



UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE QUIXADÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO
MESTRADO ACADÊMICO EM COMPUTAÇÃO

FRANCISCO GILBERTO LIMA PAIVA

**UMA ASSISTENTE VIRTUAL PARA APOIAR USUÁRIOS FINAIS NAS ATIVIDADES
DE ENGENHARIA DE SOFTWARE UTILIZANDO O APP INVENTOR**

QUIXADÁ

2022

FRANCISCO GILBERTO LIMA PAIVA

UMA ASSISTENTE VIRTUAL PARA APOIAR USUÁRIOS FINAIS NAS ATIVIDADES DE
ENGENHARIA DE SOFTWARE UTILIZANDO O APP INVENTOR

Dissertação apresentada ao Curso de Mestrado Acadêmico em Computação do Programa de Pós-Graduação em Computação do Campus de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em computação. Área de Concentração: Ciência da Computação

Orientadora: Profa. Dra. Ingrid Teixeira Monteiro

QUIXADÁ

2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

- P168a Paiva, Francisco Gilberto Lima.
Uma assistente virtual para apoiar usuários finais nas atividades de Engenharia de Software utilizando o App Inventor / Francisco Gilberto Lima Paiva. – 2022.
100 f. : il. color.
- Dissertação (mestrado) – Universidade Federal do Ceará, Campus de Quixadá, Programa de Pós-Graduação em Computação, Quixadá, 2022.
Orientação: Profa. Dra. Ingrid Teixeira Monteiro.
1. Desenvolvimento de usuário final. 2. Assistente virtual inteligente. 3. Aplicativos móveis. 4. Engenharia de software. I. Título.

CDD 005

FRANCISCO GILBERTO LIMA PAIVA

UMA ASSISTENTE VIRTUAL PARA APOIAR USUÁRIOS FINAIS NAS ATIVIDADES DE
ENGENHARIA DE SOFTWARE UTILIZANDO O APP INVENTOR

Dissertação apresentada ao Curso de Mestrado Acadêmico em Computação do Programa de Pós-Graduação em Computação do Campus de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em computação. Área de Concentração: Ciência da Computação

Aprovada em: __/__/__

BANCA EXAMINADORA

Profa. Dra. Ingrid Teixeira Monteiro (Orientadora)
Universidade Federal do Ceará (UFC)

Profa. Dra. Andréia Libório Sampaio
Universidade Federal do Ceará (UFC)

Profa. Dra. Carla Ilane Moreira Bezerra
Universidade Federal do Ceará (UFC)

Profa. Dra. Marcelle Pereira Mota
Universidade Federal do Pará (UFPA)

À Deus, que sempre esteve ao meu lado. À
minha minha família, que me inspira todos os
dias

AGRADECIMENTOS

Agradeço primeiramente a Deus, que me permitiu chegar nesse momento e sem ele nada seria possível.

Aos meus pais, irmãos e avô, que sempre me apoiaram durante minhas caminhadas e me incentivaram a buscar esse sonho, tentando ao máximo impedir que preocupações externas me atrapalhasse. À minha companheira Leiliane, que mostrou o caminho quando eu me senti mais perdido e que me ajudou em todas as etapas dessa jornada. Agradeço pela paciência e apoio e por se fazerem presentes mesmo quando estavam distantes fisicamente.

A minha orientadora Ingrid Monteiro, pelo incentivo, paciência e conhecimentos que foram de fundamental importância para chegar até aqui. Sei que dei muito trabalho, mas como seu primeiro orientando de mestrado devo dizer, você fez um ótimo trabalho.

Aos meus amigos do programa de pós-graduação, em especial a Warlles e José Neto, que me acompanharam por muitas madrugadas de estudo e pelas muitas caronas e partidas de futebol assistidas. Aos meus amigos externos ao programa pelas sugestões, revisões e contribuições para este trabalho.

As professoras Andréia Libório, Carla Ilane e Marcelle Pereira pela participação na banca avaliadora e pelas contribuições extremamente relevantes durante nossos encontros. Aos demais professores e professoras do Programa de Pós-Graduação em Computação da UFC Campus Quixadá por cada conhecimento transmitido.

"Tudo que você faz na sua vida só será legendário se seus amigos estiverem lá para verem."

(Barney Stinson, *How I Met Your Mother*, 2005)

RESUMO

Um dos focos da área de Interação Humano-Computador nos últimos anos tem sido pesquisas voltadas para o campo de *End-User Development* (EUD). Diversos trabalhos vêm sendo realizados para tentar apoiar esses usuários que não são desenvolvedores profissionais a criarem seus próprios sistemas. Conseqüentemente, assim como ocorre em sistemas desenvolvidos por profissionais, os artefatos criados por usuários finais também precisam apresentar um certo nível de qualidade. A fim de contribuir com isso, pesquisas em *End-User Software Engineering* (EUSE) propõem adaptar práticas da Engenharia de Software (ES) tradicional ao desenvolvimento pelo usuário final, de modo que suas prioridades não sejam modificadas. Porém, introduzir a consciência de adotar boas práticas no usuário final não é tão simples, devido ao seu nível de conhecimento ou à sua própria intenção de empregar o esforço necessário para isso. Diante disso, este trabalho apresenta o percurso em direção ao desenvolvimento de uma assistente virtual com interface por voz integrada ao *App Inventor*, uma plataforma para o desenvolvimento por usuários finais de aplicativos por meio de programação visual com blocos, visando auxiliar os usuários finais a praticarem atividades de EUSE em um ambiente de desenvolvimento EUD. Foram realizados estudos com usuários finais utilizando o *App Inventor* assim como uma inspeção das telas dessa plataforma. Os conhecimentos resultantes serviram como base para a construção de uma assistente virtual que foi avaliada posteriormente com novas avaliações envolvendo usuários finais. Os resultados permitiram identificar como as atividades de ES podem ser abordadas pelos usuários finais e como esse processo pode ser apoiado por uma ferramenta. Por fim, discutimos como os usuários consideraram positiva a ajuda dessa assistente virtual e quais seus pontos a serem melhorados a partir de suas perspectivas. Abordamos também, como a plataforma do *App Inventor* consegue apoiar as atividades de requisitos, design, reuso, testes e debug, além de demonstrar algumas lacunas que podem vir a ser melhoradas nesse processo de apoio a EUSE.

Palavras-chave: Desenvolvimento pelo usuário final; Assistente virtual inteligente; Aplicativos móveis; Engenharia de software.

ABSTRACT

One of the focuses of the Human-Computer Interaction area in recent years has been research focused on the field of End-User Development (EUD). Several works have been carried out to try to support users who are not professional developers to create their own systems. Consequently, as in systems developed by professionals, artifacts created by end users also need to present a certain level of quality. In order to contribute to this, research in End-User Software Engineering (EUSE) proposes to adapt Software Engineering (ES) traditional practices to end user development, so that their priorities are not changed. However, introducing awareness of best practices to the end-user is not so simple, due to their level of knowledge or their own intention to use the necessary effort for this. Therefore, this work presents the path towards the development of a virtual assistant with a voice interface integrated to App Inventor, a platform for the development of applications by end users through visual programming with blocks, aiming to help end users to practice EUSE activities in a EUD development environment. Studies were carried out with end users using App Inventor as well as an inspection of the screens of this platform. The resulting knowledge served as a basis for the construction of a virtual assistant that was later evaluated with new evaluations involving end users. The results allowed us to identify how higher education activities can be addressed by end users and how this process can be supported by a tool. Finally, we discuss how users considered the help of this virtual assistant to be positive and what points they need to improve from their perspectives. We also address how the App Inventor platform can support requirements, design, reuse, testing and debug activities, in addition to demonstrating some gaps that can be improved in this EUSE support process.

Keywords: End-User Development; Virtual Assistant; Mobile apps; Software Engineering.

LISTA DE FIGURAS

Figura 1 – Tela do <i>Designer</i> do <i>App Inventor</i>	22
Figura 2 – Tela do <i>Blocks Editor</i> do <i>App Inventor</i>	23
Figura 3 – Metodologia de pesquisa	30
Figura 4 – Comunicação entre pesquisador e participante	33
Figura 5 – Representação do Ambiente da avaliação (imagens meramente ilustrativas) .	33
Figura 6 – Representação do funcionamento do <i>ngrok</i>	36
Figura 7 – Exemplo do participante reformulando um comando	39
Figura 8 – Telas desenvolvidas pelo participante P1	39
Figura 9 – Blocos desenvolvidos pelo participante P1	40
Figura 10 – Telas desenvolvidas pelo participante P2 - Parte 1	42
Figura 11 – Tela desenvolvida pelo participante P3	43
Figura 12 – Exemplo de avisos em destaque nos blocos	44
Figura 13 – Proporção dos achados durante a etapa de análise da interface	46
Figura 14 – A interface do <i>App Inventor</i> mostrando a tela do <i>Designer</i> com elementos em destaque	47
Figura 15 – A interface do <i>App Inventor</i> mostrando a tela do <i>Blocks Editor</i> com elementos em destaque	47
Figura 16 – Caixa de ajuda sobre o componente	49
Figura 17 – Menu Projetos	50
Figura 18 – Menu Compilar	51
Figura 19 – Menu Conectar	52
Figura 20 – Mensagens sobre falhas no código	52
Figura 21 – Estrutura do assistente virtual	53
Figura 22 – Interface modificada do <i>App Inventor</i>	55
Figura 23 – Estrutura do Agente de Diálogos	56
Figura 24 – Arquitetura do Rasa Framework	58
Figura 25 – Exemplo do arquivo de intenções	59
Figura 26 – Exemplo de possível conversa entre usuário e assistente	60
Figura 27 – Exemplo do arquivo <i>domain.yml</i>	60
Figura 28 – Exemplo do arquivo de respostas	63
Figura 29 – Diálogo com <i>bot</i> por texto	63

Figura 30 – Tutorial de uso da assistente virtual	64
Figura 31 – Resultado do questionário com base no fator de satisfação	67
Figura 32 – Resultado do questionário com base no fator de suporte	68
Figura 33 – Resultado do questionário com base no fator de conhecimento	69
Figura 34 – Relação de conhecimentos adquiridos pelos usuários	70
Figura 35 – Autoavaliação de práticas conhecidas previamente	71
Figura 36 – Categorias relacionadas aos pontos a melhorar	72
Figura 37 – Tela representando o momento de espera entre uma comando do usuário e o termino da ação	89
Figura 38 – Telas desenvolvidas pelo participante P1	95
Figura 39 – Telas desenvolvidas pelo participante P2	95
Figura 40 – Telas desenvolvidas pelo participante P3	96
Figura 41 – Tela desenvolvida pelo participante P4	96
Figura 42 – Telas desenvolvidas pelo participante P5	97
Figura 43 – Tela desenvolvida pelo participante P6	97
Figura 44 – Telas desenvolvidas pelo participante P7	98
Figura 45 – Telas desenvolvidas pelo participante P8	99
Figura 46 – Tela desenvolvida pelo participante P9	99
Figura 47 – Tela desenvolvida pelo participante P10	100

LISTA DE TABELAS

Tabela 1 – Diferenças qualitativas entre a engenharia de software profissional e do usuário final	19
Tabela 2 – Comparação entre os trabalhos relacionados e o atual	28
Tabela 3 – Visão geral dos estudos 1 e 2	37
Tabela 4 – Conhecimentos dos participantes em relação aos temas abordados	38
Tabela 5 – Relação de artefatos criados no Estudo 1	45
Tabela 6 – Relação entre área x dimensão abordada	54
Tabela 7 – Exemplos do formato de uma intenção	58
Tabela 8 – Exemplo do formato de um <i>story</i>	61
Tabela 9 – Exemplo de diálogo sobre requisitos	61
Tabela 10 – Exemplo de diálogo sobre <i>design</i>	62
Tabela 11 – Exemplo de diálogo sobre reuso	62
Tabela 12 – Exemplo de diálogo sobre testes	62
Tabela 13 – Exemplo de diálogo sobre debug	62
Tabela 14 – Visão geral do estudo 3	65
Tabela 15 – Relação de artefatos criados e dimensões abordadas no Estudo 3	66
Tabela 16 – Relação de uso da assistente	66
Tabela 17 – Questões de pesquisa	74

SUMÁRIO

1	INTRODUÇÃO	14
2	REFERENCIAL TEÓRICO	17
2.1	<i>End-User Development (EUD)</i>	17
2.2	<i>End-User Software Engineering (EUSE)</i>	18
2.3	<i>App Inventor</i>	21
2.4	Interface de Linguagem Natural	24
3	TRABALHOS RELACIONADOS	26
4	METODOLOGIA	29
4.1	Objetivos e Questões de pesquisa	29
4.2	Procedimentos	30
4.3	Etapa 1 - Wizard of Oz	31
4.4	Etapa 2 - Inspeção do App Inventor	34
4.5	Etapa 3 - Desenvolvimento da assistente virtual	34
4.6	Etapa 4 - Avaliação da assistente virtual	35
5	ESTUDOS INICIAIS	37
5.1	Visão geral dos resultados iniciais	37
5.2	Estudo 1 (Wizard of Oz)	37
5.3	Estudo 2 (Inspeção do App Inventor)	45
6	DESENVOLVIMENTO DA ASSISTENTE VIRTUAL	53
6.1	Melhorias no planejamento da assistente virtual	53
6.2	Implementação da estrutura do assistente virtual	54
6.2.1	<i>Modificação na interface do App Inventor</i>	54
6.2.2	<i>Agente de Diálogos</i>	56
6.2.3	<i>Gerenciador de Diálogos</i>	57
7	ESTUDO FINAL	65
8	DISCUSSÃO DOS RESULTADOS	74
9	CONSIDERAÇÕES FINAIS	81
9.1	Contribuições da pesquisa	81
9.2	Trabalhos futuros	82
9.3	Limitações conhecidas	82

REFERÊNCIAS	84
APÊNDICES	88
APÊNDICE A–TERMO DE CONSENTIMENTO LIVRE E ESCLA- RECIDO	88
APÊNDICE B–IMAGEM CRIADA PARA O ESTUDO 1	89
APÊNDICE C–CENÁRIO DO ESTUDO	90
APÊNDICE D–FORMULÁRIO DE RECRUTAMENTO	91
APÊNDICE E–QUESTIONÁRIO UTILIZADO PARA AVALIAÇÃO DE USO DA ASSISTENTE VIRTUAL	92
APÊNDICE F–ARTEFATOS DO ESTUDO 3	95

1 INTRODUÇÃO

Desde o primeiro programa de computador digital desenvolvido na década de 1940 até a indústria atual de software que cresce cada vez mais, a programação de computadores tornou-se uma habilidade técnica de milhões de pessoas. À medida que o número desses profissionais cresce, outra segunda tendência começou a tomar forma. As pessoas começaram a explorar softwares de uma maneira mais pessoal e mais direcionada a resolver problemas no trabalho ou em outras atividades diárias. É nesse contexto que surge a área de *End-User Development* (EUD).

O que torna esses "*end-user developers*" diferentes de seus colegas profissionais são seus objetivos: os profissionais são pagos para construir e manter o software ao longo do tempo; os usuários finais, por outro lado, escrevem programas para apoiar algum objetivo em seus próprios domínios de especialização (KO *et al.*, 2011). Para isso, algumas ferramentas se tornaram bastante populares por serem usadas por programadores não profissionais, entre elas: Wordpress¹, MIT App Inventor², Scratch³, Arduino⁴, *If This Then That* (IFTTT)⁵ e uma variedade de ferramentas de mineração de dados, análises e visualização, como o OpenRefine⁶ e o R Project⁷ para computação estatística. Isso é um exemplo de como a EUD vem ganhando relevância no contexto das práticas de pesquisa e demais áreas. Com essa expansão, a qualidade do software não profissional tornou-se crítica, pois o produto gerado nesse processo é frequentemente compartilhado e distribuído por terceiros e seu uso pode afetar outras pessoas (MONTEIRO *et al.*, 2017).

As necessidades dos usuários finais por trás desses sistemas desenvolvidos faz com que tais sistemas demandem uma atenção maior para o artefato produzido. Com base nisso, a *End-User Software Engineering* (EUSE) surgiu devido à preocupação com erros e outras questões de qualidade nos sistemas produzidos por *end-users*. Novas práticas começaram a ser estudadas no campo de EUD em conjunto com práticas da Engenharia de Software (ES) tradicional como uma forma de colaborar com a melhoria da qualidade desses sistemas (KO *et al.*, 2011; BARRICELLI *et al.*, 2019).

¹ <https://wordpress.com>

² <https://appinventor.mit.edu/>

³ <https://scratch.mit.edu/>

⁴ <https://www.arduino.cc/>

⁵ <https://ifttt.com/>

⁶ <https://openrefine.org/>

⁷ <https://www.r-project.org/>

Isso é importante uma vez que, evidências do paradigma de planilhas⁸, uma das mais amplamente usadas das linguagens de programação do usuário final, constataram que usuários finais programadores são extremamente propensos a introduzir falhas em seus programas (PANKO, 1998). Esse é um grave problema, porque, embora alguns programas de usuários finais sejam simplesmente explorações e cálculos de bloco de notas, outros podem ser muito importantes para seu sustento pessoal ou comercial, como programas de cálculo de imposto de renda, páginas da web de comércio eletrônico e previsão financeira (BURNETT *et al.*, 2006).

Paralelo a isso, a tecnologia por voz está cada vez mais presente no cotidiano das pessoas. Os veículos estão explicando como chegar ao supermercado mais próximo, os termostatos estão sendo instruídos a aumentar o calor e os micro-ondas estão sendo comandados para preparar as refeições das pessoas (WEISE, 2018). Empresas, como Amazon, Google e Apple, desenvolveram dispositivos de voz para facilitar o acesso a dados, automação e interação natural com a tecnologia. De acordo com uma pesquisa de 2022, quase 35% dos adultos americanos possuem pelo menos um alto-falante inteligente, algo em torno de 100 milhões de pessoas atualmente. Número que em 2021 foi de aproximadamente 82 milhões de usuários, mostrando o contínuo aumento no uso desses dispositivos inteligentes (RADIO, 2022).

Com base nisso, e buscando facilitar a forma de programação e melhorar as práticas de desenvolvimento para usuários finais, este trabalho visa prover uma ferramenta capaz de estimular atividades de EUSE por meio de interface de voz para ambientes EUD. O foco do nosso estudo é a plataforma do *App Inventor*, uma plataforma EUD com cerca de 14,9 milhões de usuários registrados até o início de 2022 (INVENTOR, 2022). Em nosso percurso investigativo, realizamos estudos de base qualitativa, cujos resultados mostraram como a plataforma do *App Inventor* disponibiliza recursos para que o usuário final empregue atividades de EUSE em seus projetos e como esses usuários podem trabalhar essas atividades ao longo do desenvolvimento. Além disso, desenvolvemos e avaliamos a inclusão de uma assistente virtual na plataforma para apoiar ainda mais o processo de Engenharia de Software desses usuários finais.

Este trabalho está organizado da seguinte forma. O próximo capítulo apresenta os conceitos básicos associados à pesquisa. O capítulo 3 descreve os trabalhos relacionados. O capítulo 4 dedica-se à metodologia da pesquisa, apresentando a abordagem utilizada. O capítulo 5 capítulo expõe os resultados obtidos nos primeiros estudos e que serviram de base para a criação da assistente virtual. No capítulo 6 descrevemos como ocorreu o desenvolvimento da

⁸ Formato utilizado por planilhas no qual os dados são apresentados em tabelas que podem ter seu comportamento definido por meio de equações

assistente e o planejamento para as avaliações. No capítulo 7 apresentamos os resultados obtidos na avaliação. No capítulo 8 realizamos algumas discussões sobre os dados apresentados nos capítulos anteriores. Por fim, o capítulo 9 apresenta as considerações finais e aborda um pouco as limitações e próximos passos para este trabalho.

2 REFERENCIAL TEÓRICO

Esta seção apresenta o *background* necessário em diferentes conceitos e tecnologias e que são importantes ao longo deste trabalho.

2.1 *End-User Development* (EUD)

Uma inevitável consequência da inclusão dos computadores nas mais diversas atividades cotidianas é o crescimento do número de usuários que assumem uma posição ativa em relação ao consumo de sistemas computacionais. Muitas vezes, a interface original de um sistema não apoia as intenções reais de uso de um determinado usuário. Então, esse usuário é conduzido para atividades como customização, personalização e extensão do sistema, até o extremo de, no caso de alguns, criar um software específico que atenda a suas necessidades. Em todos os casos, embora em diferentes proporções, o usuário atua como produtor de tecnologia (MONTEIRO, 2015).

Ao longo do tempo, as pesquisas relacionadas a este tipo especial de atividade foi modificando seu nome conforme níveis mais abrangentes vinham sendo considerados. Denominada inicialmente de *End-User Programming* (EUP), essa área era vista exclusivamente como atividades de programação (codificação). Ko *et al.* (2011) definem EUP como “*programming to achieve the result of a program primarily for personal, rather than public use.*” (p.21). Pois, segundo os autores, um programa desenvolvido desta forma não tem como principal objetivo ser destinado ao uso por um grande número de usuários com necessidades variadas e específicas. Percebe-se então o contraste entre a EUP e a programação “tradicional”, que, segundo os autores, está na finalidade de produzir código para o uso de outras pessoas, como ocorre tradicionalmente, enquanto a EUP visava a construção de sistemas para atender as próprias necessidades do usuário programador. Ko *et al.* (2011) ainda alertam para não confundir a programação do usuário final com a inexperiência. Por exemplo, desenvolvedores profissionais com décadas de experiência podem se engajar na programação do usuário final escrevendo código para uso pessoal, sem intenção de compartilhar seu programa com outras pessoas.

Posteriormente, surgiu a expressão *End-User Development* (EUD), utilizada para definir uma área mais abrangente, com outras atividades além da codificação abordada na EUP. Lieberman *et al.* (2006) expressam uma definição mais detalhada, afirmando que “*EUD can be defined as a set of methods, techniques, and tools that allow users of software systems, who*

are acting as non-professional software developers, at some point to create, modify, or extend a software artifact". A partir desta visão, compreende-se que os usuários finais estão envolvidos no *design* do sistema, mas podem ou não estar envolvidos na codificação real.

2.2 End-User Software Engineering (EUSE)

Seja desenvolvendo planilhas ou utilizando bases de dados, escrevendo fórmulas ou *queries* de buscas para facilitar atividades diárias (SCAFFIDI *et al.*, 2005), gerando simulações no MATLAB (GULLEY, 2006) ou *scripts* para processar dados científicos (FISCHER *et al.*, 2009), em todos esses casos, usuários finais frequentemente inserem erros que eles não têm conhecimento ou são difíceis de descobrir. Isso começou a levantar dúvidas da comunidade de engenharia de software sobre os valores da EUD e destacando o papel de desenvolvedores profissionais e qualificados (KO *et al.*, 2011).

À medida que os sistemas desenvolvidos por usuários finais começam a ser compartilhados para outras pessoas, surgiram preocupações com questões mais profundas de desenvolvimento. Produzir sistemas para terceiros implica que o usuário final terá que considerar cada vez mais as preocupações de engenharia de software para satisfazer restrições cada vez mais complexas e diversas (KO *et al.*, 2011). A partir disso, um outro termo surgiu, *End-User Software Engineering* (EUSE), definido como "*a highly integrated and incremental concept of software engineering support for end-users*" (BURNETT *et al.*, 2006).

Para Ko *et al.* (2011), a principal diferença entre a engenharia de software profissional e a EUSE é justamente a quantidade de atenção dada às questões de qualidade do software, envolvendo atividades como testes e manutenção de código. Tentar produzir um sistema de qualidade demanda tempo, por vezes, equipes e ferramentas bem estruturadas. O esforço empregado na engenharia de software profissional é muito maior, uma vez que o programa, muitas vezes, é destinado para uso por milhões de usuários. Por outro lado, a EUSE ainda envolve atividades sistemáticas que tratam de questões de qualidade de software, mas essas atividades são secundárias aos objetivos que o programa está ajudando a alcançar.

Devido a essa diferença de prioridades e à natureza oportunista do desenvolvimento pelo usuário final, as pessoas envolvidas nesse tipo de programação raramente têm tempo ou interesse em atividades sistemáticas e disciplinadas de engenharia de software (BRANDT *et al.*, 2008). Em seu trabalho, Segal (2007) resumiu as diferenças presentes na EUSE (Tabela 1), mostrando que ela pode ser caracterizada por sua natureza não planejada, implícita e oportu-

nista, devido principalmente às prioridades e intenções do programador (mas talvez também à inexperiência).

Tabela 1 – Diferenças qualitativas entre a engenharia de software profissional e do usuário final

Atividades de ES	Profissional de ES	End-user SE
Requisitos	explícito	implícito
Especificações	explícito	implícito
Reuso	planejado	não planejado
Verificação e Testes	cauteloso	excesso de confiança
Depuração	sistemático	oportunista

Fonte: Segal (2007).

Dadas essas diferenças, o desafio da pesquisa em EUSE é encontrar maneiras de incorporar atividades de engenharia de software no fluxo de trabalho existente dos usuários, sem exigir que as pessoas alterem substancialmente a natureza de seu trabalho ou suas prioridades.

A EUSE é uma área multidisciplinar que envolve ciência da computação, engenharia de software, interação humano-computador, educação, psicologia e outras disciplinas (KO *et al.*, 2011). Burnett (2014) cita que EUSE é uma área de pesquisa que visa criar novas formas de tecnologia que colaboram com o *end-user* para melhorar a qualidade final do seu software. Para isso, existem diversas dimensões a serem abordadas nesse área, incluindo ferramentas, paradigmas da linguagem, abordagens de pesquisa e assim por diante. Com base no foco desse trabalho, resolvemos abordar o estudo da EUSE em torno das atividades comumente listadas na literatura (KO *et al.*, 2011). São elas: Requisitos, *Design*, Reúso, Testes e *Debug*. Para cada uma delas, é apresentada uma discussão tentando apresentar o ponto de vista do *end-user*.

Em relação à atividade de **Requisitos**, Valente (2020) diz que os requisitos definem o que um sistema deve fazer e sob quais restrições. Sendo que quando um requisito está relacionado com o que um sistema deve fazer, ou seja, suas funcionalidades, eles são chamados de Requisitos Funcionais. Já os requisitos relacionados com as restrições são chamados de Requisitos Não-Funcionais. Valente (2020) ainda explica que Engenharia de Requisitos é o termo utilizado para nomear o conjunto de atividades que são destinadas a descobertas, análise, especificação e manutenção dos requisitos. As atividades relacionadas com a descoberta e entendimento dos requisitos de um sistema são chamadas de Elicitação de Requisitos, e, que uma boa especificação é obtida através de esforço e tempo empregados a fim de diminuir problemas futuros.

Enquanto isso, no contexto de EUSE, essa é uma questão completamente diferente. Como já mencionado, desenvolvedores não profissionais não estão acostumados a empregar

muito esforço nessa etapa. O desafio, neste caso, é induzir os usuários finais a um maior comprometimento com a identificação e compreensão das necessidades, seja para si próprio ou para outros usuários. Uma tarefa particularmente difícil, pois o usuário final geralmente desconhece toda a extensão de seus próprios requisitos (KO *et al.*, 2011).

Design refere-se às decisões sobre o comportamento interno do sistema, afetando confiabilidade, manutenibilidade, desempenho, entre outros fatores. Sendo o elo crítico entre o projeto e as atividades de requisitos, pois identifica os principais componentes estruturais de um sistema e os relacionamentos entre eles (SOMMERVILLE, 2011). Para Valente (2020), abordando o conceito de arquitetura, descreve que a arquitetura de um software inclui as decisões de projeto mais importantes em um sistema. Essas decisões são tão importantes que, uma vez tomadas, dificilmente poderão ser revertidas no futuro.

Entretanto, processos de *design* são aprendidos por profissionais por meio de experiências ou treinamento, o que não é o caso dos usuários finais. Uma maneira de trabalhar esta etapa é permitir que os usuários façam suas atividades como costumavam fazer, porém, introduzindo boas decisões de *design* através de catálogos de padrões e práticas recomendadas. Outra forma é restringir o que pode ser projetado baseado no conhecimento específico do domínio (KO *et al.*, 2011).

Segundo Sommerville (2011), os softwares mais modernos são construídos por meio da atividade de **Reúso**. Pacotes de componentes, APIs⁹, bibliotecas e trechos de código escritos por outras pessoas podem ser usados para construir programas e sistemas (KO *et al.*, 2011). Os principais motivos para reutilizar código são a redução do tempo de desenvolvimento, diminuição da taxa de erros por utilizar código já testado e melhoria da manutenibilidade do sistema (SOMMERVILLE, 2011). Um dos grandes desafios para incorporar o reúso no desenvolvimento não profissional é fazer o usuário final abstrair do código o que pode ser reusado, ou pelo menos entender que isso pode ser feito. Algumas alternativas de reúso são abordagens baseadas em *mashups* (CHOWDHURY, 2012; MASSA; SPANO, 2016) ou Linha de Produtos de Software (LPS) (TZEREMES; GOMAA, 2015; TZEREMES; GOMAA, 2018).

Nas atividades de **Verificação e Testes**, desenvolvedores garantem que um programa faz o que foi proposto e descobrem defeitos antes que o sistema chegue nos usuários finais (SOMMERVILLE, 2011). Através da execução de uma variedade de tipos de atividades fundamentais para evitar que erros e inconsistências cheguem aos usuários finais e causem prejuízos, podendo

⁹ Interface de Programação de Aplicação, cuja sigla API provém do Inglês *Application Programming Interface*.

ser feitos manualmente ou de forma automatizada (VALENTE, 2020).

Em relação a EUSE, talvez o principal problema nesse caso é o excesso de confiança por parte do usuário final na qualidade do seu software, isto é, por vezes eles não se dedicam a imaginar as falhas que podem ocorrer com seus sistemas (KO *et al.*, 2011). Para isso, umas das abordagens para ajudar no desenvolvimento é o apoio fornecido pelas próprias ferramentas de testes que auxiliam na execução de rotinas de testes para validar o que foi produzido. Isso vale tanto para usuários finais como para desenvolvedores profissionais. Em Ko *et al.* (2011) são comentadas algumas abordagens e ferramentas projetadas para abordar testes por meio da visualização de comportamentos dos programas.

Uma vez que a fase de verificação e testes detecta a presença de erros, a atividade de *Debug* é responsável por localizar e corrigir os problemas encontrados no código (SOMMERVILLE, 2011). Em ambientes profissionais, as próprias ferramentas de desenvolvimento apresentam recursos para depuração do código (PERSCHEID *et al.*, 2017). É particularmente difícil para usuários finais gerarem possíveis explicações para as falhas de um programa sem uma boa ferramenta (KO *et al.*, 2011). Algumas pesquisas apresentam abordagens para auxiliar o usuário final nesse desafio, por exemplo, na detecção de falhas em planilhas (HERMANS *et al.*, 2012; ABREU *et al.*, 2015).

Todos esses conceitos são importantes, pois é em torno deles que este trabalho se desenvolve. Neste caso, ao desenvolver uma ferramenta para EUSE, geralmente, foca-se em alguns desses pontos como principal objetivo, ou então, vai além e tenta-se trabalhar mais de uma atividade em conjunto para produzir uma ferramenta mais completa.

2.3 *App Inventor*

No passado, desenvolver tecnologias como aplicativos móveis ou sistemas autônomos só podia ser alcançado por engenheiros e cientistas da computação altamente especializados. Agora, entretanto, existem ferramentas como o *App Inventor*, um software web que permite desenvolver sistemas tecnológicos complexos, contando com mais de 14,9 milhões de usuários em 195 países diferentes e mais de 67 milhões de aplicações desenvolvidas (INVENTOR, 2022).

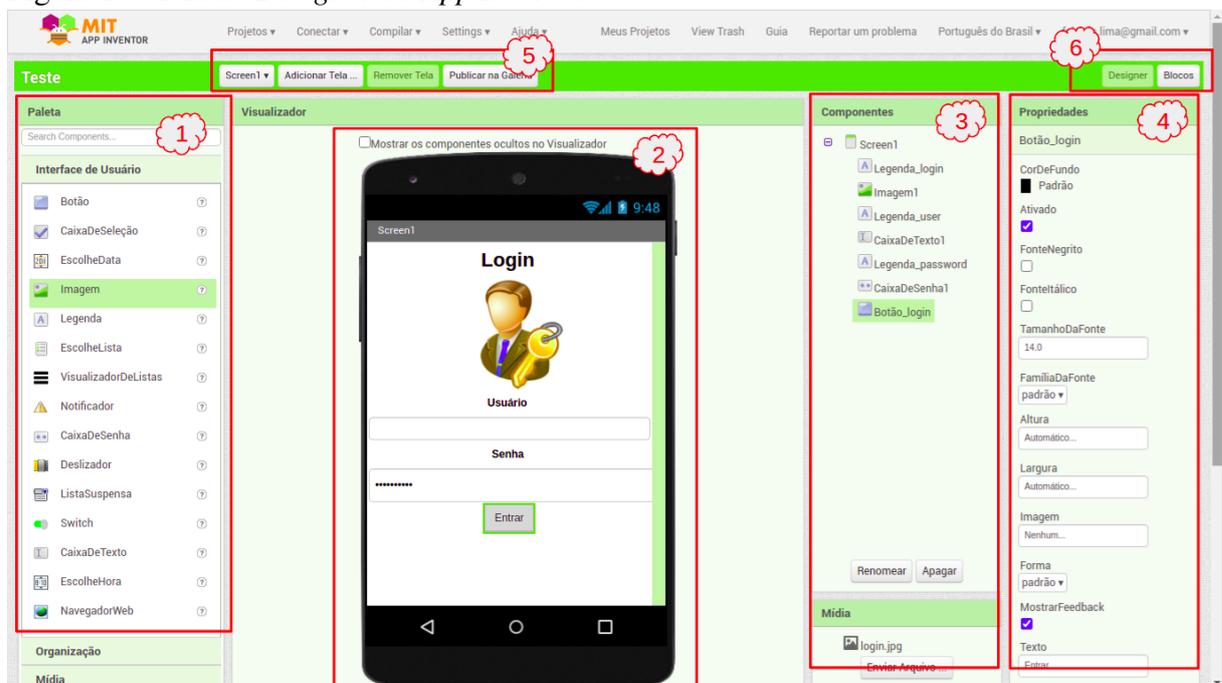
O *App Inventor* é uma plataforma que oferece uma linguagem visual de programação para criação de aplicativos para dispositivos móveis baseados no sistema *Android*. O *App Inventor* usa codificação baseada em blocos, na qual os usuários conectam blocos semelhantes a peças de quebra-cabeças para desenvolver programas totalmente funcionais. Segundo Gomes e

Melo (2013), a criação de aplicativos através do *App Inventor* é bastante intuitiva e não exige conhecimento prévio avançado em programação. Na plataforma, o desenvolvimento é baseado em componentes, os quais são manipulados por eventos, métodos e propriedades (e.g. *Button*: componente, *Button.Color*: propriedade, *whenButton.Clicked*: evento).

A plataforma possui duas janelas principais: o editor de *design* (*Designer*) e o editor de blocos (*Blocks Editor*). O *Designer* é uma interface no estilo *drag-and-drop*, na qual exibe diversos elementos de interface do usuário (UI) e uma área de trabalho para montagem da aplicação. Já o *Blocks Editor* é uma área onde os usuários devem descrever o comportamento do programa através da manipulação de blocos de código, como citado anteriormente.

A Figura 1 apresenta a janela do *Designer*, que é executada a partir do navegador e permite a criação da interface do usuário através da ação de clicar e arrastar os componentes da Paleta (item 1 da Figura 1), tais como botões, caixas de texto, imagens, entre outros, para o Visualizador (item 2 da Figura 1). Depois de adicionar os componentes no Visualizador, o *App Inventor* monta uma estrutura desses elementos que foram adicionados na seção de componentes para facilitar a organização e acesso a eles (item 3 da Figura 1). O usuário também pode alterar as propriedades desses elementos, como cores, tamanhos, textos, alinhamentos, entre outros, através da seção de propriedades (item 4 da Figura 1).

Figura 1 – Tela do *Designer* do *App Inventor*



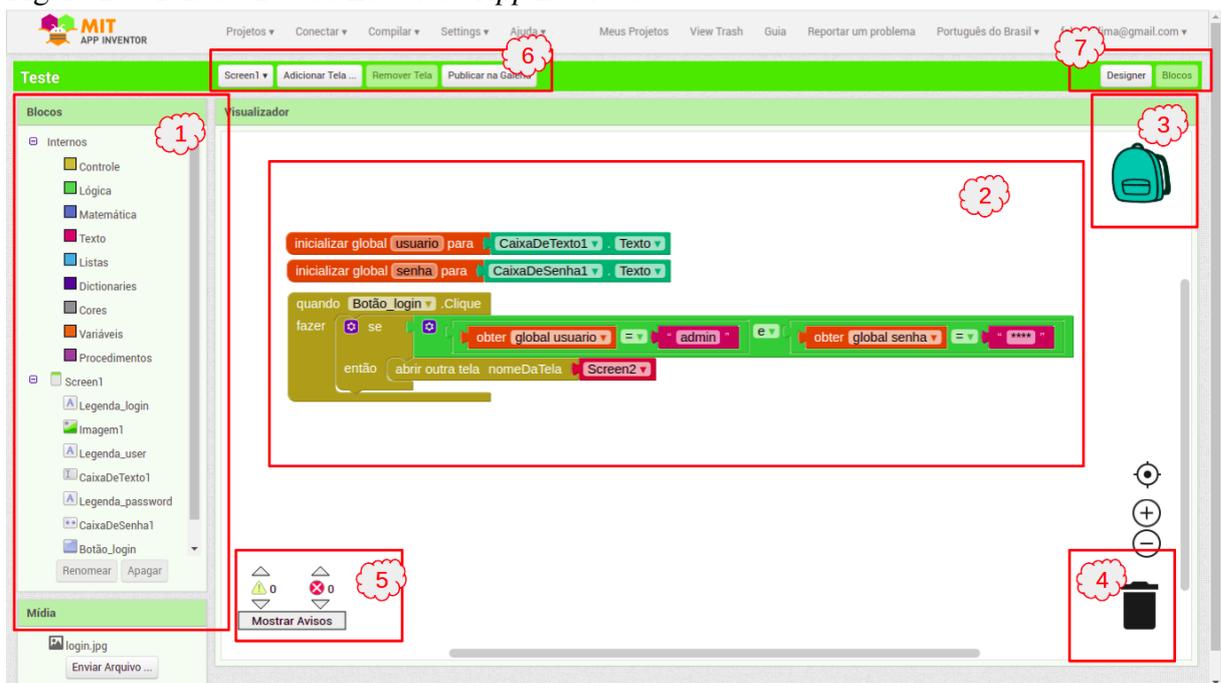
Fonte: Próprio autor (2022).

O usuário tem ainda a opção de criar novas telas, transitar entre as telas criadas, e/ou

remover alguma tela através do menu apresentado no item 5 da Figura 1. Cada tela criada no *App Inventor* tem automaticamente sua área dedicada à criação da interface do usuário e uma outra parte dedicada à edição de blocos, onde é construída a lógica por trás do aplicativo. O usuário pode alternar entre as janelas do designer e do editor de blocos através dos botões contendo o nome de cada janela como mostra o item 6 da Figura 1.

Como já foi antecipado, a janela do editor de blocos permite controlar os comportamentos dos componentes que foram definidos anteriormente na tela do *designer*. Neste ambiente, o usuário encontra blocos conectáveis, que podem ser eventos ou métodos, em uma interface do tipo arrastar e soltar (item 1 da Figura 2). Estes blocos operam *strings* e listas, realizam ações de controle (*e.g. if, else, foreach, etc.*) e operações matemáticas, entre muitas outras funcionalidades (INVENTOR, 2022).

Figura 2 – Tela do *Blocks Editor* do *App Inventor*



Fonte: Próprio autor (2022).

O item 2 da Figura 2 mostra um trecho de blocos conectados para definir o comportamento em uma ação de tentativa de *login* quando ocorre um clique no botão "*Botão_login*", definido da tela do *designer*. O usuário tem ainda a opção de adicionar esses trechos de código em área que possibilita que esse mesmo trecho já gerado possa ser utilizado em outras telas sem a necessidade de agrupar os mesmos blocos novamente para obter um resultado semelhante. Isso pode ser feito arrastando os blocos desejados para a área representada pelo ícone de uma mochila, e posteriormente, recuperados ao clicar sobre o mesmo ícone (item 3 da Figura 2). Já para deletar

os blocos, o usuário pode clicar e arrastar os elementos para a área determinada pela lixeira (item 4 da Figura 2). O item 5 é a caixa de avisos para orientar os usuários com possíveis problemas na construção dos blocos, enquanto que os itens 6 e 7 apresentam as mesmas funcionalidades presentes na tela de *Designer*.

Para auxiliar no desenvolvimento e testes, o *App Inventor* fornece um aplicativo móvel disponível para sistemas *Android* chamado *App Inventor Companion*¹⁰ no qual os desenvolvedores podem testar o comportamento de seus aplicativos em tempo real à medida que vão desenvolvendo. Dessa forma, os usuários podem rapidamente criar uma aplicação móvel e começar a interagir com ela no próprio celular.

O uso do *App Inventor* é constantemente direcionado para o ensino e aprendizado de conhecimentos e habilidades de programação como mostra alguns autores (MORELLI *et al.*, 2011; GRAY *et al.*, 2012; GROVER; PEA, 2013; XIE; ABELSON, 2016; PAPADAKIS *et al.*, 2019; CHENG *et al.*, 2021), demonstrando que essa é uma ferramenta importante quando tratamos desse tema. Além disso, a plataforma oferece recursos que apoiam os conceitos que tentamos transmitir nesse estudo, tais como requisitos, *design*, reuso, testes e *debug*, o que corrobora para a escolha dessa ferramenta para o nosso trabalho.

Desta forma, o *App Inventor* foi escolhido como ferramenta para criação de aplicações durante este estudo, é nela que a assistente virtual desenvolvida nesse trabalho vai auxiliar o usuário final para tentar introduzir conceitos de EUSE usando os elementos que a plataforma já disponibiliza.

2.4 Interface de Linguagem Natural

No campo de IHC, principalmente nos últimos anos, novos desafios surgiram para ambientes EUD quando se trata de melhorar ambientes e formas de interação, um desses desafios são interfaces por meio de linguagem natural.

O processamento de linguagem natural é uma subárea de Inteligência Artificial que visa ajudar computadores a entender e reproduzir a linguagem humana, falada e escrita (MACHIRAJU; MODI, 2018). A ideia da programação por meio de linguagem natural surgiu a mais de cinquenta anos, porém, apenas recentemente grandes avanços aconteceram nessa área, como podemos observar nos atuais assistentes virtuais (BARRICELLI *et al.*, 2019).

Nos últimos anos, *software bots* tornou-se um tópico movimentado tanto na área

¹⁰ <https://appinventor.mit.edu/explore/ai2/setup-device-wifi.html>

de pesquisa como na indústria, principalmente pelo fato de seu desenvolvimento ter se tornado cada vez mais acessível para milhões de desenvolvedores. Atualmente esse termo é utilizado para descrever interfaces de usuário no estilo conversação ou a um agente que automatiza tarefas rotineiras e tediosas. Embora a intenção não seja necessariamente se passar por uma pessoa real, muitos assistentes apresentam uma personalidade agradável e envolvente. Normalmente, esses robôs residem em plataformas nas quais os usuários trabalham ou jogam com outros usuários (LEBEUF *et al.*, 2017).

Com a chegada dos Assistentes Virtuais Pessoais (VPAs), foram desenvolvidas interfaces de usuário inteligentes por meio de linguagem natural (NLUI), também conhecidas como uma evolução dos assistentes de voz, as NLUI são responsáveis por responder a simples comandos como configuração de alarmes ou realização de chamadas, por exemplo, assim como os *chatbots* (interação por texto) e *voicebots* (interação por a voz), comunicando-se com o usuário para prover uma solução para um determinado problema, diante deste contexto, a forma como novos sistemas são desenvolvidos ou modificados caminha para um novo estilo de programação, auxiliados por voz (BARRICELLI *et al.*, 2019). Porém, segundo Azaria *et al.* (2016), a interação com alguns agentes é limitada a comandos pré-programados pelos desenvolvedores, não permitindo ao usuário a definição de um novo comando e uma sequência de ações a serem executadas.

Para Abdellatif *et al.* (2021) os *chatbots* estão sendo projetados para mudar drasticamente o futuro da ES, permitindo que os profissionais conversem e perguntem sobre seus projetos de software e interajam com diferentes serviços usando linguagem natural. Em meio a isso, Abdellatif *et al.* (2021) avaliou quatro dos NLU's mais usados para a criação de *chatbots*, dentre eles IBM Watson ¹¹, Dialogflow ¹², Rasa ¹³ e LUIS ¹⁴ para tentar esclarecer qual NLU deve ser usado em *chatbots* baseados em Engenharia de Software. Utilizando dois conjuntos de dados que refletem tarefas comuns realizadas por profissionais de Engenharia de Software, como perguntas sobre repositórios de software e perguntas de desenvolvimento em fóruns de perguntas e respostas (por exemplo, Stack Overflow ¹⁵). Onde o IBM Watson e o Rasa demonstraram o melhor desempenho ao considerar aspectos de classificação de intenções, pontuações de confiança e extração de informações. O que justifica o uso do *Rasa framework* neste trabalho.

¹¹ <https://www.ibm.com/br-pt/watson>

¹² <https://cloud.google.com/dialogflow/docs?hl=pt-br>

¹³ <https://rasa.com/>

¹⁴ <https://learn.microsoft.com/pt-br/azure/cognitive-services/luis/what-is-luis>

¹⁵ <https://stackoverflow.com/>

3 TRABALHOS RELACIONADOS

Existem várias abordagens destinadas a auxiliar usuários finais a desenvolver ou modificar seus próprios sistemas. Neste Capítulo são apresentados alguns trabalhos recentes que relacionam a área de EUD ou EUSE com abordagens utilizando o *App Inventor* para ensino de conceitos envolvendo usuários finais ou então com propósito de envolver linguagem natural para melhorar a forma de desenvolvimento em ambientes EUD.

Brummelen (2019) apresenta uma forma de democratizar o desenvolvimento de aplicações envolvendo Inteligência Artificial (IA), permitindo que pessoas que antes eram apenas consumidoras de tecnologias se tornem desenvolvedores dessa tecnologia. Através de dispositivos de conversação como a Alexa, os autores capacitaram alunos a produzir *skills* (comandos/rotinas) para esses dispositivos por meio da programação em blocos. Para isso, uma extensão da plataforma do *App Inventor* foi desenvolvida, na qual foram inseridos novos blocos de programação com estruturas que permitiam fazer a integração com dispositivos de voz. A nossa proposta difere em alguns pontos, entre eles, o principal é a finalidade da pesquisa. Enquanto Brummelen (2019) visa permitir que usuários não profissionais possam criar sistemas envolvendo IA através da plataforma do *App Inventor*, nosso trabalho visa utilizar essa mesma plataforma, com a adição de uma assistente para introduzir conceitos de EUSE.

Tentando identificar como estratégias de engenharia de software podem ser aplicadas para usuários finais em um contexto de desenvolvimento, Fronza *et al.* (2020) apresentam um estudo em ambientes de Eventos Baseados em Projetos (PBE), como *bootcamps*. Nesse estudo, os autores utilizaram a plataforma do *App Inventor* e consideraram duas populações diferentes para o experimento, uma formada por alunos do ensino médio e a outra por alunos de pós-graduação. Optando por uma estratégia de não impor qualquer prática ou lição específica, foi adotado um conjunto de atividades que levassem os participantes a refletir e raciocinar quando houvesse necessidade de planejamento. A avaliação do experimento foi feita através de observação dos participantes, análise de *code smells*¹⁶ e da aplicação de métricas ao avaliar os artefatos produzidos. A partir dos resultados, os autores conseguiram extrair *insights* que permitiram a criação de um modelo de ensino adaptado para usuários não profissionais. A principal diferença para nossa proposta é o fato de que Fronza *et al.* (2020) buscam adaptar um modelo estratégico de adoção de ES em eventos baseados em projetos, enquanto o nosso

¹⁶ Escolha de *design* abaixo do ideal que pode degradar diferentes aspectos da qualidade do código (FOWLER, 2018)

trabalho apresenta uma ferramenta que ajuda a adotar conceitos de ES em diferentes contextos.

Monteiro *et al.* (2017) apresentam um estudo do ponto de vista semiótico de como elementos da engenharia de software são abordados em programas criados por usuários finais envolvendo as ferramentas *AgentSheets* e *PoliFacets*. Através de uma análise dos signos presentes na interface dos sistemas estudados, os resultados apontaram quais conteúdos de EUSE eram mais abordados durante o projeto em estudo. A principal semelhança entre este trabalho e o de Monteiro *et al.* (2017) é o fato de ambos tentarem abordar os conceitos de EUSE que são e que podem ser explorados por usuário finais.

Lizcano *et al.* (2013) propõem um modelo de ciclo de vida baseado nas atividades da engenharia de software tradicional, mas adaptado aos problemas dos usuários finais em sistemas web. Os autores discutem que a principal fraqueza da EUSE não são as ferramentas, linguagens visuais ou técnicas de desenvolvimento, mas a ausência de um ciclo de vida do software suportado por uma estrutura para guiar os usuários por todas as atividades e fases de desenvolvimento. Com base nisso, o trabalho apresenta a definição de um ciclo com tarefas categorizadas entre algumas fases de EUSE (Requisitos, Análise, *Design* e Implementação, Testes e Compartilhamento) que guiam os usuários finais ao longo do desenvolvimento fazendo com que ele aborde atividades desejadas sem desviar de seu foco principal. O que difere o trabalho de Lizcano *et al.* (2013) deste trabalho é o fato de não abordar a plataforma do *App Inventor* como ferramenta de desenvolvimento.

Li (2019) apresenta um trabalho focado em empoderar usuários finais a programar agentes para realização de tarefas em ambientes inteligentes através de uma combinação entre programação por demonstração e instruções por linguagem natural. O trabalho citado faz um avanço na pesquisa já mostrada em Li *et al.* (2017), no qual um modelo de programação por demonstração¹⁷ e linguagem natural é apresentado. Neste novo estudo, o autor visa fazer com que a ferramenta possa quebrar uma tarefa definida pelo usuário em comandos menores e aprender esses comandos para serem generalizados em diferentes tarefas. O que difere o trabalho de Li (2019) da nossa proposta é o fato de não abordar conceitos relacionados a EUSE e também não abordar a ferramenta do *App Inventor* como instrumento de desenvolvimento. Por outro lado, entender como ferramentas atuais capturam e trabalham comandos por linguagem natural enriquecem a base deste trabalho.

Finalmente, Weigelt *et al.* (2018) mostram como integrar um componente de diálogo

¹⁷ Paradigma baseado em interface gráfica, no qual os usuários especificam “o que” deverá ser feito, sem ter a preocupação de “como” será feito (KURLANDER *et al.*, 1993)

em um sistema de processamento de linguagem natural. Os autores concentram-se em explicar a separação entre o PARSE, que é um *framework* que visa facilitar a programação de usuário final por meio da linguagem natural falada, e o componente de diálogo, apresentando quais partes de um sistema de diálogo são implementadas no componente de diálogo, quais partes são fornecidas pelo *framework* e como a interação funciona. O que difere da nossa proposta é o fato da abordagem utilizada por Weigelt *et al.* (2018) ser focada em trabalhar linguagem natural para ambientes EUD, porém não envolve conceitos de EUSE. Outra diferença entre as abordagens é que o presente trabalho aborda a plataforma do App Inventor como ferramenta de desenvolvimento.

A Tabela 2, apresentada a seguir, mostra resumidamente a relação dos conceitos abordados entre os trabalhos relacionados e o nosso trabalho.

Tabela 2 – Comparação entre os trabalhos relacionados e o atual

Trabalho	App Inventor	EUD	EUSE	Linguagem Natural
Brummelen (2019)	Sim	Sim	Não	Não
Fronza <i>et al.</i> (2020)	Sim	Sim	Sim	Não
Monteiro <i>et al.</i> (2017)	Não	Sim	Sim	Não
Lizcano <i>et al.</i> (2013)	Não	Sim	Sim	Sim
Li (2019)	Não	Sim	Não	Sim
Weigelt <i>et al.</i> (2018)	Não	Sim	Não	Sim
Nosso trabalho	Sim	Sim	Sim	Sim

Fonte: Elaborado pelo autor (2022).

4 METODOLOGIA

Seguindo um enfoque exploratório, procuramos observar, analisar e compreender formas de incorporar as atividades de ES no processo de desenvolvimento por usuários finais, ao longo das etapas que transcorreram esse percurso investigativo. A pesquisa realizada durante o mestrado foi desenvolvida baseando-se em abordagens qualitativas. Realizamos estudos que envolveram a participação de usuários, e também, inspeções apenas com o pesquisador envolvido.

Após apresentar os objetivos da proposta deste trabalho, são definidas as questões de pesquisa que direcionam os estudos apresentados nas próximas seções, é esclarecida a escolha das metodologias e como elas auxiliaram ao longo da pesquisa. Em seguida, são apresentados cada um dos estudos realizados, descrevendo o perfil dos participantes, os métodos e procedimentos realizados e como os dados foram coletados e analisados.

4.1 Objetivos e Questões de pesquisa

Conforme antecipado em capítulos anteriores, nosso interesse de investigação é uma forma de apoiar usuários finais a trabalharem atividades de EUSE. Desta forma, este trabalho tem como principal objetivo definir e avaliar uma assistente virtual, com interação por voz, vinculada à plataforma do *App Inventor* para auxiliar usuários finais durante o ciclo de desenvolvimento ao construírem uma aplicação, de modo a abordar atividades relacionadas a EUSE. Diferentemente de outros trabalhos que abordam uma fase específica de EUSE, nossa proposta opta por investigar e construir uma assistente que permita ao usuário final transitar entre todas as fases de desenvolvimento abordadas em Lizcano *et al.* (2013), Monteiro *et al.* (2017) e Ko *et al.* (2011)

Para iniciar nossa jornada, começamos este estudo guiado pela seguinte questão de pesquisa:

QP1: Como usuários finais podem trabalhar conceitos de EUSE dentro da plataforma do App Inventor?

Assim como dito anteriormente, as ferramentas EUD, em sua maior parte, dedicam-se a dar suporte ao usuário que precisa programar ou desempenhar alguma outra função relacionada ao desenvolvimento computacional para benefício próprio ou para outros usuários, porém, trabalhando essas atividades muito separadamente em cada abordagem. Pensando em favorecer esses usuários, queremos entender melhor como a plataforma escolhida possibilita trabalhar um

conjunto mais completo de atividades em um único ambiente.

A partir disso, decorrem subquestões que são necessárias de serem respondidas para se obter um panorama satisfatório desse assunto:

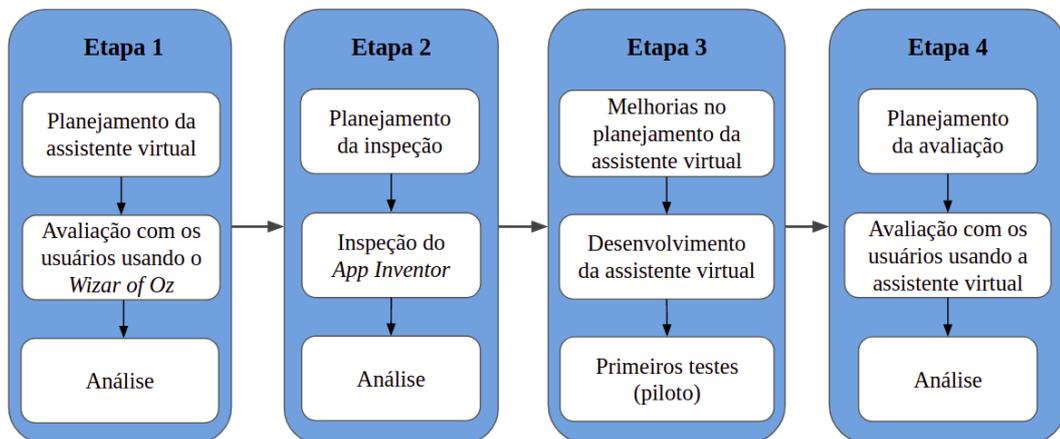
- QP1.1: Como usuários finais trabalham atividades diretamente ligadas a EUSE?
- QP1.2: Quais atividades de EUSE a plataforma do *App Inventor* aborda ou apoia?
- QP1.3: Como os usuários finais adotariam a ajuda de uma assistente virtual?
- QP1.4: Antes e depois dos estudos, o que os usuários finais sabiam de ES?

As respostas para estas perguntas foram buscadas a partir de estudos empíricos e discussões ao longo deste mestrado.

4.2 Procedimentos

A Figura 3 mostra um resumo da metodologia que foi executada para este trabalho. Apresentando as atividades realizadas separadas por etapas que foram definidas ao longo do nosso processo de investigação. Essas etapas iam se fazendo necessárias a medida que o estudo avançava para nos aproximarmos do objetivo final.

Figura 3 – Metodologia de pesquisa



Fonte: Próprio autor (2022).

As etapas 1 e 2 correspondem aos estudos iniciais que foram realizados com os objetivos de identificar informações a respeito da forma de desenvolvimento dos usuários finais e para conhecer sentenças utilizadas por esses usuários ao comunicarem-se com um “robô”. Além de entender e extrair o máximo de recursos que o *App Inventor* oferece para apoiar as atividades de requisitos, *design*, reuso, testes e *debug*. Já a etapa 3, corresponde ao planejamento,

desenvolvimento e melhorias da assistente virtual. Na etapa 4, tratamos da avaliação realizada com o *App Inventor* integrado à assistente virtual que foi desenvolvida.

4.3 Etapa 1 - *Wizard of Oz*

Buscando entender melhor como a EUSE pode ser abordada dentro do contexto de desenvolvimento e utilizando a plataforma do *App Inventor*, antes de aplicar qualquer esforço na construção de uma versão inicial do assistente, era necessário um método que pudesse ajudar a identificar *insights* relacionados a forma como o usuário final tenta desenvolver uma aplicação ao mesmo tempo que se comunica com um assistente de voz para auxiliar nessa tarefa. Sendo assim, este trabalho inicia apresentando o primeiro estudo realizado que utilizou o método do *Wizard of Oz*.

Esse método é baseado na história de L.F. Baum (1900), *O Mágico de Oz*, na qual uma pessoa produzia imagens do “mágico” manipulando um conjunto de controles enquanto se escondia atrás de uma cortina (GREEN; WEI-HAAS, 1985). O método aqui utilizado fornece praticamente o mesmo cenário, pois as imagens fornecidas são controladas por um “mágico” nos bastidores. Para Simple Usability (2018), este método permite que pesquisadores testem algum conceito, de forma que um moderador conduza a sessão junto com o usuário, enquanto alguém controla as respostas enviadas ao usuário através do dispositivo escolhido.

É considerado um método de prototipação rápida para sistemas com custo elevado de produção ou que exigem tecnologias ainda não disponíveis, permitindo assim, avaliar as reações e a interação dos usuários com o sistema antes mesmo de pensar em desenvolvimento, ideal para sistemas que exigem um alto esforço para ser desenvolvido (MAULSBY *et al.*, 1993). Assim como as outras metodologias, a criação de um protótipo no *Wizard of Oz* começa com a necessidade de determinar o que queremos testar ou explorar. Depois, precisamos descobrir como simular a funcionalidade necessária para fornecer ao usuário uma experiência realista do ponto de vista dele.

A fase de planejamento começou com a definição do que queríamos testar. Para este trabalho, não se tratava apenas de testar o conceito de uma assistente virtual, incluía também a necessidade de identificar falas e comandos utilizados por usuários finais ao longo do desenvolvimento para ajudar na criação de uma base de conhecimento baseada em situações reais de uso. Além disso, a técnica foi importante para tentar extrair formas de como esses usuários poderiam trabalhar atividades relacionadas a EUSE.

Após a definição do que seria avaliado, foi necessário criar o material para ser utilizado durante a execução da avaliação. Entre eles, o formulário de recrutamento, o termo de consentimento, o cenário a ser trabalhado, as imagens necessárias e, por fim, preparar o ambiente da avaliação e elaborar as perguntas para a entrevista ao final do teste.

A preparação do material começou com a elaboração do termo de consentimento (Apêndice A), o qual deveria deixar clara a participação do usuário e quais deveriam ser seus direitos e as atividades que ele deveria realizar. O outro documento elaborado foi o cenário, entregue aos participantes ao aceitarem participar da avaliação. Nesse documento, era descrita uma situação hipotética, porém, tentando buscar pontos em comum com a realidade dos participantes para melhor ambientá-los, e, desta forma, aumentar seu engajamento no desenvolvimento da aplicação. Além disso, foi necessário criar uma imagem para mostrar ao participante durante o que chamamos de “momento de espera”, que era o espaço de tempo entre um comando do usuário e a finalização da ação pelo responsável por simular os comandos (Apêndice B).

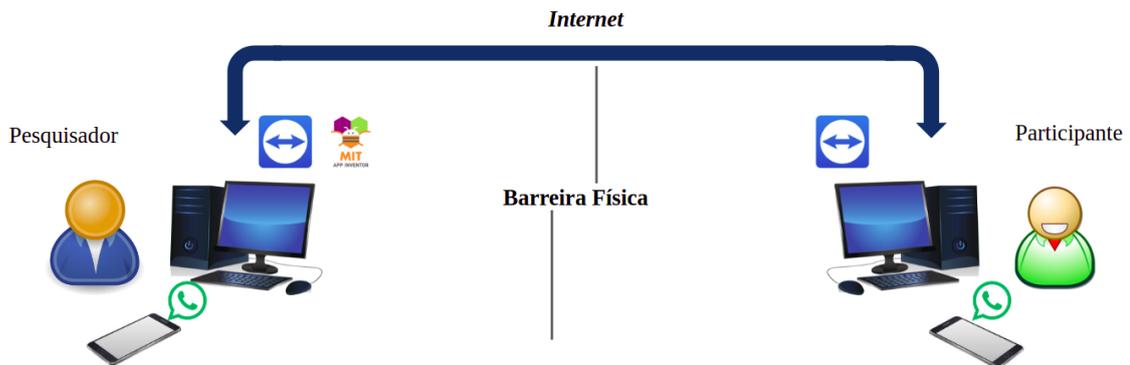
O cenário proposto descrevia uma situação na qual o participante era um funcionário de um campus da Universidade Federal do Ceará e fazia uso dos serviços do restaurante universitário. A tarefa a ser realizada por ele consistia em desenvolver uma aplicação que possibilitasse avaliar as refeições do restaurante, substituindo a tradicional avaliação no papel como era comumente utilizada. O cenário completo é apresentado no Apêndice C.

Em relação ao planejamento do ambiente, a Figura 4 apresenta a estrutura montada para a realização da avaliação com o *Wizard of Oz*. Sendo definido que o participante teria acesso a um computador que estaria diretamente conectado ao computador do pesquisador, através da ferramenta *TeamViewer*¹⁸, um software para acesso e controle de outros dispositivos remotamente. Por sua vez, o computador do pesquisador era o responsável por acessar a janela do *App Inventor* e conseqüentemente, realizar as ações que eram recebidas através dos comandos de voz que vinham do participante. Os dois computadores eram conectados pela internet e deveriam ficar em salas separadas para conseguir trazer mais realidade no uso da assistente virtual. Com a ajuda do *TeamViewer*, o participante poderia não somente observar o que acontecia no computador do pesquisador ao ter seus comandos executados através da sua interação por voz (neste caso, foi utilizada a ligação pelo aplicativo *Whatsapp*¹⁹ como forma de comunicação entre pesquisador e participante), como também poderia executar ações que eram refletidas direto no *App Inventor*.

¹⁸ <https://www.teamviewer.com/pt-br/>

¹⁹ <https://www.whatsapp.com/>

Figura 4 – Comunicação entre pesquisador e participante



Fonte: Próprio autor (2022).

O ambiente da avaliação foi organizado de maneira similar ao demonstrado na Figura 5. O espaço destinado ao participante (Figura 5a) continha um computador para o usuário “interagir diretamente” com o sistema do *App Inventor*, além de um smartphone para comunicação alocado atrás do computador. Na sala do pesquisador, similar a Figura 5b, existiam dois computadores, onde um estava diretamente ligado ao computador na sala do participante enquanto o outro era utilizado para o pesquisador fazer as modificações no *App Inventor* conforme o participante comunicava suas ações.

Figura 5 – Representação do Ambiente da avaliação (imagens meramente ilustrativas)



(a) Ambiente do participante.



(b) Ambiente do "wizard".

Fonte: *Google Imagens*

Após preparar o material necessário e realizar os ajustes que faltavam, era o momento de recrutar os participantes para a nossa avaliação preliminar. O perfil de usuário determinado para o teste foi de uma pessoa que tivesse pouca ou nenhuma habilidade em programação. Além

disso, deveria ser servidor da Universidade Federal do Ceará - Campus Quixadá e fazer uso dos serviços do restaurante universitário.

Após definido o perfil, a escolha dos participantes foi por conveniência, cada um foi convidado pessoalmente. Aos que se mostraram interessados em participar da avaliação foi enviado um formulário eletrônico para o preenchimento de algumas informações como nome, e-mail e os dias em que ela estaria disponível para realização da atividade (Apêndice D). A partir das respostas desse formulário, o termo de consentimento e o cenário elaborado foram enviados a cada um.

4.4 Etapa 2 - Inspeção do *App Inventor*

Com o objetivo de ampliar o conhecimento sobre como o *App Inventor*, em sua versão atual, permite e expõe recursos que estão diretamente relacionados com os conceitos de EUSE, foi elaborado um estudo para investigar como a plataforma oferece meios para auxiliar os usuários finais a melhorar o seu processo de desenvolvimento.

Nesse estudo foi realizada uma análise das duas janelas principais da plataforma em busca de elementos relacionados a cada uma das cinco dimensões de EUSE abordadas em Monteiro *et al.* (2017) e que são consideradas aqui. Esse estudo envolveu principalmente a figura do pesquisador como explorador da ferramenta e foi composto por duas etapas.

A primeira parte desse estudo consistia na análise das telas em busca de elementos que apresentassem algum relacionamento direto com o desenvolvimento de um aplicativo. Após obter uma lista com os elementos, a segunda parte era a análise da funcionalidade de cada um, para que dessa forma, fosse possível relacioná-lo com alguma das seguintes dimensões: Requisitos, *Design*, Reuso, Testes ou *Debug*. Os achados referentes a esse estudo serão abordados futuramente neste trabalho.

4.5 Etapa 3 - Desenvolvimento da assistente virtual

Na etapa 3 foram realizadas melhorias no planejamento da assistente virtual com base nos resultados dos estudos anteriores. Dentre essas melhorias estavam a criação de novas frases para serem agregadas na base de conhecimento da assistente e que poderiam ser adotadas pelos usuários durante atividades de desenvolvimento. Além de entender como trabalhar os recursos que o *App Inventor* oferece para também incluir esse conhecimento na nossa base.

A escolha pelas tecnologias utilizadas para o desenvolvimento foram orientadas pelo *App Inventor*, que utiliza a linguagem de programação Java ²⁰ em sua maior parte e pela escolha do Rasa framework que utiliza a linguagem Python ²¹, além de uma linguagem de serialização de dados que é usada na escrita de arquivos de configuração conhecida por YAML. A construção da assistente virtual está detalhada no capítulo 6. Nele descrevemos as alterações realizadas na plataforma atual, a construção dos novos componentes necessários para a concepção da assistente, as tecnologias utilizadas, além de demonstrar como é o seu funcionamento.

4.6 Etapa 4 - Avaliação da assistente virtual

A etapa 4 deste trabalho é composta pelo planejamento e execução da avaliação utilizando o *App Inventor* integrado à assistente virtual construída na etapa anterior. Também está contida nesta etapa a análise dos dados obtidos através das respostas do questionário, análise dos vídeos da avaliação e pela análise dos artefatos construídos pelos usuários finais. Nesta seção serão apresentados o planejamento e execução do estudo, chamado de Estudo 3.

Para o planejamento dessa avaliação, já que ela seria realizada durante o período de pandemia e devido a necessidade de um distanciamento social, as atividades para esse estudo ocorreram de forma remota. Desta forma, foi preciso disponibilizar a versão do *App Inventor* construída para este trabalho de forma online para que os participantes pudessem acessar de suas próprias casas. Para isso, foi utilizado o *ngrok*²², uma ferramenta simples e gratuita que possibilita a criação de conexões (túneis) seguras entre uma máquina local e a internet. Através de seus servidores, o *ngrok* disponibilizava uma url permitindo aos usuários conectados na internet se conectarem na máquina que estava rodando a instância do *App Inventor* (Figura 6).

Com a implementação concluída da nossa instância do *App Inventor*, da lógica por trás da assistente virtual e com um servidor online para acessos remotos, o próximo passo era realizar os testes com usuários reais. O cenário proposto foi o mesmo do estudo com o *Wizard of OZ* (Apêndice C) no qual os participantes deveriam desenvolver parte de uma aplicação para avaliação de refeições dentro de um restaurante universitário, ressaltando que foi explicado que não era necessário desenvolver uma aplicativo por completo e que eles poderiam ficar tranquilos em relação a isso.

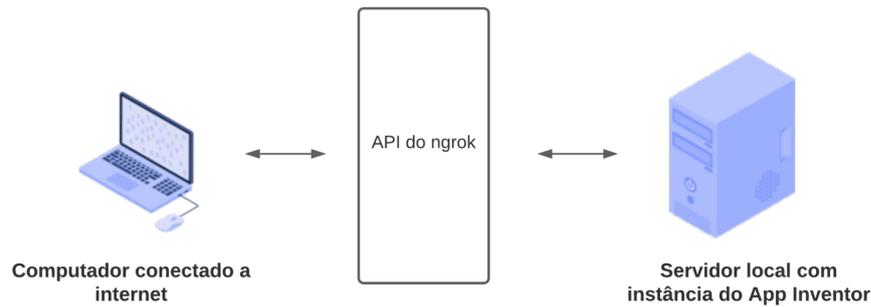
Devido às medidas de distanciamento social, os participantes foram convidados por

²⁰ https://www.java.com/en/download/help/whatis_java.html

²¹ <https://www.python.org/doc/essays/blurb/>

²² <https://ngrok.com/>

Figura 6 – Representação do funcionamento do *ngrok*



Fonte: Próprio autor (2022).

e-mail e informados que poderiam fazer a atividade de suas próprias casas, a amostragem foi escolhida por conveniência devido ao momento descrito anteriormente. O perfil escolhido foi o de estudantes de graduação ou mestrado nas áreas de engenharia que faziam ou fizeram uso de restaurantes universitários em algum momento de sua carreira acadêmica.

Antes da realização do estudo, um e-mail contendo o cenário proposto foi enviado inicialmente. O restante das informações necessárias foram repassadas através de uma conversa individual por videochamada com aqueles que aceitaram participar do teste antes de iniciarem as atividades. Esse estudo teve um total de 10 participantes e as avaliações ocorreram em datas determinadas pela disponibilidade das pessoas e com tempo aproximado de 90 minutos de duração para cada um.

Após a realização da parte prática os participantes eram convidados a responder um questionário (Apêndice E) de forma anônima enviado diretamente no e-mail deles. A escolha pela aplicação de um questionário online ocorreu para tentar evitar um viés de desejabilidade social²³ para com o aplicador da avaliação. As perguntas do questionário também foram baseadas no SUS (do inglês, *System Usability Scale*), uma forma de avaliação subjetiva para usabilidade de sistemas (BROOKE *et al.*, 1996). Os resultados obtidos durante este estudo serão descritos no capítulo 7.

²³ Propensão por parte das pessoas a dar respostas consideradas como socialmente mais aceitáveis

5 ESTUDOS INICIAIS

Neste capítulo apresentamos os resultados dos dois primeiros estudos realizados ao longo desse trabalho, organizando os achados com foco nos usuários finais e na Engenharia de Software, para assim, direcionar o leitor rumo as informações necessárias para responder as questões de pesquisa abordadas anteriormente. Além de servir como base para o desenvolvimento da assistente virtual que será descrito no próximo capítulo.

5.1 Visão geral dos resultados iniciais

Antes de iniciar a exposição das evidências encontradas em cada estudo, apresentamos na Tabela 3, algumas informações a respeito dos estudos que serão referenciados nas próximas seções.

Tabela 3 – Visão geral dos estudos 1 e 2

	Estudo 1	Estudo 2
Participantes	3 estudantes	1 pesquisador
Foco	Avaliar como usuários finais abordavam tópicos de desenvolvimento e ES no <i>App Inventor</i>	Entender quais ferramentas relacionadas a ES estão disponíveis no <i>App Inventor</i>
O que foi feito	Avaliação com usuários utilizando o método do <i>Wizard of Oz</i> Análise das observações	Inspeção das telas Classificação dos achados

Fonte: Elaborado pelo autor (2022).

5.2 Estudo 1 (*Wizard of Oz*)

Esse primeiro estudo teve o objetivo de tentar identificar como os usuários trabalhavam o desenvolvimento dentro do *App Inventor* através de ações e comandos, e como seria possível abordar atividades de EUSE por meio da ajuda de um assistente virtual. Além de aumentar a base de possíveis ações e falas que poderiam ser ditas pelos usuários. Detalhe a ser ressaltado, apenas nesse estudo 1 a ferramenta incorporada no *App Inventor* é referenciada como do sexo masculino, pois o responsável por simular o assistente era do sexo masculino. Enquanto que no restante do trabalho optou-se por uma representação feminina.

Para buscar melhorar a simulação do assistente, de início, foi realizado um teste

com um voluntário (colega de mestrado) que embora tivesse noções de programação, não tinha conhecimentos de ES ou da plataforma do *App Inventor*. Outras duas pessoas, com perfil semelhante também foram recrutadas para esse primeiro estudo. Devido a impossibilidade de dar continuidade com as avaliações, apesar de não ser o ideal, optou-se por incorporar os resultados do primeiro participante nessa análise, desta forma, o estudo totalizou três participantes. O perfil de cada participante está descrito na Tabela 4.

Tabela 4 – Conhecimentos dos participantes em relação aos temas abordados

	Sexo	Conhecimentos		
		Programação	<i>App Inventor</i>	Engenharia de Software
P1	M	Sim	Não	Não
P2	M	Não	Não	Não
P3	F	Não	Não	Não

Fonte: Elaborado pelo autor (2022).

Os resultados apresentados a seguir foram derivados das observações e das respostas obtidas durante a realização dos estudos. Abordando uma visão para cada participante individualmente, tentando entender seus objetivos, suas dúvidas e suas percepções. Cada participante realizou a atividade dentro do tempo de 1-2 horas.

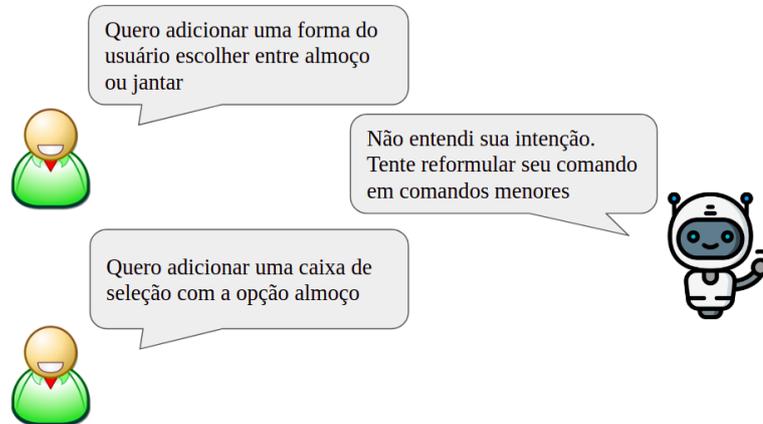
Participante P1

O primeiro participante (P1) iniciou o desenvolvimento sem nenhum tipo de “cola” ou esboço do que queria desenvolver. No início, o participante demonstrou curiosidade na plataforma e nos elementos que ela apresentava antes mesmo de começar a concepção do software. No primeiro contato com o assistente virtual, P1 foi questionado sobre o sistema que iria desenvolver e o que havia planejado para o software, começando assim o seu processo de criação. Quando o assistente perguntou sobre as funcionalidades da aplicação e sobre como deseja armazenar aquilo, o participante dedicou uma parte do tempo que tinha para digitar os requisitos em um bloco de notas para facilitar futuras consultas.

Ao longo do estudo, o participante apresentou dúvidas sobre alguns elementos da interface do *App Inventor* ao mesmo tempo que solicitava a ajuda do assistente para inserir elementos na tela, desde elementos gráficos para visualização quanto elementos de lógica. Além disso, P1 utilizava comandos de muito alto nível para se comunicar com o assistente, fazendo-se necessário induzir o participante a quebrar seu comando em intenções menores para definir uma

funcionalidade, como mostra o exemplo da Figura 7.

Figura 7 – Exemplo do participante reformulando um comando



Fonte: Próprio autor (2022).

P1 iniciou o desenvolvimento pelas telas da aplicação, definindo o *design* delas com base em conhecimentos prévios. Ao final, 3 telas foram desenvolvidas (Figura 8).

Figura 8 – Telas desenvolvidas pelo participante P1



Fonte: Próprio autor (2022).

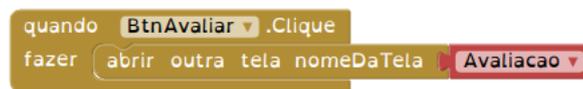
A primeira tela desenvolvida foi a tela de seleção da refeição, apresentada na Figura 8a, na qual o participante inseriu um elemento para escolha da data da avaliação e a possibilidade de escolher o tipo de refeição, almoço ou jantar. Já na tela de avaliação (Figura 8b), o participante identificou um caso de reuso no elemento Deslizador²⁴ escolhido para avaliar cada

²⁴ Deslizador é uma barra de progresso que possui um indicador arrastável e permite atribuir um valor/nota

aspecto da refeição. A última tela feita foi a tela de *feedback* (Figura 8c), onde o participante queria dar alguma resposta ao usuário após concluir a avaliação.

Depois de finalizar as telas, P1 desejava testar a interação entre elas. Mais uma vez o participante recorreu a ajuda do assistente para auxiliar nesse processo. Após perceber que precisaria definir a lógica para interação, P1 entrou na fase de montagem dos blocos onde reusou a mesma lógica de navegar entre as telas para desenvolver seu *feedback* pós avaliação (Figura 9). Fazendo uso da ferramenta da mochila, o participante incluiu os blocos da Figura 9a na mochila e depois alterou alguns blocos para gerar a lógica do *feedback* (Figura 9b).

Figura 9 – Blocos desenvolvidos pelo participante P1



(a) Blocos para a tela 1.



(b) Blocos para a tela 2.

Fonte: Próprio autor (2022).

Ao ser questionado no final do estudo, o participante P1 revelou ter ficado satisfeito com o resultado obtido e que desejava mostrar o que tinha desenvolvido a seus colegas. Além disso, o participante declarou que não conhecia as práticas abordadas pelo assistente ao longo do desenvolvimento, tais como requisitos e reuso, mas pode perceber sua importância no decorrer de suas atividades.

Em relação a linguagem utilizada pelo assistente, o participante considerou clara e objetiva. P1 também declarou ter sido fundamental a ajuda que recebeu, o que fez ele se sentir mais seguro ao desenvolver uma aplicação mesmo sem tantos conhecimentos. Em alguns momentos, sentiu que o assistente não o ajudou em determinadas tarefas (casos em que usava comandos muito complexos), atribuindo a responsabilidade a si mesmo por não saber expressar mais claramente as suas intenções para um “robô”.

Participante P2

O segundo participante (P2), adotou uma abordagem diferente de P1, antes mesmo da atividade de desenvolvimento, P2 esboçou algumas telas em uma folha de papel para guiá-lo nesse processo. Como já tinha trabalhado nos conceitos do seu sistema, o participante tinha uma ideia mais bem formulada das características do que seria desenvolvido. Assim como aconteceu com P1, o participante P2 iniciou seu desenvolvimento pelas telas do sistema. No início, P2 apresentou muitas dúvidas sobre diversos pontos, como nomear elementos e arquivos por exemplo, isso fez com que ele solicitasse diversas vezes a ajuda do assistente para adicionar ou modificar elementos, algo que foi mudando ao longo do experimento.

No final, P2 criou mais telas do que os outros participantes, sendo 8 no total (Figura 10). O participante optou por criar telas com objetivos mais simples e específicos, gerando assim um número maior de artefatos. Embora P2 solicitasse muito a ajuda do assistente no início da atividade, isso foi mudando à medida que o participante ia absorvendo informações, até o ponto que ele deixou de pedir apoio ao assistente e continuou a construir o sistema independente de ajuda.

Com suas maiores dificuldades no *design* do sistema, P2 teve que ir modificando a solução que havia esboçado antes ao longo do projeto, uma vez que, alguns elementos do *design* feito anteriormente não estavam disponíveis na plataforma. Ao final do experimento, P2 afirmou já ter ouvido conceitos relacionados a engenharia de software, mas nunca tinha se aprofundado ou aplicado em algum projeto. Afirmou ainda que voltaria a usar essas práticas em outros projetos quando necessário.

Figura 10 – Telas desenvolvidas pelo participante P2 - Parte 1

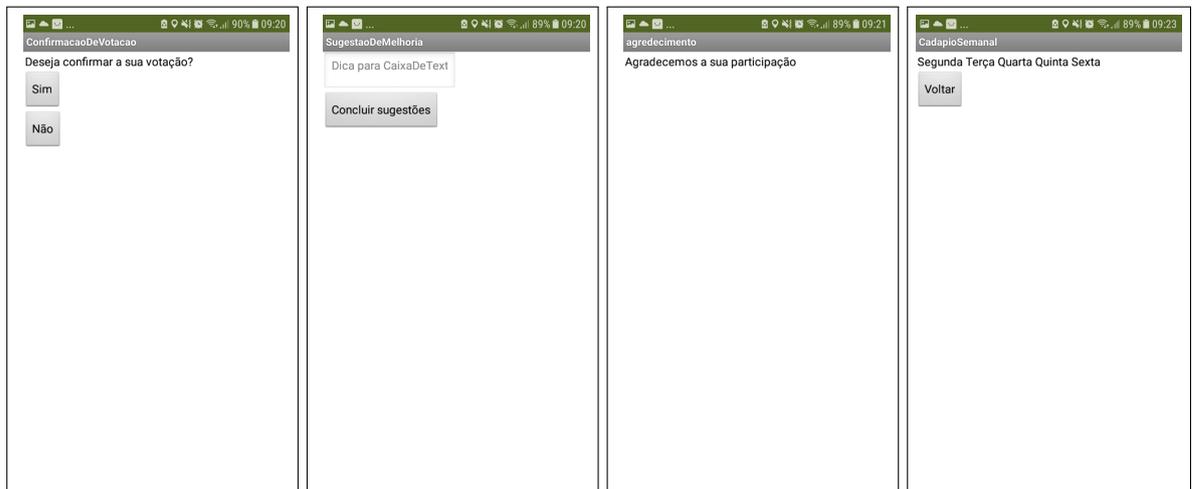


(a) Tela de login.

(b) Tela de menu.

(c) Tela de escolha da opção.

(d) Tela de avaliar.



(e) Tela de confirmação.

(f) Tela de sugestão.

(g) Tela de agradecimento.

(h) Tela do cardápio semanal.

Fonte: Próprio autor (2022).

Participante P3

O participante P3, assim como fez P2, começou a trabalhar as atividades de requisitos e *design* antes mesmo do experimento, levando consigo um esboço da tela que desejava desenvolver. Ao ser questionada pelo assistente virtual sobre os requisitos da aplicação, P3 preferiu falar todos ao invés de digitar, com a intenção de que o assistente armazenasse aquelas informações.

O participante começou a observar a tela do *App Inventor* procurando pelos recursos que deveria utilizar, chegando a identificar alguns elementos que utilizou em seu esboço. Porém, assim como aconteceu com os outros participantes, P3 não conseguiu localizar todos os elementos planejados para sua aplicação e como não conseguia formular uma ação clara do que pensava

para o assistente, acabou mudando seu *design* inicial para facilitar o desenvolvimento. Ao longo do experimento, P3 mostrava sua preferência por comunicar uma ação ao assistente virtual do que executar ele mesmo essa ação. Diferente de P1 e P2, o participante P3 construiu apenas uma tela (Figura 11) e não chegou a produzir código.

Figura 11 – Tela desenvolvida pelo participante P3



(a) Tela de avaliação.

Fonte: Próprio autor (2022).

Embora podemos notar um possível caso de reuso nas partes da tela produzida para avaliar, tais como botões e caixa de comentário, P3 não identificou essa possibilidade, adicionando assim elemento por elemento, conseqüentemente, aumentando o seu tempo para o desenvolvimento. Caso o “*assistente*” tivesse tomado uma abordagem mais reativa em relação as ações ao invés de apenas quando solicitado, poderia ter sugerido a prática de reuso nessa situação na qual o participante não descobriu sozinho.

Foi possível notar que P3 dedicou muito tempo para as tarefas de requisitos e *design*, o participante apresentou um pensamento focado no usuário do seu sistema e em como diminuir o seu esforço ao avaliar uma refeição, desejando deixar todas as informações necessárias em uma única tela. Devido ao tempo destinado para o experimento, não foi possível entrar na fases de geração de código e testes, mesmo assim, P3 se mostrou satisfeito com o resultado.

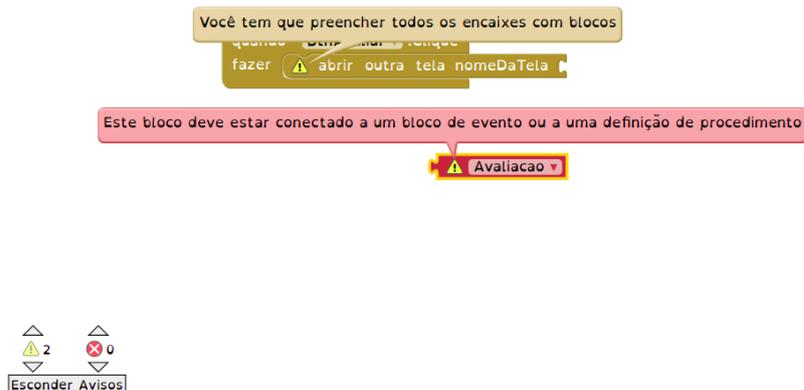
Discussão dos resultados do Estudo 1

Uma das características importantes do método *Wizard of Oz* são as observações feitas pelos pesquisadores, tanto pelo que se mantém na sala com o usuário como facilitador, mas principalmente pelo “mágico”. Em relação a essas observações, alguns pontos podem ser destacados.

O primeiro foi o *design* inicial da aplicação que alguns participantes já trouxeram para o estudo com base no cenário que foi enviado previamente para eles, revelando que eles dedicaram tempo e esforço já na fase inicial e sem que isso fosse solicitado. Além disso, um aspecto notado foi a facilidade dos usuário em modelar novas soluções para as dificuldades apresentadas. Quando um elemento não estava disponível ou então um comando imaginado não era possível de ser realizado, rapidamente eles planejavam uma outra forma de substituir o que tinha pensado anteriormente. O mesmo ocorria quando eles queriam realizar uma ação muito complexa, o que seria difícil para um assistente virtual entender, eles tinham que estruturar comandos menores e mais simples de serem entendidos por uma máquina.

Outro ponto a destacar foi como alguns participantes identificaram e fizeram reuso de partes que já estavam prontas, tais como elementos da tela ou blocos com código para a troca delas. Já a atividade de testes era atividade difícil de auxiliar, afinal é algo que os participantes deveriam fazer fora da plataforma. Por outro lado, antes mesmo dos testes, era possível auxiliar na detecção de possíveis falhas no que foi produzido através dos avisos que a plataforma oferecia, ao chamar a atenção para esses elementos quando os participantes procuravam ajudam para poder testar seus aplicativos. Os participantes conseguiam identificar pontos de falha através dos destaques que são exibidos pelo *App Inventor* (Figura 12).

Figura 12 – Exemplo de avisos em destaque nos blocos



Fonte: Próprio autor (2022).

Para finalizar, em relação ao assistente virtual, ao longo do teste os participantes foram se mostrando mais confiantes nas ações do assistente, e conseqüentemente, deixavam de executar as ações diretamente na interface para solicitar, de modo falado, o que deveria ser feito. Com exceção do participante P2 que ao adquirir mais confiança deixou de solicitar ajuda. Por vezes, os participantes utilizavam linguagem informal, comandos extensos e complexos, além de solicitar ações que não estavam disponíveis pela plataforma utilizada.

A respeito das dificuldades, um bom ponto para ser trabalhado no assistente foi que algumas vezes os participantes solicitavam ações que não eram possíveis, como adicionar elementos que não existiam na ferramenta (como um sistema de avaliação por estrelas) e que gerariam um grande esforço para reproduzir algo semelhante, ou então, tentar modificar propriedades que não estavam disponíveis (como trocar a borda de um botão). Na percepção deles, o assistente era responsável por auxiliá-los em qualquer situação, considerando assim, uma falha no suporte em não conseguir realizar tudo que eles desejavam.

Por fim, a Tabela 5 mostra uma visão dos artefatos criados e das dimensões de EUSE que foram trabalhadas nesse primeiro estudo.

Tabela 5 – Relação de artefatos criados no Estudo 1

Participante	Telas	Blocos	Dimensões abordadas				
			Requisitos	Design	Reuso	Verificação e Testes	Debug
P1	3	2	X	X	X	X	X
P2	8	7	X	X	X	X	X
P3	1	0	X	X			

Fonte: Elaborado pelo autor (2022).

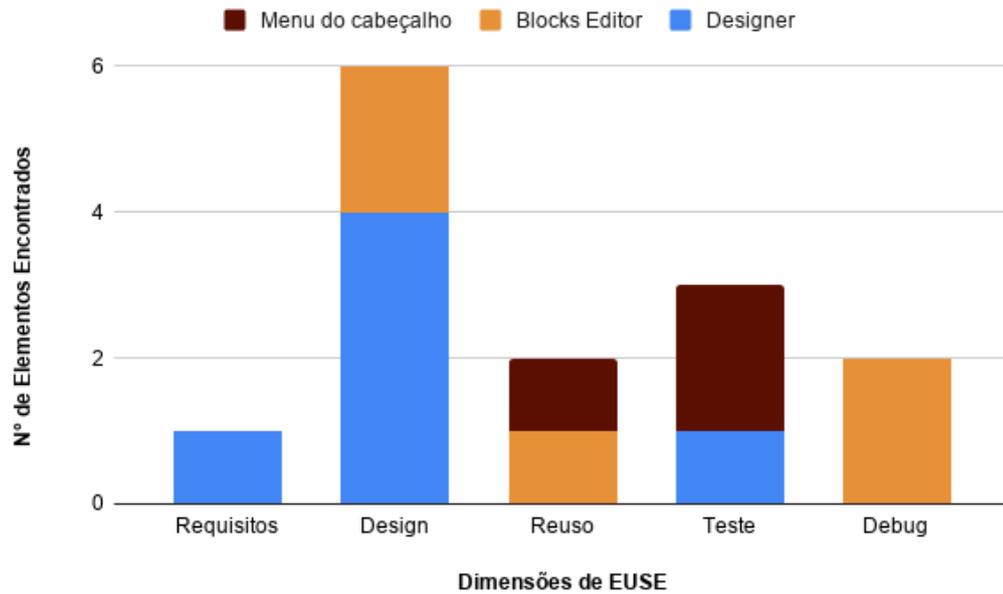
5.3 Estudo 2 (Inspeção do *App Inventor*)

O segundo estudo teve foco na identificação e categorização dos recursos que o *App Inventor* oferece para viabilizar que os usuários finais executem atividades relacionadas a ES dentro da plataforma. A descrição inicia apresentando uma visão geral dos achados e faz, posteriormente, um detalhamento com base em cada dimensão de ES.

A Figura 13 sintetiza os achados da inspeção sobre a interface do *App Inventor*. Cada barra representa uma dimensão avaliada de EUSE em relação ao total de elementos observados em cada uma das telas analisadas. É importante ressaltar que a barra marrom no gráfico representa os elementos presentes no menu do cabeçalho das páginas, que é uma parte invariável das telas.

Isso significa que um mesmo elemento está disponível na tela de *Designer* assim como está no *Blocks Editor*.

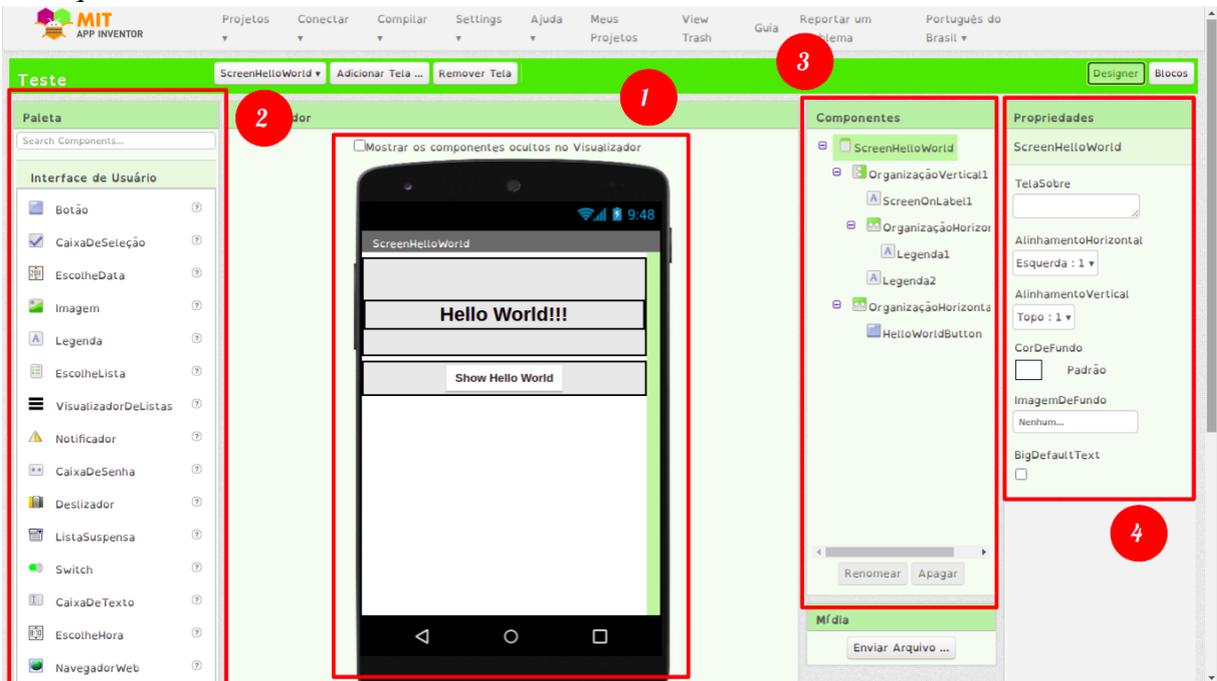
Figura 13 – Proporção dos achados durante a etapa de análise da interface



Fonte: Próprio autor (2022).

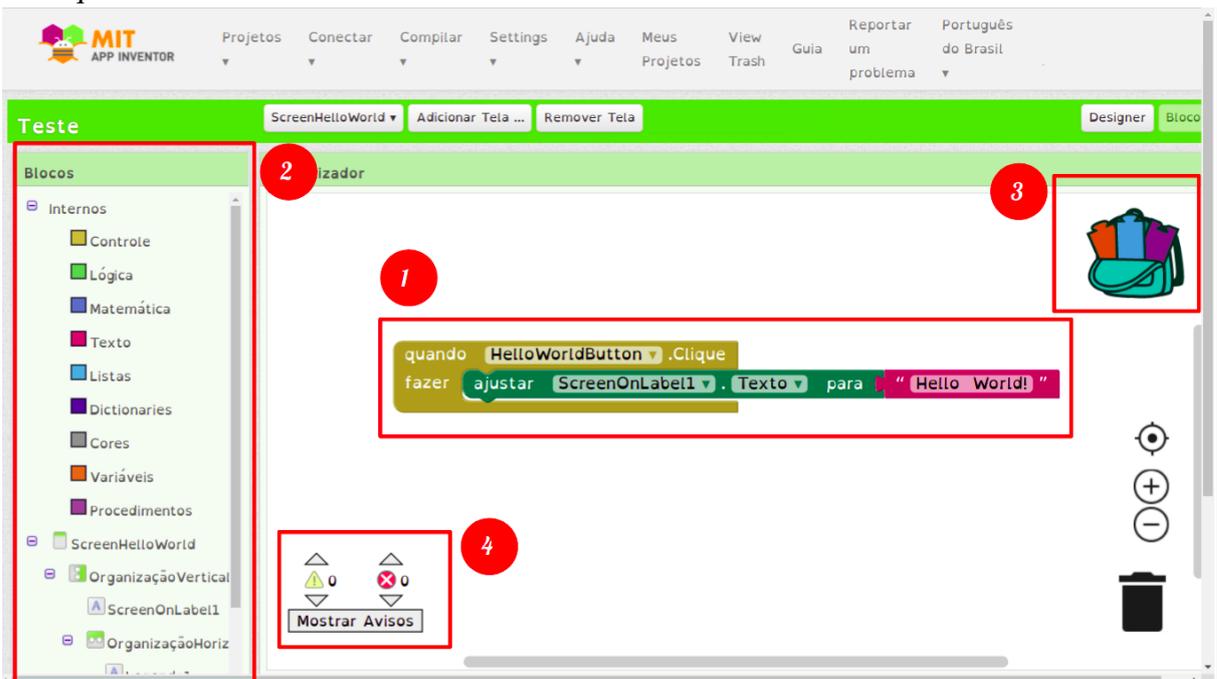
As Figuras 14 e 15 representam as janelas do *Designer* e do *Blocks Editor* respectivamente. Destacamos algumas regiões nas imagens para facilitar a identificação dos elementos que já são visíveis naturalmente. Aqueles que precisam de alguma interação para aparecer são mostrados em imagens separadas ao longo da exposição dos resultados. Em seguida, os elementos são descritos focados em cada dimensão de EUSE apresentada anteriormente.

Figura 14 – A interface do *App Inventor* mostrando a tela do *Designer* com elementos em destaque



Fonte: Próprio autor (2022).

Figura 15 – A interface do *App Inventor* mostrando a tela do *Blocks Editor* com elementos em destaque



Fonte: Próprio autor (2022).

a) Requisitos

Iniciando pela atividade de requisitos, é possível notar que esta atividade não recebe tanto apoio da ferramenta, a análise identificou apenas um elemento relacionado a esta dimensão. Isso significa que o usuário deve trabalhar esta etapa mais detalhadamente de uma outra forma, fora do ambiente da ferramenta.

- Tela de Smartphone. A plataforma exibe na sua área de trabalho um *container* no formato da tela de um smartphone para o usuário poder arrastar e soltar os componentes em seu interior, como mostrado no destaque 1 da Figura 14. Desta forma, é possível visualizar como a aplicação vai se moldando à medida que o usuário adiciona, modifica ou remove componentes. Assim, a ilustração do smartphone está relacionada à dimensão de requisitos porque permite que o usuário visualize e reflita sobre os elementos que vão compor o seu sistema.

b) Design

Em relação a atividade de *design*, esta análise identificou o maior número de elementos relacionados, sendo quatro achados na tela do *Designer* e três no *Blocks Editor*.

- *Paleta de Componentes*. Oferece uma variedade de componentes para o usuário utilizar no *design* da aplicação, como mostra o destaque 2 da Figura 14. Este elemento permite ao usuário selecionar e arrastar o componente desejado para a área de trabalho/visão. A paleta disponibiliza componentes como botão, legenda, caixa de texto, estilos de organização de layout, além de fornecer acesso a recursos nativos como câmera, microfone, entre diversos outros. Estes recursos são dispostos em um menu do tipo acordeon, facilitando a organização dos componentes por tipo e permitindo que o usuário final visualize apenas os componentes do grupo que ele deseja no momento.

- *Ajuda dos Componentes*. Cada componente oferece uma opção de informações sobre ele, auxiliando o usuário sobre a finalidade daquele componente (Figura 16). Esta ajuda é importante no apoio ao processo de aprendizado ao mesmo tempo que auxilia no processo de *design*. Este elemento permite que o usuário conheça o componente ou saiba mais sobre ele, influenciando assim nas suas decisões sobre como o seu sistema irá se comportar.

- *Árvore de Componentes*. Este item apresenta os elementos utilizados pelo usuário para compor a aplicação. Os elementos são dispostos em forma de hierarquia, facilitando a identificação de como eles estão organizados entre si, como mostra o destaque 3 da Figura 14. Através da árvore de componentes também é possível renomear os elementos, para melhorar a identificação ao longo do desenvolvimento, ou então, remover componentes que foram

Figura 16 – Caixa de ajuda sobre o componente



Fonte: Próprio autor (2022).

adicionados na aplicação.

- *Paleta de Propriedades*. Possibilita a modificação de uma variedade de propriedades dos elementos incluídos na interface da aplicação, como texto, alinhamento e posição, entre outras coisas, alterar suas propriedades visuais como vemos no quarto destaque da Figura 14.

Analisando a segunda tela, o *Blocks Editor*, ainda em relação a dimensão de *design*, destacamos os seguintes elementos que consideramos estar diretamente relacionados a essa dimensão.

- *Formato dos Blocos*. O formato de encaixe dos blocos é uma forma de orientar o usuário na lógica de programação e evitar erros. Como mostrado na Figura 15 no primeiro destaque, quando a lógica se mostra correta, os blocos se encaixam perfeitamente. O mesmo não ocorre quando há um erro na lógica de programação. Este elemento por sua vez, pode ser relacionado não apenas às atividades de *design* como também às de depuração, pois uma falha no encaixe dos blocos pode ser facilmente identificada e corrigida.

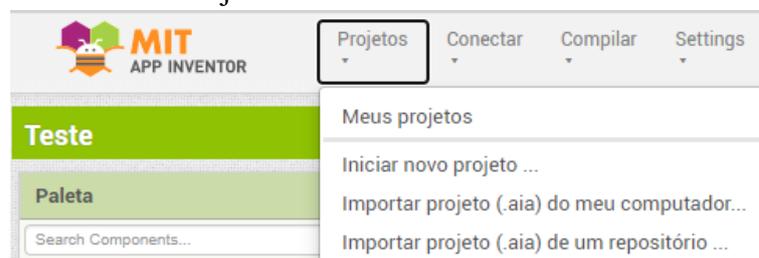
- *Separação dos Blocos*. A paleta de blocos mantém os blocos separados por tipo e com cores diferentes conforme a lógica de programação que os blocos apresentam, como mostra o destaque número 2 da Figura 15, facilitando na identificação e no raciocínio de programação do usuário, já que, a partir daí, ele pode refletir sobre variáveis, estruturas de controle, entre outros.

c) Reúso. Os achados relacionados a reuso estão principalmente associados a código (blocos) já produzidos.

- *Mochila*. Localizado na parte superior direita da tela de edição de blocos, este elemento inicialmente se apresenta como uma mochila fechada, à medida que se adiciona blocos dentro dela, o ícone se transforma, exibindo uma mochila cheia de blocos como mostra o terceiro destaque da Figura 15. Entretanto, quando algum bloco é arrastado sobre esse ícone, ele se modifica, e exibe uma mochila aberta e vazia, indicando que alguma coisa pode ser adicionada lá dentro. Este elemento permite que os usuários armazenem trechos de código produzidos para serem reutilizados em outras telas ou outros projetos.

- *Importar Projetos*. As opções que estão presentes na barra de menu permitem que o usuário importe um projeto já desenvolvido anteriormente através de duas alternativas. Como apresentado na Figura 17, o usuário pode carregar um projeto presente na própria máquina ou a partir de um repositório de projetos existentes, seja ele disponibilizado pela plataforma ou através de repositório de seu interesse. Desta forma, o usuário pode reutilizar algo que já fez ou que outra pessoa tenha feito e modificá-lo de forma a atender suas necessidades, podendo fazer reuso de algo que já foi testado e aprovado anteriormente.

Figura 17 – Menu Projetos



Fonte: Próprio autor (2022).

d) Verificação e Testes

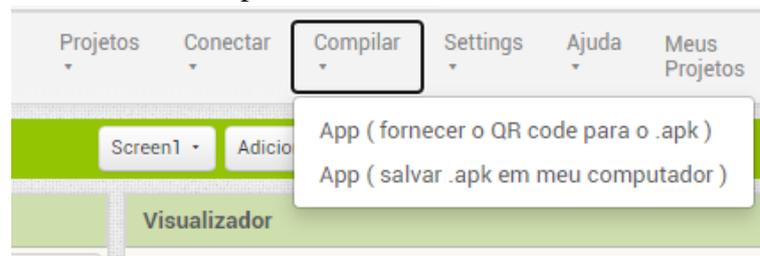
Relacionado a verificação e testes, a inspeção identificou três elementos que apoiam o usuário em atividades associadas a esta dimensão de ES. Os elementos estão presentes no cabeçalho e na tela do *Designer*. Porém, estes itens direcionam o usuário a uma atividade fora do ambiente da ferramenta, já que se relacionam com o *deploy* da aplicação, isto é, o usuário vai avaliar a execução do seu sistema em um dispositivo que está externo ao App Inventor.

- *Informações de Versão*. A paleta de propriedades, já comentada anteriormente, além de conter campos que permitem modificar as propriedades dos componentes, alguns campos são usados para realizar o controle do versionamento das telas da aplicação. Para isso, o usuário pode determinar algumas informações, tais como o número da versão e um nome para ela, facilitando a identificação de versões dos artefatos produzidos, assim como os desenvolvedores

profissionais fazem na gerência de configuração dos sistemas. Desta forma, o usuário pode ter o controle de quando uma determinada tela foi adicionada ou alterada e isto pode ser útil caso o usuário queira resgatar uma versão anterior.

- *Menu de Compilação*. Este menu está presente no cabeçalho de ambas as telas do App Inventor, fornecendo duas opções distintas para o usuário construir o arquivo de instalação da aplicação (Figura 18). Uma delas é através de um QR code onde é possível fazer a leitura e instalação automática da aplicação diretamente no *smartphone*, enquanto a outra opção permite a geração de um arquivo no formato apk para download no computador. A partir disso, os usuários podem utilizar e testar suas aplicações com a finalidade de validar o sistema construído e encontrar possíveis falhas.

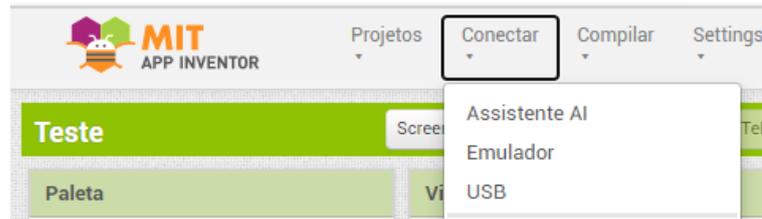
Figura 18 – Menu Compilar



Fonte: Próprio autor (2022).

- *Menu de Conexão*. Este elemento, também presente no menu do cabeçalho do App Inventor, oferece diferentes opções para que o usuário defina a forma como deseja executar de forma rápida a aplicação desenvolvida. Como mostra a Figura 19, o usuário pode escolher entre utilizar o *companion*, uma aplicação desenvolvida pelo próprio MIT, no qual é possível realizar a emulação da aplicação por meio da leitura de um código ou QR code gerado pela plataforma. Isto é feito através de um sistema emulador instalado no próprio computador ou utilizando um aparelho de *smartphone* conectado por um cabo USB. Similar ao menu de compilação, este elemento permite que o usuário utilize a aplicação desenvolvida e possa procurar por defeitos que venham a dificultar o seu uso. Podemos notar que os elementos relacionados a teste direcionam o usuário para um ambiente externo ao App Inventor de modo que os resultados referentes às ações feitas durante os testes não podem ser identificados dentro da plataforma.

Figura 19 – Menu Conectar



Fonte: Próprio autor (2022).

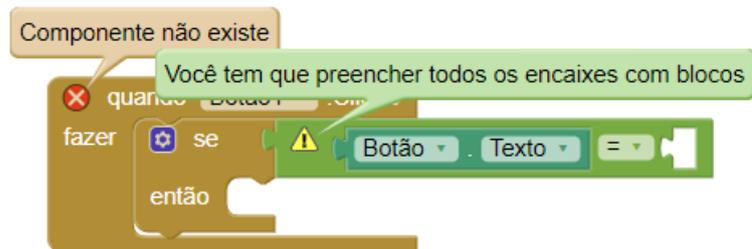
e) Debug

Por sua vez, os achados relacionados a debug apresentam resultados visíveis dentro das telas do sistema. Em relação a esta atividade, foram encontrados dois elementos que auxiliam o usuário nessa fase do desenvolvimento. Esses elementos se encontram na tela do editor de blocos, já que estão diretamente relacionados à lógica montada através dos blocos.

- *Ícones de Warnings e Erros*. Presentes na tela de montagem dos blocos, estes ícones são usados para indicar que tem algum problema na lógica dos blocos, seja um simples aviso, representado pelo triângulo amarelo, ou destacando erros, por meio do círculo vermelho, como apresentado no destaque 4 da Figura 15.

- *Mensagens de Avisos*. Essas informações podem ser acessadas quando o usuário clica sobre “Mostrar Avisos” presente nos ícones de *warning* e erros comentados anteriormente. Após isso, o sistema destaca um pequeno triângulo amarelo de aviso em blocos que apresentam problemas. Ao clicar sobre esse novo símbolo em algum bloco, o sistema exibe um texto de ajuda explicando o que deve ser feito para corrigir o problema, como é apresentado na Figura 20.

Figura 20 – Mensagens sobre falhas no código



Fonte: Próprio autor (2022).

Esse elemento tem a finalidade de ajudar o usuário não apenas na identificação de problemas presentes ao longo do código, como auxiliar sobre as ações que o usuário deve realizar para corrigir as falhas.

6 DESENVOLVIMENTO DA ASSISTENTE VIRTUAL

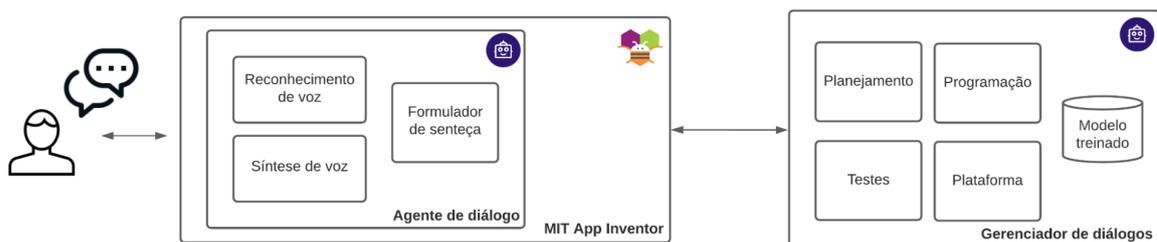
O planejamento para a assistente virtual, descrito nas próximas seções, foi baseado na arquitetura apresentada em Weigelt *et al.* (2018), que foi integrada a uma estrutura já existente apresentada em Weigelt e Tichy (2015). Nossa principal diferença é a integração de uma nova estrutura que seja capaz de abordar atividades de EUSE de forma a auxiliar os usuários finais a melhorar sua forma de desenvolvimento.

6.1 Melhorias no planejamento da assistente virtual

Com base nas descobertas dos primeiros estudos, foi possível obter *insights* que refletiram na construção da assistente virtual. O primeiro estudo serviu para trabalhar e melhorar a parte de interação e comunicação do usuário com o assistente, assim como do assistente para com o usuário. Já o segundo estudo, forneceu uma base para entender o que poderia ser feito na plataforma e como organizar o assistente para melhorar a forma de apoiar o usuário final nos seus objetivos com as ferramentas que o *App Inventor* oferece.

Desta forma, foi elaborado uma arquitetura a ser desenvolvida para este trabalho (Figura 21). A estrutura é composta por duas partes principais: o **Agente de Diálogos**, estrutura anexada dentro do sistema do *App Inventor*; e o **Gerenciador de Diálogos**, uma estrutura externa responsável por fornecer a lógica por trás da comunicação.

Figura 21 – Estrutura do assistente virtual



Fonte: Próprio autor (2022).

Para entender melhor o modelo apresentado, deve-se começar pelo Agente de Diálogos, que é responsável por receber e processar a conversa com o usuário final. Após reconhecer a entrada fornecida pelo usuário, esse agente deve usar o formulador de sentença para processar a instrução e repassar o comando gerado para o Gerenciador de Diálogos por meio de uma Interface de programação de aplicações (API), onde ele irá fornecer as informações necessárias para

auxiliar o usuário nos seus objetivos. Ao receber uma informação, o Gerenciador de Diálogos consulta um modelo treinado previamente e fornece uma resposta como resultado referente a planejamento, programação, testes ou informações gerais sobre a plataforma. A Tabela 6 mostra a relação dessas áreas com as dimensões de EUSE abordadas neste estudo.

Tabela 6 – Relação entre área x dimensão abordada

Área	Dimensão	EUSE
Planejamento	Requisitos/ <i>Design</i>	Sim
Programação	Reuso	Sim
Testes	Verificação e testes/ <i>Debug</i>	Sim
Plataforma	<i>App Inventor</i>	Não

Fonte: Elaborado pelo autor (2022).

A implementação e funcionamento destes dois componentes, o Agente de Diálogos e o Gerenciador de Diálogos, serão detalhadas a seguir.

6.2 Implementação da estrutura do assistente virtual

Como uma forma de permitir que mais pessoas contribuam com seu projeto e/ou que essas pessoas modifiquem o sistema com base em suas necessidades, é possível criar novas instâncias da aplicação do *App Inventor* utilizando o código-fonte disponibilizado no repositório do *github* através da url “<https://github.com/mit-cml/appinventor-sources>”.

Partindo desse código original, para possibilitar o uso do assistente virtual dentro do *App Inventor* e assim conseguir auxiliar os usuários finais no desenvolvimento, foram desenvolvidos os seguintes componentes:

- Modificação na interface do *App Inventor* para permitir acionar o assistente virtual
- Modificação no *back-end* do *App Inventor* para troca de informações com a API da assistente virtual
- Criação do gerenciador de diálogos (assistente virtual)

Com o objetivo de familiarizar os leitores com as atividades desenvolvidas, as seções a seguir descrevem esses recursos de forma detalhada.

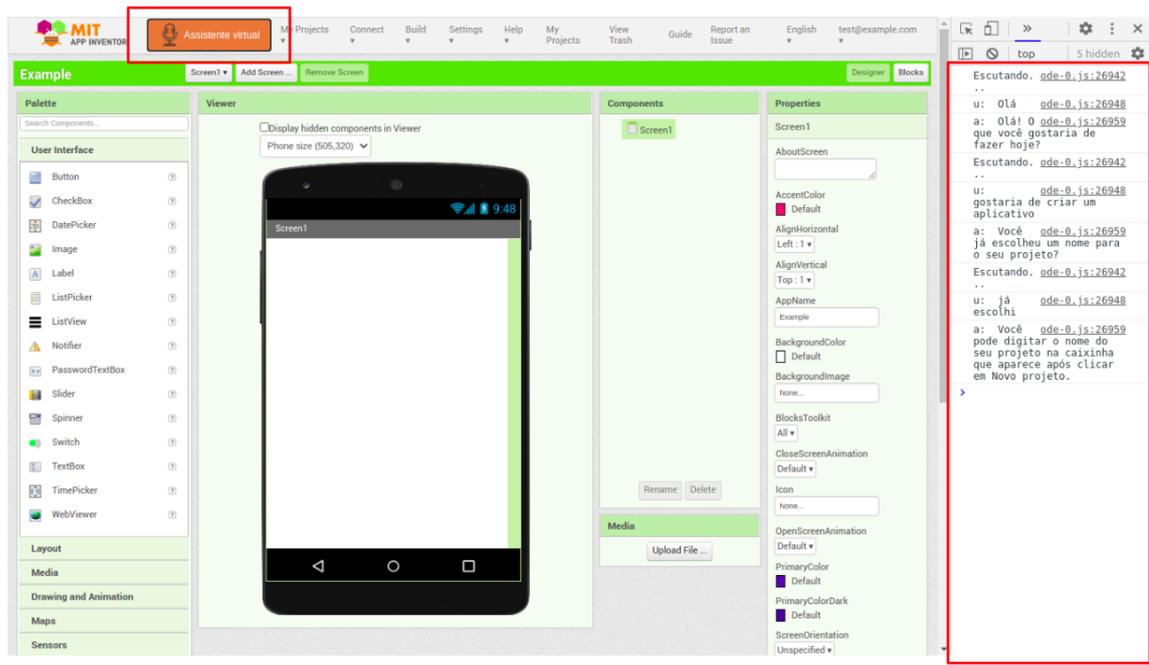
6.2.1 Modificação na interface do *App Inventor*

De início, duas alternativas foram levadas em consideração. A primeira delas era a possibilidade da assistente permanecer escutando qualquer fala do usuário durante todo o tempo

de uso da plataforma. Porém, devido ao fato de que o Agente de diálogos poderia interpretar qualquer comunicação que não fosse direcionada para ele como uma solicitação de ajuda e fornecer respostas que fossem desconexas para a situação não seria o ideal, além da falta de um indicador visual da existência de tal recurso dentro da plataforma. A segunda alternativa era a inclusão de um botão que ficasse facilmente visível ao usuário e que ele poderia solicitar ajuda apenas quando sentisse a necessidade de apoio.

A Figura 22 mostra a interface do *App Inventor* após a inclusão do novo botão, localizado no menu superior da tela, tentando tornar mais explícita a existência dessa nova funcionalidade. Quando o botão é clicado, o sistema ativa o microfone do computador permitindo que o usuário se comunique com a assistente.

Figura 22 – Interface modificada do *App Inventor*



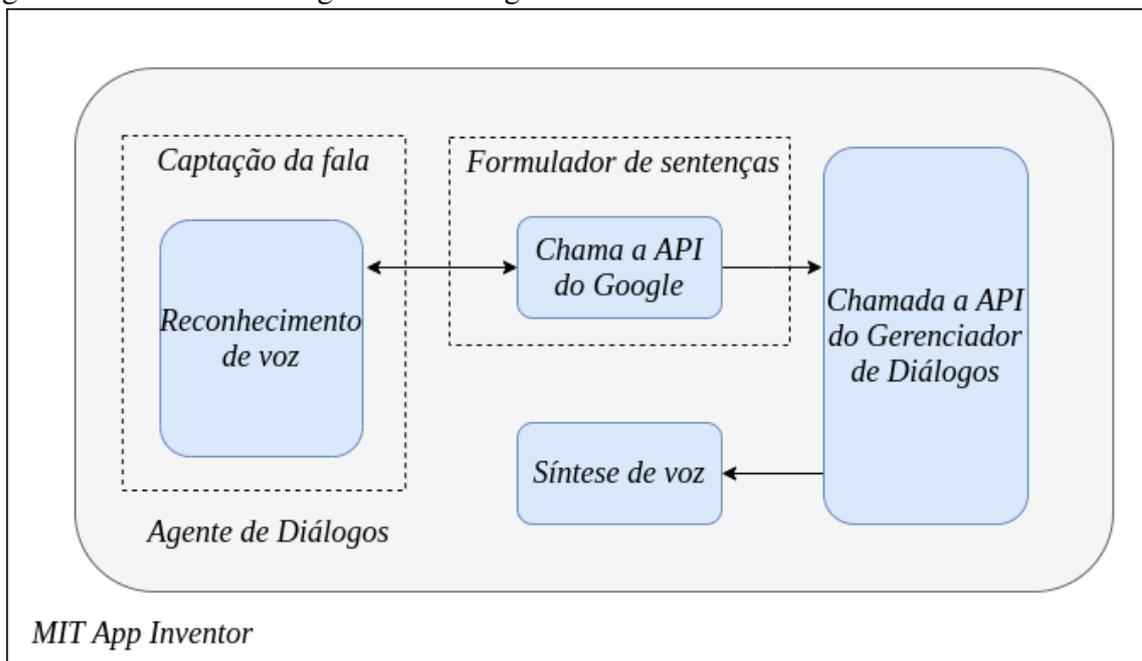
Fonte: Próprio autor (2022).

Outro ponto a ser destacado foi a inclusão de um histórico da conversa entre o usuário e a assistente, como pode ser visto do lado direito da Figura 22. Desta forma, caso sentisse necessidade, o usuário poderia revisitar algum ponto da conversa que fosse do seu interesse. De início, esse histórico fica disponível apenas no console do navegador, com a possibilidade de futuramente ser incluído em alguma parte do próprio *App Inventor*.

6.2.2 *Agente de Diálogos*

O Agente de Diálogos é uma estrutura dentro do código do próprio *App Inventor* que executa as seguintes etapas: captação da fala; formulação da sentença; repasse da sentença para o gerenciador e devolução de resposta para o usuário. A Figura 23 apresenta a estrutura desenvolvida para esse agente.

Figura 23 – Estrutura do Agente de Diálogos



Fonte: Próprio autor (2022).

Após o usuário clicar no botão e terminar sua fala, os dados são captados pelo navegador e enviados para uma API do *Google* que faz a transcrição da fala do usuário e retorna uma sentença de texto para a nossa aplicação. Essa sentença é então enviada para a nossa API da assistente virtual (Gerenciador de Diálogos) que irá retornar alguma informação. Essa informação chega em forma de texto para o Agente de Diálogos que utiliza uma ferramenta nativa do navegador para transformar o texto em fala. Vale ressaltar que o histórico da conversa entre usuário e assistente fica disponível no console do navegador, o que permite ao usuário revisitar alguma parte da interação se assim desejar. Por mais que a existência desse histórico não fique explícita para o usuário, foi um ponto apresentado pelo pesquisador durante a realização das avaliações como mais uma forma de apoio ao desenvolvimento.

6.2.3 Gerenciador de Diálogos

Como a intenção de direcionar o diálogo da forma mais adequada para os objetivos deste projeto, o Gerenciador é responsável não apenas por fornecer respostas para as dúvidas que os usuários apresentam, seu dever também é de agregar novas informações que possibilitem a melhoria do processo de desenvolvimento. Para isso, além de responder as dúvidas, é de responsabilidade do Gerenciador introduzir os conceitos de EUSE abordados neste trabalho.

Para entender o desenvolvimento desse componente, devemos iniciar pela elaboração do modelo de linguagem natural que serve como base para os diálogos da nossa assistente virtual. Para construção desse modelo, foi utilizado o Rasa, um *framework* de criação de bots para diversas plataformas, sendo um *framework* eficiente, simples e gratuito.

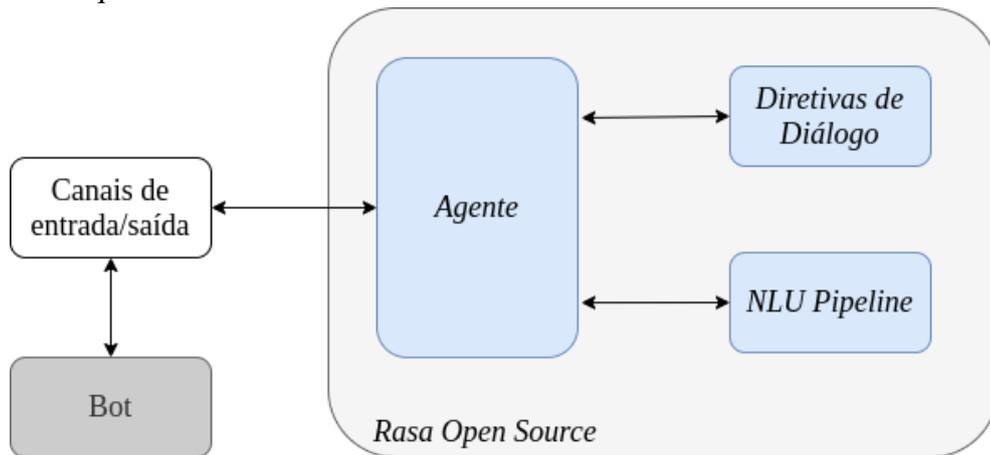
Rasa Framework

O *Framework* Rasa é uma estrutura de aprendizado de máquina de código aberto para automatização de conversas, consistindo de componentes para pré-processamento do conjunto de dados de treinamento, classificação de intenção, extração de entidades e seleção de respostas. Contendo um *pipeline* relativamente flexível, onde componentes personalizados podem ser adicionados ou modelos existentes podem ser modificados dependendo dos requisitos dos usuários.

A Figura 24 apresenta uma simplificação da arquitetura do Rasa *Framework* com seus componentes principais, o Entendimento de Linguagem Natural (NLU) e o Gerenciamento de diálogo. o NLU é a parte responsável por lidar com a classificação de intenções e recuperação de respostas. Na imagem ele é mostrado como o *NLU Pipeline* porque processa os enunciados do usuário usando um modelo NLU que é gerado pelo pipeline treinado. Por sua vez, o componente de gerenciamento de diálogo decide a próxima ação em uma conversa com base no contexto, ele é exibido como as Diretivas de Diálogo no diagrama. Já a figura do Agente, é o responsável por fazer a ligação entre os processamentos e os canais de entrada e saída.

Em sistemas de linguagem natural, um dos grandes desafios é a correta identificação da intenção ou desejo de consulta do usuário, esta é uma etapa crítica pois reflete na qualidade da resposta retornada e consequentemente na satisfação do usuário. No Rasa, uma intenção é definida por uma coleção de conjuntos de sentenças de treinamento. Cada conjunto tem o mesmo propósito mas pode apresentar grande variação na forma como está estruturada a sentença, por exemplo: “Como fazer para testar a minha aplicação?” e “Gostaria de testar o

Figura 24 – Arquitetura do Rasa Framework



Fonte: Próprio autor (2022).

aplicativo”. Embora as sentenças sejam diferentes, elas possuem o mesmo objetivo que seria testar a aplicação. Exemplos de intenções são apresentados na Tabela 7.

Tabela 7 – Exemplos do formato de uma intenção

Intenção/ <i>Intent</i>	Sentença/ <i>Query</i>
greet	“Oi”, “Olá”
initial_dialog	“Quero criar um aplicativo”

Fonte: Elaborado pelo autor (2022).

Nosso *dataset* de intenções foi construído baseado nos dados obtidos nos primeiros estudos apresentados anteriormente, envolvendo diálogos, observações e inspeções. Embora as intenções tenham sido geradas pensando em cada dimensão de EUSE, algumas dimensões receberam uma atenção maior, como requisitos e *design*, devido ao escopo inicial que seria trabalhado nos testes. Mesmo assim, os casos mais comuns para reuso, testes e *debug* foram adotados para não deixar totalmente sem respostas caso os usuários considerassem falar sobre esses pontos.

As intenções definidas foram alocadas no arquivo chamado *nlu.md* como determina o *framework* do Rasa. A Figura 25 mostra um exemplo de intenção para saudação inicial com suas possíveis variações de sentenças que poderiam ser utilizadas pelos usuários para expressar um mesmo significado. Esse é um ponto importante para tentar não limitar a comunicação entre usuário e *bot* a determinadas frases, sendo assim, definir um maior número possível de sentenças para cada intenção acarreta na qualidade do treinamento e posteriormente na qualidade da interação com os usuários.

Para cada intenção identificada do usuário foi criada uma resposta para ser devolvida

Figura 25 – Exemplo do arquivo de intenções

```

nlu.md
data > nlu.md > ## intent:affirm_2
    You, a year ago | 1 author (You)
1  ## intent:greet
2  - oi
3  - olá
4  - oie
5  - bom dia
6  - boa tarde
7  - boa noite
8  - tem alguém ai
9  - ooi
10 - oii
11
    You, a year ago | 1 author (You)
12 ## intent:goodbye
13 - tchau
14 - adeus
15 - xau
16 - até mais
17 - te vejo depois
18

```

Fonte: Próprio autor (2022).

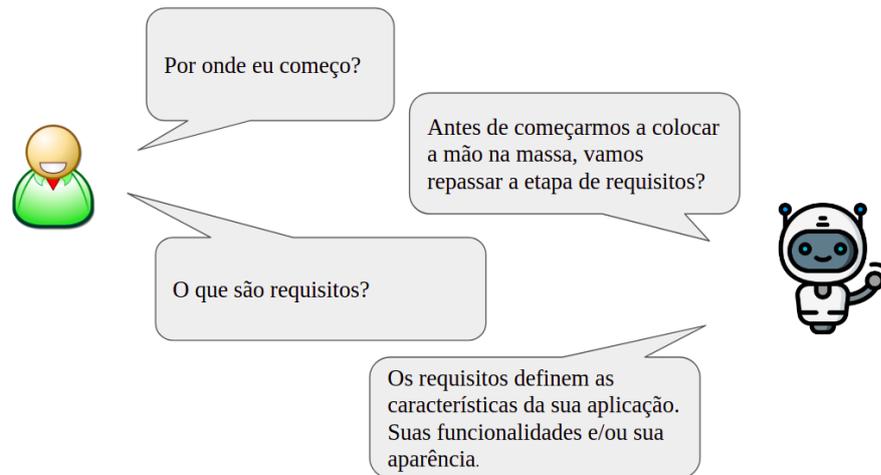
pelo assistente. Essas respostas eram compostas por uma chave de identificação, que por uma questão de padrão do *framework* é iniciada com *utter* (uma abreviação do termo *utterance*), seguida pelas palavras que completam essa chave, e em seguida, pelo texto a ser retornado como mostra o exemplo de resposta a seguir:

utter_remember_requirements:

- text: “Antes de começarmos a colocar a mão na massa, vamos repassar a etapa de requisitos?”

Para incentivar os usuários a pensarem nas dimensões de EUSE, as respostas foram criadas para mencionar práticas ou assuntos relacionados a esse tema à medida que fosse possível, pensando em diálogos naturais que poderiam ocorrer ao longo do processo de desenvolvimento como mostrado no exemplo de uma conversa na Figura 26.

Figura 26 – Exemplo de possível conversa entre usuário e assistente



Fonte: Próprio autor (2022).

Nesse estudo, 108 respostas foram utilizadas com foco nas etapas de requisitos e *design*, além de respostas gerais sobre o uso da ferramenta, compondo o arquivo *domain.yml* do *framework* (Figura 27).

Figura 27 – Exemplo do arquivo *domain.yml*

```

domain.yml x
domain.yml
09
70 responses:
71
72 #dialogo inicial
73 utter_greet:
74 - text: Olá! O que você gostaria de fazer hoje?
75
76 utter_question_about_name:
77 - text: Você já escolheu um nome para o seu projeto?
78
79 utter_suggest_name_to_project:
80 - text: Entendi. Vamos chamá-lo de 'MyApp' enquanto pensa em um nome legal para ele, pode ser?
81
82 utter_confirm_type_project_name:
83 - text: Você pode digitar o nome do seu projeto na caixinha que aparece após clicar em Novo projeto.
84

```

Fonte: Próprio autor (2022).

As histórias (ou *Stories*) representam outro componente importante do Rasa *framework*, sendo usadas para projetar o fluxo de conversação do bot para o treinamento dos módulos de gerenciamento de diálogo. As histórias são essencialmente trechos de possíveis conversas que um usuário pode ter com um bot e, portanto, são importantes no treinamento das respostas do bot.

Stories, no Rasa, são uma coleção de intenções, respostas e ações, que podem ser organizadas em qualquer ordem para criar um fluxo de conversação. Um *story* é formado pela chave de identificação da intenção do usuário seguida pela chave de identificação da resposta do *bot*, como mostra a Tabela 8. Podendo ter diferentes tamanhos, desde os mais curtos como apenas uma pergunta e uma resposta, até diálogos maiores sobre como trabalhar requisitos, por exemplo.

Tabela 8 – Exemplo do formato de um *story*

<i>story_short</i>	<i>story_long</i>
*Greet - Bot responds with greet	*Greet - Bot responds with greet *Ask general or contextual question - Bot returns answer *Goodbye - End session

Fonte: Elaborado pelo autor (2022).

Diante disso, as Tabelas 9, 10, 11, 12 e 13 mostram exemplos de diálogos criados para cada uma das dimensões de EUSE que foram posteriormente transformados em *stories*. Um total de 48 *stories* foram feitos para esse estudo. Em seguida, esses *stories* foram armazenados no arquivo *stories.md* do Rasa *framework* (Figura 28).

Tabela 9 – Exemplo de diálogo sobre requisitos

Agente	Chave	Texto
User	intent:doubt_about_requirements	O que são requisitos?
Bot	utter_requeriment_is	Os requisitos definem as características da sua aplicação. Suas funcionalidades e/ou sua aparência. O que ela poderá ou não executar.
Bot	utter_need_example	Precisa de um exemplo?
User	intent:i_want	Preciso
Bot	utter_example_requeriments	Um exemplo de requisitos é "o sistema deve permitir que o usuário faça login antes de acessar a aplicação"
Bot	utter_good_practice	É sempre bom pensar bem nos requisitos da sua aplicação e registrá-los em algum lugar. Dessa forma, você garante que o sistema atendeu às suas necessidades, além de facilitar para futuras consultas.

Fonte: Elaborado pelo autor (2022).

Tabela 10 – Exemplo de diálogo sobre *design*

Agente	Chave	Texto
Bot	utter_how_is_your_planning	Como está sendo seu planejamento para construção do sistema? Pensa em fazer todas as telas de uma vez e só depois ir para o código ou vai criar uma tela completa com código por vez?
User	intent:my_planning_is_one_screen_at_time	Vou criar uma tela com código por vez
Bot	utter_this_is_a_good_idea	Esse é um ótimo pensamento, vai facilitar no momento do teste e em possíveis mudanças que venham a aparecer.

Fonte: Elaborado pelo autor (2022).

Tabela 11 – Exemplo de diálogo sobre reuso

Agente	Chave	Texto
User	intent:how_create_programming_logic	Como criar a lógica da aplicação?
Bot	utter_how_create_programming_logic	A lógica deve ser construída na tela do editor de blocos ao unir os blocos para criar as respostas para as ações na tela. Ao criar um trecho com uma lógica comum para outras telas, você pode adicionar esse trecho na mochila e reutilizá-lo outras vezes.

Fonte: Elaborado pelo autor (2022).

Tabela 12 – Exemplo de diálogo sobre testes

Agente	Chave	Texto
User	intent:fix_all_warnings	Já corrigi todos os avisos
Bot	utter_do_you_test_now	Isso é muito bom. Deseja testar agora?
User	intent:affirm	Sim
Bot	utter_tip_verify_spend_requirements	Ao testar sua aplicação, verifique se ela atende aos requisitos que você planejou inicialmente.
Bot	utter_explain_how_test	O <i>app inventor</i> oferece três formas para testar sua aplicação. Deseja conhecer todas?

Fonte: Elaborado pelo autor (2022).

Tabela 13 – Exemplo de diálogo sobre debug

Agente	Chave	Texto
User	intent:how_do_I_fix_warnings	como eu corrijo esses avisos?
Bot	utter_how_fix_warnings	Ao clicar sobre a caixinha de avisos, o sistema irá destacar onde estão os problemas no seu código. Basta ler e saberá o que deve fazer.

Fonte: Elaborado pelo autor (2022).

Depois de estabelecer as intenções, respostas e *stories*, o próximo passo foi realizar o treinamento do nosso modelo baseado em linguagem natural, isso ocorreu em uma máquina com processador Intel(R) Core(TM) i5-4460 CPU @2.5GHz, 4 GB de RAM utilizando a base criada. Foram executadas um total de 100 épocas²⁵ como é definido por padrão nos arquivos de configuração do Rasa *framework*. Ao final desse treinamento, já era possível iniciar uma

²⁵ Época refere-se a um ciclo completo através do conjunto de dados de treinamento.

Figura 28 – Exemplo do arquivo de respostas

```

M+ stories.md X
data > M+ stories.md > ...
1 <!-- diálogo inicial -->
2   ## path unknow name for project
3   * greet
4     - utter_greet
5   * initial_dialog
6     - utter_question_about_name
7   * negative_response_to_project_name
8     - utter_suggest_name_to_project
9   * affirm
10    - utter_confirm_type_project_name
11
12
13   ## happy path
14   * greet
15     - utter_greet
16   * initial_dialog
17     - utter_question_about_name
18   * positive_response_to_project_name
19     - utter_confirm_type_project_name
20

```

Fonte: Próprio autor (2022).

conversa com a assistente em modo texto através do terminal como mostra a Figura 29.

Figura 29 – Diálogo com *bot* por texto

```

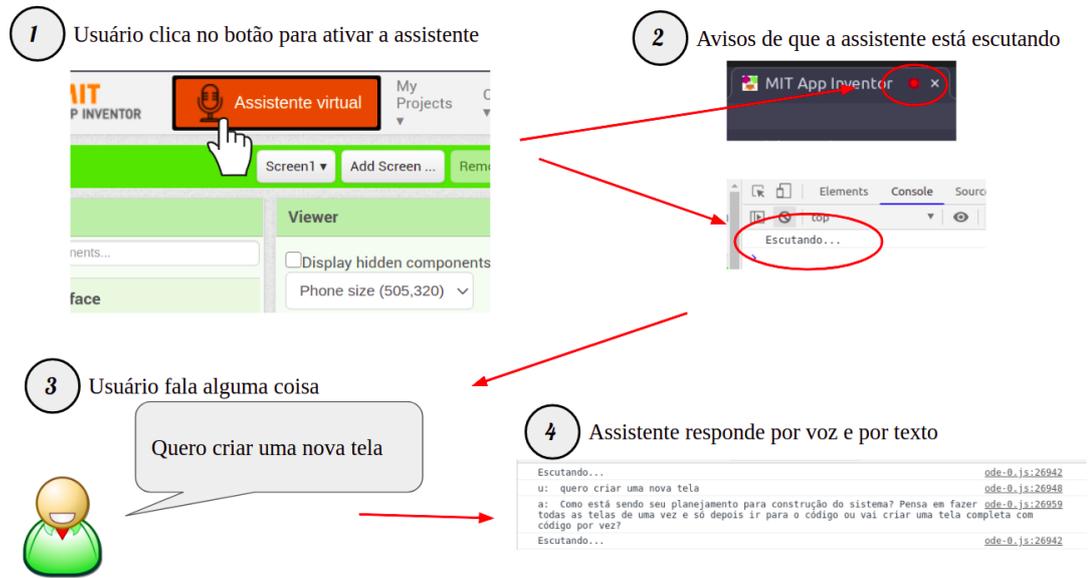
Bot loaded. Type a message and press enter (use '/stop' to exit):
Your input -> olá
Olá! O que você gostaria de fazer hoje?
Your input -> Quero criar um app
Você já escolheu um nome para o seu projeto?
Your input -> Sim
Você pode digitar o nome do seu projeto na caixinha que aparece após clicar em Novo projeto.
Your input -> ok
Your input -> █

```

Fonte: Próprio autor (2022).

O processo de treinamento e adição de novas sentenças para tentar identificar as intenções dos usuários foi repetido algumas vezes até se obter um resultado aceitável de precisão durante as conversas. Após essa etapa, o passo seguinte foi fazer a conexão entre a plataforma do *App Inventor* e o servidor com o modelo treinado. A Figura 30 apresenta o passo a passo do uso da assistente virtual por meio do *App Inventor*. Quando o usuário comunica alguma ação, a frase que é recuperada e enviada para o Gerenciador de Diálogos fica registrada no histórico da conversa, desta forma o usuário pode saber exatamente o que a assistente recebeu com intenção, ajudando nos casos em que a resposta da assistente possa estar fora de contexto. As respostas que são retornadas para o usuário, além de serem expressas pela interface de voz, também ficam armazenadas em modo texto no histórico.

Figura 30 – Tutorial de uso da assistente virtual



Fonte: Próprio autor (2022).

Os arquivos de recursos e configurações do Rasa utilizados para a criação do Gerenciador de Diálogos podem ser encontrados no repositório do *github* através do link <https://github.com/beto22ufc/voice-bot>. Já as alterações do *App Inventor* estão disponíveis pelo link <https://github.com/beto22ufc/appinventor-sources/tree/feature/voice-bot>.

7 ESTUDO FINAL

Este capítulo apresenta os resultados do estudo 3, envolvendo o uso do *App Inventor* integrado com a assistente virtual desenvolvida neste trabalho. A Tabela 14 mostra algumas informações sobre o estudo 3.

Tabela 14 – Visão geral do estudo 3

Estudo 3	
Participantes	10 estudantes
Foco	Avaliar o apoio de uma assistente virtual integrada ao <i>App Inventor</i> nas atividades de desenvolvimento
O que foi feito	Avaliação com os usuários utilizando a assistente desenvolvida Análise do questionário Análise dos vídeos Análise dos artefatos

Fonte: Elaborado pelo autor (2022).

O estudo 3 proporcionou *insights* a respeito do pensamento e das formas como um usuário final pode e precisa usar o apoio de um assistente virtual em relação a atividades de programação. Um total de 10 participantes realizaram o experimento, com idades entre 18 e 23 anos, sendo 70% autodeclarados do sexo masculino e 30% autodeclaradas do sexo feminino. Os resultados obtidos foram extraídos por meio das respostas do questionário online (Apêndice E) criado na plataforma *Google Forms*, das observações realizadas ao longo dos testes, das análises dos vídeos e dos artefatos gerados.

Iniciando pela análise dos artefatos produzidos durante a avaliação. A Tabela 15 apresenta uma síntese das informações obtidas ao analisar os aplicativos criados. Nela estão expressas a quantidade de interfaces de usuários (telas) criadas e a quantidade de telas com a lógica dos blocos montadas (blocos). Mostrando ainda as dimensões que foram abordadas durante o desenvolvimento desses artefatos.

Podemos observar que mesmo durante o curto período que envolve a avaliação, foi possível abordar boa parte das dimensões com os participantes. Ressaltando que as abordagens nessas dimensões não necessariamente tinham como ponto de partida uma fala do usuário sobre aquele tema, por exemplo, uma dúvida em como testar já poderia ser um gatinho para a assistente abordar temas como requisitos ou atividades de depuração.

A Tabela 16 mostra uma relação entre o número de vezes que o participante solicitou ajuda para a assistente virtual e o número de vezes que cada dimensão foi abordada durante essas

Tabela 15 – Relação de artefatos criados e dimensões abordadas no Estudo 3

Participante	Telas	Blocos	Dimensões abordadas				
			Requisitos	Design	Reuso	Verificação e Testes	Debug
P1	2	1	X	X		X	X
P2	2	0		X		X	
P3	2	2	X	X	X	X	X
P4	1	0	X	X		X	
P5	2	1	X	X		X	
P6	1	0		X		X	
P7	3	0	X	X	X	X	
P8	2	0	X	X		X	
P9	1	0	X	X		X	
P10	1	0	X	X		X	
Total	17	4	-	-	-	-	-

Fonte: Elaborado pelo autor (2022).

conversas com base na análise dos vídeos da avaliação. Desta forma podemos identificar o uso da assistente virtual na construção dos artefatos citados anteriormente.

Tabela 16 – Relação de uso da assistente

Participante	Chamadas a AV	Dimensões abordadas				
		Requisitos	Design	Reuso	Verificação e Testes	Debug
P1	8	1	4	0	1	2
P2	5	0	4	0	1	0
P3	9	1	3	1	3	1
P4	4	0	3	0	1	0
P5	6	1	3	0	2	0
P6	5	0	3	0	2	0
P7	6	1	2	1	2	0
P8	6	1	4	0	1	0
P9	5	2	2	0	1	0
P10	5	1	3	0	1	0
Total	59	8	31	2	15	3

Fonte: Elaborado pelo autor (2022).

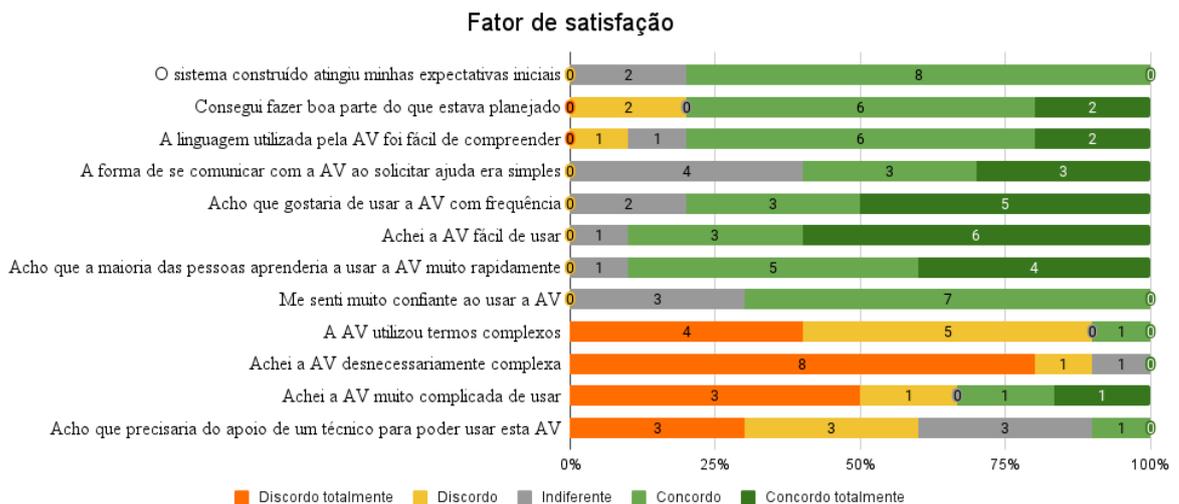
Os resultados mostram que P1, P2, P5, P7 e P8 foram os participantes que mais solicitaram ajuda para a assistente virtual, e que esses mesmos participantes, foram os que mais produziram artefatos levando em consideração interfaces de usuário e montagem de blocos. Outro ponto a destacar é a dimensão de *design* que se mostrou a mais abordada, isso pode ser explicado pelo próprio paradigma da plataforma por ter essa abordagem mais visual para se trabalhar os componentes, fazendo com que as maiores dúvidas dos usuários sejam refletidas nessa dimensão. Como por exemplo, dúvidas desde a definição do que vai compor as telas até a forma correta de usar os componentes da interface.

Em relação aos dados extraídos do questionário, estes foram agrupados em 3 cate-

gorias para entender melhor alguns pontos relacionados às questões de pesquisa deste trabalho. Dentre essas categorias estão fator de satisfação, fator de suporte e fator de conhecimento. Fator de satisfação está relacionado ao sentimento do usuário com relação a interação, alcance de suas expectativas e como ele se sentiu com a ajuda dessa ferramenta. Fator de suporte diz respeito à assistência prestada pela assistente virtual. Por último, o fator de conhecimento está relacionado a percepção de aprendizagem do usuário final.

A Figura 31 apresenta os resultados conforme as respostas do questionário em relação às perspectivas e percepções dos participantes em relação à satisfação no uso da assistente virtual. Como já mencionado, para avaliar a satisfação, levamos em consideração as perguntas relacionadas às expectativas em relação ao desenvolvimento e a interação do participante com a assistente virtual.

Figura 31 – Resultado do questionário com base no fator de satisfação



Fonte: Próprio autor (2022).

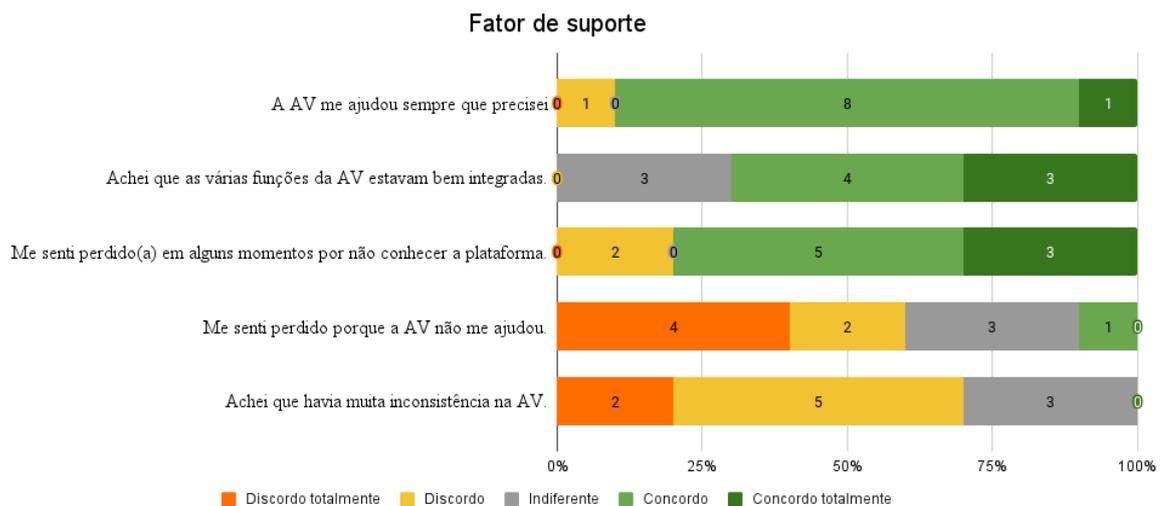
A avaliação da satisfação dos participantes foi relativamente positiva. Avaliando os aspectos de expectativas e interação, a grande maioria das respostas se concentrou em faixas positivas ou de neutralidade como mostrado na imagem anterior.

Em relação às expectativas dos participantes “O sistema construído atingiu minhas expectativas iniciais” e “Consegui fazer boa parte do que estava planejado”, podemos destacar que grande parte declarou que o sistema construído atingiu suas expectativas iniciais e que conseguiram fazer o que estava planejado. Além disso, destaque para a afirmação “Me senti muito confiante ao usar o assistente virtual”, o resultado positivo mostra que a ajuda da assistente impactou na forma como o participante se sentiu ao longo do desenvolvimento.

Nos aspectos relacionados à interação “Achei o assistente virtual fácil de usar”, “A linguagem utilizada pelo assistente virtual foi fácil de compreender” e “a forma de se comunicar com o assistente virtual ao solicitar ajuda era simples”, o gráfico mostra que a maioria dos participantes não demonstrou dificuldades em interagir com a assistente, exceto um único participante que discordou no ponto em relação à linguagem e termos utilizados pela assistente. Por outro lado, a avaliação na forma de se comunicar, do participante para com a assistente se manteve, na maior parte das respostas, em faixas positivas enquanto o restante permaneceu neutro.

Com relação aos resultados sobre o Fator de suporte, a Figura 32 apresenta os dados referentes à perspectiva dos participantes em relação à ajuda obtida ao longo do experimento. Considerando os dados, pode-se notar uma avaliação positiva, uma vez que, a maioria das respostas se concentrou em faixas positivas ou de neutralidade.

Figura 32 – Resultado do questionário com base no fator de suporte



Fonte: Próprio autor (2022).

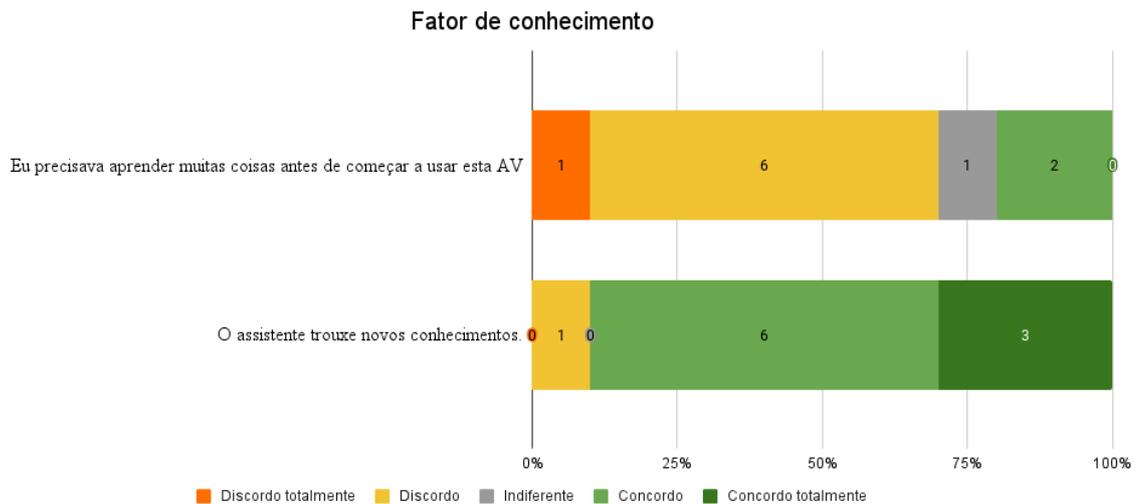
Em relação ao sentimento de apoio “A AV me ajudou sempre que precisei”, apenas 1 participante discordou dessa afirmação. Já em relação ao sentimento de não saber o que fazer na plataforma “Me senti perdido(a) em alguns momentos por não conhecer a plataforma” mostra um contraste com “Me senti perdido porque a AV não me ajudou”, afirmação da qual boa parte dos participantes discordou. Sendo uma possível indicação de que esse sentimento de não saber o que fazer por falta de conhecimento prévio pode ter sido alterado com a ajuda da assistente virtual.

Já em relação a lógica desenvolvida para o suporte “Achei que havia muita incon-

sistência na AV” e “Achei que as várias funções da AV estavam bem integradas”, os dados mostraram um aspecto positivo, onde boa parte dos participantes discordaram da afirmação negativa e concordaram ou se mantiveram neutros na afirmação positiva.

Na percepção em relação aos conhecimentos obtidos, as Figuras 33 e 34 apresentam os dados referentes a perspectivas dos participantes em relação ao conhecimento pré e pós experimento.

Figura 33 – Resultado do questionário com base no fator de conhecimento



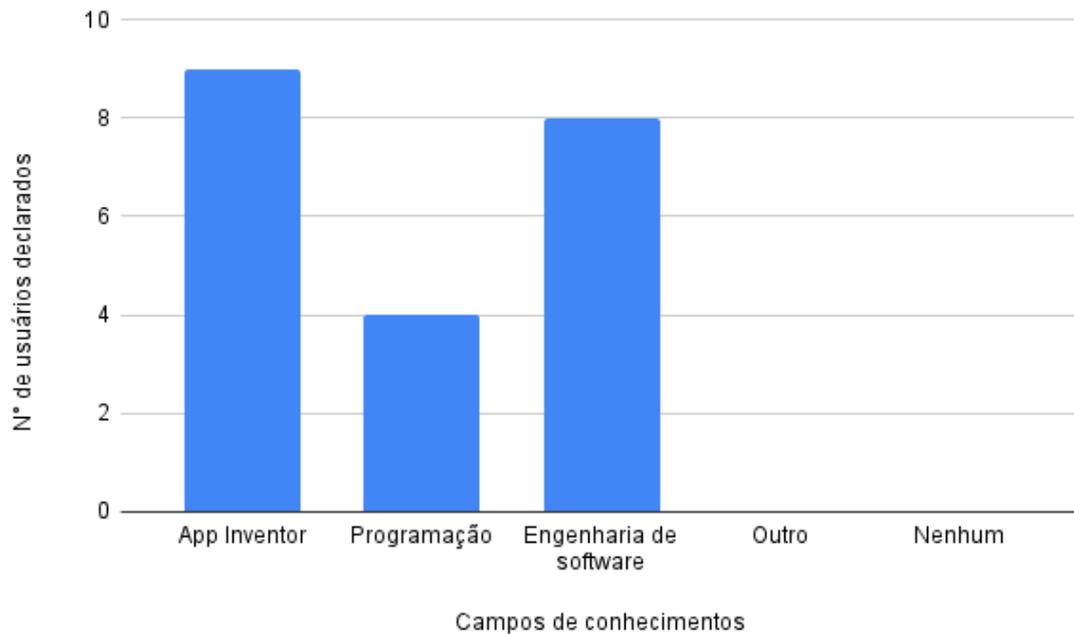
Fonte: Próprio autor (2022).

No aspecto de conhecimento prévio para utilizar a assistente “Eu precisava aprender muitas coisas antes de começar a usar esta AV” seja em relação a programação ou ao próprio *App Inventor*, 2 participantes consideraram que precisavam de mais conhecimentos antes de interagir com a ferramenta, enquanto o restante em grande parte se opôs a essa afirmação e 1 participante se manteve neutro. Já em relação aos conhecimentos pós experimento “O assistente trouxe novos conhecimentos”, somente 1 participante mostrou discordância, enquanto o restante se manteve nas faixas positivas de resposta.

Ao serem questionados a respeito desses conhecimentos, os participantes avaliaram alternativas com pontos abordados durante os experimentos, dentre eles, conhecimentos sobre a própria plataforma, algo que o participante não conhecesse; programação; ou sobre Engenharia de Software (ES), incluindo requisitos, *design*, reuso, verificação e testes, e *debug*; e teriam ainda a possibilidade de apontar algo que não estava previamente identificado. A Figura 34 mostra esse resultado.

Com base na imagem, podemos ver que em relação a plataforma, 9 dos 10 partici-

Figura 34 – Relação de conhecimentos adquiridos pelos usuários



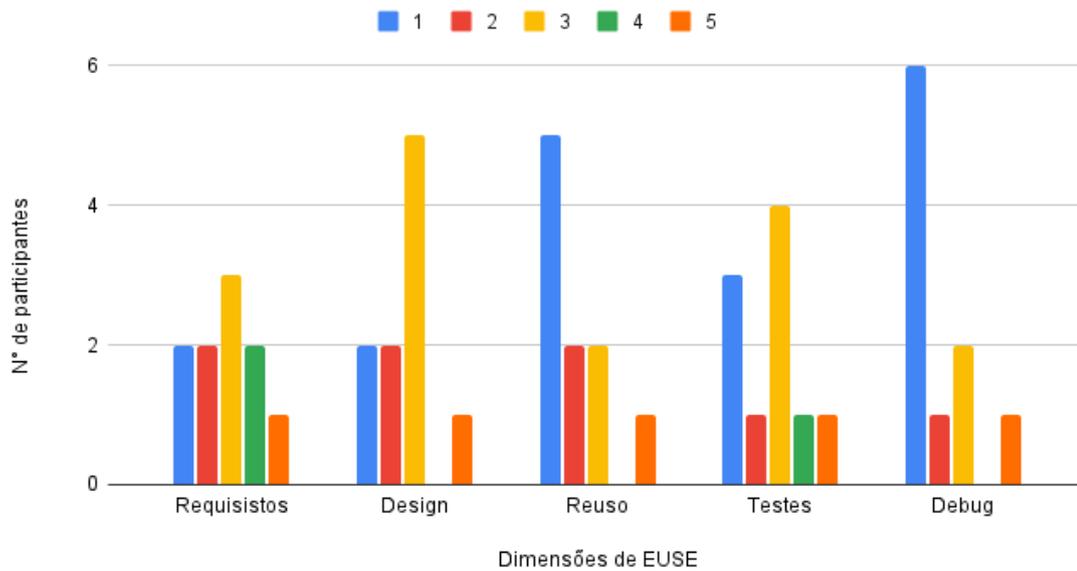
Fonte: Próprio autor (2022).

pantes informaram que obtiveram novos conhecimentos sobre o *App Inventor*. Já em relação a programação, apenas 4 participantes relataram obter novos conhecimentos a respeito desse assunto. Esse número pode ser uma retratação de casos em que eles não ingressaram na etapa mais profunda de programação, uma vez que isso não era exigido para a avaliação, tornando-se opcional. Por outro lado, 8 de 10 participantes informaram adquirir novos conhecimentos em relação às dimensões de ES. Enquanto que as alternativas de nenhum e a inclusão de outro conceito que não havia sido destacado na pergunta não foram utilizadas.

Outro ponto abordado com os participantes foi se eles já conheciam/teriam utilizado algumas das práticas de ES comentadas pela assistente virtual. Nesse ponto, eles teriam que preencher uma matriz relacionando cada dimensão de ES com uma nota de 1 a 5 com base em uma autoavaliação de conhecimento prévio sobre esse tema. A Figura 35 mostra o resultado dessa avaliação, onde cada cor representa uma nota diferente para cada uma das dimensões e na vertical temos o número de participantes que concedeu aquela nota.

Percebe-se que ao menos um participante avaliou com nota máxima o conhecimento prévio para cada uma das etapas, enquanto que a maior parte ficou na faixa entre 1 e 3, com nota média de 2.38 considerando todas as dimensões do gráfico, indicando pouco conhecimento em relação às dimensões de ES. Destaque para Reuso e Debug que tiveram metade ou mais dos participantes avaliando com nota mínima, destacando-se como as dimensões menos conhecidas

Figura 35 – Autoavaliação de práticas conhecidas previamente



Fonte: Próprio autor (2022).

pelos participantes, e também, para Requisitos, que se mostrou uma dimensão balanceada em relação ao número de participantes em cada nota apresentada.

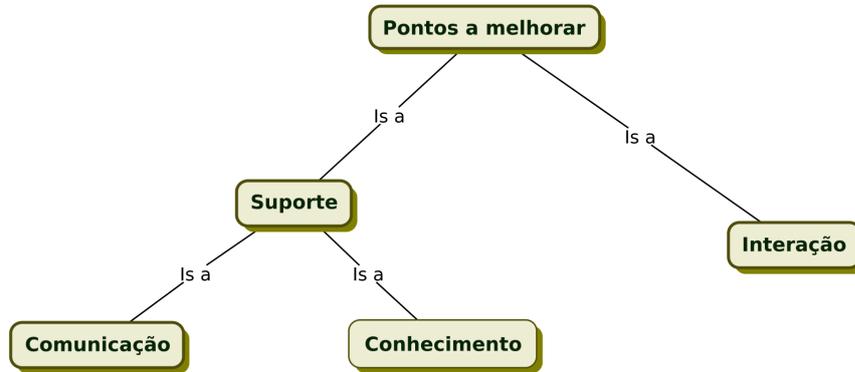
Focando agora um pouco mais na parte qualitativa, para tentar entender os aspectos da assistente virtual que precisavam ser melhorados na percepção dos participantes, foram analisadas as respostas dadas na questão aberta do questionário: “Como você acha que a assistente virtual pode melhorar?”. Dos 10 participantes, 9 fizeram comentários em relação as características que poderíamos melhorar, enquanto 1 fez apenas um comentário positivo.

A partir dos comentários, foi realizada uma etapa de codificação baseada no processo do *Grounded Theory*, uma metodologia sistemática que, através de uma coleção e análise de dados, pode concentrar as informações coletadas para a construção de uma teoria (MARTIN; TURNER, 1986). Tendo como base a codificação dos dados coletados, essa técnica procura elaborar através da identificação de códigos e categorias dos dados, uma teoria geral sobre as informações (CORBIN; STRAUSS, 1990).

Com base nisso, foram identificados 5 códigos e 4 categorias, sendo que destas, duas são subcategorias assim como mostra a Figura 36 que apresenta as categorias identificadas sobre os pontos a serem melhorados.

A análise das respostas começou utilizando um modelo de codificação mais dedutivo, tendo em foco dois pontos inicialmente definidos, o suporte oferecido e a relação da comunicação entre usuário e assistente. A partir dessa análise, duas novas categorias foram identi-

Figura 36 – Categorias relacionadas aos pontos a melhorar



Fonte: Próprio autor (2022).

ficadas, Comunicação e Conhecimento, sendo elas subcategorias de Suporte que foi inicialmente considerada. No Quadro 1 estão detalhadas as categorias e os códigos identificados.

Quadro 1 – Detalhamento dos códigos associados às categorias de pontos a melhorar

Categoria	Subcategoria	Códigos relacionados
Interação	-	Reformular frases Uso de sinônimos
Suporte	Comunicação	Uso de termos complexos Resposta fora do contexto
Suporte	Conhecimento	Não soube responder

Fonte: Elaborado pelo autor (2022).

O quadro 2 apresenta as citações dos participantes que foram extraídas juntamente com suas categorias.

Com relação a interação, o principal ponto levantado foi na melhoria da identificação das intenções dos usuários, como estrutura das sentenças e independente dos termos utilizados através do uso de sinônimos. Além disso, observou-se uma concordância entre os participantes em relação a aumentar o escopo de respostas para às dúvidas apresentadas. Por último, melhorar a forma como a assistente responde algumas perguntas.

Embora os participantes tenham deixado a entender, que em seus pensamentos, a assistente virtual devesse entender todos os conceitos e possibilidades a serem trabalhados no *App Inventor*, ressaltamos que esse não era o escopo do estudo.

Para finalizar a apresentação dos dados do questionário, a última pergunta foi relacionada ao nível de interesse dos participantes em relação a programação com base na experiência

Quadro 2 – Citações baseadas nas categorias de pontos a melhorar

Categoria	Citação dos participantes
Interação	“Necessitei reformular a mesma pergunta algumas vezes”, “Precisei reformular as frases algumas vezes para ela entender”, “Parece que tenho que acertar na escolha das palavras”, “Entender melhor as perguntas que são feitas”, “As vezes precisa usar umas palavras mais formais”, “Entender melhor o sinônimo das palavras”, “Poderia ter tipo uns sinônimos”, “Uma forma dela entender algumas palavras simples”
Comunicação	“Algumas falas podem ter ficado fora de contexto”, “Respondendo melhor algumas perguntas”
Conhecimento	“Aumentar um pouco a gama de possíveis respostas”, “Principalmente em perguntas mais complexas”, “Entendo que uma máquina não pode entender tudo”, “As vezes parecia que não sabia tudo a respeito do sistema”

Fonte: Elaborado pelo autor (2022).

de uso da plataforma do *App Inventor* e da assistente virtual. Dos resultados, 60% das respostas afirmaram que o interesse pela programação aumentou, enquanto que para os outros 40% permaneceu igual. Estes 40% não necessariamente refletem um desapeço pela programação, apenas que o experimento não acarretou no aumento de um desejo que poderia ou não já estar presente no participante.

8 DISCUSSÃO DOS RESULTADOS

Neste capítulo discutimos os resultados da pesquisa como um todo, tentando responder às questões de pesquisa formuladas para este trabalho, apresentadas novamente na Tabela 17. Iniciando pelas subquestões e, ao final, depois de ter respondido essas 4 perguntas (**QP1.1 - QP1.4**), buscou-se sintetizar as informações de cada estudo para responder a questão principal (**QP1**).

Tabela 17 – Questões de pesquisa

QP1	Como usuários finais podem trabalhar conceitos de EUSE dentro da plataforma do App Inventor?
QP1.1	Como usuários finais trabalham atividades diretamente ligadas a EUSE?
QP1.2	Quais atividades de EUSE a plataforma do App Inventor aborda ou apoia?
QP1.3	Como os usuários finais adotariam a ajuda de uma assistente virtual?
QP1.4	Antes e depois dos estudos, o que os usuários finais sabiam de ES?

Fonte: Elaborado pelo autor (2022).

QP1.1 - Como usuários finais trabalham atividades diretamente ligadas a EUSE?

Para tentar responder a primeira subquestão de pesquisa, devemos analisar os dados de 2 estudos, em especial, o estudo 1 com o *Wizard of Oz*, e o estudo 3, com a assistente virtual. Fazendo uma avaliação direcionada a cada uma das dimensões de EUSE destacadas neste trabalho, para entender como e quais perspectivas foram abordadas pelos usuários em algum momento. Com base nos dados coletados desses estudos, destacamos alguns pontos que se mostraram relevantes para nos direcionar no caminho da resposta que buscamos.

Começando pela dimensão de Requisitos, pode-se observar no estudo 1 que os usuários iniciavam seus testes com algum tipo de registro onde se encontravam características que deveriam estar presentes no sistema a ser construído. Desta forma, demonstrando que dedicaram tempo para formular previamente algumas funcionalidades que deveriam ser trabalhadas durante o experimento. Embora não fosse um documento muito complexo, já havia uma manifestação em direção a identificação e compreensão das necessidades do usuário. Já com base no estudo 3, os participantes demonstraram que estão mais dispostos a se dedicarem mais profundamente nessa etapa de requisitos quando são apoiados de alguma forma, seja interativamente como por meio de conversas como foi o caso abordado nos estudos ou mesmo pela disponibilização de ferramentas que o auxiliem nesta etapa.

Isso pode indicar que os usuário finais não abordam a dimensão de requisitos de

forma mais abrangente, talvez por falta de conhecimento do que fazer e como fazer, ou mesmo por falta de apoio ferramental que deixe essa atividade explícita.

Em relação a *Design*, os participantes mostraram um aspecto interessante durante os testes, o de modelar soluções ao longo do projeto frente às dificuldades enfrentadas que o impediam de seguir a ideia original, seja por falta de conhecimento ou pelo fato da plataforma não disponibilizar os meios necessários para atingir essas ideias. Fato observado em ambos os estudos, levando a acreditar que esses usuários estavam aptos a modificar suas soluções de *design* e encontrar novas soluções de estruturas para seus projetos ao longo do desenvolvimento. Embora essas soluções não fossem as melhores do ponto de vista profissional, elas eram capazes de suprir suas necessidades.

Voltando-se para a dimensão de Reuso, alguns participantes identificaram a possibilidade de reutilizar partes que já haviam sido criadas, fosse elementos visuais ou mesmo blocos de código como ocorreu no estudo 1. Os casos de reusos se repetiam à medida que os participantes tomaram conhecimento dessa possibilidade, fosse através de suporte externo, como ajuda do assistente ou do pesquisador, ou mesmo descobrindo sozinho. Um indicativo que essa atividade poderia ser muito mais abordada se os usuários tivessem conhecimento sobre essas possibilidades.

Na dimensão de Verificação e Testes, os participantes demonstraram interesses diferentes durante os experimentos. Enquanto alguns optaram por tentar desenvolver o máximo de telas que conseguiam antes de passar por qualquer etapa de teste, outros mostravam o interesse de testar a cada nova tela que era feita. Embora as atividades de testes tivessem que ser realizadas fora da plataforma utilizada no estudo, durante as avaliações quando a figura do "wizard" no estudo 1 ou a própria assistente virtual no estudo 3 eram questionados a respeito dessa atividades, foi possível explicar quais passos os participantes deveriam seguir para conseguir testar suas aplicações mesmo fora do *App Inventor* através das opções do menu que ele disponibiliza.

Porém, antes mesmo de testar, alguns participantes conseguiam identificar através da própria plataforma do *App Inventor* e de seus ícones e mensagens de avisos, possíveis problemas que estavam presentes naquele artefato que estava sendo desenvolvido. A dimensão de Debug era abordada através das tentativas de interpretação desses avisos e na tentativa de removê-los.

Fazendo uma visão geral do conteúdo utilizado para tentar responder nossa primeira subquestão de pesquisa, ao longo dos estudos, vimos como os usuários abordam atividades de EUSE em seu desenvolvimento mesmo que não seja de um forma pré-estabelecida ou tão

estruturada, esse processo que pode ou não ser intuitivo mostrou que com o apoio correto, esse usuários finais podem ser aproximar cada vez mais dessas atividades com a finalidade de melhorar tanto o seu processo de desenvolvimento quanto o seu artefato construído.

Embora esses usuários, inicialmente, não tenham tanto conhecimento dessas técnicas ou de como aplicá-las, quando lhes são apresentadas, é possível notar uma inclinação em seguir essas atividades, agora de forma mais consciente.

QP1.2 - Quais atividades de EUSE a plataforma do *App Inventor* aborda ou apoia?

Para responder essa questão, podemos analisar principalmente o estudo 2, baseado na inspeção da interface da plataforma do *App Inventor*, no qual permitiu aprofundar ainda mais nas ferramentas que a plataforma oferece.

Conforme foi retratado anteriormente na seção 5.3, podemos ver que a plataforma disponibiliza ferramentas para todas etapas do processo de desenvolvimento. Em especial, para as atividades relacionadas a dimensão de *Design*, na qual foi registrada a maioria das funcionalidades identificadas. Além disso, a plataforma também oferece funções para se trabalhar requisitos, reuso, testes e atividades de depuração.

Durante o estudo foi possível notar que o *App Inventor* aborda de forma satisfatória, mas não completa, a maioria das dimensões, levantando uma observação para a falta de suporte maior nas atividades de requisitos e testes que devem ser executadas principalmente fora da plataforma. Com apenas 1 elemento encontrado com relação a dimensão de Requisitos, o *App Inventor* não apresenta um foco nessa atividade. Já em relação a testes, embora a plataforma consiga oferecer diferentes formas para o usuário conseguir testar sua aplicação, nenhuma delas promove uma maneira que ele faça isso sem ter que sair para outro dispositivo.

Contudo, isso não significa que a plataforma falha em não cobrir de forma adequada todas as dimensões, afinal, não podemos garantir que esse é o seu objetivo. Porém, se colocando no papel de usuário, devemos considerar como pontos a serem melhorados. Embora ferramentas de desenvolvimento profissionais não permitam que o desenvolvedor trabalhe seus requisitos na própria ferramenta, por muitas vezes esse usuário já tem o conhecimento dessa parte do desenvolvimento e já entende esse processo, podendo inclusive ter uma equipe dedicada somente para trabalhar essa dimensão. Quando focamos em usuários finais, podemos supor que o único recurso do usuário é a própria ferramenta, sendo assim, é ideal que ela inclua o máximo de

recursos para apoiar um processo mais robusto além deixar bem claro todos os recursos que podem ser trabalhados.

QP1.3 - Como os usuários finais adotariam a ajuda de uma assistente virtual?

Para tentar responder essa questão, devemos analisar as observações e respostas obtidas nos estudos 1 e 3 para identificar aspectos como experiência de uso e suporte fornecido. Com base nas observações, podemos iniciar destacando que tivemos dois tipos de usuários com características diferentes em relação a requisição de ajuda da nossa assistente virtual. Enquanto um buscava a assistência logo no início das atividades, o outro por sua vez, era mais relutante a fazer isso. Esse segundo tipo de usuário relatado adotava uma abordagem mais exploratória, e quando não satisfeito, requisitava a ajuda da assistente.

A ideia de ter alguém a sua disposição para tirar dúvidas a qualquer momento e ainda trazer novos conhecimentos parece ter sido bem aceita pelos participantes, embora sabemos de alguns pontos a serem ainda melhorados. Ao analisar as respostas do questionário podemos ver relatos sobre o aumento de confiança no processo e no produto desenvolvido pelo fato de ter um suporte mais presente. Além de abordagem de novos conhecimentos, facilidade de interação, entre outros pontos.

Em contraponto, em alguns casos em que os usuários solicitaram ajuda e não obtiveram uma resposta coerente, destacavam-se novamente dois perfis diferentes. O primeiro é aquele que ao receber uma resposta caracterizada como errada, desistia da ajuda momentaneamente e partia para uma abordagem exploratória, quando não encontrava a solução, retornava a solicitar ajuda para a assistente. O segundo perfil foi do usuário que, ao receber uma resposta caracterizada como errada, tentava reformular a pergunta até conseguir uma resposta satisfatória.

Vale ressaltar que a maioria dos participantes do estudo 3 relatou aumento no interesse em aprender e construir aplicativos no *App Inventor* após a experiência com a assistente virtual, refletindo os resultados mostrados no capítulo anterior e comentados anteriormente a respeito da comunicação, suporte e aprendizado que foram trabalhados durante as avaliações.

QP1.4 - Antes e depois dos estudos, o que os usuários finais sabiam de ES?

A resposta para a questão é difícil de ser mensurada, afinal não foi aplicada nenhum tipo de questionário avaliativo nem antes e nem após os experimentos. Para esse ponto, vamos levar em consideração a percepção dos próprios participantes sobre o que eles conheciam antes e o que eles consideraram como um novo conhecimento adquirido em relação às atividades que

foram realizadas.

Quando questionados, embora alguns participantes tenham relatado ter mais conhecimento sobre os conceitos de ES, a maioria deles responderam que possuíam pouco conhecimento sobre as atividades de ES antes de iniciar os experimentos. Após os estudos, esses participantes relataram que o contato com a assistente resultou em novos conhecimentos, tanto em relação a própria ES como em conceitos de programação e da plataforma do *App Inventor*. Essa afirmação pode ser constatada nos casos em que os participantes solicitavam a ajuda para uma determinada atividade e depois das informações repassadas pela assistente eles continuavam seguindo as mesmas indicações, mesmo sem a necessidade da assistente revisitar a mesma conversa.

Acreditamos que, embora esse resultado possa não ser muito conclusivo em relação à real aprendizagem dos participantes, tomamos a percepção demonstrada nas respostas como válidas levando em conta que nem todas as respostas do questionário tiveram inclinação a serem 100% positivas, e também, considerando a avaliação das ações dos participantes pelas análises dos vídeos.

QP1 - Como usuários finais podem trabalhar conceitos de EUSE dentro da plataforma do *App Inventor*?

Conforme visto antes, a plataforma do *App Inventor* permite que os usuários finais trabalhem os conceitos de EUSE para todas as dimensões. Nas etapas iniciais, os usuários podem trabalhar requisitos de forma mais modesta dentro do sistema, conforme mostraram os estudos. Atualmente os usuários trabalham essa etapa principalmente fora da plataforma. Neste ponto, identificamos como uma forma de melhoria, uma área reservada dentro da plataforma para permitir que os usuários listassem seus requisitos, fazendo-se visível para o usuário final identificar sua existência e a importância de preenchê-los, deixando assim os requisitos acessíveis no próprio projeto, facilitando consultas de forma mais prática nas etapas seguintes do desenvolvimento.

A dimensão de *Design* já é bem suportada pelas ferramentas atuais da plataforma, como mostrou o estudo 2. Desde os componentes que ela disponibiliza para criação das telas e dos blocos até as estruturas hierárquicas com as disposições de elementos que compõem o artefato em desenvolvimento. Os usuários podem, de maneira descomplicada, estruturar seus aplicativos como acharem melhor apenas arrastando componentes, alterando suas propriedades e validando sua arquitetura visualmente exposta na tela da plataforma. Considerando a criação da

lógica dos blocos como *design*, vemos que a plataforma oferece estruturas de controle e repetição (*if, else, for, while*), estruturas lógicas (e, ou), fórmulas matemáticas, manipulação de listas, entre muitos outros recursos. Em relação a *design*, não vemos ponto de melhoria, uma vez que, as atividades relacionadas a essa dimensão já são bem trabalhadas no *App Inventor*.

O reuso pode ser trabalhado de maneira um pouco ultrapassada para desenvolvedores profissionais, porém bastante simples de ser aplicado. Além do tradicional copiar e colar na criação de telas e no desenvolvimento dos blocos, a plataforma ainda oferece a opção da mochila, que por sinal, faz uma boa referência a sua utilidade de armazenar coisas para uso futuro. Um bom ponto de melhoria seria possibilitar que o usuário entenda que isso pode ser feito ao longo de todo o projeto, seja pela disponibilização de trechos de blocos já produzidos e validados com algumas funcionalidades comuns, por exemplo para trocar de tela ou armazenar informações em base de dados, ou algum tipo de tutorial com breve explicação desse recurso quando o usuário acessar a tela ou adicionasse o primeiro bloco lá. Consequentemente isso poderia diminuir seu esforço nas outras etapas do desenvolvimento.

Diretamente dentro da plataforma não há formas de o usuário trabalhar os testes da sua aplicação, atualmente, o *App Inventor* permite diferentes formas de conectar a dispositivos externos para que isso seja feito fora da plataforma. O que leva o usuário a ter que instalar o aplicativo em outro dispositivo para aplicar os testes, tornando esse trabalho mais tedioso e cansativo. Como melhoria, um ponto não trivial, seria que a plataforma apresentasse meios do usuário conseguir testar partes da sua aplicação sem precisar voltar-se externamente para outro dispositivo, a não ser apenas para testes de cenários mais completos.

O motivo da melhoria ser tratada como não trivial seria pela complexidade da solução ser incorporada dentro da plataforma. Diferentemente de outros levantamentos nas dimensões já comentadas, nas quais uma melhoria para os usuários finais trabalharem de forma mais completa as atividades de ES fossem modificações relativamente mais simples, permitir realizar testes dentro da própria plataforma teria um grau de complexidade em relação a implementação da solução muito maior.

Por último, na dimensão de Debug, podemos perceber que embora o *App Inventor* disponibilize poucos elementos com relação a essa atividade, os usuários conseguem se empenhar de maneira satisfatória na busca por pontos problemáticos na sua aplicação. Graças aos avisos apresentados nos componentes, que auxiliam os usuários com os erros presentes no produto que está sendo desenvolvido, fazendo com que ele dedique um pouco mais da sua atenção para essa

atividade tão importante da EUSE.

Através da própria interface, é possível identificar imediatamente os erros mais simples, como por exemplo, erros na montagem dos blocos. Por outro lado, a plataforma não aborda formas de investigar erros mais complexos, como por exemplo, erros em tempo de execução, deixando a cargo somente do usuário de tentar resolver esse problema. Provavelmente, adotando a recomendação sugerida para a etapa de testes, veríamos impactos refletidos nessa área de depuração de códigos também.

De um modo geral, a utilização de um meio de suporte “externo” como foi o caso do uso da assistente virtual, mostrou que o usuário final busca executar atividades básicas de desenvolvimento que sejam importantes, no seu conhecimento, e que isso possibilita o direcionamento desse usuário para outras atividades que a EUSE considera importante, desde que isso seja apoiado por alguém ou por alguma ferramenta.

9 CONSIDERAÇÕES FINAIS

Este trabalho abordou a área de *End-User Software Engineering* (EUSE) inserida no contexto do *App Inventor* como plataforma para usuários finais produzirem suas aplicações. Seguimos uma visão de entender como os usuários finais abordam e podem ser apoiados por ferramentas para introduzir mais qualidade em seus processos e produtos.

Nosso objetivo de pesquisa foi entender como usuários finais poderiam trabalhar atividades de EUSE inseridos na plataforma do *App Inventor*, além de desenvolver uma assistente virtual para auxiliar esses usuários nas atividades de desenvolvimento ao mesmo tempo que, por meio de conversas, tenta abordar conceitos sobre requisitos, *design*, reuso, testes e *debug*.

Discutimos como os usuários finais podem abordar esses temas atualmente e como adotariam a colaboração de uma assistente virtual que funcionaria por comandos de voz, também tentamos entender como uma plataforma consegue apoiar essas atividades com base nos seus elementos de tela. Desenvolvemos e avaliamos uma assistente virtual por meio de avaliações com os usuários, análise de questionários, vídeos e artefatos.

Confirmamos uma ideia inicial de que a inclusão de uma assistente virtual engajaria os usuários finais nesse processo tão complexo que é incluir atividades sistemáticas da engenharia com objetivos de tornar o desenvolvimento mais simples. Sem surpresas, os participantes dos estudos demonstraram receber de forma positiva a abordagem apresentada, além de trazerem pontos importantes do processo comunicativo para serem melhorados em relação a comunicação e suporte.

9.1 Contribuições da pesquisa

Acreditamos que através desta pesquisa, com o fator de transformar o processo de desenvolvimento em uma caminhada comunicativa na qual o usuário final não está sozinho como programador, damos um passo importante para engajar ainda mais usuários finais em processos de desenvolvimento com o adicional de poder levantar conceitos e práticas afim de melhorar a qualidade do processo e do produto gerado. Uma vez que a plataforma oferece ferramentas para adotar as atividades de EUSE, o desafio se torna exercitar o processo de incorporar tais atividades.

Desta forma, como contribuições da pesquisa, listamos os seguintes itens:

- Identificamos como os usuários finais podem trabalhar atividades de EUSE;

- Identificamos e classificamos os recursos presentes no *App Inventor* para apoiar as atividades de EUSE;
- Desenvolvemos e avaliamos uma assistente virtual para apoiar os usuários finais no *App Inventor*.

9.2 Trabalhos futuros

Há muitas oportunidades de complementar este trabalho. As sugestões a seguir são baseadas nos conhecimentos até o momento, mas a dimensão de cada trabalho pode mudar a medida que for se aprofundando no tema. Dentre eles, o desenvolvimento de uma base de conhecimento mais robusta para ser utilizada pelo Gerenciador de Diálogos. Nesse ponto, podemos listar aspectos a melhorar em relação a conceitos de EUSE, conceitos de programação em geral e até mesmo os conceitos sobre a própria plataforma. Não apenas a introdução de novos conceitos e conhecimentos, incluir também sinônimos e sentenças que representam a mesma finalidade, fazendo com que o usuário não precise ficar formulando novas frases até atingir seu objetivo.

O foco da base da assistente virtual poderia ser alterado para abordar outras questões de ES, ou se aprofundando mais em determinada dimensão. Ajudando a entender melhor como os usuários finais poderiam trabalhar cada dimensão isoladamente. Além disso, aplicar a assistente virtual em ambientes de aprendizado real, como em salas de aula por exemplo.

Outro ponto seria implementar melhorias na interface do *App Inventor* levando em consideração as observações expostas no capítulo de discussão dos resultados. Tendo em vista que uma ferramenta EUD, diferente de uma ferramenta de desenvolvimento profissional, deveria comunicar claramente seus recursos com a intenção de orientar os usuários no seu processo de adoção dessas atividades de EUSE. Podemos ainda não nos restringir ao *App Inventor* e buscar desenvolver um processo similar para outras ferramentas que também apoiam usuários finais.

9.3 Limitações conhecidas

Destacamos ainda algumas limitações desta pesquisa, dentre elas, podemos listar o tamanho e escopo da base utilizada nos experimentos, uma vez que, a base foi criada totalmente pelo pesquisador a partir de conhecimentos prévios e incrementada ao longo dos estudos. Essa limitação, restringiu um pouco o escopo dos testes e pode ter influenciado nos resultados,

provavelmente, uma base mais robusta retornaria resultados ainda mais conclusivos. O número de participantes, a amostragem por conveniência e a época em que os experimentos ocorreram podem ter afetado a qualidade dos resultados de alguma forma, deixando o desejo de realizar novos experimentos em um outro momento.

REFERÊNCIAS

- ABDELLATIF, A.; BADRAN, K.; COSTA, D. E.; SHIHAB, E. A comparison of natural language understanding platforms for chatbots in software engineering. **IEEE Transactions on Software Engineering**, IEEE, v. 48, n. 8, p. 3087–3102, 2021.
- ABREU, R.; HOFER, B.; PEREZ, A.; WOTAWA, F. Using constraints to diagnose faulty spreadsheets. **Software Quality Journal**, Springer, v. 23, n. 2, p. 297–322, 2015.
- AZARIA, A.; KRISHNAMURTHY, J.; MITCHELL, T. M. Instructable intelligent personal agent. In: **Thirtieth AAAI Conference on Artificial Intelligence**. [S. l.: s. n.], 2016.
- BARRICELLI, B. R.; CASSANO, F.; FOGLI, D.; PICCINNO, A. End-user development, end-user programming and end-user software engineering: A systematic mapping study. **Journal of Systems and Software**, Elsevier, v. 149, p. 101–137, 2019.
- BRANDT, J.; GUO, P. J.; LEWENSTEIN, J.; KLEMMER, S. R. Opportunistic programming: How rapid ideation and prototyping occur in practice. In: **Proceedings of the 4th international workshop on End-user software engineering**. [S. l.: s. n.], 2008. p. 1–5.
- BROOKE, J. *et al.* Sus-a quick and dirty usability scale. **Usability evaluation in industry**, London–, v. 189, n. 194, p. 4–7, 1996.
- BRUMMELEN, J. J. R. V. **Tools to create and democratize conversational artificial intelligence**. Tese (Doutorado) – Massachusetts Institute of Technology, 2019.
- BURNETT, M. End-user software engineering: Toward a future beyond the silos. In: SPRINGER. **International Conference on Human-Centred Software Engineering**. [S. l.], 2014. p. 315–316.
- BURNETT, M.; ROTHERMEL, G.; COOK, C. An integrated software engineering approach for end-user programmers. In: **End User Development**. [S. l.]: Springer, 2006. p. 87–113.
- CHENG, S.-C.; CHENG, Y.-P.; HUANG, Y.-M.; YANG, Y. Combining flipped learning and formative assessment to enhance the learning performance of students in programming. In: SPRINGER. **International Conference on Innovative Technologies and Learning**. [S. l.], 2021. p. 498–507.
- CHOWDHURY, S. R. Assisting end-user development in browser-based mashup tools. In: IEEE. **2012 34th International Conference on Software Engineering (ICSE)**. [S. l.], 2012. p. 1625–1627.
- CORBIN, J. M.; STRAUSS, A. Grounded theory research: Procedures, canons, and evaluative criteria. **Qualitative sociology**, Springer, v. 13, n. 1, p. 3–21, 1990.
- FISCHER, G.; NAKAKOJI, K.; YE, Y. Metadesign: Guidelines for supporting domain experts in software development. **IEEE software**, IEEE, v. 26, n. 5, p. 37–44, 2009.
- FOWLER, M. **Refactoring: improving the design of existing code**. [S. l.]: Addison-Wesley Professional, 2018.

FRONZA, I.; CORRAL, L.; PAHL, C. End-user software development: Effectiveness of a software engineering-centric instructional strategy. **Journal of Information Technology Education: Research**, Informing Science Institute. 131 Brookhill Court, Santa Rosa, CA 95409, v. 19, p. 367–393, 2020.

GOMES, T. C.; MELO, J. C. de. App inventor for android: Uma nova possibilidade para o ensino de lógica de programação. In: **Anais dos Workshops do Congresso Brasileiro de Informática na Educação**. [S. l.: s. n.], 2013. v. 2, n. 1.

GRAY, J.; ABELSON, H.; WOLBER, D.; FRIEND, M. Teaching cs principles with app inventor. In: **Proceedings of the 50th Annual Southeast Regional Conference**. [S. l.: s. n.], 2012. p. 405–406.

GREEN, P.; WEI-HAAS, L. The rapid development of user interfaces: Experience with the wizard of oz method. In: SAGE PUBLICATIONS SAGE CA: LOS ANGELES, CA. **Proceedings of the Human Factors Society Annual Meeting**. [S. l.], 1985. v. 29, n. 5, p. 470–474.

GROVER, S.; PEA, R. Using a discourse-intensive pedagogy and android's app inventor for introducing computational concepts to middle school students. In: **Proceeding of the 44th ACM technical symposium on Computer science education**. [S. l.: s. n.], 2013. p. 723–728.

GULLEY, N. Improving the quality of contributed software and the matlab file exchange. In: **2nd Workshop on End User Software Engineering**. [S. l.: s. n.], 2006. p. 8–9.

HERMANS, F.; PINZGER, M.; DEURSEN, A. van. Detecting and visualizing inter-worksheet smells in spreadsheets. In: IEEE. **2012 34th International Conference on Software Engineering (ICSE)**. [S. l.], 2012. p. 441–451.

INVENTOR, M. A. **Stories**. [S. l.], 2022. Disponível em: <https://appinventor.mit.edu>. Acesso em: 01 nov. 2022.

KO, A. J.; ABRAHAM, R.; BECKWITH, L.; BLACKWELL, A.; BURNETT, M.; ERWIG, M.; SCAFFIDI, C.; LAWRENCE, J.; LIEBERMAN, H.; MYERS, B. *et al.* The state of the art in end-user software engineering. **ACM Computing Surveys (CSUR)**, ACM, v. 43, n. 3, p. 21, 2011.

KURLANDER, D.; CYPHER, A.; HALBERT, D. C. **Watch what I do: programming by demonstration**. [S. l.]: MIT press, 1993.

LEBEUF, C.; STOREY, M.-A.; ZAGALSKY, A. Software bots. **IEEE Software**, IEEE, v. 35, n. 1, p. 18–23, 2017.

LI, T. J.-J. End user programming of intelligent agents using demonstrations and natural language instructions. In: **Proceedings of the 24th International Conference on Intelligent User Interfaces: Companion**. [S. l.: s. n.], 2019. p. 143–144.

LI, T. J.-J.; AZARIA, A.; MYERS, B. A. Sugilite: creating multimodal smartphone automation by demonstration. In: **Proceedings of the 2017 CHI conference on human factors in computing systems**. [S. l.: s. n.], 2017. p. 6038–6049.

LIEBERMAN, H.; PATERNÒ, F.; KLANN, M.; WULF, V. End-user development: An emerging paradigm. In: **End user development**. [S. l.]: Springer, 2006. p. 1–8.

LIZCANO, D.; ALONSO, F.; SORIANO, J.; LOPEZ, G. A web-centred approach to end-user software engineering. **ACM Transactions on Software Engineering and Methodology (TOSEM)**, ACM New York, NY, USA, v. 22, n. 4, p. 1–29, 2013.

MACHIRAJU, S.; MODI, R. Natural language processing. In: **Developing Bots with Microsoft Bots Framework**. [S. l.]: Springer, 2018. p. 203–232.

MARTIN, P. Y.; TURNER, B. A. Grounded theory and organizational research. **The journal of applied behavioral science**, JAI PRESS INC. 36 Sherwood Place PO Box-1678 Greenwich, Connecticut 06836-1678, v. 22, n. 2, p. 141–157, 1986.

MASSA, D.; SPANO, L. D. Facemashup: an end-user development tool for social network data. **Future Internet**, Multidisciplinary Digital Publishing Institute, v. 8, n. 2, p. 10, 2016.

MAULSBY, D.; GREENBERG, S.; MANDER, R. Prototyping an intelligent agent through wizard of oz. In: **ACM. Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems**. [S. l.], 1993. p. 277–284.

MONTEIRO, I. T. **Autoexpressão e engenharia semiótica do usuário-designer**. [S. l.: s. n.], 2015.

MONTEIRO, I. T.; SALGADO, L. C. de C.; MOTA, M. P.; SAMPAIO, A. L.; SOUZA, C. S. de. Signifying software engineering to computational thinking learners with agentsheets and polifacets. **Journal of Visual Languages & Computing**, Elsevier, v. 40, p. 91–112, 2017.

MORELLI, R.; LANEROLLE, T. D.; LAKE, P.; LIMARDO, N.; TAMOTSU, E.; UCHE, C. Can android app inventor bring computational thinking to k-12. In: **Proc. 42nd ACM technical symposium on Computer science education (SIGCSE'11)**. [S. l.: s. n.], 2011. p. 1–6.

PANKO, R. R. What we know about spreadsheet errors. **Journal of Organizational and End User Computing (JOEUC)**, IGI Global, v. 10, n. 2, p. 15–21, 1998.

PAPADAKIS, S.; KALOGIANNAKIS, M.; ORFANAKIS, V.; ZARANIS, N. The appropriateness of scratch and app inventor as educational environments for teaching introductory programming in primary and secondary education. In: **Early childhood development: Concepts, methodologies, tools, and applications**. [S. l.]: IGI Global, 2019. p. 797–819.

PERSCHEID, M.; SIEGMUND, B.; TAEUMEL, M.; HIRSCHFELD, R. Studying the advancement in debugging practice of professional software developers. **Software Quality Journal**, Springer, v. 25, n. 1, p. 83–110, 2017.

RADIO, N. P. **The smart audio report, spring 2022**. [S. l.]: NPR and Edison Research, 2022.

SCAFFIDI, C.; SHAW, M.; MYERS, B. Estimating the numbers of end users and end user programmers. In: **IEEE. 2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05)**. [S. l.], 2005. p. 207–214.

SEGAL, J. Some problems of professional end user developers. In: **IEEE. IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2007)**. [S. l.], 2007. p. 111–118.

SIMPLE USABILITY. **Wizard of Oz testing – a method of testing a system that does not yet exist**. 2018. Disponível em: <https://www.simpleusability.com/inspiration/2018/08/wizard-of-oz-testing-a-method-of-testing-a-system-that-does-not-yet-exist/>. Acesso em: 21 nov. 2019.

SOMMERVILLE, I. **Software Engineering, 9/E**. [S. l.]: Pearson Education India, 2011.

TZEREMES, V.; GOMAA, H. A software product line approach for end user development of smart spaces. In: IEEE. **2015 IEEE/ACM 5th International Workshop on Product Line Approaches in Software Engineering**. [S. l.], 2015. p. 23–26.

TZEREMES, V.; GOMAA, H. Applying end user software product line engineering for smart spaces. In: **Proceedings of the 51st Hawaii International Conference on System Sciences**. [S. l.: s. n.], 2018.

VALENTE, M. T. **Engenharia de software moderna: Princípios e práticas para desenvolvimento de software com produtividade**. [S. l.: s. n.], 2020. v. 1. 24 p.

WEIGELT, S.; HEY, T.; LANDHÄUSSER, M. Integrating a dialog component into a framework for spoken language understanding. In: ACM. **Proceedings of the 6th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering**. [S. l.], 2018. p. 1–7.

WEIGELT, S.; TICHY, W. F. Pronat: an agent-based system design for programming in spoken natural language. In: IEEE PRESS. **Proceedings of the 37th International Conference on Software Engineering-Volume 2**. [S. l.], 2015. p. 819–820.

WEISE, K. **Hey, Alexa, why is Amazon making a microwave?** 2018. Disponível em: <https://www.nytimes.com/2018/09/20/technology/amazon-alexa-new-features-products.html>. Acesso em: 19 abr. 2020.

XIE, B.; ABELSON, H. Skill progression in mit app inventor. In: IEEE. **2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)**. [S. l.], 2016. p. 213–217.

APÊNDICE A – TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

Você está sendo convidado(a) a participar, como voluntário(a), do estudo intitulado “Proposta de um voicebot para auxiliar usuários finais”, conduzida por Francisco Gilberto Lima Paiva. Este estudo tem por objetivo avaliar um modelo inicial de um assistente virtual proposto para auxiliar pessoas a desenvolver aplicações através de linguagem natural utilizando interface por voz.

Você foi selecionado(a) por se encaixar no perfil desejado para esta pesquisa. Sua participação não é obrigatória. A qualquer momento, você poderá desistir de participar e retirar seu consentimento. Sua recusa, desistência ou retirada de consentimento não acarretará prejuízo.

Sua participação nesta pesquisa consistirá em tentar desenvolver uma aplicação com a ajuda do assistente virtual que está sendo avaliado, e em seguida, responder a algumas questões que o pesquisador irá fazer sobre sua experiência na avaliação. Os dados obtidos por meio desta pesquisa serão confidenciais e não serão divulgados em nível individual, visando assegurar o sigilo de sua participação.

O pesquisador responsável se comprometeu a tornar públicos nos meios acadêmicos e científicos os resultados obtidos de forma consolidada sem qualquer identificação de indivíduos participantes.

Caso você concorde em participar desta pesquisa, assine ao final deste documento, que possui duas vias, sendo uma delas sua, e a outra, do pesquisador responsável pela pesquisa. Caso você tenha qualquer dúvida sobre o projeto e sua participação nele, poderá entrar em contato com o pesquisador responsável Francisco Gilberto Lima Paiva através do número de telefone: **[telefone do pesquisador]**, ou pelo e-mail: fgberto.lima@gmail.com.

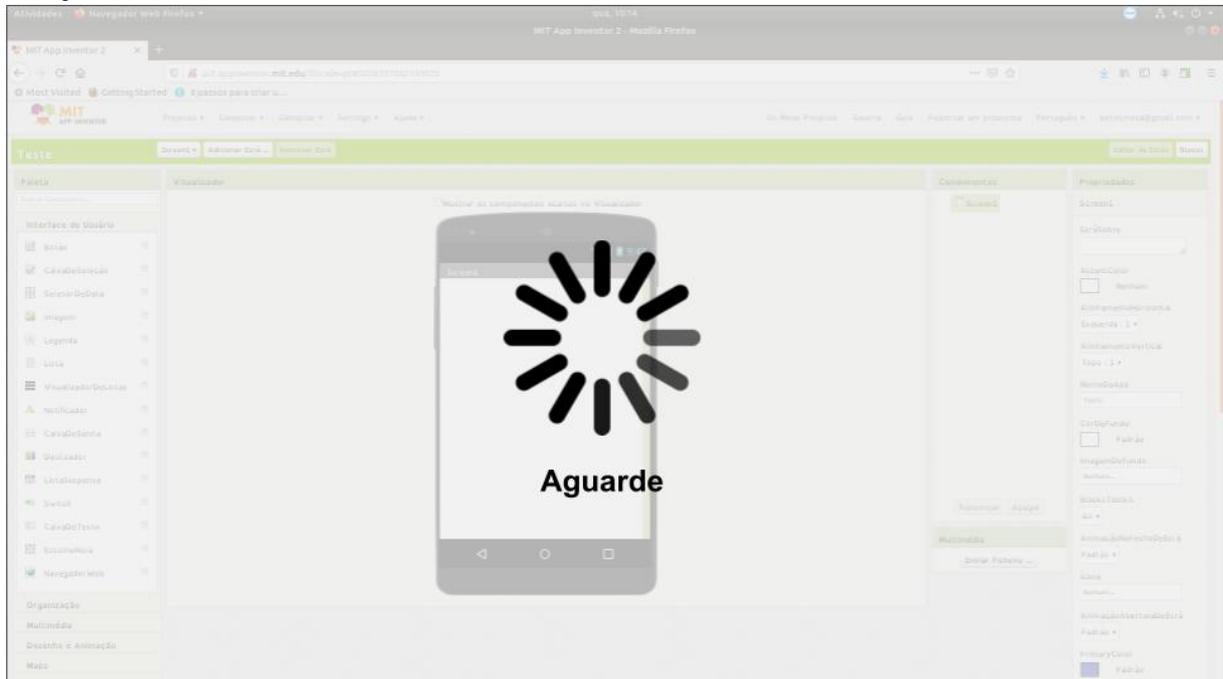
Quixadá, __ de __ de ____.

Assinatura do(a) participante:

Assinatura do pesquisador:

APÊNDICE B – IMAGEM CRIADA PARA O ESTUDO 1

Figura 37 – Tela representando o momento de espera entre uma comando do usuário e o termino da ação



Fonte: Próprio autor (2022).

APÊNDICE C – CENÁRIO DO ESTUDO

Você é aluno da UFC e almoça todos os dias no Restaurante Universitário (RU). Seus colegas sempre falam de como estava a refeição e do quanto gostariam de opinar mais sobre ela, porém o método de avaliação atual desagrada um pouco, muitos dos alunos também desejam fazer o mesmo. Os alunos de tecnologia do campus estão muito ocupados para desenvolver um aplicativo que resolva este problema. Mesmo não sendo formado em TI, você se interessa muito pelo assunto. Um dia, seu colega de sala contou sobre o *App Inventor*, um sistema para criação de aplicativos para celular de forma facilitada. Diante desta situação você resolve tomar uma atitude e mostrar que é capaz de criar um app para que seus colegas e toda a comunidade acadêmica possam avaliar cada refeição do RU usando o próprio smartphone.

Pense um pouco sobre a aplicação que irá criar, o que ela deve ter, como será a aparência, e, quando você estiver pronto(a), mãos à obra.

Tarefa

Desenvolver um aplicativo simples para avaliação das refeições do Restaurante Universitário utilizando o *App Inventor*. Para isso, você pode utilizar da ajuda da assistente virtual que está disponível na plataforma ao longo do desenvolvimento.

APÊNDICE D – FORMULÁRIO DE RECRUTAMENTO

Formulário destinado a receber os dados dos interessados em participar do experimento “Proposta de um agente conversacional para EUSE”.

Após responder este questionário, entendo que você aceita participar do experimento e enviarei para o seu e-mail o restante dos detalhes.

Não se preocupe, você poderá abandonar (caso queira) a qualquer momento, sem nenhum problema.

Nome:

E-mail:

Disponibilidade de dias (pode marcar mais de uma opção). Obs: Você vai definir seu melhor horário.

- Segunda
- Terça
- Quarta
- Quinta
- Sábado
- Domingo

APÊNDICE E – QUESTIONÁRIO UTILIZADO PARA AVALIAÇÃO DE USO DA ASSISTENTE VIRTUAL

Antes de tudo, gostaríamos de começar agradecendo pela sua participação nesse estudo. Para finalizar pedimos que você responda as questões abaixo sobre a sua percepção no uso da assistente virtual como apoio no *App Inventor* para nos ajudar a melhorá-la. Todos os dados são coletados anonimamente e somente serão utilizados no contexto desta pesquisa.

Informações demográficas	
Gênero:	<input type="checkbox"/> Masculino <input type="checkbox"/> Feminino <input type="checkbox"/> Outro
Faixa etária:	<input type="checkbox"/> 18-20 <input type="checkbox"/> 21-23 <input type="checkbox"/> 24-26 <input type="checkbox"/> 26-28 <input type="checkbox"/> 28-30 <input type="checkbox"/> Acima de 30 anos
Grau de conhecimento em informática:	<input type="checkbox"/> Nenhum <input type="checkbox"/> Básico <input type="checkbox"/> Intermediário <input type="checkbox"/> Avançado
Grau de conhecimento em programação:	<input type="checkbox"/> Nenhum <input type="checkbox"/> Básico <input type="checkbox"/> Intermediário <input type="checkbox"/> Avançado

Por favor, marque uma opção de acordo com o quanto você concorda ou discorda de cada afirmação abaixo.

Afirmações	Marque uma opção conforme sua avaliação				
	Discordo totalmente	Discordo	Nem discordo, nem concordo	Concordo	Concordo totalmente
O sistema construído atingiu minhas expectativas iniciais.					
Consegui fazer boa parte do que estava planejado.					
A assistente trouxe novos conhecimentos.					
A linguagem utilizada pela assistente foi fácil de compreender					
A assistente utilizou termos complexos.					
A forma de se comunicar com a assistente ao solicitar ajuda era simples.					
Me senti perdido(a) em algum momento por não conhecer a plataforma.					
Me senti perdido porque a assistente não me ajudou.					
A assistente me ajudou sempre que precisei.					

Já conhecia/tinha utilizado as práticas comentadas pela assistente. (Marque uma opção para cada atividade de acordo com seu conhecimento)

	avaliação				
Requisitos	<input type="checkbox"/> 1	<input type="checkbox"/> 2	<input type="checkbox"/> 3	<input type="checkbox"/> 4	<input type="checkbox"/> 5
Design	<input type="checkbox"/> 1	<input type="checkbox"/> 2	<input type="checkbox"/> 3	<input type="checkbox"/> 4	<input type="checkbox"/> 5
Reuso	<input type="checkbox"/> 1	<input type="checkbox"/> 2	<input type="checkbox"/> 3	<input type="checkbox"/> 4	<input type="checkbox"/> 5
Testes	<input type="checkbox"/> 1	<input type="checkbox"/> 2	<input type="checkbox"/> 3	<input type="checkbox"/> 4	<input type="checkbox"/> 5
Debug	<input type="checkbox"/> 1	<input type="checkbox"/> 2	<input type="checkbox"/> 3	<input type="checkbox"/> 4	<input type="checkbox"/> 5

Quais novos conhecimentos a assistente trouxe?

- App Inventor
- Programação
- Atividades de Engenharia de Software (Requisitos, Design, Reuso, Testes e/ou Debug)
- Nenhum

Outro: _____

Afirmações	Marque uma opção conforme sua avaliação				
	Discordo totalmente	Discordo	Nem discordo, nem concordo	Concordo	Concordo totalmente
Acho que gostaria de usar a assistente virtual com frequência.					
Achei a assistente virtual desnecessariamente complexo.					
Achei o assistente virtual fácil de usar.					
Acho que precisaria do apoio de um técnico para poder usar esta assistente virtual.					
Achei que as várias funções da assistente virtual estavam bem integradas.					
Achei que havia muita inconsistência na assistente virtual.					
Eu imagino que a maioria das pessoas aprenderia a usar essa assistente virtual muito rapidamente.					
Achei a assistente virtual muito complicado de usar.					
Eu me senti muito confiante ao usar a assistente virtual.					
Eu precisava aprender muitas coisas antes de começar a usar esta assistente virtual.					

A partir desta experiência com o agente e o *App Inventor*, como está seu interesse

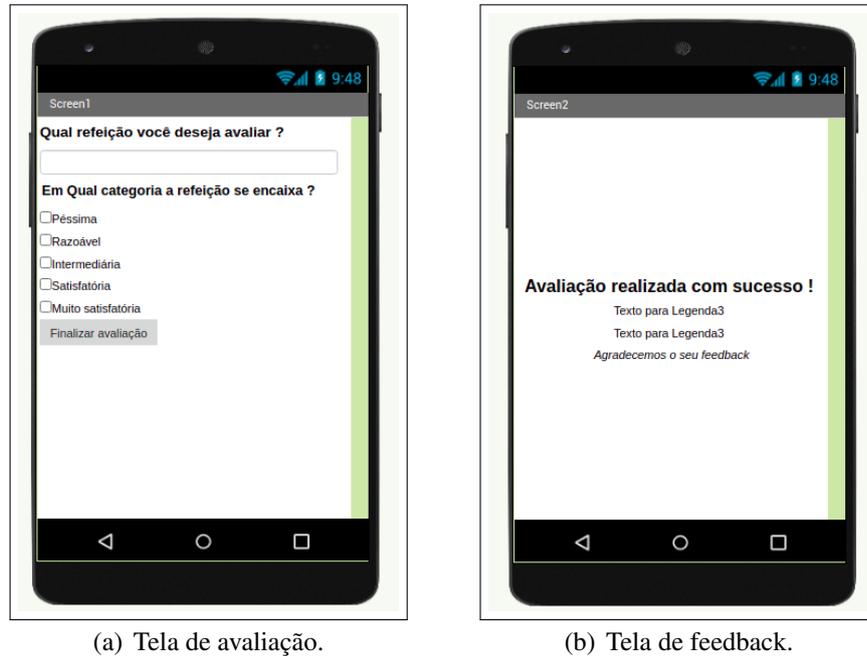
em aprender e construir aplicativos na plataforma?

Aumentou Diminuiu Permaneceu igual

Como você acha que a assistente virtual pode melhorar?

APÊNDICE F – ARTEFATOS DO ESTUDO 3

Figura 38 – Telas desenvolvidas pelo participante P1



Fonte: Próprio autor (2022).

Figura 39 – Telas desenvolvidas pelo participante P2



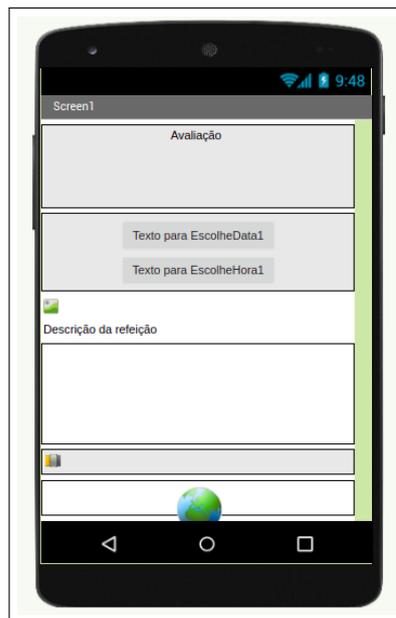
Fonte: Próprio autor (2022).

Figura 40 – Telas desenvolvidas pelo participante P3



Fonte: Próprio autor (2022).

Figura 41 – Tela desenvolvida pelo participante P4



Fonte: Próprio autor (2022).

Figura 42 – Telas desenvolvidas pelo participante P5



Fonte: Próprio autor (2022).

Figura 43 – Tela desenvolvida pelo participante P6



(a) Tela de de avaliação.

Fonte: Próprio autor (2022).

Figura 44 – Telas desenvolvidas pelo participante P7



(a) Tela de seleção da refeição.

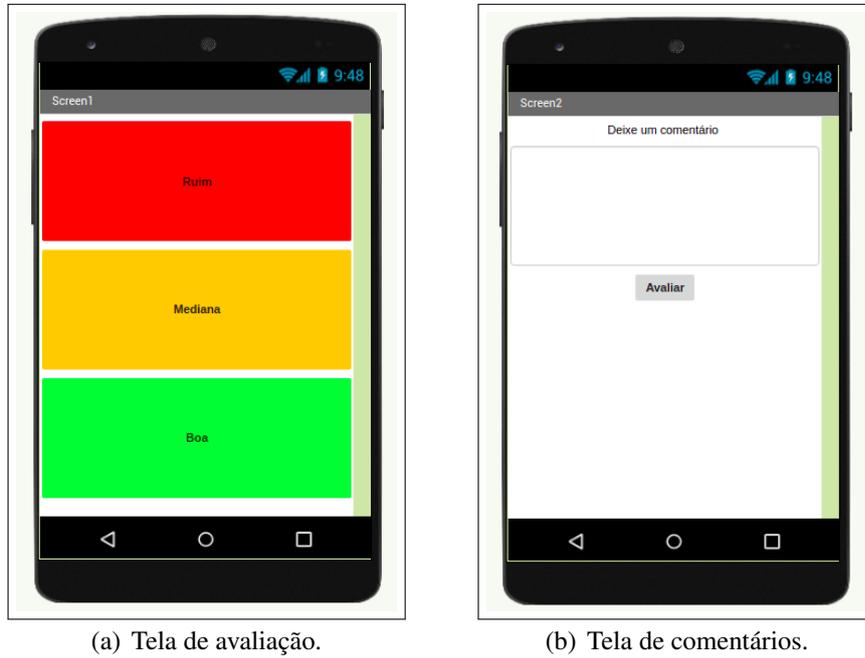
(b) Tela de avaliação.



(c) Tela de feedback.

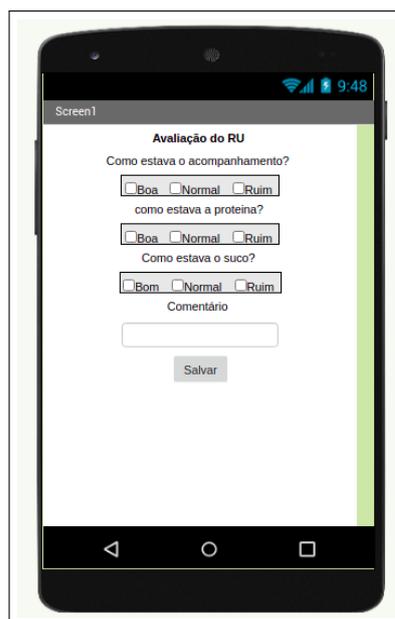
Fonte: Próprio autor (2022).

Figura 45 – Telas desenvolvidas pelo participante P8



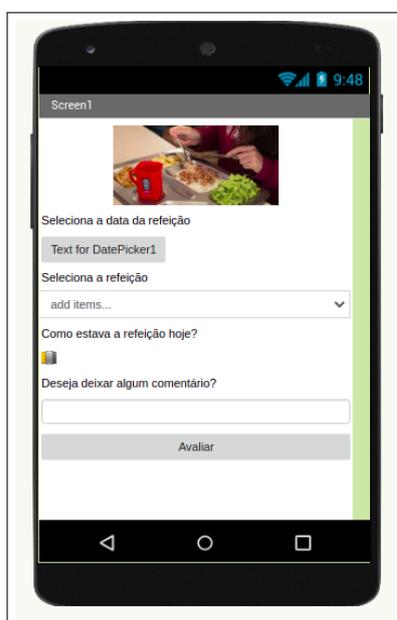
Fonte: Próprio autor (2022).

Figura 46 – Tela desenvolvida pelo participante P9



Fonte: Próprio autor (2022).

Figura 47 – Tela desenvolvida pelo participante P10



(a) Tela de avaliação.

Fonte: Próprio autor (2022).