

5.4 API

5.4.1 Tecnologias

A Tabela 10 descreve as principais tecnologias utilizadas no desenvolvimento da *API*, além de uma breve descrição e as versões utilizadas.

Tabela 10 – Tecnologias e Ferramentas para Desenvolvimento da *API*

Tecnologia	**Descrição**	**Versão**
NodeJS	Software multiplataforma que permite a execução de JavaScript fora do navegador	14.17.3
Express	Framework que oferece recursos para criação de serviços web	4.17.1
Sequelize	Um OMR (*Object-Relational Mapper*) para NodeJS que pode ser utilizado para PostgreSQL e outros bancos. Faz o mapeamento de dados relacionais de banco de dados para JavaScript	6.6.5

5.4.2 Aplicação do Padrão de Projeto

Para uma melhor adequação de um padrão arquitetural aplicado ao caso de uso de um servidor, o padrão arquitetural da API é uma adaptação de alguns conceitos do MVC.

Como principal ajuste, a camada de visualização torna-se inexistente, considerando que não há interface retornada da API.

Aplicando o conceito a um servidor, um controlador pode ser um item de manipulação de um modelo ou possuir a responsabilidade de receber uma requisição, realizar um tratamento e retornar uma resposta para o cliente no formato *JavaScript Object Notation* (JSON).

Quanto à camada de modelo, não só representa uma entidade no contexto da aplicação, como é responsável pela representação de uma tabela da base de dados.

5.4.3 Diagrama de Pacotes

A Figura 20 mostra a organização de pastas do projeto. A seguir, serão explanados os objetivos dos pacotes e arquivos que estão presentes na imagem:

- *controllers*: composto pelos arquivos de controle, já citados anteriormente;
- *models*: possui as classes já relacionadas com suas respectivas tabelas no banco de dados através do Sequelize;
- *middlewares*: são responsáveis por interceptar as requisições, e atuam entre as rotas e os controladores;
- *config*: arquivos de configuração;
- *migrations*: pacote utilizado para agrupar arquivos relacionados às criações e alterações no banco de dados. Muito utilizado para versionamento do banco de dados, quando ele deve ser mantido em mais de uma máquina;
- *routes.js*: possui o mapeamento das *urls*, apontando cada uma para o seu respectivo controlador e aplicando os *middlewares*;
- *server.js*: inicializa a aplicação de acordo com a porta definida na variável de ambiente;
- *app.js*: responsável por aplicar os *middlewares* e rotas à aplicação;

- *node-modules*: pacote de armazenamento das bibliotecas de dependência;

- *Dockerfile* e *docker-compose*: responsáveis pela conteinerização da API e do banco de dados, assim como o compartilhamento da rede entre eles;

- *Makefile*: usado para mapear e facilitar a execução de comandos muito utilizados, atribuindo-lhes um atalho acessível pelo terminal, através do comando "make <nome-do-atalho>", e

- *package.json*: mapeia dependências e configuração do projeto.

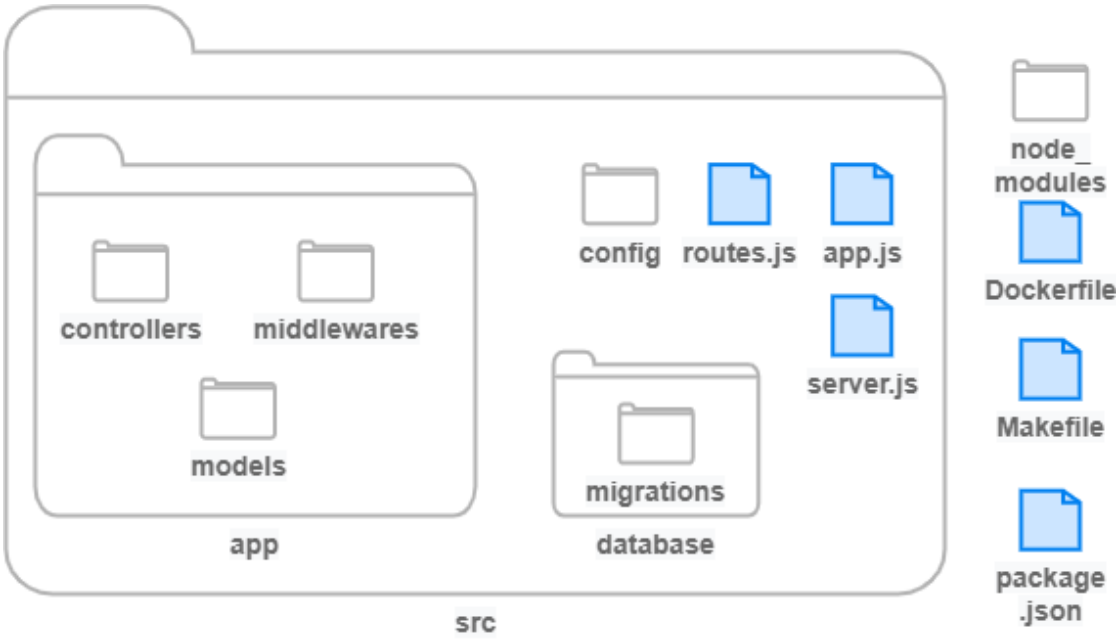
5.4.4 Fluxo de Dados

O fluxo de uma requisição para o servidor inicia-se nas rotas. Elas mapeiam que função de qual *controller* deve ser chamada, quando determinado *endpoint* for utilizado pelo cliente. Além disso, é responsável por indicar quais *middlewares* devem ser aplicados a cada rota, sendo que uma rota pode acumular mais de um *middleware*, e um mesmo *middleware* pode ser aplicado a várias rotas.

Middlewares são funções que interceptam o meio do caminho entre as rotas e os controladores. Eles possuem a atribuição de verificar a presença de cabeçalhos de autenticação, e adicionar informações utilizadas no contexto da requisição.

O controlador apontado por uma rota irá utilizar funções que acessam o banco de dados, fornecidas pelo Sequelize através das *models*. Após o banco de dados recuperar as informações requisitadas, elas são retornadas para o controlador que fez o pedido, tratadas para o formato esperado pelo cliente e, caso tudo tenha sido executado com sucesso, enviadas como resposta da requisição. Caso ocorra um erro em qualquer uma das etapas, esse erro é capturado e enviado com uma mensagem amigável para o cliente.

Figura 20 – Diagrama de Pacotes da API



Fonte: a autora

5.5 Base de Dados

5.5.1 Tecnologias

A Tabela 11 descreve as principais tecnologias utilizadas na modelagem e implementação da base de dados.

Tabela 11 – Tecnologias e Ferramentas para Desenvolvimento da *API*

Tecnologia	**Descrição**	**Versão**
PostgreSQL	Gerenciador de Sistema de Banco de Dados Relacional de código aberto	11.2
DBeaver	Aplicativo de cliente SQL capaz de realizar uma conexão com vários bancos de dados (incluindo o PostgreSQL)	21.1.0.0

5.5.2 Diagrama Entidade-Relacionamento

O Diagrama Entidade-Relacionamento é uma representação gráfica do modelo de dados, que é usado no banco de dados da aplicação. Ele descreve as entidades do *software*, seus atributos, e como elas interagem entre si. O DER não é a representação final do banco de dados. Trata-se de um formato mais abstrato, sem tabelas, chaves (exceto a primária) e atributos de relacionamento (RODRIGUES, 2014).

A Figura 21 mostra quais objetos são utilizados para fazer essa representação e o que cada um deles significa.

Figura 21 – Descrição de Elementos do DER

Fonte: a autora

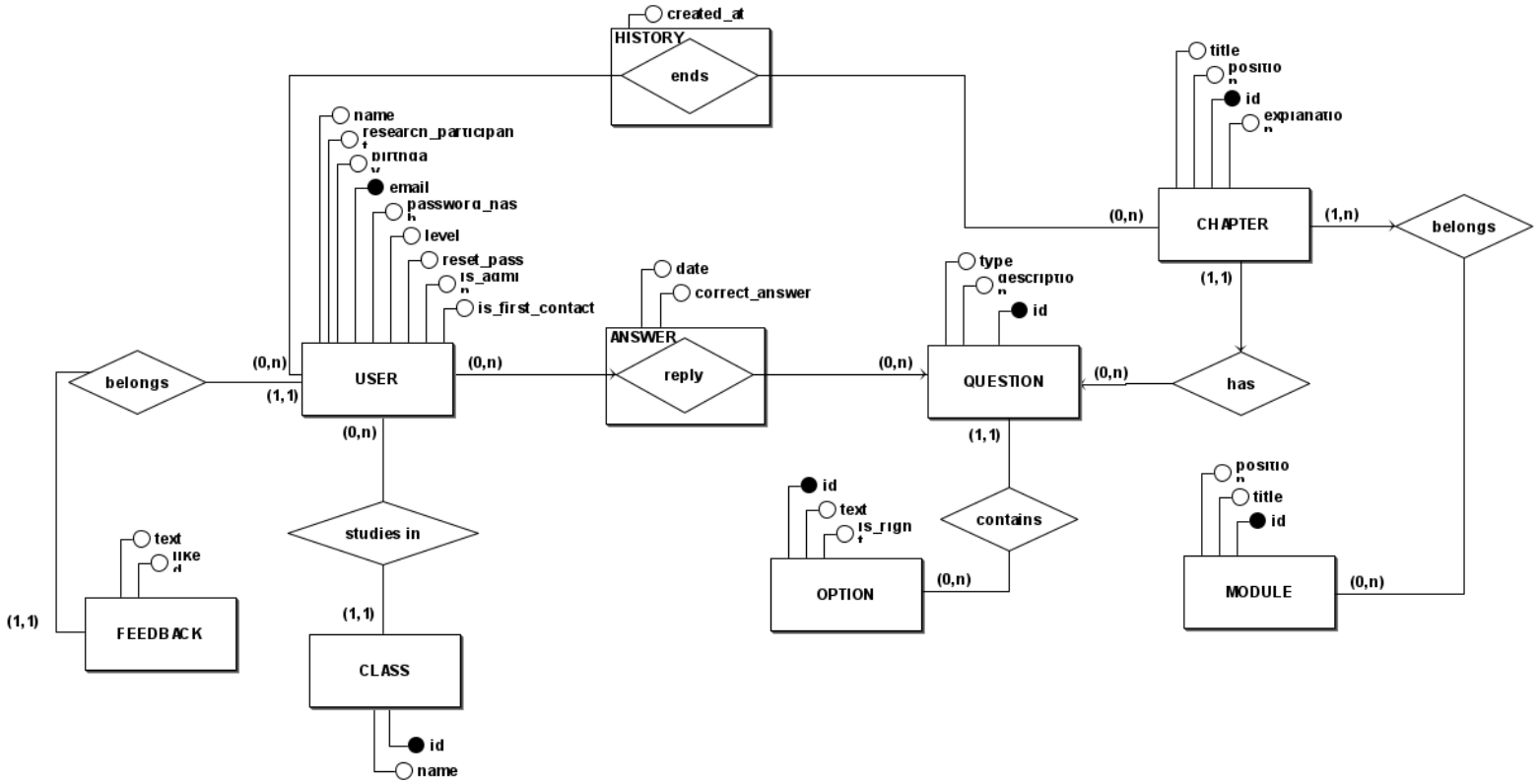
A Figura 22 mostra como o banco de dados desse sistema foi projetado para atender todos os requisitos citados anteriormente.

5.5.3 Diagrama Lógico

O diagrama lógico é um diagrama de mais baixo nível quando comparado com o DER. Ele representa exatamente o que será implementado no banco de dados, incluindo os tipos de dados, as chaves primárias, as chaves estrangeiras, os atributos e as tabelas de relacionamento.

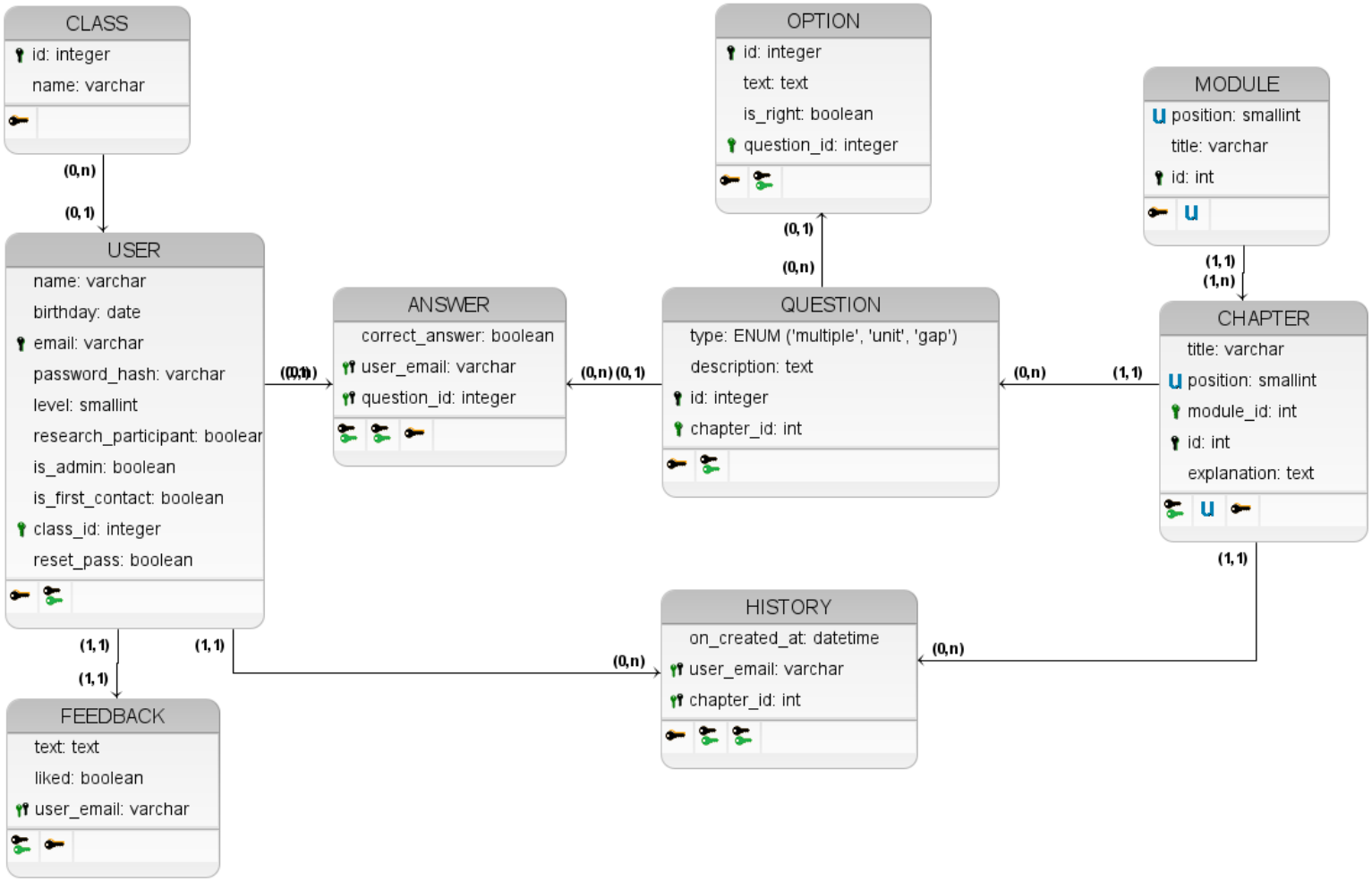
Na Figura 23, pode-se ver de forma mais clara como esse sistema será mantido na base de dados.

Figura 22 – Diagrama Entidade-Relacionamento



Fonte: a autora

Figura 23 – Diagrama Lógico do Banco de Dados



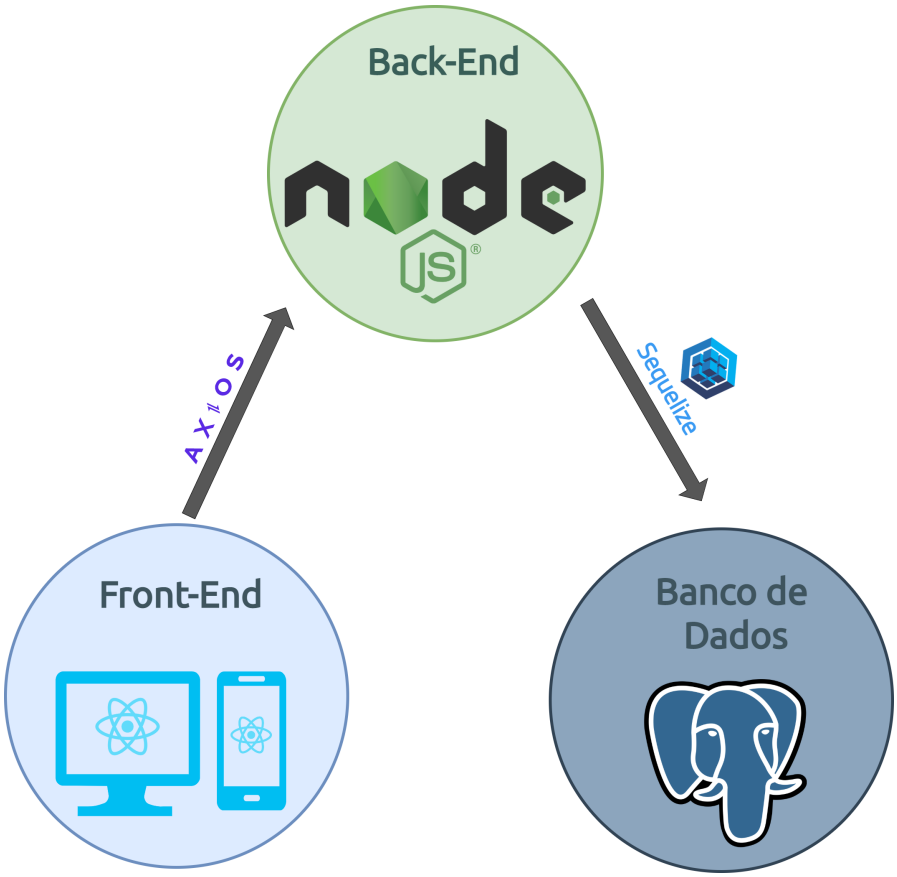
Fonte: a autora

5.6 Integração dos Sistemas

Agora que já foram citados todos os módulos do sistema, será apresentado como eles interagem entre si, e quais as ferramentas foram utilizadas para isso.

Na Figura 24, pode-se notar os três módulos do sistema: API, cliente (Aplicação WEB/Mobile) e Banco de Dados.

Figura 24 – Diagrama de Integração dos Módulos



Fonte: a autora

O fluxo do usuário inicia-se no *Front-End* da aplicação. Um *login*, uma nova tarefa concluída ou até mesmo o ato de abrir a tela principal da aplicação são fatores que disparam uma requisição para o *Back-End*. Para realizar essa ação, foi utilizada a biblioteca *Axios*.

No site oficial da *Axios*, ele é identificado como "um cliente *HTTP* baseado-

em-promessas para o node.js e para o navegador"(AXIOS, 2022). Resumindo, o Axios é responsável por fazer as requisições *HTTP* e retornar *promises*, funções assíncronas que possibilitam o carregamento do resultado em *background*, liberando o *Front-End* para continuar sua execução.

Feita a requisição, ela é enviada para o *Back-End*, que fará uma conexão com o banco de dados para retornar as informações requisitadas pelo *Front-End*. De modo a abstrair o processo de *queries* e conexão com o banco, faz-se o uso do *Sequelize*, um *Object-Relational Mapper*(ORM) para NodeJS, que também é baseado em promessa, e que possui suporte a transações, relações, e outras funcionalidades que facilitam a interação com o Banco de Dados.

Caso haja algum problema com a recuperação de dados pelo banco, ele gerará uma exceção, que será capturada pelo *Back-end*, tratada, e enviada para o cliente de maneira que o usuário tenha o *feedback* do ocorrido.

Caso todas as etapas sejam executadas com sucesso, o banco retornará os dados para o *Back-end*, que os tratará e converterá em JSON, para retorná-los ao cliente e mostrar a informação ao usuário.

5.7 *Deploy* e Integração Contínua

Nessa sessão, é mostrado o fluxo de integração e *deploy* das aplicações que compõem o sistema.

Para esteira de integração contínua, foi utilizado o *GitHub Actions*, uma ferramenta de *Continuous Integration* (CI) *open source* e gratuita, que possui integração com o GitHub.

Os arquivos da *pipeline* de integração podem ser encontrados no Apêndice D desse documento e no repositório do GitHub.

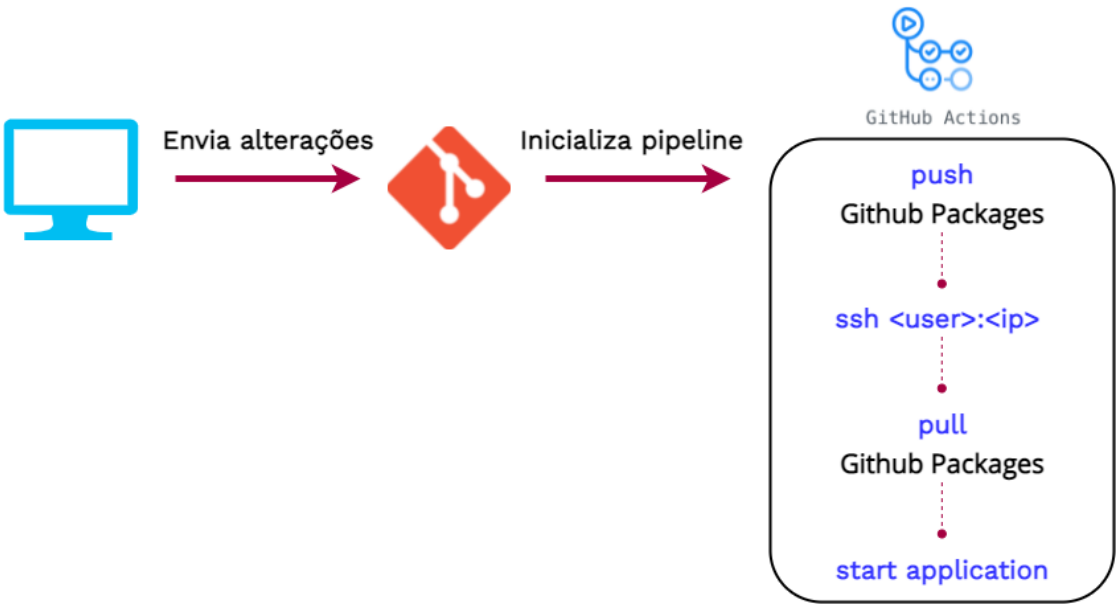
5.7.1 Aplicação WEB e API

As aplicações WEB (API e Aplicação de Gerenciamento) possuem o mesmo fluxo de integração e *deploy*, sendo que há apenas a esteira de produção. O motivo

disso, é a falta da necessidade de haver um ambiente separado (conhecido como ambiente de homologação) para sistemas que podem ser testados localmente com condições semelhantes ao ambiente de produção.

Conforme pode ser visto na Figura 25, o fluxo inicia-se ao subir uma nova alteração para a *branch* "master", inicializando a esteira de integração. A partir disso, ocorre a atualização do contêiner no *GitHub Packages*; faz-se o acesso à máquina que hospeda a aplicação a partir do seu IP; baixa-se o contêiner com as novas alterações, e a aplicação no servidor é reiniciada. Se tudo ocorre bem, as alterações que estavam no último *push* passam pela esteira e já podem ser utilizadas no ambiente.

Figura 25 – Diagrama da *Pipeline* de CI das Aplicações WEB



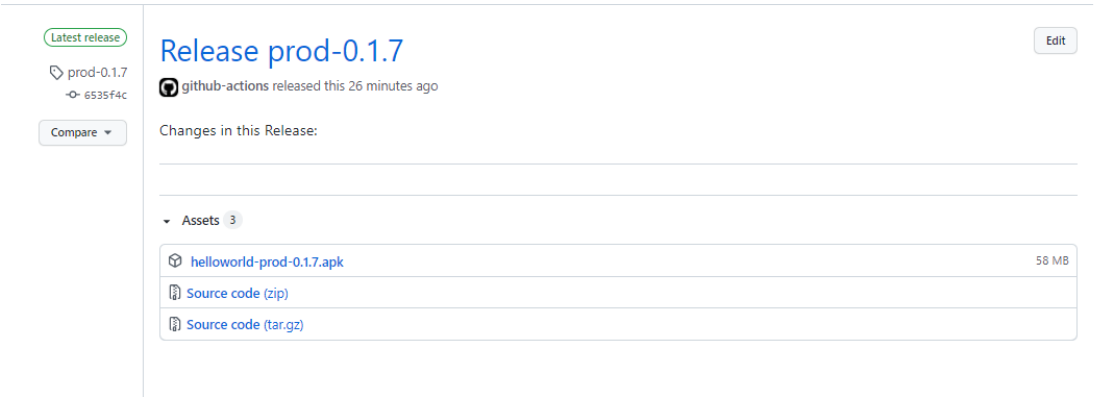
Fonte: a autora

5.7.2 Aplicação *Mobile*

A aplicação móvel possui dois fluxos de integração, um para homologação e testes e outro para produção (Figura D). O fluxo de homologação utiliza o Expo para inicializar a aplicação, e se resume a publicar uma versão no Expo, que pode ser acessada pelo celular utilizando o aplicativo e escaneando o QRCode gerado.

Já o fluxo de publicação em produção é inicializado a partir do *push* de uma *tag* no repositório do *Git* com o formato "(prod-<número>.<número>.<número>)". Ao final da *pipeline*, é gerada uma *release* no repositório com o modelo mostrado na Figura 26.

Figura 26 – *Release* Gerada pela *Pipeline* de Integração Contínua



Fonte: a autora

5.8 Arquitetura da Informação

Nesse tópico, será abordada a disposição das informações para o usuário do aplicativo móvel.

5.8.1 Protótipo de Alta Fidelidade da Aplicação para Dispositivos Móveis

Para facilitar o desenvolvimento da aplicação móvel e possibilitar uma análise prévia das telas do aplicativo, foi feito um protótipo de alta fidelidade, no qual podem ser visualizadas as informações com um alto grau de similaridade com o que foi desenvolvido.

Para desenvolvimento desse protótipo, foi utilizado o Figma, uma plataforma WEB de prototipação, que permite o desenho de telas e componentes, a execução do protótipo navegável, e o compartilhamento e gerenciamento de aces-

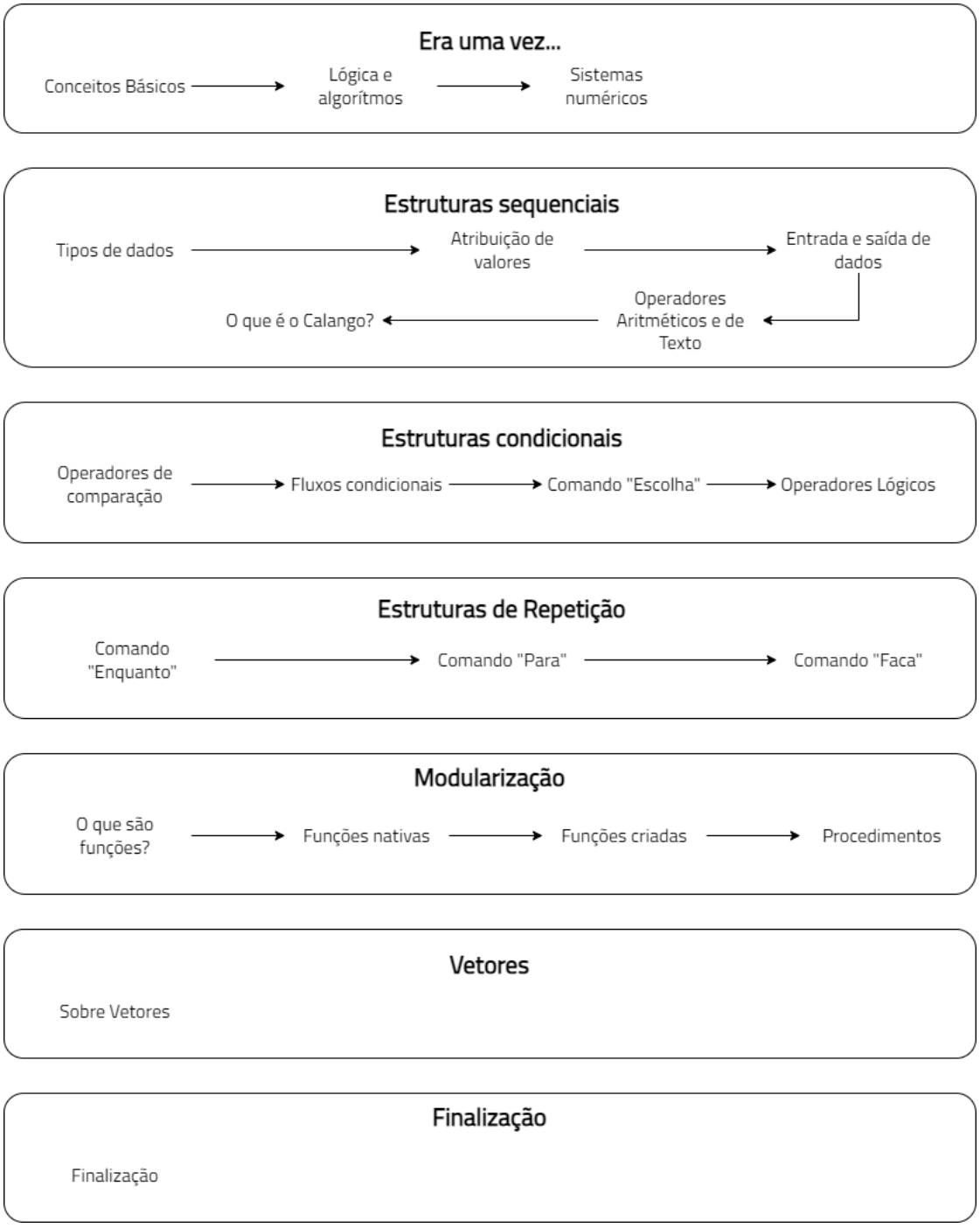
sos, caso assim se faça necessário. As imagens das telas geradas como protótipo podem ser observadas no Apêndice C, com as suas atribuições.

5.8.2 Fluxo de Aprendizado

A Figura 27 mostra o fluxo de conteúdo que o usuário irá percorrer através da tela "Sua História" (Figura 84). Os retângulos maiores indicam os módulos, e os textos seguidos de setas dentro deles representam os capítulos.

Apesar de mostrar um modelo pré determinado de conteúdo, todos os dados relacionados ao conteúdo da aplicação poderão ser modificados posteriormente através da aplicação WEB.

Figura 27 – Fluxo de Aprendizado - Módulos e Capítulos



Fonte: a autora

6 Resultados

Esse capítulo expõe as telas finais das aplicações, o formato de validação das funcionalidades do aplicativo móvel, e os resultados obtidos a partir dela.

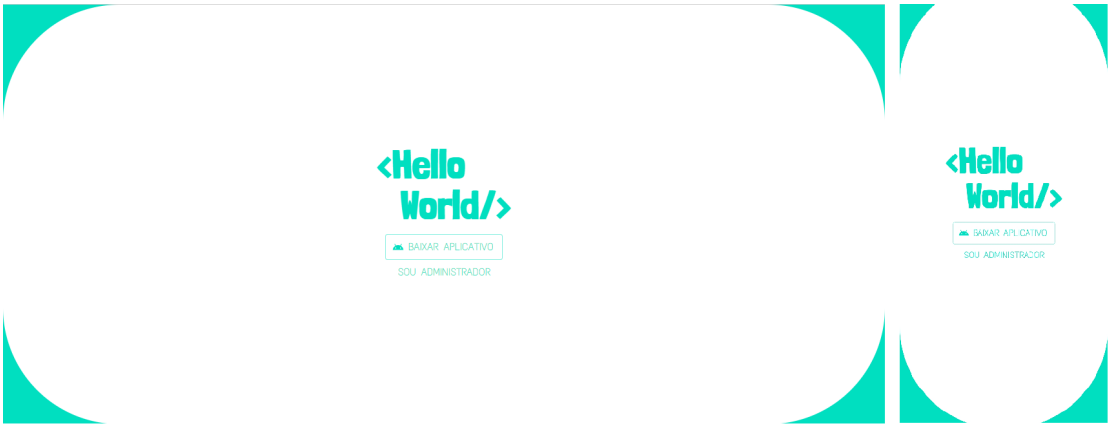
6.1 Aplicações Finais

A aplicação Web pode ser acessada através desse link, onde também pode ser baixado o arquivo para instalação da aplicação em dispositivos móveis.

É possível visualizar as telas das aplicações a seguir, sendo que a aplicação Web possui a exposição em duas telas de tamanhos diferentes, visando a demonstração do requisito de responsividade.

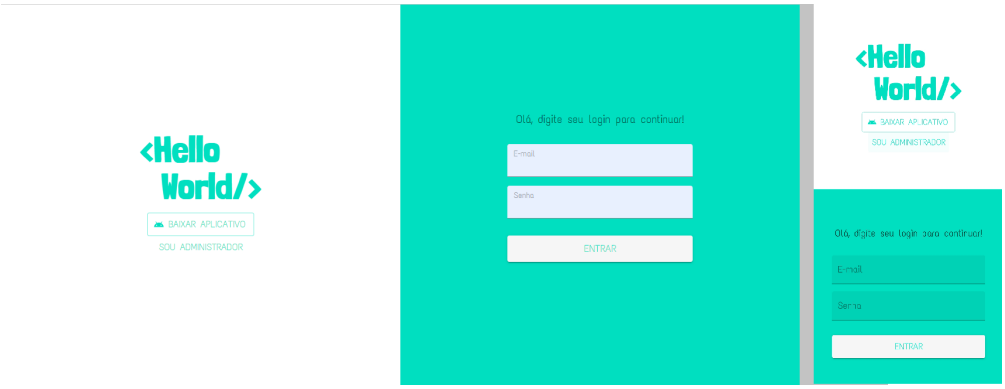
6.1.1 Aplicação Web

Figura 28 – Tela Inicial



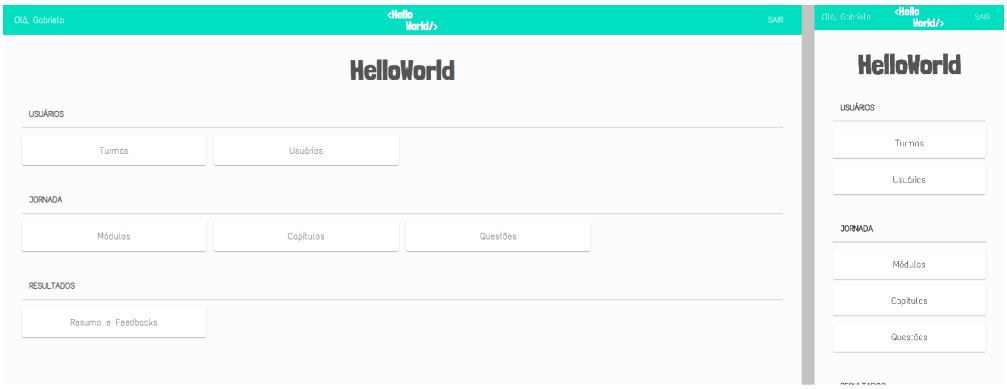
Fonte: a autora

Figura 29 – Tela de Login



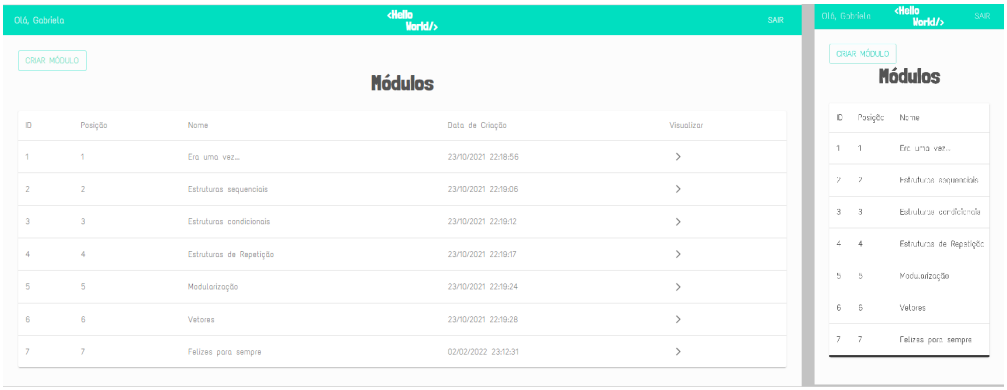
Fonte: a autora

Figura 30 – Tela Inicial - Área Logada



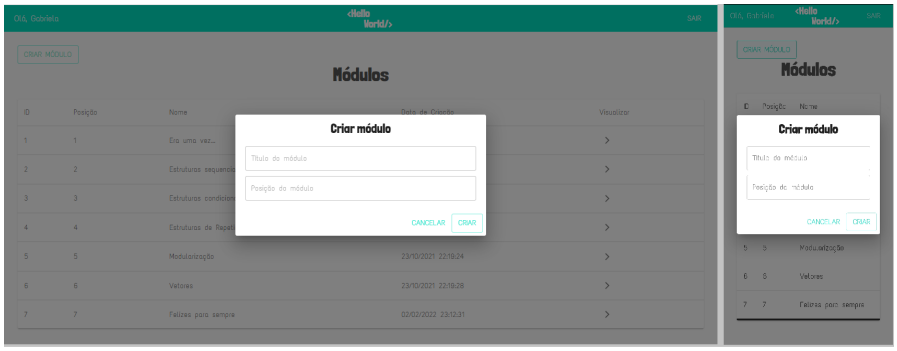
Fonte: a autora

Figura 31 – Tela de Listagem de Itens



Fonte: a autora

Figura 32 – Tela de Criação de Item

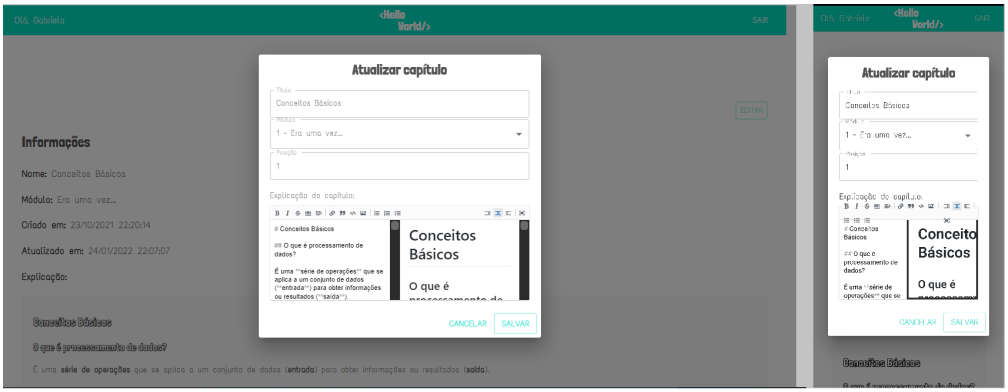


Fonte: a autora

Figura 33 – Tela de Descrição de Item

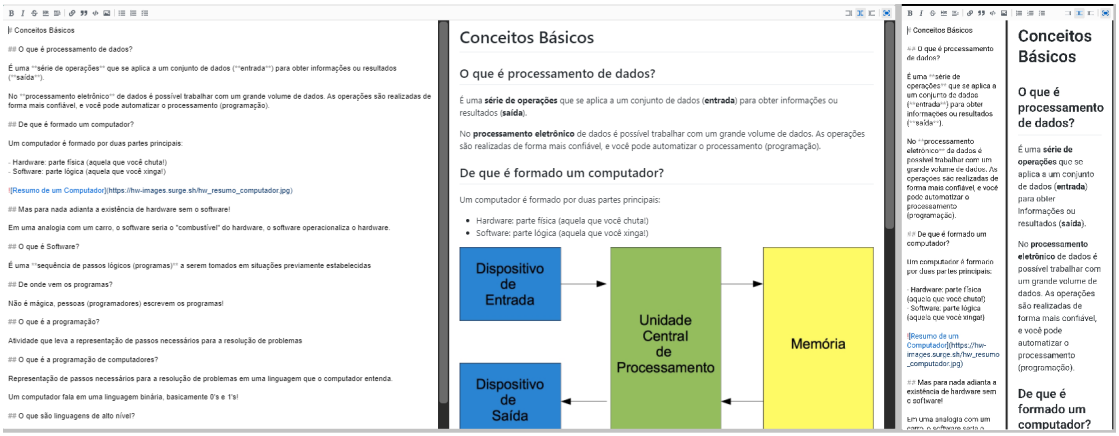
Fonte: a autora

Figura 34 – Tela de Edição de Item



Fonte: a autora

Figura 35 – Tela de Edição de Markdown



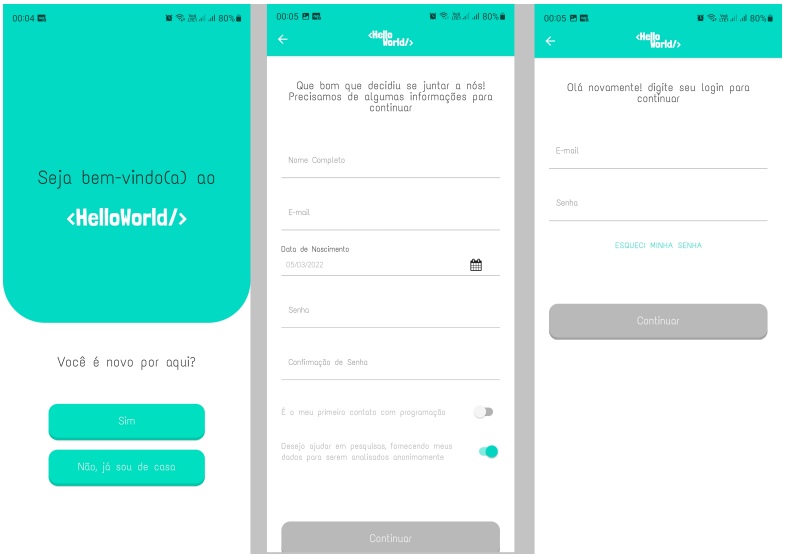
Fonte: a autora

Figura 36 – Tela de Resultados

Fonte: a autora

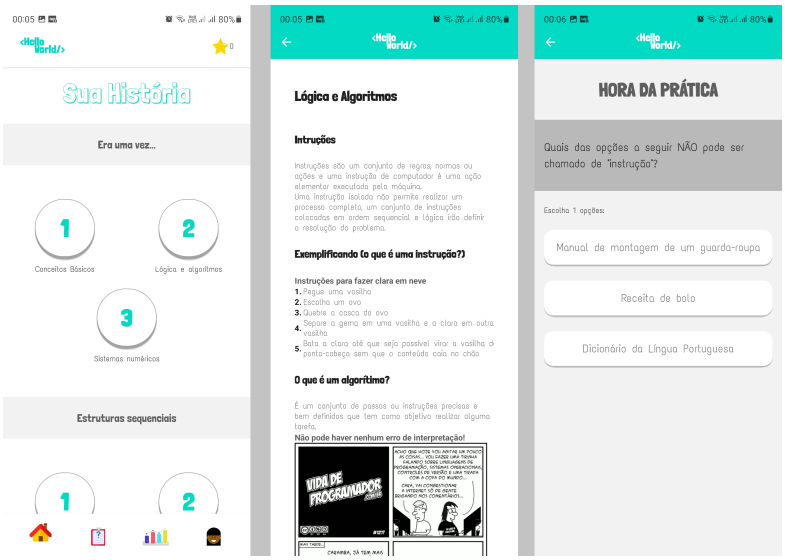
6.1.2 Aplicação Móvel

Figura 37 – Tela Inicial, Cadastro e *Login*



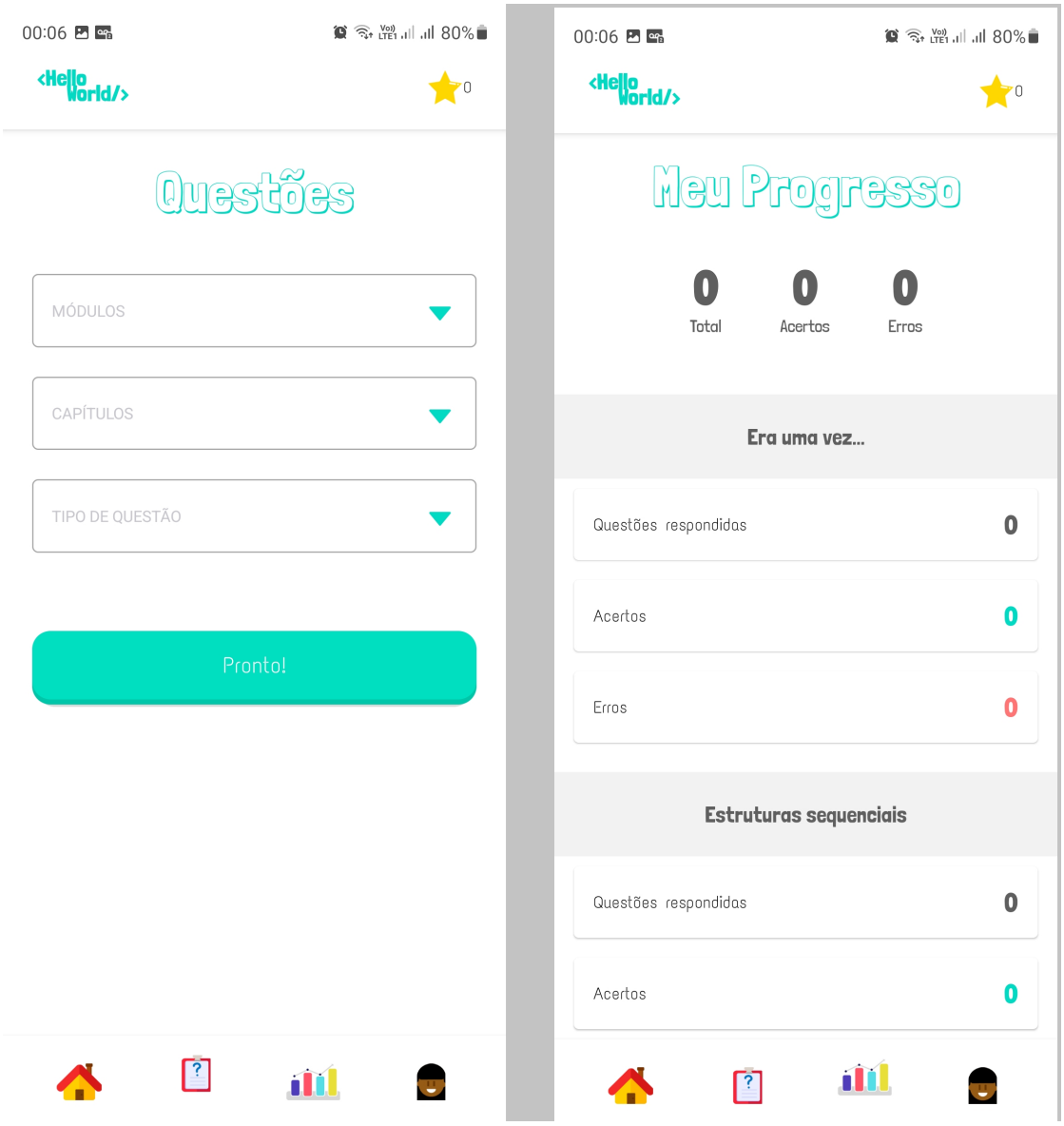
Fonte: a autora

Figura 38 – Tela de Jornada, de Explicação e de Atividade



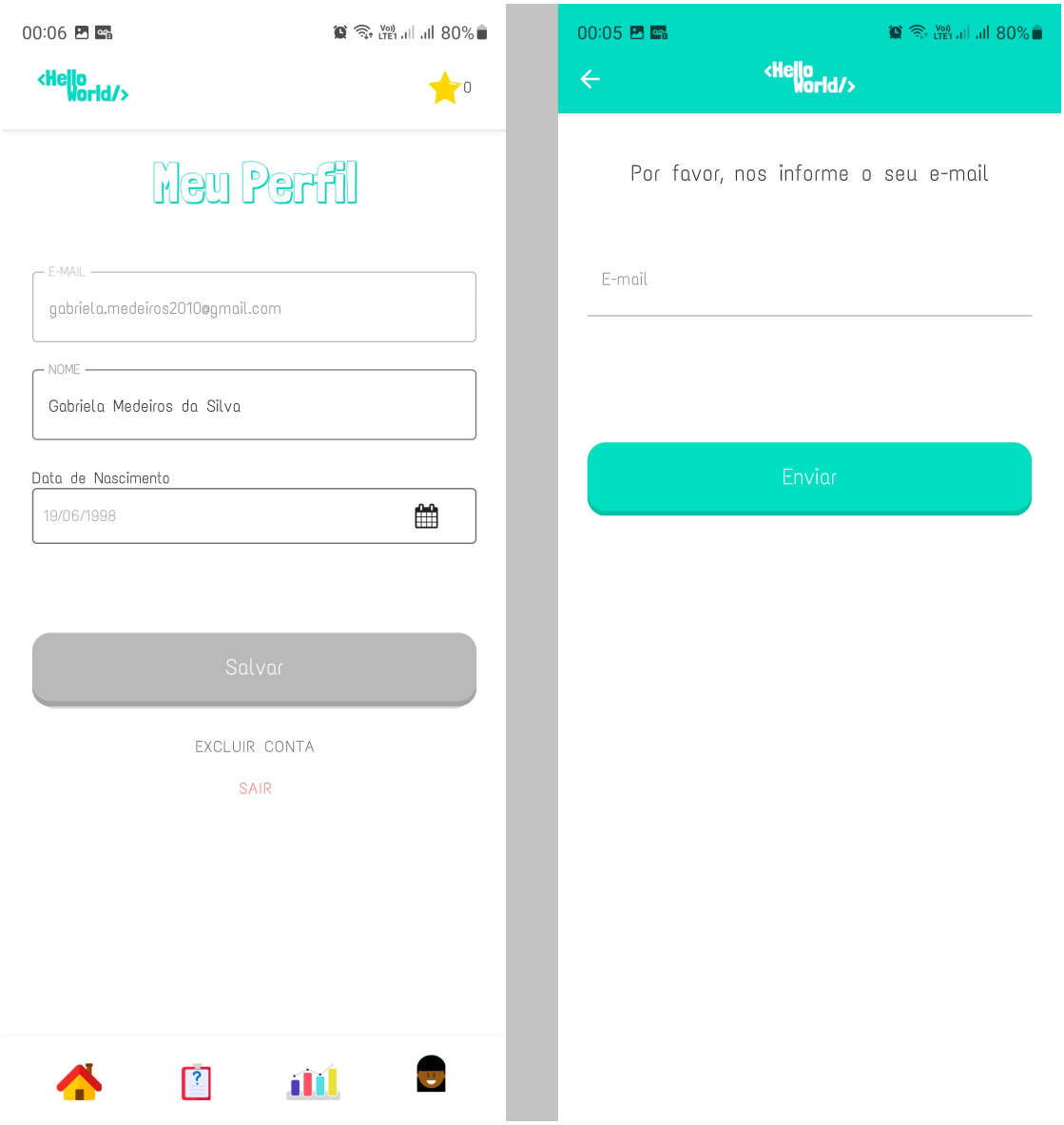
Fonte: a autora

Figura 39 – Tela de Questões Avulsas e de Progresso



Fonte: a autora

Figura 40 – Tela de Perfil e Recuperação de Senha



Fonte: a autora

6.2 Método de Validação

Visando buscar resultados assertivos sobre as funcionalidades da aplicação e a sua usabilidade em um espaço de tempo relativamente curto, foi desenvolvido um roteiro, em formato de formulário, para ser divulgado para os usuários interessados,

com o objetivo de levá-los a explorarem a aplicação em um tempo estimado entre 15 (quinze) e 20 (vinte) minutos. No total, 7 (sete) usuários realizaram o teste de avaliação guiados pelo roteiro, sendo que o formulário ficou aberto para respostas durante 2 (duas) semanas.

O fluxo do formulário pode ser visualizado na Figura 41, e a legenda dos seus objetos pode ser vista a seguir:

- Quadrado amarelo: texto explicativo;
- Quadrado branco: enunciado de uma questão;
- Círculo azul: questão com resposta de "sim"ou "não";
- Círculo verde: questão com resposta única e opções numéricas, e
- Círculo vermelho: questões de escrita aberta.

6.3 Resultados Obtidos

A primeira parte da validação consistiu em enviar o formulário para a primeira pessoa que demonstrasse interesse em respondê-lo. Isso foi feito para que problemas impeditivos no uso da aplicação, fossem identificados e tratados a tempo do restante das respostas. Após isso, o formulário foi compartilhado para os outros usuários.

Os gráficos gerados pelo Google, representando as respostas das questões, podem ser observados no Apêndice B.1.

6.3.1 Oportunidades de Melhoria

Através dos testes com os usuários, foram encontrados alguns problemas na execução do aplicativo e algumas melhorias que poderiam ser aplicadas às funcionalidades.

Na Tabela 12, podem ser visualizadas a descrição de cada problema encontrado, sua relevância (baixa ou alta), e se foi ou não resolvido até o término desse trabalho.

Os problemas que impedem a utilização de uma funcionalidade são classificados com alta relevância, e todos os outros são classificados com baixa relevância.

Tabela 12 – Problemas Identificados no Aplicativo Durante a Etapa de Validação

Descrição	**Relevância**	**Resolvido**	**Justificativa**
Os campos de e-mail apresentam o erro de formato inválido quando são preenchidos automaticamente pelo dispositivo	Alta	Sim	-
A mensagem gerada após o envio da avaliação não fecha	Alta	Sim	-
Ao tentar atualizar seu perfil, o usuário recebe uma mensagem de erro não tratada	Alta	Sim	-
Adequar as caixas de opções de resposta para letras maiores	Alta	Sim	-
As setas dos campos de seleção não estão clicáveis (apenas o restante do conteúdo)	Baixa	Sim	-
Os usuários clicam nos botões acidentalmente ao realizar o *scroll* na tela	Baixa	Não	Necessário mudança de biblioteca usada nos botões de todo o aplicativo

Além dos problemas identificados, os usuários tiveram a oportunidade de sugerir melhorias e funcionalidades que poderiam aumentar seu interesse pelo uso do aplicativo. Nos parágrafos a seguir, pode-se visualizar as sugestões feitas pelos usuários, e quais delas já foram desenvolvidas no momento direcionado para melhoria e correções de problemas no cronograma.

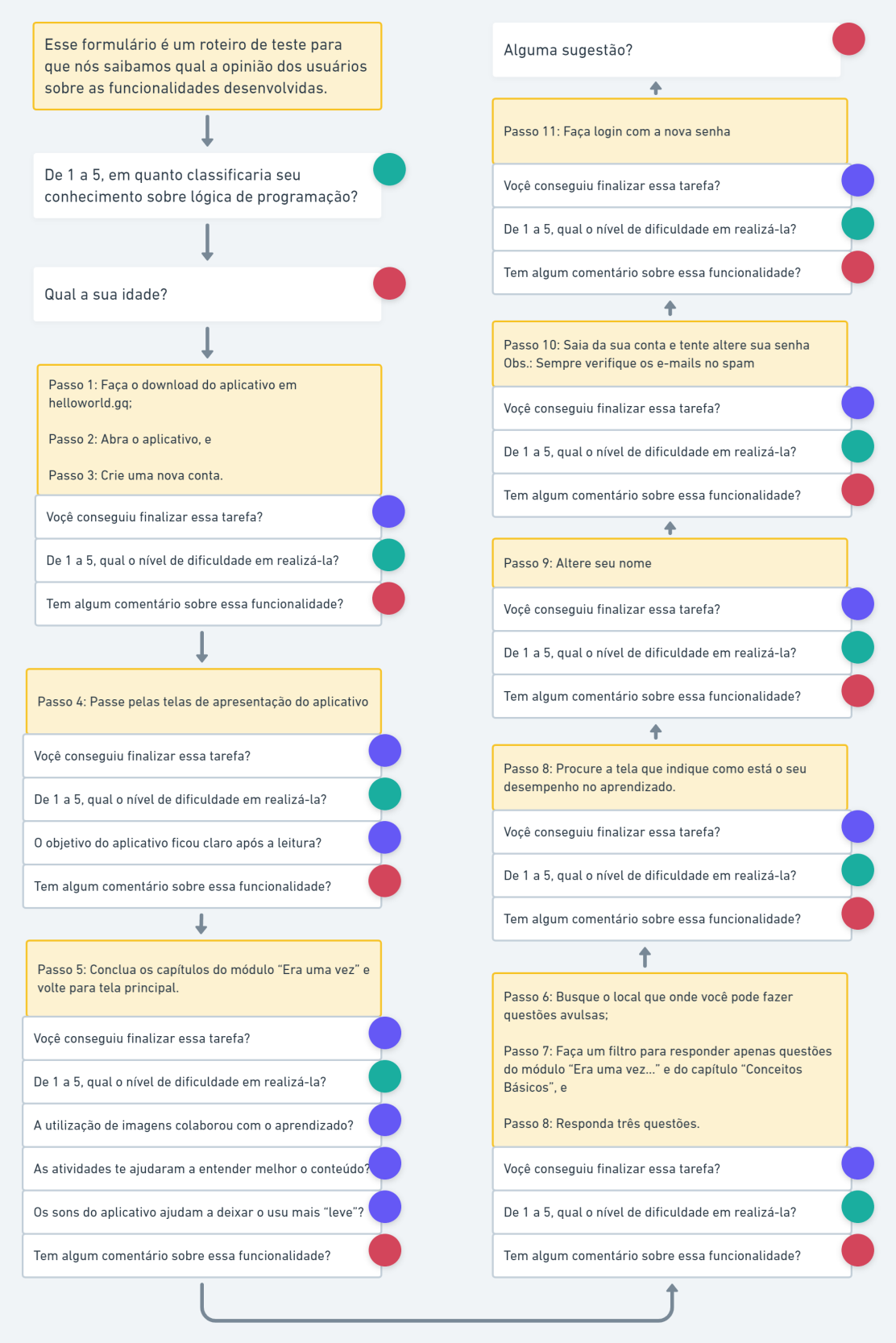
Permitir que o usuário refaça os capítulos que ele já concluiu:

Quando um usuário finaliza um capítulo, ele não consegue retornar para o seu conteúdo novamente, pois o botão fica desabilitado. Essa melhoria não foi desenvolvida, pois, requer uma alteração na lógica de finalização do capítulo que alteraria muitas funcionalidades.

Adicionar botão de mostrar texto nos campos de senha:
Os usuários sentiram falta da opção de mostrar e esconder a senha nos campos de segurança. Esse item foi desenvolvido, e está na versão final do aplicativo.

Adicionar um *feedback* textual quando o usuário errar ou acertar uma questão:
A cor de *feedback* positivo pode ser confundida com a cor principal do aplicativo no contexto das atividades. Isso leva o usuário a não perceber que a cor sinaliza um acerto. Esse item foi desenvolvido adicionando um texto acima da opção selecionada, informando se o usuário acertou ou errou a questão. Essa melhoria está na versão final do aplicativo.

Figura 41 – Roteiro de Validação de Requisitos



Fonte: a autora

7 Conclusão

O objetivo desse trabalho, de gerar um sistema integrado, que inclui uma aplicação gerenciadora e um aplicativo móvel em português de ensino de lógica de programação para dispositivos *Android*, mostrou-se necessário através da análise dos resultados de pesquisa e da exploração de aplicações similares. O estudo do escopo dessas aplicações mostrou a escassez de aplicativos voltados para pessoas que possuem o português como o seu idioma nativo, dado que, durante a etapa de pesquisa, foi revelado que o uso de um idioma não nativo pode dificultar o aprendizado de lógica de programação.

A partir da confirmação da justificativa, foram levantados potenciais usuários da aplicação, os quais foram utilizados como agentes das introspecções, técnica responsável por fornecer parte dos requisitos do sistema, sendo que muitos requisitos foram levantados a partir da análise de outros aplicativos educativos para dispositivos Android.

O desenvolvimento desse trabalho possibilitou a utilização de muitos dos conhecimentos relacionados à engenharia de software. Foram aplicados padrões de projeto, técnicas de levantamento de requisitos, metodologias de desenvolvimento e conceitos de padrão e configuração de software.

7.1 Funcionalidades Desenvolvidas

Todos os requisitos funcionais listados no Tópico 4.2 foram cumpridos a partir do desenvolvimento das funcionalidades listadas nos Tópicos 3 e 4.

7.2 Trabalhos Futuros

Para dar continuidade a essa aplicação, estão listadas, na Tabela 13, algumas funcionalidades e conteúdos que poderiam ser acrescentados a esse sistema.

Tabela 13 – Trabalhos Futuros

Descrição	**Tipo de Modificação**
Adaptação do aplicativo móvel para o sistema IOS	Arquitetura
Notificações e lembretes de estudo	Funcionalidade
Aumentar a gamificação, adicionando marcos e prêmios por conquistas e acertos	Grupo de Funcionalidades
Acrescentar um espaço de dúvidas dentro de cada questão, onde os usuários interagem entre si	Grupo de Funcionalidades
Fornecer opção de salvar os dados em uma base de dados local, para não ser necessário o consumo de *internet* para o uso do aplicativo	Funcionalidade e Arquitetura

Referências

AHMAD, M. O.; MARKKULA, J.; OIVO, M. Kanban in software development: A systematic literature review. In: *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*. [S.l.: s.n.], 2013. p. 9–16. Citado na página 31.

AURUM, A.; WOHLIN, C. *Engineering and Managing Software Requirements*. Springer Berlin Heidelberg, 2006. ISBN 9783540282440. Disponível em: <https://books.google.com.br/books?id=JRWrGLuWpLsC>. Citado na página 35.

AXIOS. *Introdução*. 2022. Disponível em: <https://axios-http.com/ptbr/docs/intro>. Citado na página 80.

CHUNG, L. et al. The nfr framework in action. In: _____. *Non-Functional Requirements in Software Engineering*. Boston, MA: Springer US, 2000. p. 15–45. ISBN 978-1-4615-5269-7. Disponível em: <https://doi.org/10.1007/978-1-4615-5269-7_2>. Citado na página 37.

CRNKOVIC, I. Component-based software engineering — new challenges in software development. *Software Focus*, v. 2, n. 4, p. 127–133, 2001. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/swf.45>. Citado 2 vezes nas páginas 34 e 68.

CYSNEIROS, L. M.; LEITE, J. Requisitos não funcionais: da elicitação ao modelo conceitual. *PhDTese, PUC-RJ*, 2001. Citado na página 37.

FREITAS, C. de. *Metodologia do Trabalho Científico: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico - 2ª Edição*. Editora Feevale, 2013. ISBN 9788577171583. Disponível em: <https://books.google.com.br/books?id=zUDsAQAAQBAJ>. Citado 2 vezes nas páginas 29 e 30.

GOGUEN, J.; LINDE, C. Techniques for requirements elicitation. In: *[1993] Proceedings of the IEEE International Symposium on Requirements Engineering*. [S.l.: s.n.], 1993. p. 152–164. Citado 2 vezes nas páginas 35 e 36.

NING, W. et al. Research on the web information system development platform based on mvc design pattern. In: *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*. [S.l.: s.n.], 2008. v. 3, p. 203–206. Citado na página 32.

RODRIGUES, J. *Modelo Entidade Relacionamento (MER) e Diagrama Entidade-Relacionamento (DER)*. 2014. Disponível em: <https://www.devmedia.com.br/modelo-entidade-relacionamento-mer-e-diagrama-entidade-relacionamento-der/14332>. Citado na página 76.

SILVA, G.; SANTOS, G.; RISSOLI, V. A importância da linguagem nativa para a aprendizagem significativa em lógica de programação. In: *Anais do XXXI Simpósio Brasileiro de Informática na Educação*. Porto Alegre, RS, Brasil: SBC, 2020. p. 1803–1812. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/sbie/article/view/12936>. Citado na página 35.

SRIVASTAVA, A.; BHARDWAJ, S.; SARASWAT, S. Scrum model for agile methodology. In: *2017 International Conference on Computing, Communication and Automation (ICCCA)*. [S.l.: s.n.], 2017. p. 864–869. Citado na página 32.

STORE, P. *Duolingo: Inglês e muito mais! - Apps no Google Play*. 2022. Disponível em: <https://play.google.com/store/apps/details?id=com.duolingo&hl=pt_BR&gl=US>. Citado na página 47.

Apêndices

APÊNDICE A – Repositórios

A organização pode ser acessada pelo *link* https://github.com/HelloWorld-Repo

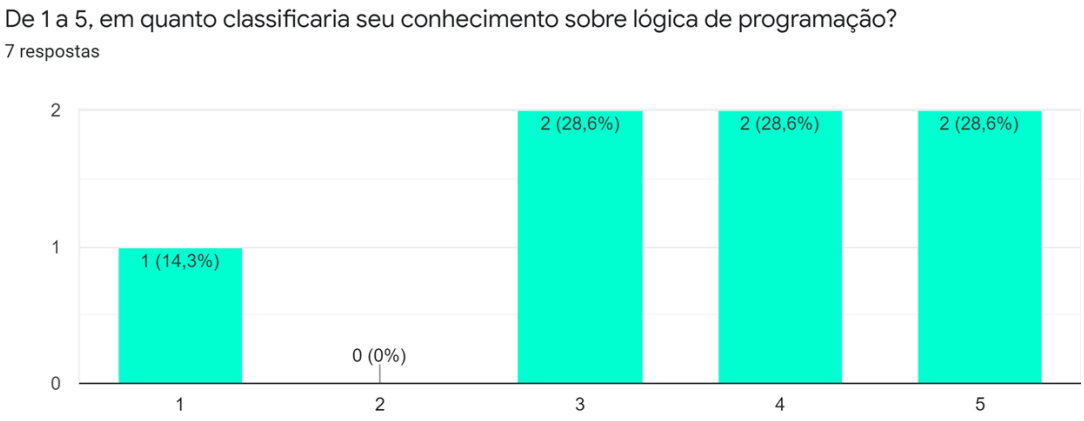
Tabela 14 – Link dos Repositórios no GitHub

Módulo do Sistema	**Repositório**
Aplicação WEB - Front-End	https://github.com/HelloWorld-Repo/HelloWorld-Web
API	https://github.com/HelloWorld-Repo/HelloWorld-API
Aplicação *Mobile*	https://github.com/HelloWorld-Repo/HelloWorld-App

APÊNDICE B – Resultados de Pesquisas

B.1 Pesquisa de Avaliação Guiada

Figura 42 – Pergunta 1 - Conhecimento Sobre Lógica de Programação



Fonte: a autora

B.2 Entrevista de Questionário para Elicitação de Requisitos

64 (sessenta e quatro) pessoas responderam a esse formulário, montado através do *Google Forms*. O formulário ficou aberto para respostas do dia 7 (sete) ao dia 29 (vinte e nove) do mês de julho do ano de 2021 (dois mil e vinte e um).

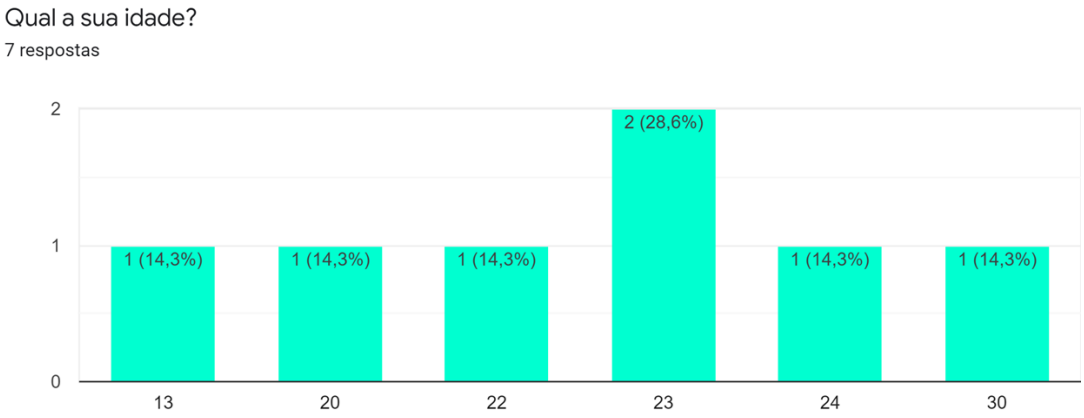


Figura 43 – Pergunta 2 - Idade
Fonte: a autora

CADASTRO

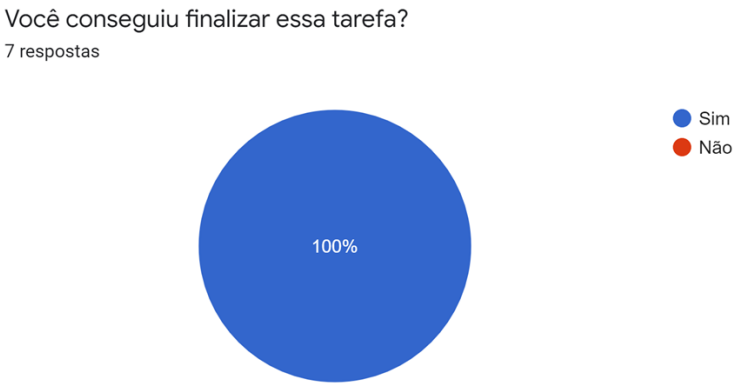


Figura 44 – Cadastro - Finalização da Tarefa
Fonte: a autora

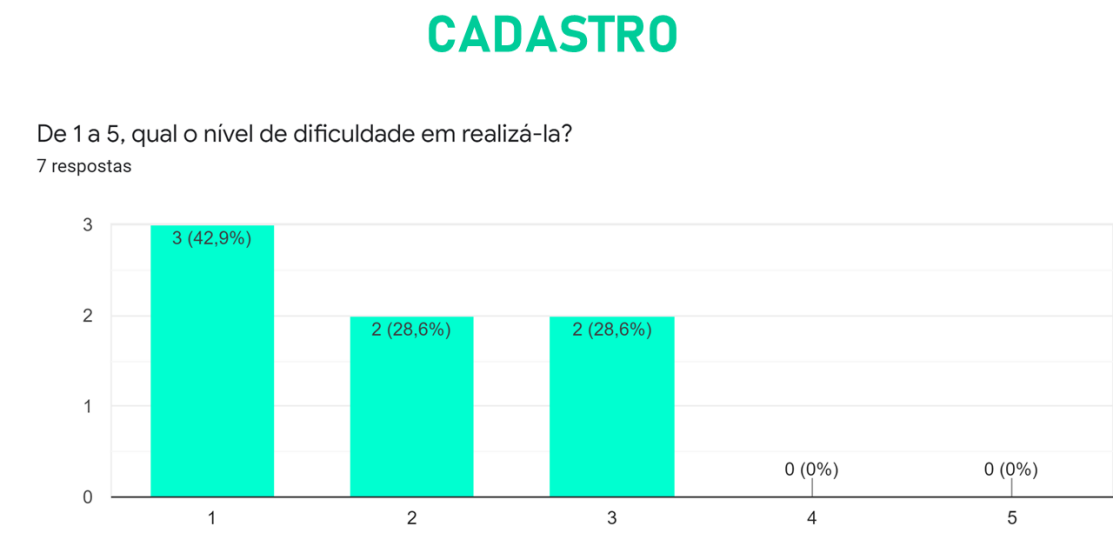


Figura 45 – Cadastro - Nível de Dificuldade
Fonte: a autora

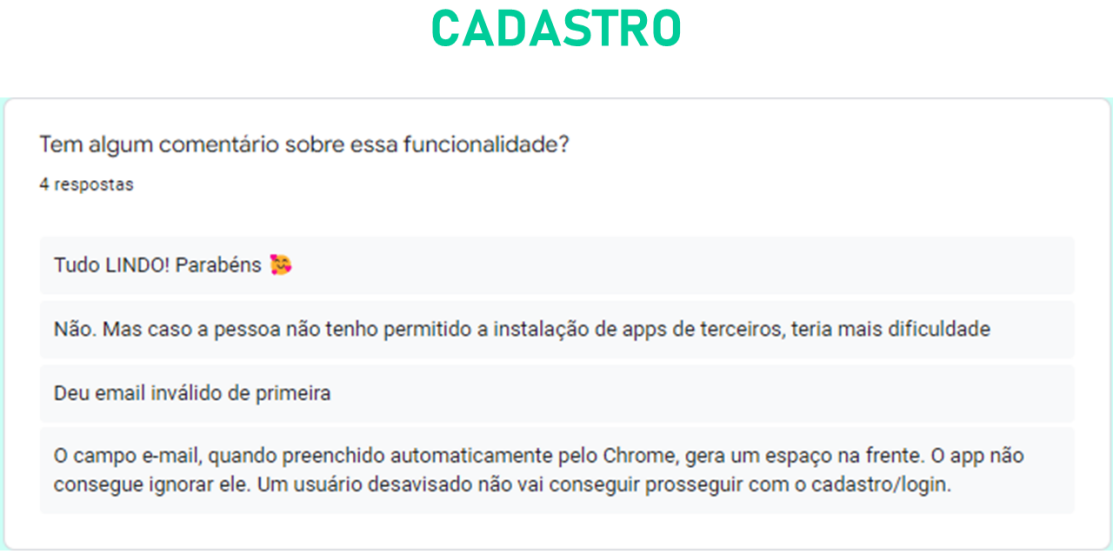


Figura 46 – Cadastro - Comentários
Fonte: a autora

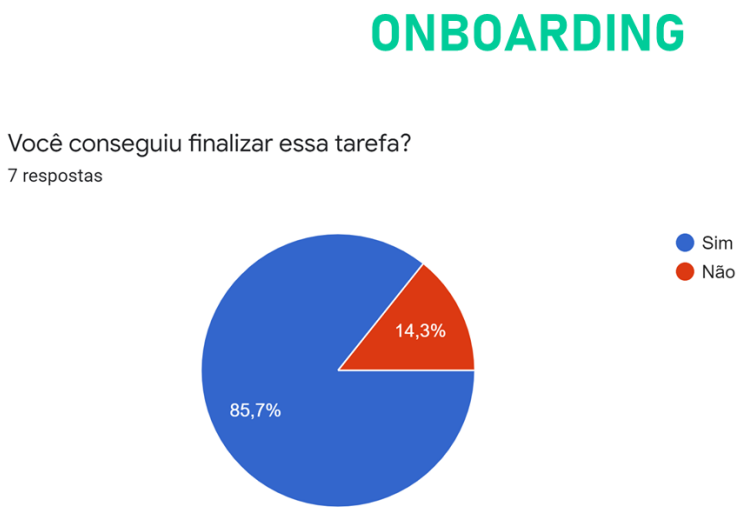


Figura 47 – Onboarding - Finalização da Tarefa
Fonte: a autora

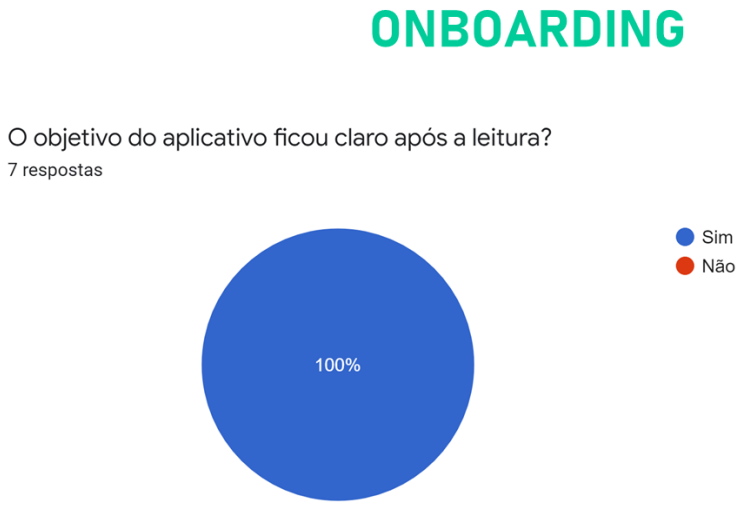


Figura 48 – Onboarding - Entendimento do Objetivo do Aplicativo
Fonte: a autora

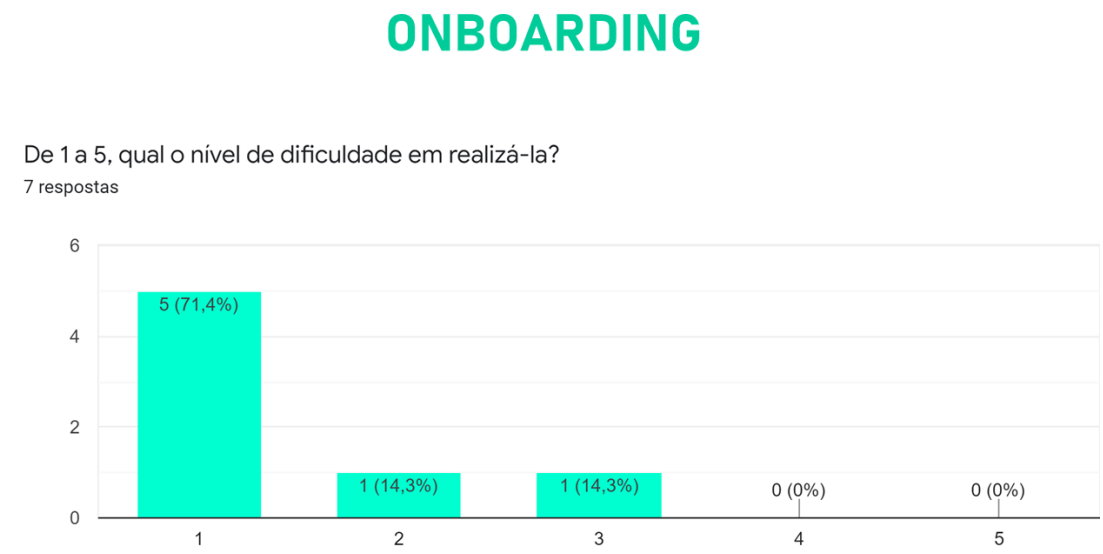


Figura 49 – Onboarding - Nível de Dificuldade
Fonte: a autora

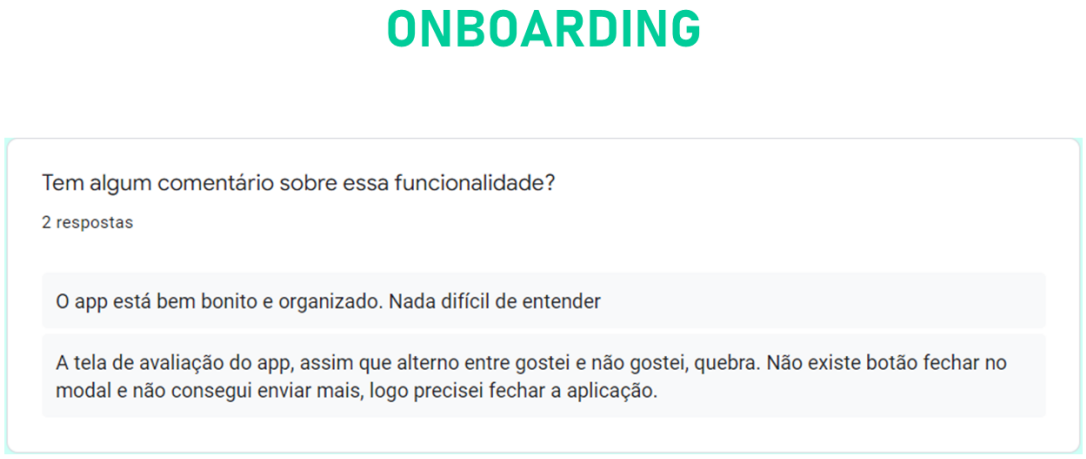


Figura 50 – Onboarding - Comentários
Fonte: a autora

Figura 51 – Jornada - Finalização da Tarefa
Fonte: a autora

Figura 52 – Jornada - Utilização de Imagens
Fonte: a autora

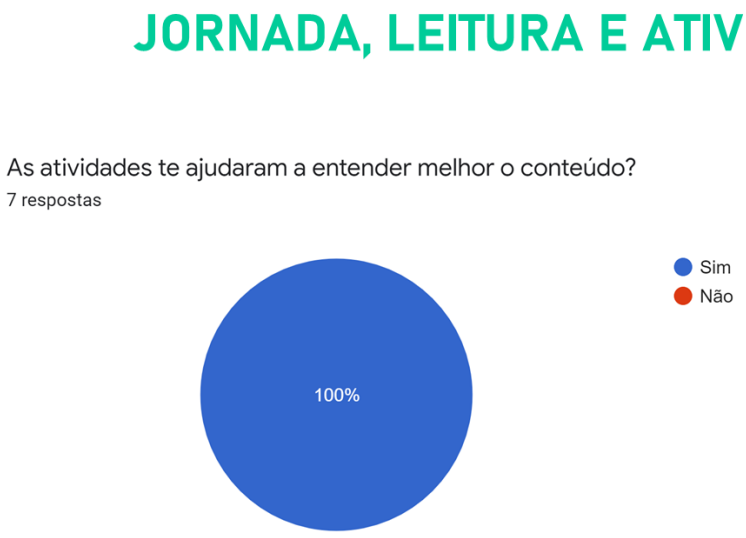


Figura 53 – Jornada - Sobre as Questões
Fonte: a autora

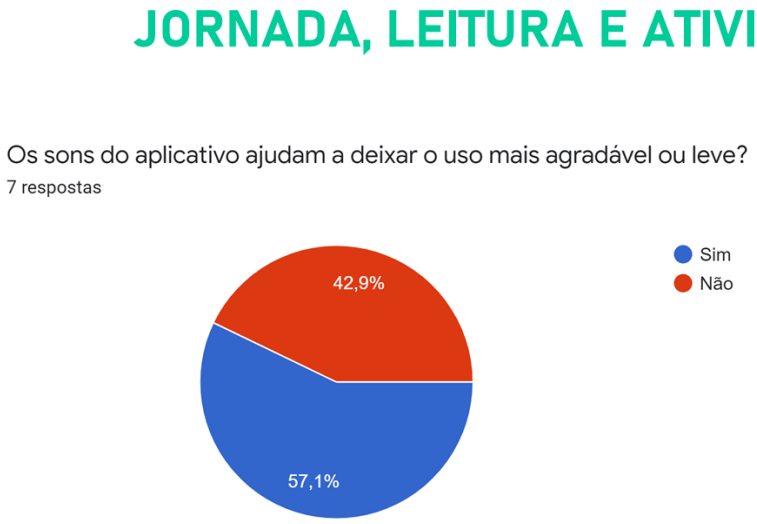


Figura 54 – Jornada - Sons do Aplicativo
Fonte: a autora

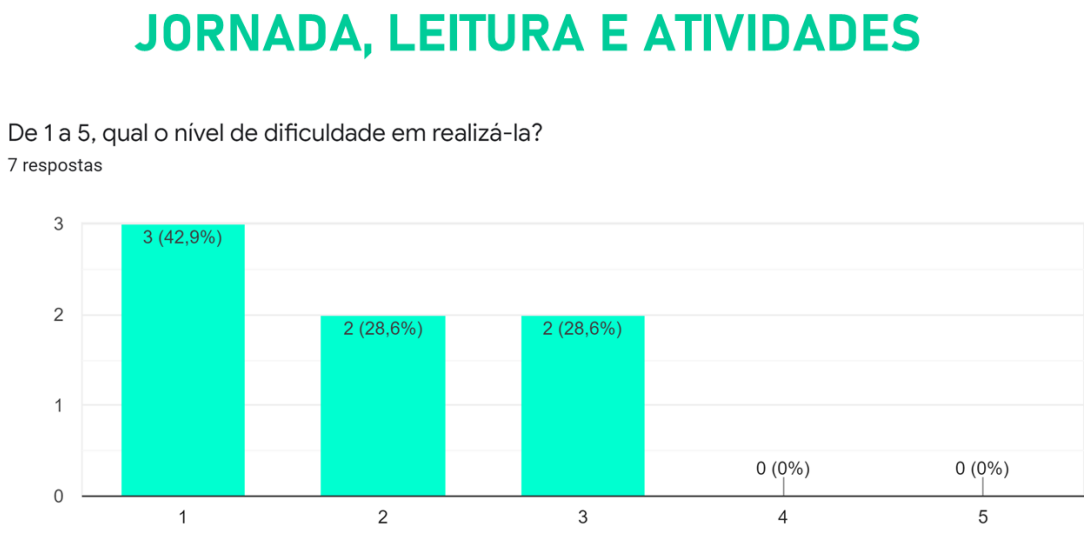


Figura 55 – Jornada - Nível de Dificuldade
Fonte: a autora

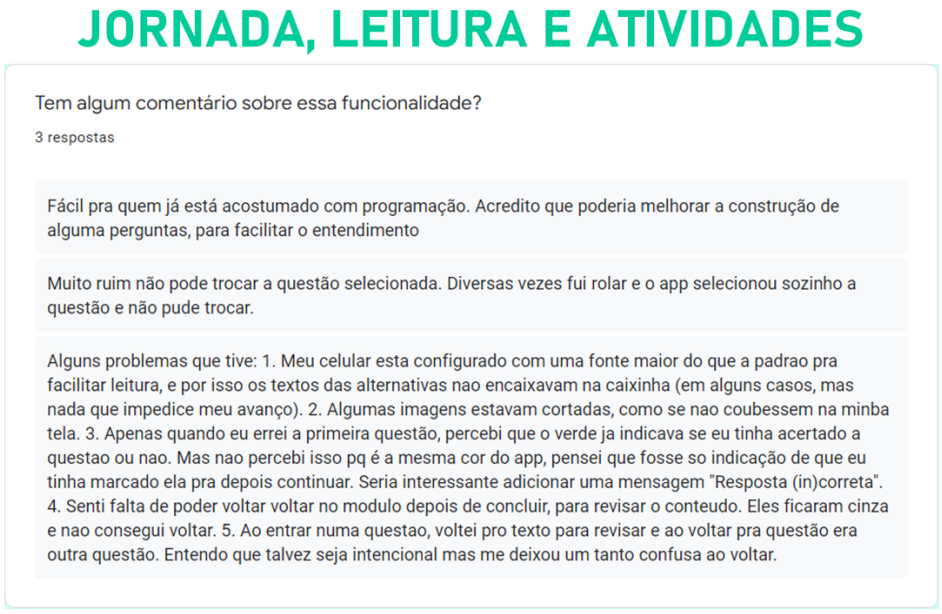


Figura 56 – Jornada - Comentários
Fonte: a autora

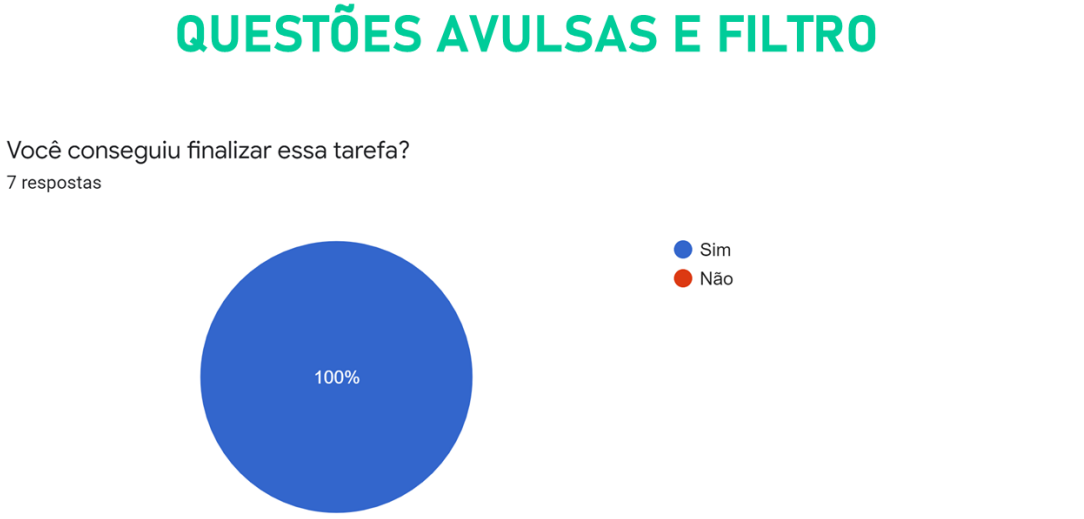


Figura 57 – Questões Avulsas - Finalização da Tarefa
Fonte: a autora

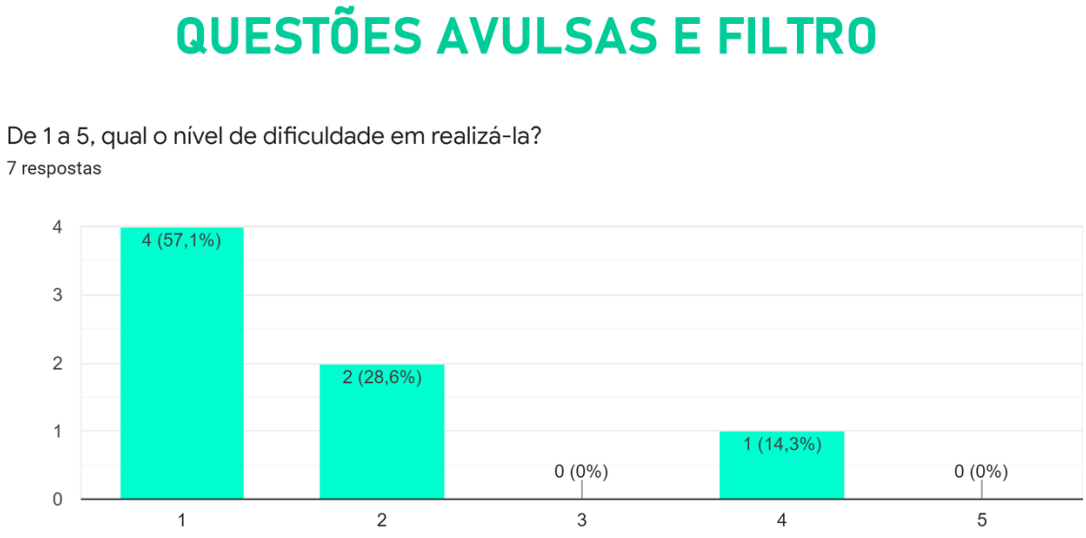


Figura 58 – Questões Avulsas - Nível de Dificuldade
Fonte: a autora

QUESTÕES AVULSAS E FILTRO

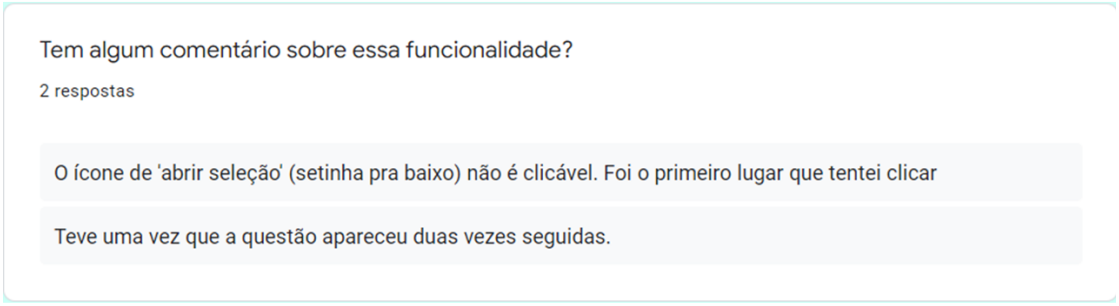


Figura 59 – Questões Avulsas - Comentários
Fonte: a autora

TELA DE RESULTADOS

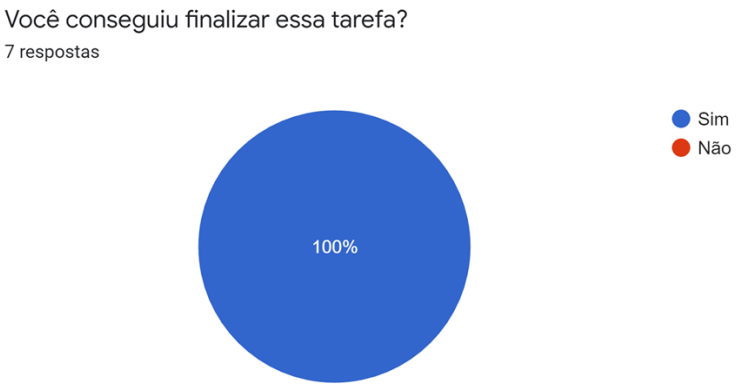


Figura 60 – Resumo - Finalização das Tarefas
Fonte: a autora

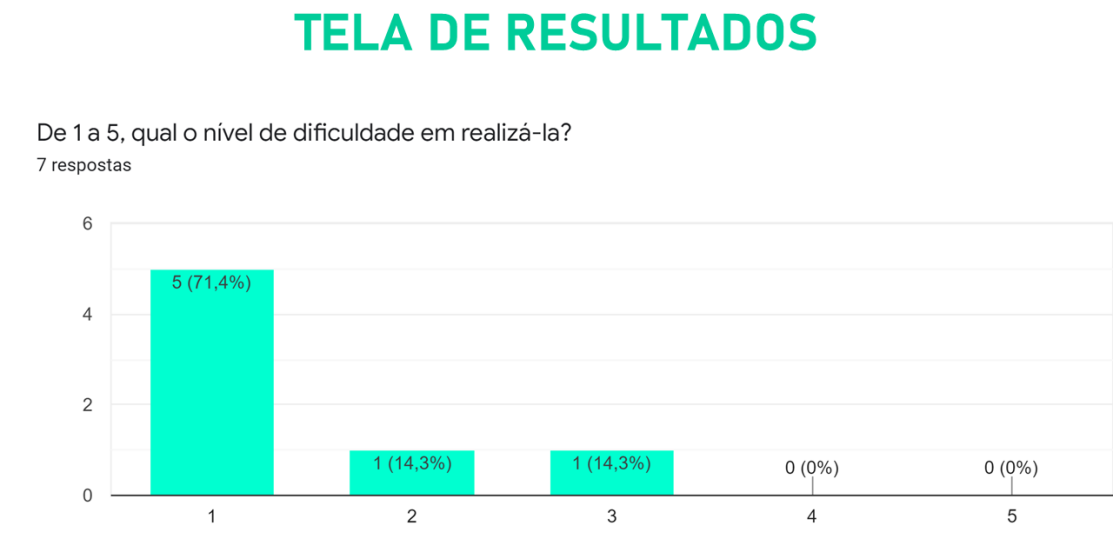


Figura 61 – Resumo - Nível de Dificuldade
Fonte: a autora

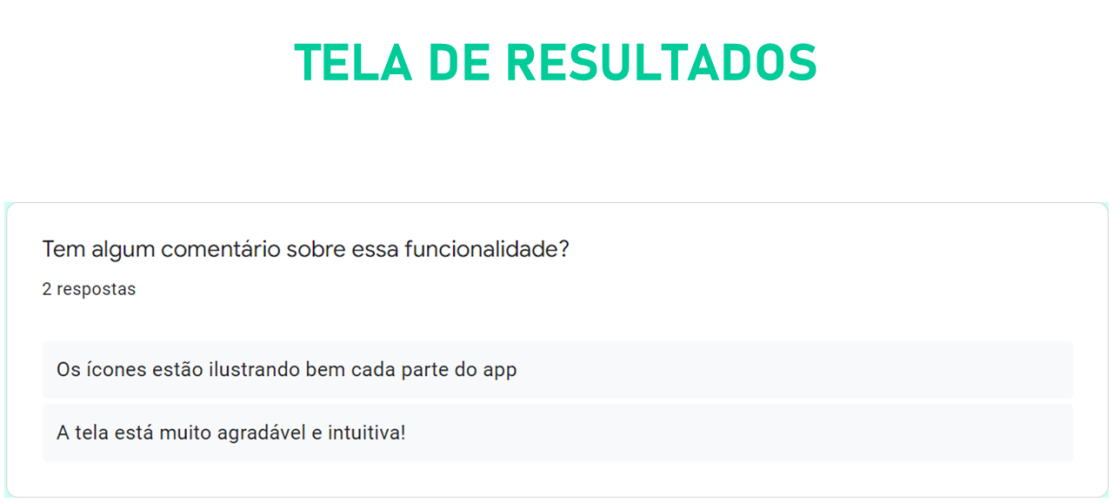


Figura 62 – Resumo - Comentários
Fonte: a autora

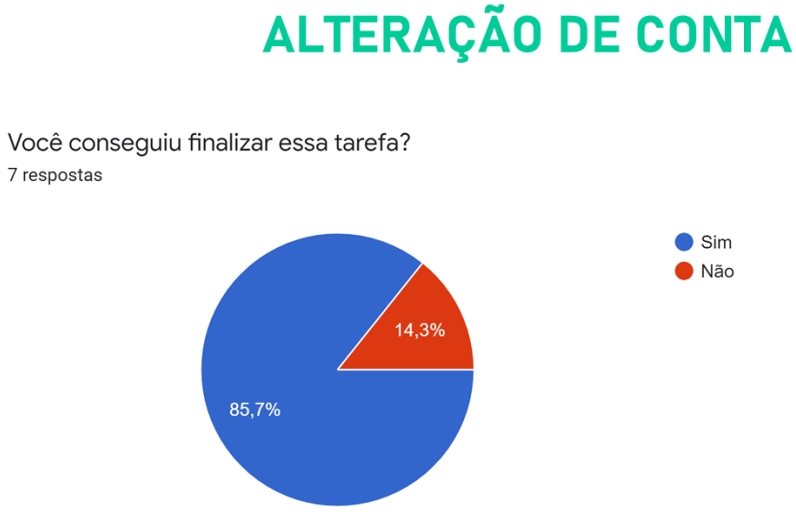


Figura 63 – Edição de Conta - Finalização da Tarefa
Fonte: a autora

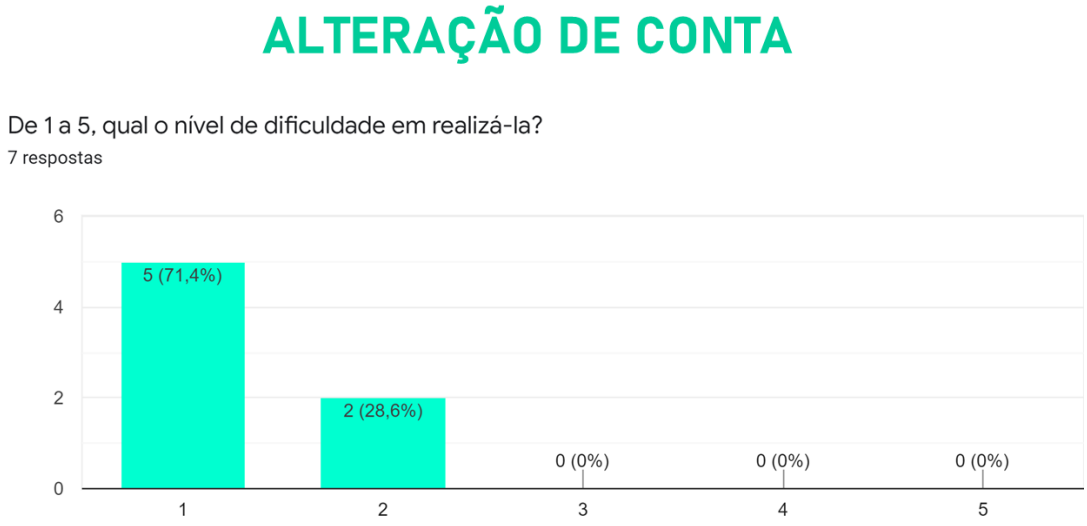


Figura 64 – Edição de Conta - Nível de Dificuldade
Fonte: a autora

ALTERAÇÃO DE CONTA

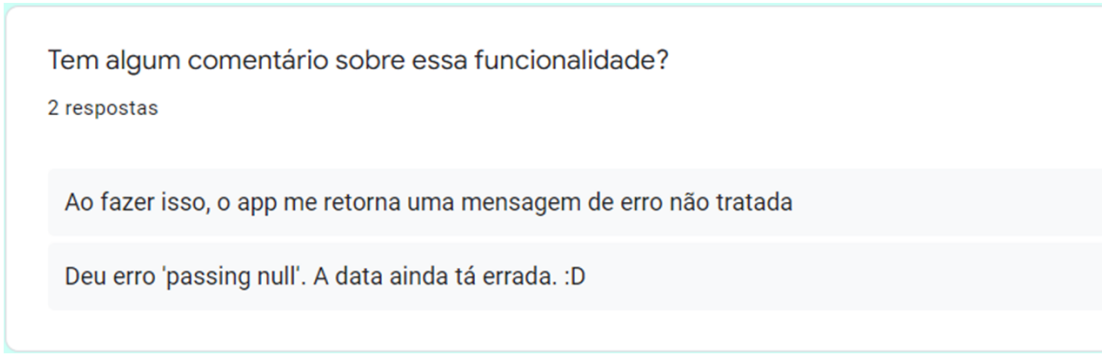


Figura 65 – Edição de Conta - Comentários
Fonte: a autora

RECUPERAÇÃO DE SENHA

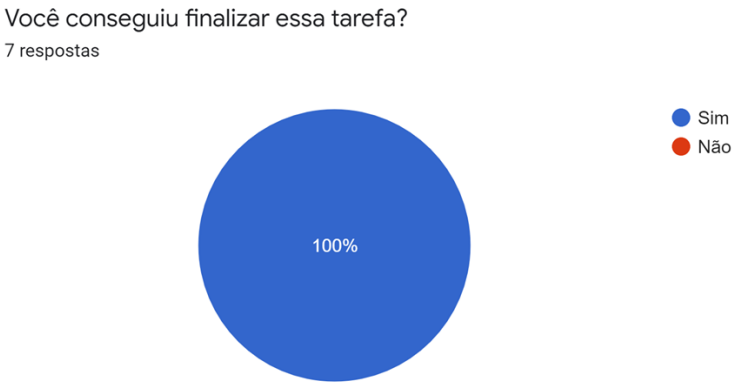


Figura 66 – Recuperação de Senha - Finalização da Tarefa
Fonte: a autora

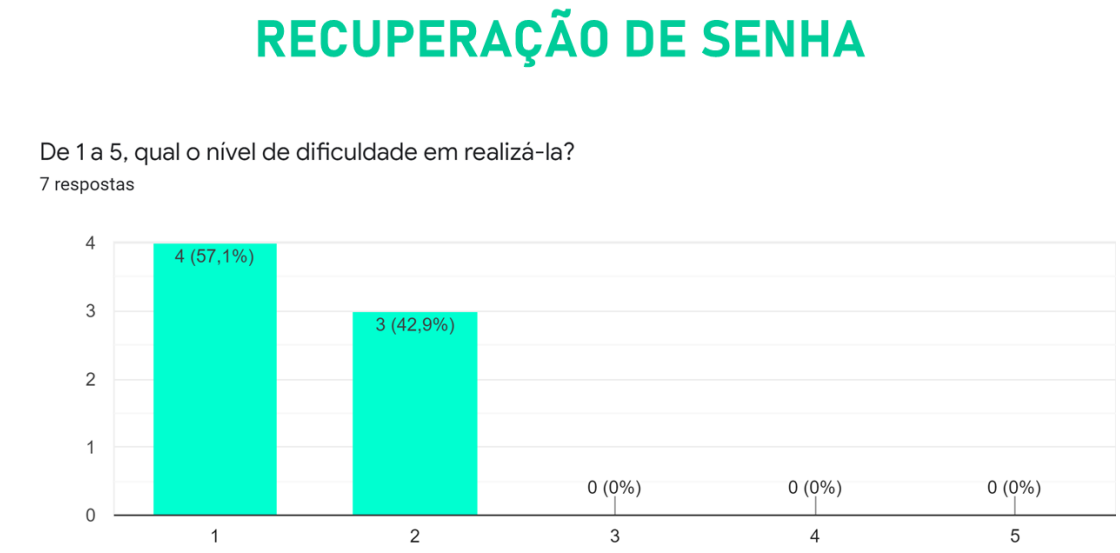


Figura 67 – Recuperação de Senha - Nível de Dificuldade
Fonte: a autora

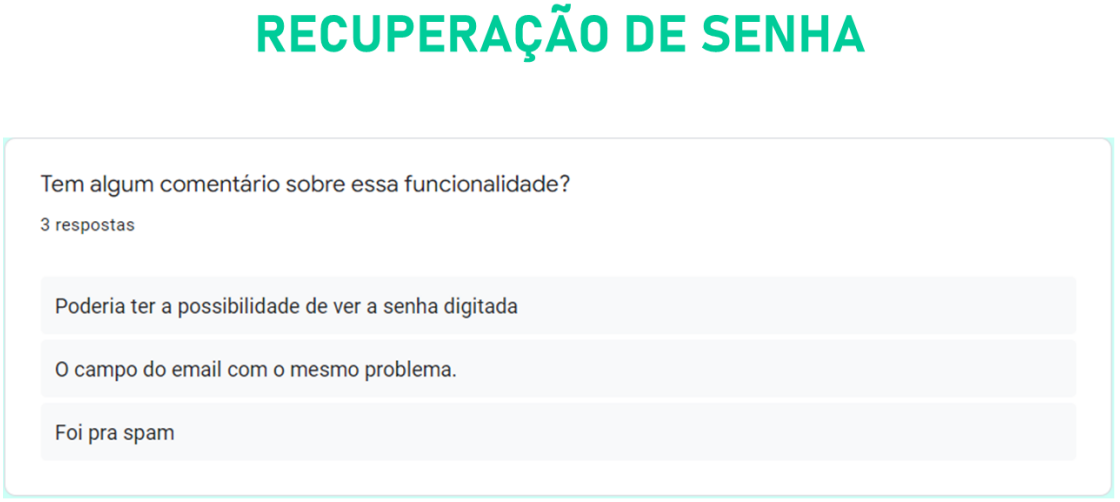


Figura 68 – Recuperação de Senha - Comentários
Fonte: a autora

Figura 69 – *Login* - Finalização da Tarefa
Fonte: a autora

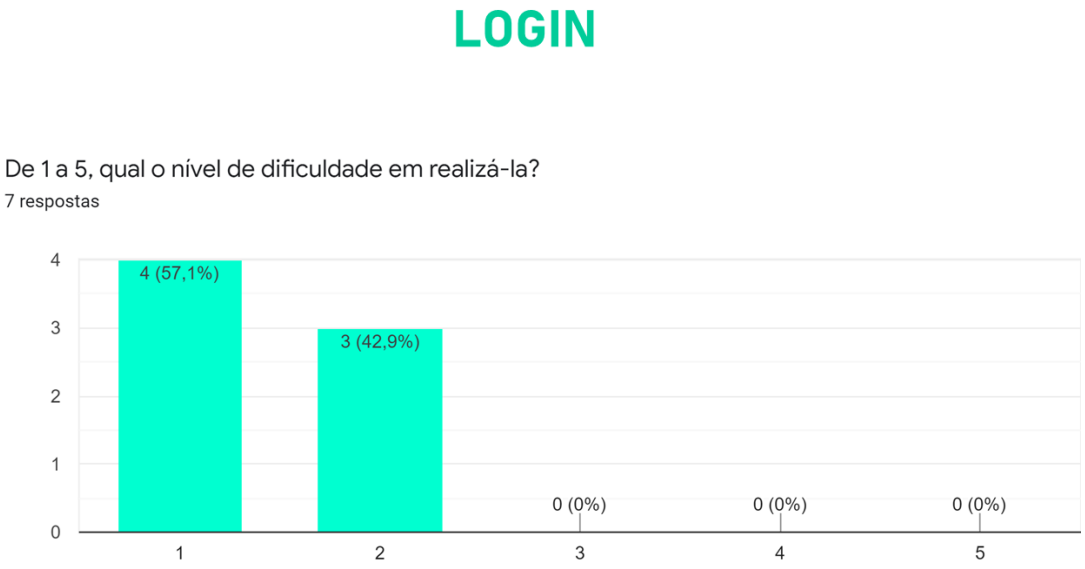
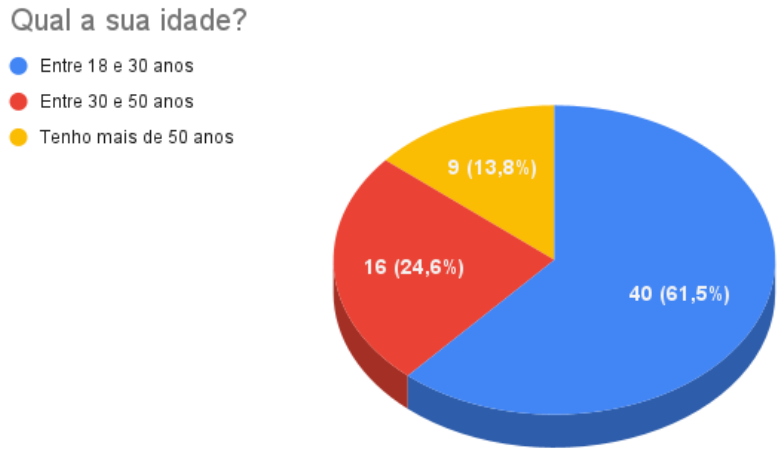


Figura 70 – *Login* - Nível de Dificuldade
Fonte: a autora

Figura 71 – (Pergunta 1) Qual a Sua Idade?



Fonte: a autora

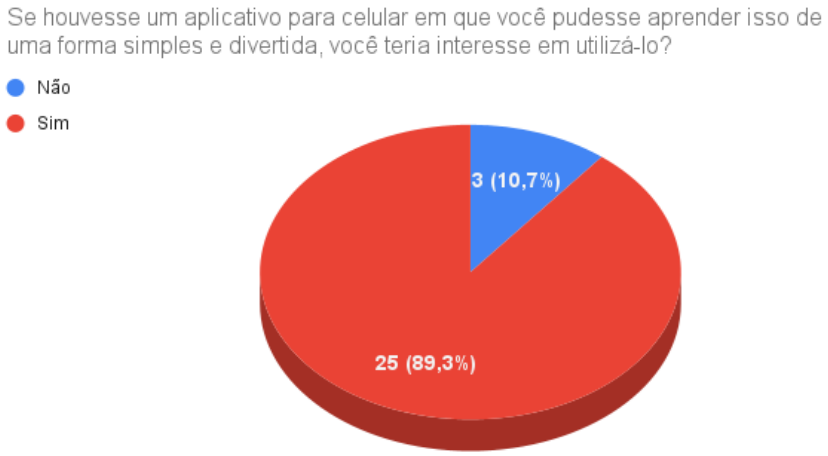
Figura 72 – (Pergunta 2) Você Já Estudou Programação?

Fonte: a autora

Figura 73 – (Pergunta 2.1.1) Por Qual Motivo?

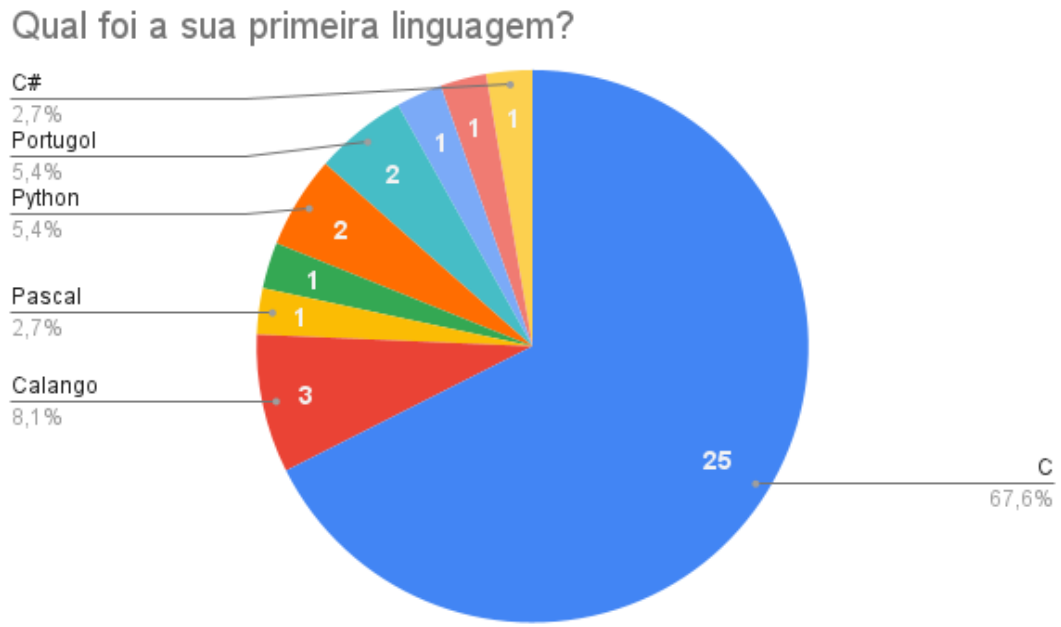
Fonte: a autora

Figura 74 – (Pergunta 2.1.2) Se Houvesse um Aplicativo para Celular em que Você Pudesse Aprender Isso de Uma Forma Simples e Divertida, Você Teria Interesse em Utilizá-lo?



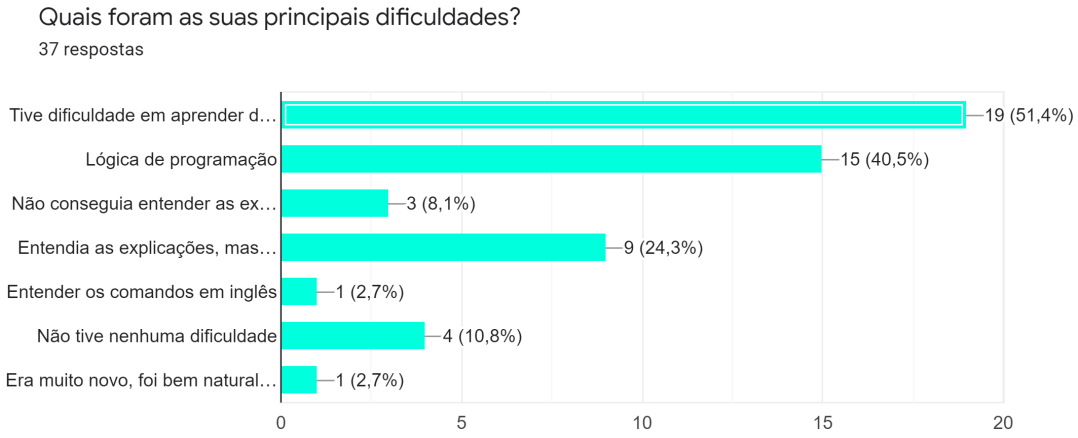
Fonte: a autora

Figura 75 – (Pergunta 2.2.1) Qual Foi a Sua Primeira Linguagem?



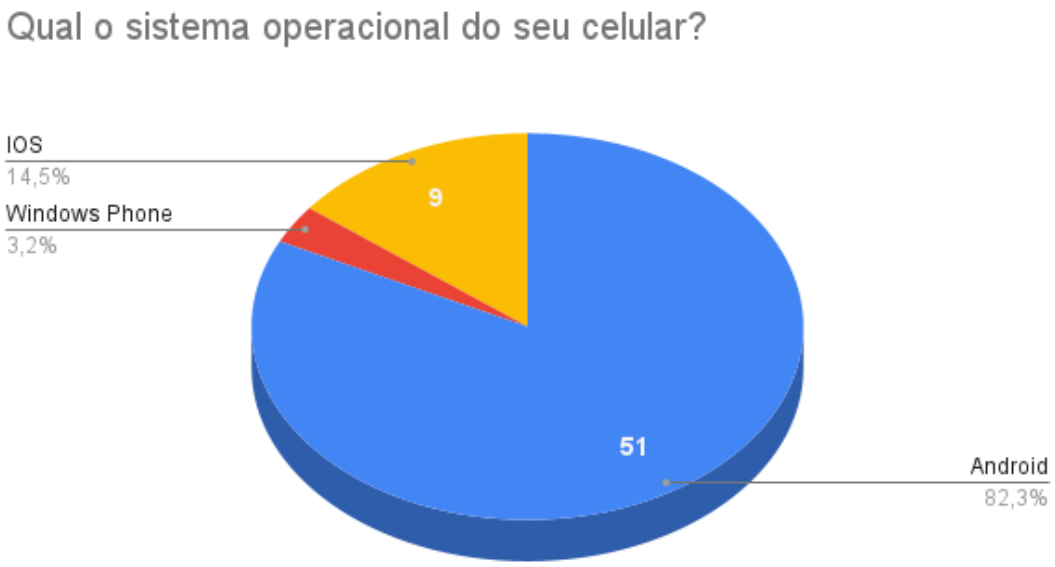
Fonte: a autora

Figura 76 – (Pergunta 2.2.2) Quais Foram as Suas Principais Dificuldades?



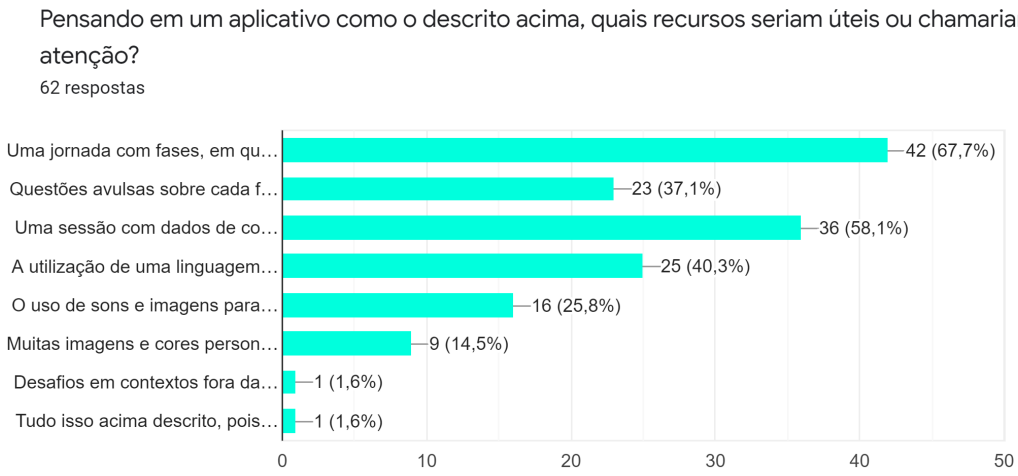
Fonte: a autora

Figura 77 – (Pergunta 3) Qual é o Sistema Operacional do Seu Celular?



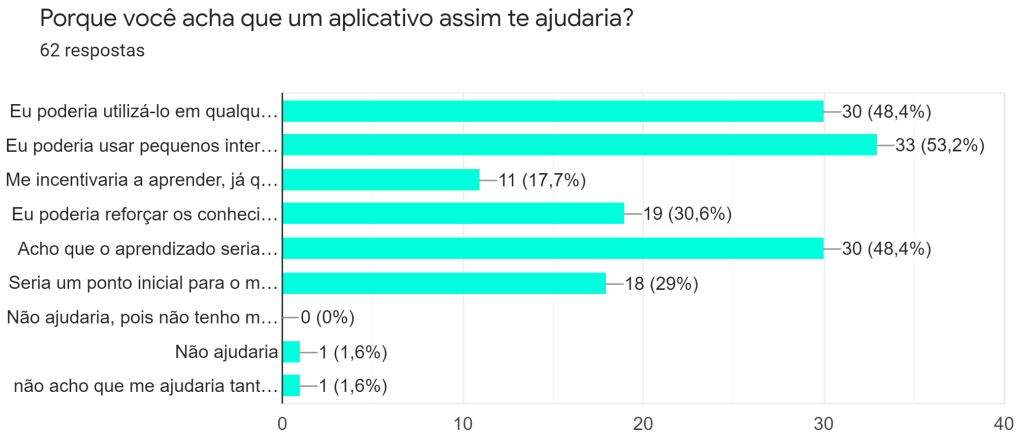
Fonte: a autora

Figura 78 – (Pergunta 4) Quais Desses Recursos Seriam Úteis ou Chamariam a Sua Atenção?



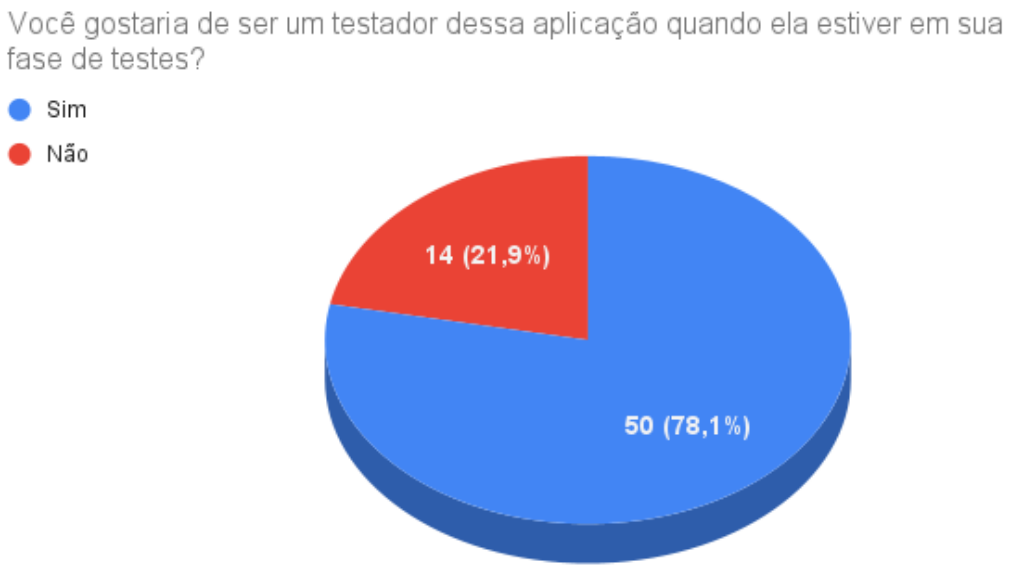
Fonte: a autora

Figura 79 – (Pergunta 5) Porque Você Acredita Que um Aplicativo Assim te Ajudaria?



Fonte: a autora

Figura 80 – (Pergunta 6) Você Gostaria de Ser um Testador Dessa Aplicação Quando ela Estiver em Sua Fase de Testes?



Fonte: a autora

APÊNDICE C – Telas do Protótipo de Alta Fidelidade

Figura 81 – Telas SPLASH, Inicial e de *Login*

Fonte: a autora

Figura 82 – Telas de Cadastro e de Envio de E-mail

Fonte: a autora

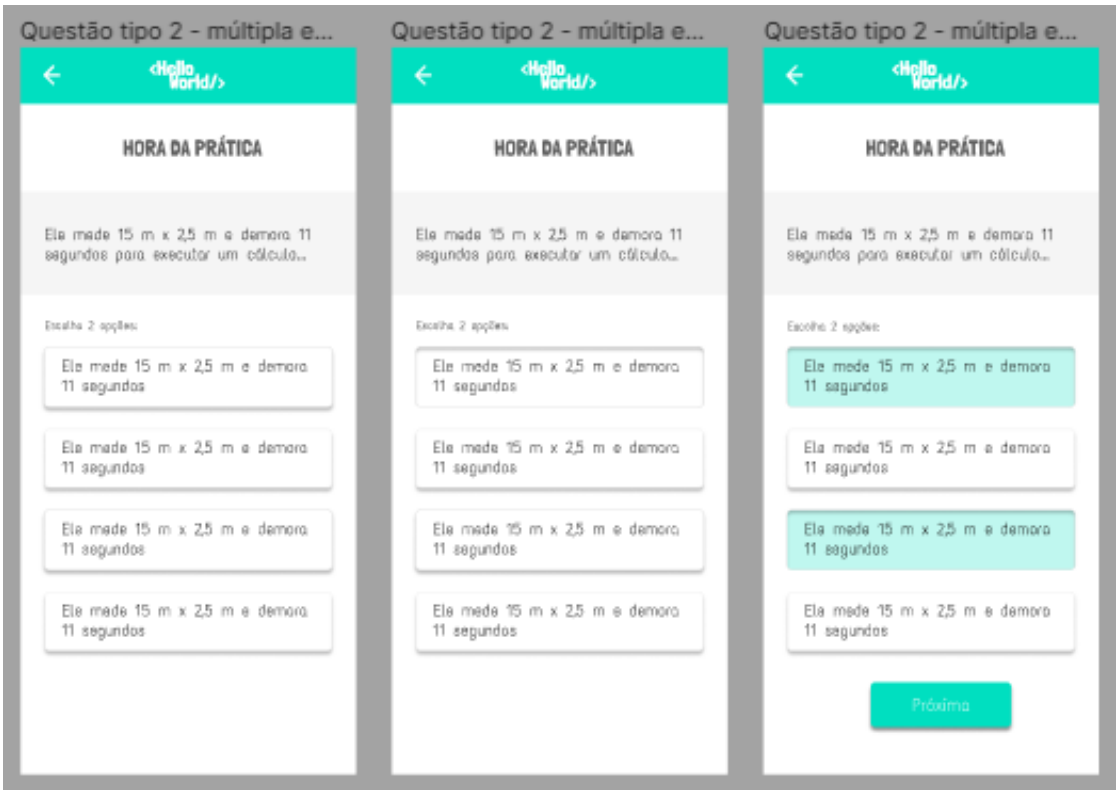
Figura 83 – Carrossel de *Onboarding*

Fonte: a autora

Figura 84 – Aba *Home*, Sessão de Explicação e Questão de Resposta Única

Fonte: a autora

Figura 85 – Questão de Múltipla Escolha

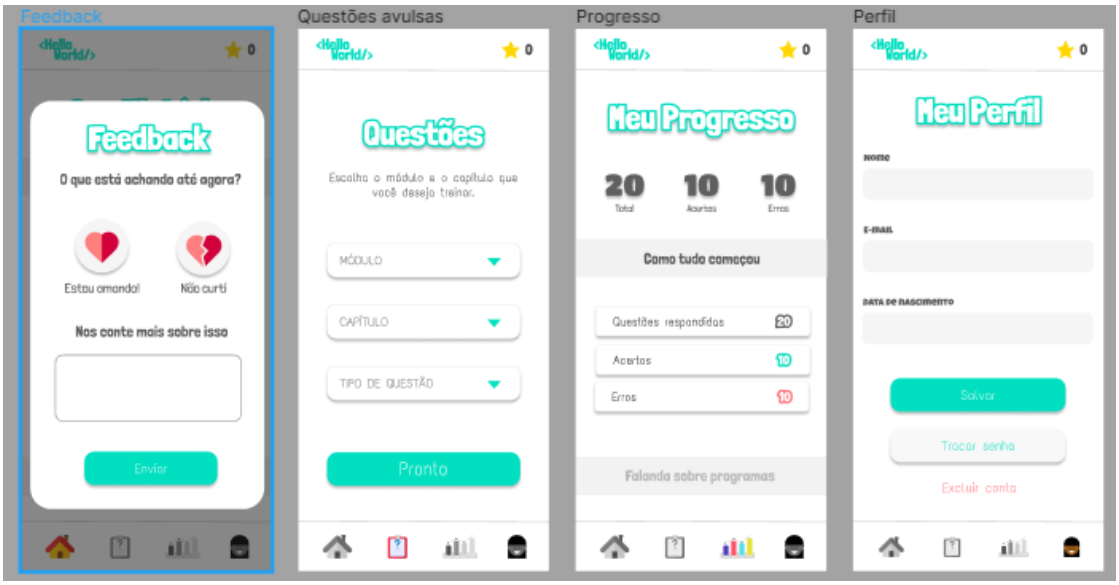


Fonte: a autora

Figura 86 – Questão de Lacuna

Fonte: a autora

Figura 87 – Telas de Pesquisa de Satisfação, Questões Avulsas, Relatório e Perfil



Fonte: a autora

APÊNDICE D – Códigos de Integração Contínua

D.1 Aplicação Web e API

Como falado anteriormente, há apenas uma esteira nessas aplicações. Ela é iniciada quando ocorre um *push* para a *branch master*.

A única diferença entre a esteira do servidor e do cliente, é o nome do contêiner dentro do *Github Packages*.

```
name: Production Continuous Delivery

on:
  push:
    branches:
      - master

jobs:
  docker-build-push:
    name: Push Docker image to GitHub Packages
    runs-on: ubuntu-latest
    permissions:
      contents: read
      packages: write
    steps:
      - name: Checkout
        uses: actions/checkout@v2
      - name: Log in to GitHub Docker Registry
        uses: docker/login-action@v1
        with:
          registry: docker.pkg.github.com
          username: ${{ github.actor }}
          password: ${{ secrets.GITHUB_TOKEN }}
      - name: Build container image
```

```
      uses: docker/build-push-action@v2
      with:
        push: true
        tags: |
          docker.pkg.github.com/helloworld-repo/helloworld-web/app:
              latest
          # Para API:
          # [...]/helloworld-repo/helloworld-api/server:latest

deploy:
  name: Deploy package to DigitalOcean
  needs: docker-build-push
  runs-on: ubuntu-latest
  steps:
    - name: Deploy package to DigitalOcean
      uses: appleboy/ssh-action@master
      env:
          GITHUB_USERNAME: ${{ github.actor }}
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
      with:
        host: ${{ secrets.PRODUCTION_IP }}
        username: ${{ secrets.PRODUCTION_USERNAME }}
        password: ${{ secrets.PRODUCTION_PASSWORD }}
        envs: GITHUB_USERNAME, GITHUB_TOKEN
        script: |
          cd hello-world
          docker login docker.pkg.github.com -u $GITHUB_USERNAME -p
              $GITHUB_TOKEN
          docker pull docker.pkg.github.com/helloworld-repo/
              helloworld-web/app:latest
          # Para API:
          # [...]/helloworld-repo/helloworld-api/server:latest
          docker-compose -f docker-compose.prod.yml up -d --build
```

D.2 Aplicação Móvel

D.2.1 Esteira de Produção

A esteira de produção para aplicação móvel, possui as seguintes etapas:

- Recuperar os *commits* desde a última *tag* gerada e criar o CHANGELOG;
- Criar uma release com os arquivos do repositório compactado;
- adicionar o CHANGELOG à descrição da *release*;
- Gerar *build* da aplicação, e
- Adicionar o arquivo com extensão ".apk"à *release*.

```
name: Create Release
on:
  push:
    tags:
      - "prod-[0-9]+.[0-9]+.[0-9]+"

jobs:
  deploy_prod:
    name: Deploy To Production
    runs-on: ubuntu-latest
    outputs:
      releaseChannel:
        ${{ steps.releaseChannel.outputs.releaseChannel }}
      latestBinaryVersion:
        ${{ steps.latestBinaryVersion.outputs.version }}
    steps:
      - uses: actions/checkout@v2
        with:
          ref: ${{ github.head_ref }}
      - name: Fetch Tags
        run: |
          git fetch --prune --unshallow --tags -f
      - uses: actions/setup-node@v1
        with:
```

```
        node-version: 12.x
    - uses: expo/expo-github-action@v5
      with:
        expo-packager: npm
        expo-username: ${{ secrets.EXPO_CLI_USERNAME }}
        expo-password: ${{ secrets.EXPO_CLI_PASSWORD }}
        expo-cache: true
    - uses: rlespinasse/github-slug-action@v3.x
    - name: Generate Release Channel
      id: releaseChannel
      run: |
        RELEASE_CHANNEL=$(echo ${{ env.GITHUB_REF_SLUG }} |
        sed -r 's/\.[0-9]+\.[0-9]+$//')
        echo
          "::set-output name=releaseChannel::$RELEASE_CHANNEL"
    - name: Install Packages
      run: npm install
    - name: Get Latest Binary Version
      id: latestBinaryVersion
      run: |
        RELEASE_TAG=$(echo $(git describe --tags --abbrev=0))
        echo "::set-output name=version::${RELEASE_TAG#*-}"
    - name: Echo Version Details
      run: |
        echo Build number is $GITHUB_RUN_NUMBER
        echo Latest release is
        ${{ steps.latestBinaryVersion.outputs.version }}
    - name: Expo Publish Channel
      run: expo publish --non-interactive --release-channel=
          ${{ steps.releaseChannel.outputs.releaseChannel }}

create_release:
  name: Create Release
  needs: deploy_prod
  runs-on: ubuntu-latest
  steps:
    - uses: actions/checkout@v2
    - uses: rlespinasse/github-slug-action@v2.x
    - name: Generate Changelog
```

```
      id: changelog
      uses: metcalfc/changelog-generator@v0.4.0
      with:
        myToken: ${{ secrets.GITHUB_TOKEN }}
    - name: Creating Release
      uses: ncipollo/release-action@v1
      with:
        body: |
          Changes in this Release:
          ${{ steps.changelog.outputs.changelog }}
        token: ${{ secrets.GITHUB_TOKEN }}
        name: Release ${{ env.GITHUB_REF_SLUG }}
        allowUpdates: true

build_android:
  needs: [deploy_prod, create_release]
  runs-on: ubuntu-latest
  steps:
    - uses: actions/checkout@v2
    - uses: rlespinasse/github-slug-action@v3.x
    - uses: expo/expo-github-action@v5
      with:
        expo-packager: npm
        expo-username: ${{ secrets.EXPO_CLI_USERNAME }}
        expo-password: ${{ secrets.EXPO_CLI_PASSWORD }}
        expo-cache: true
    - name: Install Packages
      run: npm install
    - name: Build Android Release
      env:
        APP_BUILD_VERSION: ${{ github.run_number }}
        APP_BINARY_VERSION:
          ${{needs.deploy_prod.outputs.latestBinaryVersion}}
        API_URL: ${{ secrets.API_URL }}
      run: |
        expo build:android --release-channel=
          ${{ needs.deploy_prod.outputs.releaseChannel }}
              > buildLogAndroid.txt
        cat buildLogAndroid.txt
```

```
      - name: Parse Asset URL
        id: androidUrl
        run: |
          ASSET_URL=$(cat buildLogAndroid.txt | tail |
            egrep -o 'https://expo.dev/artifacts/[^ ]+')
          echo The android url is $ASSET_URL
          echo "::set-output name=assetUrl::$ASSET_URL"
      - name: Download APK Asset
        run: wget -O helloworld-${{ env.GITHUB_REF_SLUG }}.apk
            ${{ steps.androidUrl.outputs.assetUrl }}
      - name: Upload Release Asset
        uses: svenstaro/upload-release-action@v2
        with:
          repo_token: ${{ secrets.GITHUB_TOKEN }}
          file: ./helloworld-${{ env.GITHUB_REF_SLUG }}.apk
          asset_name: helloworld-${{ env.GITHUB_REF_SLUG }}.apk
          tag: ${{ github.ref }}
```

D.2.2 Esteira de Homologação

Essa esteira inicia-se quando ocorre um *push* para a *branch development*, então é gerado um QRCode pela Expo, que pode ser usado para simular um ambiente de produção. Essa esteira não produz um arquivo com extensão ".apk", utiliza-se o próprio aplicativo da Expo para testar, bastando apenas apontar a câmera para o o QRCode gerado para iniciar a aplicação.

```
name: Deploy Staging

on:
  push:
    branches:
      - development

jobs:
  deploy_staging:
    name: Deploy to Staging
```

```
runs-on: ubuntu-latest
steps:
  - uses: actions/checkout@v2
  - uses: actions/setup-node@v1
    with:
      node-version: 12.x
  - uses: expo/expo-github-action@v5
    with:
      expo-packager: npm
      expo-username: ${{ secrets.EXPO_CLI_USERNAME }}
      expo-password: ${{ secrets.EXPO_CLI_PASSWORD }}
      expo-cache: true
  - name: Install Packages
    run: npm install
  - name: Expo Publish Channel
    run: expo publish --non-interactive
        --release-channel staging
  - name: Add Comment To PR
    uses: mshick/add-pr-comment@v1
    env:
      GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
      EXPO_PROJECT: "@gabimsilva2010/helloworld-app"
      API_URL: ${{ secrets.API_URL }}
    with:
      message: |
        ## Application
        ![Expo QR](https://api.qrserver.com/v1/
            create-qr-code/?size=250x250&data=exp://exp.host/
            ${{ env.EXPO_PROJECT }}?
            release-channel=pr${{ github.event.number }})

        Published to https://exp.host/
            ${{ env.EXPO_PROJECT }}?
            release-channel=pr${{ github.event.number }}
```

APÊNDICE E – Tabelas de Rastreamento

E.1 Rastreamento de Requisitos

Tabela 15 – Rastreamento de Modificações nos Requisitos

Data	**Versão**	**Descrição**
04/07/21	1.0	- Criação da tabela geral de requisitos
10/07/21	2.0	- Alinhamento com funcionalidades do cronograma - Removendo requisitos repetidos - Adicionando novos requisitos elicitados pelo questionário
19/07/21	2.1	- Removendo requisito de excluir questão
29/07/21	2.2	- Adição de requisitos relacionados à existência de turmas (WEB18, WEB19 e WEB20) - Edição do requisito WEB5 para incluir turma
19/03/22	2.3	- Removendo requisitos sobre a Play Store - Removendo requisito de notificações diárias

E.2 Rastreamento do Cronograma

Tabela 16 – Rastreamento de Alterações no Cronograma

Data	**Versão**	**Descrição**
09/07/2021	1.0.0	- Criação do documento
12/07/2021	1.1.2	- Adicionando tabela de rastreamento - Priorizando *deploy* do app
16/07/2021	1.1.3	- Adicionando semanas de preparação da apresentação
19/07/2021	1.2	- Removendo requisito de excluir questão
29/07/2021	1.3	- Adicionando requisitos de turma no cronograma
24/09/2021	1.4	- Aumentando o tempo de desenvolvimento das questões - Removendo modo *dark* e *light*
17/10/2021	1.4.1	- Deixando apenas uma atividade de desenvolvimento por semana para a segunda metade da tabela
12/01/2022	1.4.2	- Movendo criação de diagrama de pacotes WEB para ser feito após o desenvolvimento da aplicação
07/02/2022	1.5	- Removendo requisito de importar questão
26/02/2022	1.6	- Removendo requisito de notificações *push*
14/03/2022	1.6.1	- Movendo tarefas das semanas 17/04 e 24/04 para 1 semana antes