Universidade Federal de Pernambuco
Centro de Informática

Graduação em Ciência da Computação

Criação e execução de testes através de método no-code: Sem a necessidade de escrita de código

Germano Pires de Carvalho

Trabalho de Graduação

Recife
14 de Dezembro de 2021

Universidade Federal de Pernambuco
Centro de Informática

Germano Pires de Carvalho

Criação e execução de testes através de método no-code: Sem a necessidade de escrita de código

Trabalho apresentado ao Programa de Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientadora: *Patrícia Cabral de Azevedo Restelli Tedesco*

Recife
14 de Dezembro de 2021

Aos meu Pais
Francisco e Sandra
Aos amigos que fiz na faculdade
André, Douglas, Aurelio e Plácido
E aos amigos que fiz no trabalho
Queilison, Manu, Bea, Christian, Paulo e Bruna.

Resumo

Para o processo de teste de software, é utilizado uma grande parcela dos recursos e tempo de desenvolvimento do produto. Para melhorar a efetividade dos testes, são utilizados os testes automatizados. Para tal são necessários profissionais que criem e executem esses testes. Engenheiros de testes nem sempre estão relacionados ao teste de software diretamente, mas com a validação das funcionalidades com as expectativas dos usuários finais. Isso provoca uma discrepância entre os profissionais que desenvolvem as automatizações dos testes e aqueles que as executam, acarretando uma maior demora na automatização, portanto, menos casos de testes automatizados. Para solucionar esses problemas, esse trabalho tem o objetivo de idealizar, não implementar, uma proposta de solução com um chatbot integrado ao google chat, que permita aos engenheiros de testes criar, editar e executar testes automatizados, por método no-code, sem o conhecimento em linguagens de programação, apenas descrevendo ao chatbot com inteligência artificial, os passos que esperam que o teste automatizado execute.

Palavras-chave: Engenharia de Testes, Automatização, Chatbot, No-Code, Inteligencia artificial, Google Chat.

Abstract

A large amount of resources and development time is used in the process of software testing. To increase effectiveness, automatized tests are used, but they require professionals who can create and execute them. Not all test engineers are directly related to software testing, though. Instead, they validate features with the user's expectations in mind. This fact induces a discrepancy between those that create and those that execute automated texts, generating a time delay in the automatization process and a smaller number of automated tests. To solve these problems, this academic work will idealize and validate a potential solution using google chat, leaving the implementation for future works. The proposed tool will allow test engineers to create, edit and run automated tests via a no-code method, without requiring any knowledge of programming languages from them. Instead, they only need to be able to describe those commands to a chatbot containing artificial intelligence, and the commands will be converted to code and executed automatically.

Keywords: Test Engineering, Automation, Chat-bot, No-Code, Artificial Intelligence, Google Chat.

Sumário

CAPÍTULO 1

Introdução

1.1 Relevância

Durante o desenvolvimento de um programa, quase a metade dos recursos são destinados à realização de testes (JAN e SHAH, 2016). Atividades nessa área são extremamente essenciais para entregar um produto com qualidade aos usuários finais. A própria fase de desenvolvimento pode possuir metodologias de criação de software, como especificado por (MOE, 2019): o método orientado a testes (TDD) utiliza de uma perspectiva muito interna, a do desenvolvedor, o método orientado a procedimentos (BDD), por outro lado, já possui uma visão mais abrangente, envolvendo desenvolvedores, engenheiros de testes e usuários finais, o que nem sempre é suficiente para corresponder as expectativas de negócios e dos usuários. Por isso também é utilizada a metodologia de aceitação (ATDD) em que antes do desenvolvimento, é decidido em conjunto com as equipes do projeto, quais serão os critérios para aceitar que o programa está funcionando como esperado.

A metodologia de aceitação é ideal para guiar o desenvolvimento do programa por ser focada na qualidade externa do sistema, contrastando com o TDD que é baixo nível. Nesse método nem todos os testes de aceitação serão realizados sobre os códigos, e sim sobre as funcionalidades. Isso é, garantindo que o usuário final receba o produto funcionando como esperado, e não necessariamente possuindo conhecimento interno da execução do projeto. Devido a essa grande abrangência de categorias validações, é possível que possa haver engenheiros de testes habilitados para trabalhar na área, mesmo sem possuir o conhecimento com programação ou as linguagens utilizadas no desenvolvimento. Outra característica do ATDD é que além de buscar que todos os envolvidos com o projeto compreendam o que precisa ser feito, também automatiza os testes (MOE, 2019). A automatização para os testes de aceitação é extremamente relevante, porque além de economizar bastante tempo, também evita que o profissional realize atividades repetitivas e exaustivas, melhorando a qualidade de trabalho. Por outro lado, a automatização nem sempre está implantada de forma satisfatória, visto que os profissionais de testes, nem sempre possuem conhecimentos em programação. Fica para outras equipes a responsabilidade pela criação dos testes automatizados. E por conta disso podem ocorrer atrasos e falhas de comunicação, problemas que poderiam ser evitados caso os próprios engenheiros de testes que executam os testes automatizados, pudessem cria-los e editá-los sem a necessidade de conhecimentos em linguagens de programação.

1.2 Objetivos e resultados

Baseando-se no ambiente de trabalho do autor, o convênio Cin/Motorola e a equipe de aceitação, que verifica se as funcionalidades dos dispositivos concordam com as expectativas do cliente, no caso o Google. Este trabalho idealiza e valida um chat-bot com finalidade de permitir aos engenheiros de testes, que não possuem conhecimentos em linguagens de programação, criar e executar testes pelo método no-code. Isso é, sem o método de programação tradicional (i.e. escrita de linhas de código utilizando Java, C++, entre outras linguagens) melhorando assim a automatização, reduzindo custos na área de testes e melhorando a satisfação no trabalho dos profissionais. Para alcançar tal objetivo será necessário realizar pesquisas sobre os métodos de desenvolvimento no-code, assim como as áreas de testes e tecnologias relacionadas a chatbots, além de pesquisas com os profissionais que atuam na área da proposta, e outros trabalhos relacionados, com a finalidade de estruturar a idealização do sistema, validando, absorvendo ou excluindo ideias da proposta, de modo a gerar o maior valor possível aos olhos dos engenheiros de testes. Para auxiliar na implementação do sistema idealizado esse trabalho também fornece uma recomendação de implementação e design da proposta idealizada de modo a facilitar o desenvolvimento e gerar um bot com melhor usabilidade para os profissionais.

O objetivo desse trabalho foi atingido, obtendo os seguintes resultados:

- A idealização de um chat-bot que permite criação de testes pelo método no-code.
- Validação da ideia com engenheiros, melhorando a geração de valor da proposta.

A demonstração do processo de obtenção desses resultados é abordada nos próximos capítulos na ordem:

- Capítulo 2, a contextualização de certos termos sobre a área abordada.
- Capítulo 3, a demonstração da metodologia utilizada para a elaboração do trabalho.
- Capítulo 4, o fluxo do desenvolvimento da ideia, suas validações e alterações.
- Capítulo 5, a recomendação de integração e design do chatbot com o google chat.
- Capítulo 6, a conclusão do trabalho.

CAPÍTULO 2

Revisão Bibliográfica

Este capitulo aborda conceitos básicos para o entendimento da proposta do trabalho, facilitando a compreensão da proposta idealizada no capítulo seguinte.

2.1 Conceitos

2.1.1 Programação

De acordo com Amorim (2015) chama-se software o programa que possui a responsabilidade de coordenar os passos de execução de tarefas dentro de um sistema. Programas funcionam como os meios para utilização de diversos produtos e serviços. Logo, o desenvolvimento de novas tecnologias digitais é sinônimo de desenvolvimento de software, que requer conhecimentos em linguagens de programação, método utilizado para a escrita de programas.

Acompanhando o desenvolvimento de novas tecnologias, está a crescente busca por desenvolvedores, onde Gordon (2021) informa que a secretaria de estatística trabalhista dos Estados Unidos prevê um crescimento de 13 por cento entre 2018 e 2028 para o ramo de desenvolvedores web. Esses profissionais são alguns dos que possuem o conhecimento em linguagens de programação e.g. Java, C++, Python, e suas sintaxes, e semânticas necessárias para escrever o código dos programas especificando as sequências de instruções que um sistema irá receber ou executar.

Para simplificar o processo de desenvolvimento de software, há plataformas de abstração das linguagens de programação.Essas plataformas podem ser do tipo *low-code e no-code*. Enquanto a primeira possui uma maior abstração da programação tradicional, com a presença de linguagens de programação, em menor quantidade, a no-code, por outro lado, abstrai totalmente a escrita de código pelo modelo típico, utilizando-se de outras formas de programação. Uma descrição de uma plataforma de desenvolvimento low-code é a seguinte, com tradução do autor:

> *"Uma plataforma de desenvolvimento Low-code (LCDP) é um software na nuvem que tem como alvo não programadores desenvolver programas sem possuir conhecimento em tecnologia da informação. Migrando o estado de desenvolvimento de código manual e linguagens de programações tradicionais por interfaces gráficas interativas, usando componentes pré construídos e definindo configurações. As interfaces, lógicas de negócio, serviços de dados são construídos por diagramas visuais, abstração de alto nível e linguagens declarativas e em casos específicos por código manual."*
>
> —KHORRAM; MOTTYU; SUNYÉ (2020, p. 1)

Tais plataformas propõem alterar o método de desenvolvimento de programas, abstraindo o método convencional de programação, facilitando para pessoas com menor afinidade ou conhecimento das linguagens de programação criar, editar, executar e testar códigos.

Um exemplo de ferramenta no code é o AppInventor criado pelo google e mantido pela MIT Media Lab, que facilita a programação android. Utilizando-se de bloco pre-montados que devem ser organizados de modo a criar o programa desejado. Pode ser também utilizado como uma forma de ensinar programação para iniciantes,

2.1.2 Teste

Seja para tamanho de roupa, ou qualidade de software, testar é fundamental. Ao calçar número de sapato quarenta, e testar a proposta que um número a mais calçaria bem, é verificado se o tamanho quarenta e um é compatível com o tamanho quarenta, se os requisitos desejados, como conforto e mobilidade são satisfatórios a proposta é validada, caso contrário é rejeitada. O mesmo acontece com os testes de software onde desenvolvedores e engenheiros de testes verificam se a execução satisfaz as especificações esperadas, detectando e corrigindo erros, melhorando a qualidade do software (BERNARDO e KON, 2008).

Considerando que a fase de teste engloba cerca de 40 a 50 porcento dos recursos e tempo de desenvolvimento, e é crucial para garantir diversos padrões de qualidade como confiabilidade, disponibilidade, capacidade, eficiência, integridade e segurança, os testes de software possuem diversos objetivos, como detecção de erros e prevenção, providenciando informações sobre imperfeições para melhorar a qualidade geral do produto, verificando se os requisitos funcionais foram atendidos, e validando se o desenvolvimento foi realizado como o especificado pelo cliente (JAN e SHAH, 2016).

Para atingir esses objetivos, são utilizadas diferentes estratégias e técnicas, como relatadas por Sawant, Bari e Chawan (2012). As técnicas são divididas em manual, um processo lento de revisão informal ou técnica, inspeção e utilização do software, realizado pelo desenvolvedor, ou engenheiro de testes. Já a técnica automatizada aborda a satisfação dos requisitos mínimos por três diferentes abordagens, a primeira com conhecimento total do código interno, técnica chamada White box, a intermediaria, com conhecimento parcial, Gray Box e a técnica com ausência de conhecimento do código, a Black Box. Depois desses testes, há o teste de performance que visa identificar gargalos na velocidade de carregamento, e então a confiabilidade que busca encontrar problemas de design para garantir a satisfação dos requisitos, e o teste de

segurança que gerencia o acesso às funcionalidades do programa, evitando invasões.

As estratégias utilizadas nos testes, são metodologias que guiam sua realização, algumas delas são: os testes de unidade, que validam o funcionamento das menores partes testáveis, ou seja, em desenvolvimento de software se refere aos testes das funcionalidades mais simples do programa, como, por exemplo, divisões e outros cálculos matemáticos, testes simples, mas relevantes, visto que problemas na base de um programa acarretará problemas maiores com o passar do desenvolvimento. Os testes de integração buscam criar a estrutura do programa, entrelaçando módulos de código e testando a regressão, ou seja, que implementações novas, não danifiquem as anteriores. Os testes de sistema verificam se o programa consegue atender aos requisitos de segurança evitando o acesso a dados sigilosos, validando que a interface responde às ações de entrada, e recuperação, que engloba a robustez do sistema á falhas. Os testes de aceitação ou validação verificam se o sistema atende os requisitos, padrões e critérios especificados pelo cliente, visa muito mais a qualidade demonstrada pelo produto final do que o funcionamento interno do sistema.

Há diversas categorias de testes, estratégias, objetivos e técnicas, em algumas, desenvolvedores atuam, em outros os engenheiros de testes. Para os testes de aceitação ou validação, que buscam validar o sistema para os usuários finais, o conhecimento interno do código não é sempre necessário, pois existem testes que simulam a utilização dos clientes, validando apenas as funcionalidades que podem ser visualizadas por um usuário, ou seja, sem atuação no código interno (SAWANT; BARI; CHAWAN, 2012) por isso nem sempre os engenheiros de testes focados em testes de aceitação possuem habilidades de programação ou entendimento de linguagens de programação.

> *"A execução manual de um caso de teste é rápida e efetiva, mas a execução e repetição de um vasto conjunto de testes manualmente é uma tarefa muito dispendiosa e cansativa."*
>
> —BERNARDO E KON (2008,p. 1)

Há diversos testes manuais de aceitação, cansativos para o engenheiro de testes, por serem simples, repetitivos e sem motivação, que além de tomarem muito tempo são também passíveis a falhas humanas. Esses testes podem ser automatizados, mas ocorrem diversos problemas como a presença de falhas ou falta de abrangência dos casos de testes devido à falta de comunicação entre as equipes de execução de testes, pois engenheiros de testes nem sempre possuem conhecimentos ou afinidade com linguagens de programação, e por conta disso não entregam um feedback suficiente para aqueles que automatizam, nem são capacitados para automatizarem testes por si mesmos.

2.1.3 Automatização

Um dos maiores objetivos tecnológicos é o de melhorar a qualidade de vida. Para atingir esse objetivo no que se refere ao ambiente de trabalho, a automatização se faz necessária, visto que automatizar é substituir atividades humanas em ações robotizadas. Para os engenheiros de testes, especificamente para os testes de aceitação, é uma abordagem extremamente efetiva,

pois, por estar focada em testar requisitos básicos de qualidade: aceitação e correção, seus resultados são extremamente valiosos no desenvolvimento, ao validar as novas funcionalidades e sua compatibilidade com as já existentes (BERNARDO, 2011). Por isso deve-se evitar ao máximo serem passiveis a falhas humanas.

> *"Testes automatizados são programas ou scripts simples que exercitam funcionalidades do sistema sendo testado e fazem verificações automáticas nos efeitos colaterais obtidos. A grande vantagem desta abordagem, é que todos os casos de teste podem ser facilmente e rapidamente repetidos a qualquer momento e com pouco esforço."*
>
> —BERNARDO E KON (2008,p.2)

Essa definição descreve os scripts possuindo duas vantagens: a velocidade e a facilidade de repetição. Ambas características estão relacionadas ao tempo de desenvolvimento, visto que automatizar testes possui a função de reduzir repetições. Deve-se analisar se é vantajoso, o custo da automatização do teste, para o ganho de tempo na execução entre o método manual e o automatizado, considerando também a priorização da automatização de testes em que falhas humanas são erros críticos (IZABEL, 2014).

Para automatizar os testes é necessário escrever os scripts, que vão conter as sequências de ações que devem ser executadas, assim como as respostas esperadas pelo sistema, de modo a aceitar ou rejeitar as funcionalidades testadas em questão. Para construção desses testes automatizados é necessário um conhecimento em linguagens de programação, e desenvolvimento de software, atividades em que nem sempre está relacionada às funções exercidas pelos engenheiros de testes, gerando um atraso na automatização de novos testes ou automação em baixa quantidade em relação à quantidade de testes. Para solucionar essa problemática e permitir aos engenheiros de testes programar a automatização dos testes, há diversos métodos que buscam facilitar o aprendizado das linguagens de programação, sejam as plataformas low-code e no-code de construção de testes ou até mesmo formas diferenciadas de ensino como um chatbot que auxilia ao aprendizado (HOBERT, 2019).

2.1.4 ChatBots

São alguns dos dispositivos inteligentes, que utilizam da inteligência artificial (IA) para compreender as ações desejadas pelos usuários, como, por exemplo, preenchimento automático de frases ou pesquisas por comando de voz. Uma definição mais detalhada é a seguinte, com tradução livre do autor:

> *Um chatbot é um exemplo tipico de uma sistema inteligente, e um dos mais diversos exemplos de interação humano-computador. É um programa de computador, que responde como uma entidade inteligente quando interage por texto ou voz, e consegue compreender uma ou mais linguagens humanas, por processamento de linguagem natural (NLP), formalmente é "Um programa de computador desenvolvido para simular uma conversa entre usuários humanos, especialmente na internet"*
>
> —ADAMOPOULOU E MOUSSIADES (2020,p.1)

A funcionalidade de simulação de conversas humanas pelos chat-bots possui diversas atuações na sociedade, como assistentes virtuais, marketing, suporte ao cliente, educação e entretenimento. Devido à capacidade de entender a linguagem humana, utilizando-se do processamento de linguagem natural que extrai o contexto e significado do texto ou voz para identificar a intenção do usuário mapeando para a tarefa desejada. Por exemplo, ao falar "ok google" a identificação da expressão reservada para a tarefa de despertar o assistente virtual da Google é ativada, gerando também uma resposta no formato humano como "Boa tarde". Atividades simples, mas que requerem um bom desenvolvimento do conhecimento do banco de dados, em que serão buscadas as respostas às tarefas solicitadas (PEMBRIDGE e VERLEGER, 2018).

Para cumprir com sucesso suas atividades, um chatbot pode possuir diversas características como mencionadas por Wei, Yu e Fong (2018) : autoconsciência, detecção e simulação de sentimentos, capacidade de bloquear conteúdos sensível, assim como abstração, fundamental para interpretar e predizer as ações dos usuários, características essas relacionadas com a inteligência artificial que os chat-bots possuem.

2.1.5 Inteligência Artificial

A inteligência artificial que fornece tantas habilidades aos chat-bots para realizarem seus objetivos, como auxiliar clientes na realização de suas compras ou suporte ao consumidor, é descrita como:

> *"It is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, ... Intelligence is the computational part of the ability to achieve goals in the world."*
>
> —MCCARTY (2007,p.2)

Ou seja, a inteligência artificial é uma área da ciência da computação em que dispositivos possuem a habilidade de compreender a lógica humana, e utilizar desse entendimento para cumprir objetivos. Um objetivo a longo prazo da IA é justamente atingir o nível de inteligência de um ser humano, pelo aprendizado baseado em experiências, o que depende de uma abundância de dados disponíveis.

Interior a inteligência artificial há a aprendizagem de máquina, um fator fundamental para os chatbots detectarem os sentimentos dos usuários e simularem empatia podendo por meio do reconhecimento e correspondência de padrões, prever ou representar estímulos e respostas ao usuário expandindo a área de atuação de ferramentas para companheiros virtuais. Uma linguagem que utiliza dessa técnica é a Artificial Intelligence Markup Language (ADAMOPOULOU e MOUSSIADES, 2020) para o desenvolvimento de software utilizando marcações em categorias. padrão e modelo, como na tabela 2.1, que demonstra a entrada do usuário e o modelo de resposta.

```
<category>
<padrão> Olá meu nome é * </padrão>
<modelo> Olá * eu sou um chatbot</modelo>
</category>
```

Tabela 2.1 Exemplo de código em Artificial Intelligence Markup Language

Processamento de linguagem natural, processo que identifica e traduz a linguagem humana para o sistema, aprendizado de máquina e inteligência artificial são tecnologias que compõem os chatbots, podendo atuar em diferentes áreas de serviço como restaurantes, ou atendimento virtual, variando sua personalidade entre apenas transmitir informações ou criar empatia com o usuário, dependendo se seu objetivo é focar na realização de tarefas ou se relacionar empaticamente com os clientes. Para modelar os chatbots de modo a atingir seus objetivos são utilizados diferentes metodologias, os baseados em regras geralmente possuem compreensão limitada, utilizando de padrões fixos de entrada e saída (ADAMOPOULOU e MOUSSIADES, 2020), por exemplo, ao receber “Olá” responderia “Olá, como está”, mas ao receber “Oi” poderia não possuir uma resposta válida. Há diferentes métodos de modelagem que os chatbots utilizam para escolher as respostas para as mensagens recebidas, como, por exemplo, à modelagem de recuperação analisa entre diversas opções de respostas levantadas pela inteligência artificial, qual se encaixa melhor no contexto. Enquanto as generativas utilizam do histórico de mensagens para predizer qual será a melhor resposta, considerando também fatores como detecção e simulação de sentimentos para agir humanizadamente (ADAMOPOULOU e MOUSSIADES, 2020), logo o desenvolvimento de um chatbot leva a diversas opções de modelagem, para simplificar e focar em trazer valor aos usuários, se faz necessário uma pesquisa com aqueles que usarão o sistema.

CAPÍTULO 3

Metodologia

Este capÍtulo aborda o processo de construção do trabalho, com o intuito de esclarecer a sequência de procedimentos, desde a procura do tema até a concepção da versão final.

3.1 Definição do tema

Desde o primeiro momento da criação deste trabalho, sempre esteve determinado que iria ser abordada a área de engenharia de testes, visto que o autor possui conhecimentos e contato frequente com o tema por ser um engenheiro de testes. O primeiro passo foi realizar uma pesquisa bibliográfica sobre diversas palavras-chave: engenharia de testes, testes, automação, testes de software e testes manuais. Onde foram sintetizados nove possíveis temas e dez temas que não seriam interessantes para trabalhar. Dentre as possibilidades positivas estavam: "E se existisse uma ferramenta de execução de testes, que siga o script, mas assim como os humanos, possua curiosidade de executar ações não programadas e conferir o que aconteceria?" e "E se uma IA ao invés de apenas executar testes, identificasse o motivo da falha e corrigisse", mas a escolhida foi "Seria possível criar scripts automatizados sem código? Ideia essa gerada em sua maior parte pela interpretação dos artigos *Continuous Integration in Automation Testing* por Gota L., Gota D. e Miclea L. (2020) e *Evaluating the Impact of Different Testers on Model-based Testing* por Silva, Mattiello, Endo, Souza E., Souza S. (2018) que abordam a questão da automatização com uma menor quantidade de uso de códigos escrito de forma tradicional, i.e. por linguagens de programação e a relação dessa categoria de código com aqueles que criam e executam os testes.

3.2 Pesquisas sobre o tema

Com o tema definido, foi realizada mais uma pesquisa, agora focada no tema escolhido, onde foi descoberto diversas informações, como *Esolangs*, categorias de linguagens de programação não convencionais, que não possuem como objetivo serem práticos ou fáceis, mas serem divertidos ou desafiadores (TEMKIN, 2017). Assim como plataformas de desenvolvimento online, que abstraem a programação em blocos pré montados, facilitando a criação de sites online (KAPPI, 2021). Conhecimentos fundamentais que surgiram com a pesquisa foram a existência de métodos de programação low-code que abstrai a programação clássica, i.e. Java, C++, Python em novas modalidades mais fáceis de se utilizar, como, por exemplo, diagramas

visuais, exemplificado por Khorram, Mottu e Sunyé (2020), e no-code que se difere do low-code por ser mais completo, abolindo totalmente o uso do código tradicional. A abordagem no-code encaixou perfeitamente com o tema do trabalho, de forma que validou a existência de um método de programação sem a utilização de código, restando analisar como seriam criados os scripts automatizados. A partir deste momento, iniciou-se a procura por temas relacionados a automatização de testes, que resultou nas áreas de aprendizagem de máquina, inteligencia artificial e chat-bots.

3.3 O processo de Identificar o problema e idealizar a solução

Com esses temas em mãos, o autor realizou uma auto avaliação, de modo a procurar um ponto de dor, em seu trabalho, onde a utilização de inteligência artificial e chat-bots poderia compor uma solução. Foi identificado que, embora houvesse automação para execução de testes, o nível de automação não era suficiente para o quantitativo de testes realizados, e que muitas vezes os testes automatizados estavam desatualizados e com falhas de execução, causando desmotivação no uso da modalidade de testes automatizados e preferência pela execução manual dos testes.

Com o suposto ponto de dor, as tecnologias e o tema determinados, foi realizada uma pesquisa qualitativa, com engenheiros de testes e de software de modo a analisar a metodologia de trabalho e as ferramentas utilizadas, tal que o ponto de dor levantado pudesse ser validado. Também foi desenvolvida uma versão básica da proposta, descrita para os integrantes da pesquisa e debatida questões sobre automatização, programação e chat-bots, com a finalidade de identificar melhorias, atender a requisitos dos profissionais e possibilitar a criação do design do projeto.

Auxiliando a pesquisa com os engenheiros de testes, também foi realizada uma revisão dos trabalhos relacionados, onde foi possível interpretar e absorver para a proposta diversas características positivas, que agregam valor para aqueles que usarão o sistema, tal qual a possibilidade de exportar arquivos de testes, suprindo uma necessidade levantada pelos engenheiros que é o compartilhamento de testes. Os resultados da pesquisa com os engenheiros e a revisão de trabalhos relacionados, possibilitaram desenvolver uma nova versão para a proposta, que cumpre com os desejos dos pesquisados e integra as funcionalidades das propostas semelhantes. Nesse momento foi desenvolvida um fluxograma de uso da proposta idealizada até o momento, assim como foi demonstrado o design do chat-bot e exemplificado uma possível execução do sistema por parte de um engenheiro de testes.

Com a proposta desenvolvida, foi realizada novamente uma pesquisa com engenheiros, mas dessa vez apenas com engenheiros de testes, visto que engenheiros de software não fazem parte do público alvo deste trabalho, informação descoberta apenas após a realização da primeira pesquisa com os engenheiros. Essa segunda pesquisa, agora quantitativa, valida as características atribuídas ao sistema, ao nível de valor que agrega aos engenheiros de testes, permitindo validar se as funcionalidades idealizadas são realmente importantes para aqueles que utilizarão o sistema, de modo a remover as funcionalidades com baixo valor de relevância, simplificando o sistema.

Com a versão retrabalhada pela validação, a finalização do trabalho se deu pela instrução de

como implementar a proposta idealizada, especificando os requisitos necessários, assim como as ferramentas e tecnologias recomendadas para a construção da proposta.

CAPÍTULO 4

Desenvolvimento

4.1 Tópicos sensíveis

Neste capítulo será demonstrado o processo de criação da proposta deste trabalho, com suas fases de pesquisa, idealização e validação. Com isso aqueles que desejarem implementar a proposta deve estar de possuir o conhecimento dos seguintes tópicos.

- Manutenção

Baseada na ferramenta interna de execução de testes automatizados, utilizada no ambiente de trabalho do autor. A manutenção do chat-bot, deve considerar que o engenheiro de teste poderá observar em tempo real a execução dos passos do teste, ou seja. Caso haja um evento inesperado que quebre o teste, o engenheiro de teste deve tomar as ações correspondentes. Como, por exemplo: durante a execução de um teste que verifica a conectividade bluetooth entre dois devices, um pop-up que nunca apareceu antes nesse caso de teste e que não impacta no resultado do teste, pode ser vista. Ao observar que a ferramenta não conseguiu progredir com a execução devido a esse pop-up, o engenheiro deve utilizar da ferramenta do chat-bot "Editar"e adicionar um novo comando que atenda a essa nova situação.

- Privacidade

Outro ponto importante para os desenvolvedores, é que a proposta idealizada utilizou do google chat como plataforma de integração do chatbot. Portanto, deve-se ter a permissão da empresa a qual irá implementar, de que seus dados, que podem ser confidenciais poderão ser utilizados no google chat. Isso pode acarretar uma maior diminuição da confidencialidade e segurança dos dados, por outro lado, considerando que o google drive é uma plataforma de armazenamento de dados amplamente utilizada, e que o google chat também faz parte dos serviços Google, a diminuição da segurança pode não ser relevante. Mas se mesmo assim for de desejo da empresa, também é possível desenvolver a proposta sem a integração com o google chat, tomando como responsabilidade criar um programa específico para a proposta.

- Versionamento

Para o controle de versão da proposta idealizada é de responsabilidade do engenheiro de testes, assegurar que o produto a ser testado está na sua versão correta e que o teste a ser executado é o correspondente. Tomando como exemplo um caso real do autor: ao utilizar uma ferramenta de automação é necessário verificar que o dispositivo móvel (produto) está na build correspondente ao pedido do cliente (versão correta) e que a ferramenta de automação está com os scripts correspondentes a versão pedida, ou seja, se a versão do teste "A"que será executada automaticamente será a do android 10 ou 11.

4.2 Pesquisa com Engenheiros

Com os conceitos abordados, nesta seção se descreve o início do processo de criação da solução. Processo inicial que consta como motivação, atuar na área profissional do autor, i.e. engenharia de testes, logo uma auto avaliação foi realizada.

A auto análise foi delimitada para a área de testes de aceitação, realizada pelo autor, que abordam configuração de ambiente e execuções manuais ou automatizadas, onde para as automatizadas foi encontrado um ponto de dor onde toda a automação é realizada pela equipe de automação e não a equipe que executam os testes, com isso há um atraso na atualização dos testes automatizados e baixa quantidade de automação, tornando-se conveniente executar os testes manualmente pela vantagem de tempo. Essa foi a motivação para criação da proposta desse trabalho, permitir que os engenheiros de testes (testers), sem conhecimento de programação possam criar seus próprios testes, melhorando a automação.

A proposta inicial se tornou: idealizar, sem implementação, um sistema de chatbot onde o engenheiro de testes poderia escrever os passos de execução dos testes, como, por exemplo:

Abrir câmera
Tirar uma foto
Abrir galeria de imagens
Conferir se a foto foi tirada
Repetir o teste 2000 vezes

Tabela 4.1 Exemplo de passos de criação de um teste utilizando a proposta inicial do trabalho.

Onde uma inteligência artificial seria responsável por converter os comandos de entrada, em linguagem humana, para o código do teste e executá-lo, removendo a necessidade dos engenheiros de testes possuírem conhecimentos em linguagens de programação para criação de scripts.

Com a proposta em sua forma inicial, foi seguido o procedimento de análise agora com outros engenheiros de software, desenvolvedores e engenheiros de testes, todos com conhecimentos na área de testes. A pesquisa foi composta por questões descritivas, buscando visualizar novas informações não abrangidas pela auto análise, dividida em três tópicos: a primeira sendo questões demográficas, seguindo com a análise de ferramentas e atividades realizadas e por fim questões relacionadas a proposta inicial.

Para as questões demográficas foram encontradas as seguintes informações: A idade dos profissionais pesquisados variaram entre vinte e dois e trinta e sete anos, em graduação completa ou cursando de Sistemas de informação, ciência da computação e Analise e desenvolvimento de sistemas, 75 porcento do gênero masculino, sendo 62 porcento engenheiros de testes, vinte e cinco como engenheiros de software e treze como desenvolvedores.

Para análise de ferramentas e atividades foram os seguintes questionamentos: quais as atividades costumam realizar e quais ferramentas utilizam antes, durante e depois das atividades, ou seja, não apenas ferramentas de trabalho, mas de comunicação e afins. Em seguida foi questionado o nível de programação e afinidade com á área de criação e execução de código

convencional, sem a proposta, e então questionado se há pontos de dor nas atividades realizadas pelos entrevistados, o que sentem e se acreditam que a automação poderia ser uma solução viável.

Foi possível sintetizar os dados em informações, descritas a seguir. Para a abordagem de atividades foi visto que engenheiros de software e desenvolvedores atuam na criação de testes de unidade e de integração, assim como criam testes automatizados e os testam. Já para os engenheiros de testes foi identificado uma maior abordagem em testes, como de API, aplicativos, integração, sanidade, assim como liderança de equipe, validação de ambiente de testes, além da criação de testes e automação. Com isso foi possível perceber que os principais testes dos engenheiros de software são testes no próprio código, logo ineficaz para proposta, visto que são testes sobre código e isso implica que os profissionais já possuem conhecimento satisfatório em linguagens de programação, por outro lado, os engenheiros de testes possuem mais possibilidades de atuação para o chatbot.

Para as ferramentas foram identificadas diversas opções, que passaram por um filtro, removendo aquelas sem aplicabilidade com chat-bots, restando a porcentagem de uso das ferramentas de comunicação abaixo:

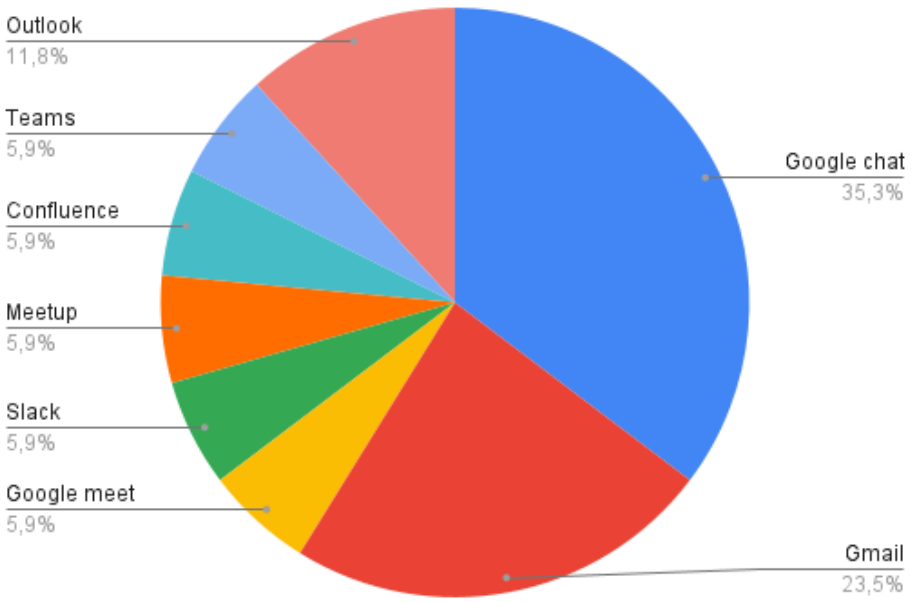


Figura 4.1 Ferramentas de comunicação mais utilizadas durante o trabalho pelos profissionais pesquisados

Fomentando a ideia que o chatbot deve ser idealizado para ser utilizado com o Google chat, o que é validado por uma pergunta da próxima seção, i.e. após a descrição da proposta, onde a maior parte dos pesquisados afirmaram que se sentiriam mais confortável utilizando o chatbot integrado a uma ferramenta que eles já utilizam se comparado a possibilidade de utilizar um

aplicativo separado unicamente para o chatbot.

Para as perguntas relacionadas com programação foi identificado que os engenheiros de software e desenvolvedores, como esperado, possuem nível bom e avançado em programação, assim como afinidade com a área, por outro lado, os engenheiros de testes possuem 20 porcento com nível baixo de programação, 60 por cento com nível intermediário e o restante no nível bom, sobre a afinidade com a área de criação de código ou possuem, ou dizem poder possuir se receberem treinamento adequado. O que fortalece o ideal de aplicabilidade do chatbot para auxiliar esses profissionais na área de programação de testes.

Entrando na seção de automação foi identificado que os engenheiros de software não creem na possibilidade de automação de suas atividades, por não haver repetições, por outro lado, os desenvolvedores e engenheiros de testes mencionam o sentimento de cansaço e repetição, por exemplo, testes de regressão e testes de câmera (dispositivos móveis), também para os engenheiros de testes foram destacados pontos de dor como: automação com nível insuficiente e presença de testes não devidamente automatizados. Identificado e validado o ponto de dor levantado na auto avaliação do autor, onde há automatização, mas não é o bastante para ser eficiente.

Por fim foi descrito na pesquisa a proposta no seu estado inicial, e solicitado que os pesquisados relatassem se conhecem propostas semelhantes, quais as formas de comunicação utilizam no trabalho e se estariam confortáveis em utiliza-los integrados ao chatbot da proposta, se esse bot deveria se comportar como ferramenta ou com simulação humana. além de verificar se há valor agregado aos profissionais, se acreditam que ao utilizar a ideia proposta, suas atividades seriam realizadas mais facilmente, e se os incentivariam a atuar na área de programação. Por fim foi analisada a questão ética se inteligencia artificial é confiável a ponto de poder julgar como aprovado ou rejeitado os testes, ou se isso deve ser trabalho de um ser humano.

No que se refere a propostas semelhantes ao deste trabalho, foram mencionados em 75 porcento dos pesquisados a falta de conhecimento de tais ferramentas, mas relatam pontos positivos da proposta, como a possibilidade de escrever a descrição dos passos sobre a utilização de interface de usuário para configuração de testes. Para as ferramentas semelhantes relatadas, Selenium e Appium, foram mencionadas como de difícil utilização, com requerimento de um alto conhecimento em automação. Sustentando a base da proposta do trabalho, permitir que qualquer engenheiro de testes possa criar e executar testes, sem conhecimento prévio em linguagens de programação. Embora a maioria não tenha expressado conhecimento de propostas semelhantes, foi relatado que entre a utilização de chatbot convencional, com simulação de ser humano, contendo sentimentos, humor e afins ou se a abordagem direta seria preferível, obtendo as seguintes respostas: A maioria preferiu utilizar o chatbot como ferramenta, facilitando o processamento da inteligência artificial e focar apenas na atividade, de modo a otimizar a eficiência no trabalho, uma minoria preferiu uma abordagem humanizada e apenas um relatou que depende da situação.

Quando analisada a atuação dos profissionais em atividades que requerem envolvimento com programação para engenharia de testes, mais da metade dos que não estão englobados nessas atividades e mencionam que se sentiriam com mais vontade de atuar na área, se existisse a proposta idealizada por esse trabalho, como exemplificado pelo engenheiro de software:

“E no meu caso, como atuei com teste de software, uma ferramenta que facilite a execução de testes automatizados seria muito bem-vinda e aumentaria a produtividade. Então sim, esse tipo de "programação"caso existisse na época em que trabalhei com testes, me deixaria mais satisfeito.”

—CRISTIANO

Para engenheiros de software que já atuam em áreas que envolvem criação de código, com linguagens de programação, é relatado que a proposta não geraria valor, pelo motivo mencionado anteriormente, que não acredita ser possível automatizar suas atividades pela falta de repetição. Já os engenheiros de testes variam entre pouco valor, visto que já há uma equipe focada em realizar automações, relato que não coincide com a informação dita anteriormente pelos engenheiros de testes se sentirem cansados e entendiados com atividades repetitivas, outro engenheiro de teste menciona que o chatbot poderia entregar valor pela possibilidade de troca de conhecimentos entre testers, o que seria possível por um sistema de compartilhamento de testes, assim o passo a passo de como realizar os testes poderiam ser distribuídos entre os profissionais.

Por fim foi perguntado uma questão ética, que envolve confiar no sistema proposto para criar, executar e julgar os testes, ou deixar a responsabilidade de julgamento para um profissional humano. Obtendo como dados que 50 porcento dos engenheiros de software confiam completamente, a outra metade prefere deixar o julgamento para o humano, assim como os desenvolvedores, por outro lado, todos os engenheiros de testes pesquisados acreditam que a inteligência artificial pode tanto criar e executar como julgar, se o resultado do teste é aprovado ou reprovado, desde que possua uma taxa de acerto muito próximo de 100 porcento. Vale ressaltar que possuindo os engenheiros de testes como publico alvo da proposta, é um ótimo achado a informação de que há uma confiança na proposta.

No final da pesquisa foi constatado ter sido essencial a sua realização, identificando as atividades realizadas pelos profissionais, o seu nível de afinidade e habilidades de programação, a identificação da maior utilização do Google Chat como ferramenta de comunicação, assim como sentimentos positivos em integração com o chatbot proposto, validação dos problemas de automação e pontos de melhoria. Tornando assim fatores que serão utilizados com os dados do estudo de caso, mostrado mais para frente, para idealização da versão final da proposta.

4.3 Trabalhos Relacionados

Para o contexto de chatbots existem diversos projetos semelhantes, as mencionadas abaixo são de maioria com funcionamento de auxiliar ao aprendizado, neste capítulo serão analisadas as suas características, isto é o que e como fazem, de modo a buscar pontos de dor e necessidades não atendidas assim como inspirações de criação e melhoria da proposta idealizada neste trabalho.

3.2.1 Programming an Educational Chatbot to Support Virtual Learning

É demonstrado por Ai, Dhar, Kohli, Maina, Manelski, Ramos e Brzycki (2020) a criação de um chatbot focado em realização de tarefas e comandado por regras. Com atuação no aplicativo de comunicação Discord, o bot pode auxiliar virtualmente a aprendizagem, servindo como ponte entre professores e alunos. O sistema possui o uso de processamento de linguagem natural, aprendizagem de máquina e banco de dados (SQL) que permite salvar os dados quando o servidor em que se está hospedado está desligado. O funcionamento desse chatbot se dá por utilização de prefixos que ativam funcionalidades do chat-bot. Uma analogia seria o "ok Google" que ativa o bot do Google, e responde à mensagem falada em seguida pelo usuário.

O projeto em questão embora utilize de aprendizagem de máquina, não possui discernimento entre palavras quaisquer, por ser fixo em regras, é necessário utilizar de comandos específicos que o bot consegue compreender. Logo a proposta para auxiliar na criação de testes, deve possui uma maior abrangência nesse quesito, onde o usuário poderá descrever de diferentes formas os passos que deseja que o bot execute.

Deste caso também é possível obter o conhecimento que o chatbot deve possuir algum sistema robusto de armazenamento do banco de dados, para evitar perda de informações. Assim como forneceu uma ideia a ser implantada na proposta: O armazenamento e reprodução de comandos, visto que para a criação de um teste é utilizado uma sequência de comandos em ordem, e para criação de outro caso de teste é utilizado a mesma sequência, apenas variando comandos anteriores ou posteriores, é viável podermos guardar comandos e reutiliza-los, como, por exemplo.

Caso de Teste 1	Caso de Teste 2
Abrir câmera	Abrir bluetooth
Tirar uma foto	Parear com dispositivo disponível
Abrir galeria de imagen	Abrir câmera
Verificar que a imagem existe	Tirar uma foto
	Abrir galeria de imagens
	Verificar que a imagem existe
	Abrir imagem
	Enviar imagem por bluetooth

Tabela 4.2 Exemplos de casos de testes recebidos pelo chatbot proposto neste trabalho.

Os comandos do caso de teste 1 poderão ser armazenados em um comando (T1) e então quando for mencionar um novo caso de teste, poderá ser aplicado no local, demonstrado abaixo:

3.2.2 Implementation Chatbot WhatsApp using Python Programming

Outro caso é o chatbot para WhatsApp criado por Ramaditiya, Rahmatia, Munawar e Samijayani (2021), utilizando inteligencia artificial e processamento de linguagem natural. Por ser criado para o WhatsApp possui baixo custo e alto alcance pela quantidade de pessoas que utilizam o aplicativo, além de estar disponível para receber entrada de informação por texto, imagem e arquivos, o destaque para esse modelo é que as perguntas dos usuários não são apenas

Caso de Teste 2 com armazenamento de T1
Abrir bluetooth
Parear com dispositivo disponível
T1
Abrir imagem
Enviar imagem por bluetooth

Tabela 4.3 Caso de teste 2 em que há a utilização do caso de teste 1 previamente armazenado.

respondidas, mas acrescentadas ao banco de dados servindo como conhecimento futuro.

De tal forma a proposta do trabalho pode absorver mais duas características, a primeira constitui-se de que a capacidade do bot de ler as entradas no chat pode também ser expandida para leitura de comandos em sequência em arquivos de texto (.txt) permitindo o compartilhamento não só do código gerado, mas dos comandos geradores. O segundo é que se o usuário obtêm a resposta em código dos comandos de entrada, deve ser possível obter os comandos de entrada, ao entrar com o código correspondente gerado anteriormente, assim permitindo a usuários que estão com acesso apenas ao código, compreender os passos que o programa segue.

3.2.3 PACT - Programming Assistant ChaTbot

O PACT desenvolvido por Yadav, Garg e Mathur (2019) é um agente inteligente que auxilia novos programadores e aqueles não acostumados com um novo ambiente de programação. Funciona fornecendo informações e documentações necessárias para o desenvolvedor que o está utilizando, buscando suas fontes em websites técnicos, por exemplo, o stackoverflow através de expressões regulares para focar nas informações desejadas da página, e utilizaram da API Wrapeer, para coletar dados de posts do reddit focados em programação e por “Neural Machine Translation” criar e avaliar a melhor resposta a ser dada, já para trabalhos futuros mencionaram um sistema de feedback às respostas do bot.

Embora o caso atual seja para auxiliar com a programação e a proposta em idealização possui uma maior automatização, de modo que o bot programa no lugar do engenheiro de testes, ainda assim é interessante a análise desse caso, pelos trabalhos futuros, que forneceu a ideia de poder dar “like” ou “deslike” nas respostas do bot, de modo a identificar falhas de conversão comando-código, onde o bot poderá focar seus conhecimentos na procura por erros ou maneiras diferentes de construir o mesmo código, de modo a melhorar a eficiência do programa gerado.

3.2.4 Say Hello to ‘Coding Tutor’!

O estudo de caso a seguir não possui uma comparação com a proposta do trabalho ou pontos positivos e negativos entre eles, o motivo para esta menção é que este projeto em questão serviu como fundamentação metodológica para idealização da proposta deste trabalho de criação e execução de testes por método no-code, mostrado na secção anterior. Foram tomadas como inspiração a sequência de criação dos objetivos que a proposta deve atingir, a necessidade dos

engenheiros de testes, a listagem de funcionalidades que devem estar presentes, características de como as funcionalidades serão abordadas e tecnologias sugeridas para implementação do chatbot, visto que a proposta é a idealização de um chatbot, e por fim uma análise de validação da proposta finalizada com os engenheiros de testes.

Nesse caso proposto por Hobert (2019) é detalhado o desenvolvimento de um chatbot que auxilia no aprendizado de estudantes em fase inicial dos cursos de programação, permitindo um auxílio individualizado quando não há um profissional de educação disponível. Sua proposta foi formulada em etapas, começando com os objetivos do trabalho: os estudantes precisariam entender a sintaxe e semântica das linguagens de programação, como aplicar algorítimos subjacentes ao problema a ser resolvido, ou seja, a quebra do problema em pequenos problemas e a solução individual delas, e o por fim absorver o conhecimento, permitindo o aplicar em novas situações.

Com os objetivos levantados, observaram as necessidades dos usuários, ou seja, os estudantes, e perceberam que foram semelhantes aos objetivos levantados. Com essa validação, foi criada a lista de funcionalidades do chatbot que cumpra os objetivos levantados, i.e. explicar o exercício, os algorítimos envolvidos, corrigir atividades, feedback individual, entre outros de modo a realizar a atividade proposta de substituir os profissionais de educação enquanto estão indisponíveis.

Possuindo a lista de funcionalidades, foi possível idealizar as características do sistema: uma interface para interação com o bot, área de editor de código (com Ace editor que permite destacamento de sintaxe, intenção automática e quebra de linha), ambiente web de acesso a qualquer momento, guia passo a passo para solução da atividade, capacidade de responder questões abertas. Para então criar um protótipo utilizando as tecnologias HTML5, CCS3, javascript e framework Bootstrap 4 no front-end clicável e responsivo, e por RESTful web para estender o front-end com javascript chamando as funções necessárias do backend via AJAX.

Com o protótipo desenvolvido foi detalhado o funcionamento do chatbot, assim como suas ramificações, onde, por exemplo, após explicar o exercício, está disponível uma opção de pedir ajuda podendo levar a uma explicação mais detalhada sobre o problema abordado. Em seguida o protótipo foi levado a teste, onde tanto os estudantes quanto os professores testaram dos seus pontos de vistas, adquirido então as informações de sugestões de melhorias, pontos falhos para então serem analisados e implementados em uma nova versão.

4.4 Chat-Bot de criação e execução de testes

Como mencionado anteriormente, a proposta será um sistema no-code, em que não há a necessidade de criar ou editar código no modelo convencional, por linguagens de programação, isso não impede que o engenheiro de teste interaja com código, visto que haverá um local para visualização e compartilhamento do código gerado pelo chatbot proposto na ideia inicial.

Primeiramente é necessário listar os objetivos que se espera atingir com o trabalho:

O1. Melhorar a automação

O2. Permitir a criação de testes automatizados, sem necessitar conhecimentos em linguagens de programação
O3. Permitir a execução dos testes automatizados
O4. Melhorar a qualidade do trabalho dos engenheiros de testes

Os objetivos acima se referem a diminuição de recursos utilizados na fase de testes, acelerando o processo de criação de testes, aumentando a abrangência da automatização, de modo que os engenheiros de testes sem conhecimentos em linguagens de programação criem e executem seus próprios testes automatizados ou testes compartilhados, diminuindo a quantidade de atividades cansativas e repetitivas.

Com os objetivos definidos é preciso listar as necessidades dos usuários do sistema, que precisam ser atendidas:

N1. Método de uso fácil
N.2 Compartilhamento de testes
N.3 Converter comandos em código
N.4 Converter código em comandos

Tais necessidades são atribuídas ao fator que qualquer um sem conhecimento em programação deve conseguir utilizar o sistema, logo precisaria ser intuitivo e simples além de possuir as características solicitadas na pesquisa com os engenheiros alvos.

Em seguida precisamos das funcionalidades que cumprirão as necessidades dos usuários e atingirão os objetivos, serão eles:

F.1 Tutorial simples
F.2 Importação e Exportação de comando e código
F.3 Abstração de comandos
F.4 Seleção de ambiente de execução
F.5 Armazenamento de comandos
F.6 Sistema de validação de execução
F.7 Julgar o teste

Essas funcionalidades servirão para apresentar o chatbot para os usuários, tornando mais simples (N1) e melhorando a automatização (O1). Assim como fornecer um modo de compartilhamento de testes (N2), por chat ou arquivo de texto para comunicação entre os engenheiros de testes, facilitando o trabalho (O2). Além disso, a proposta terá a utilização de processamento de linguagem natural para a criação e execução dos testes (O3 e O4) convertendo escrita humana em código de programa assim como revertendo o processo (N3 e N4). Para criação dos testes é viável para facilitar o uso (N1) e melhorar a automação (O1) que os comandos de testes possam ser armazenados são só como compartilhamento (N2), mas para utilização em outros casos de testes. Tais execuções devem possuir um sistema de validação que busque a melhora

continua da automação (O1) de modo a relatar problemas encontrados durante a execução. Já para a funcionalidade F.4, ela foi idealizada inicialmente para permitir a execução dos testes em ambiente web ou mobile, mas com o desenvolvimento da proposta, ela foi removida, por estar fora do escopo principal da ideia. Por fim na etapa final dos testes, deverá existir o método onde a proposta aprova ou rejeita o teste executado automaticamente, de modo a facilitar o trabalho do engenheiro de testes (O4).

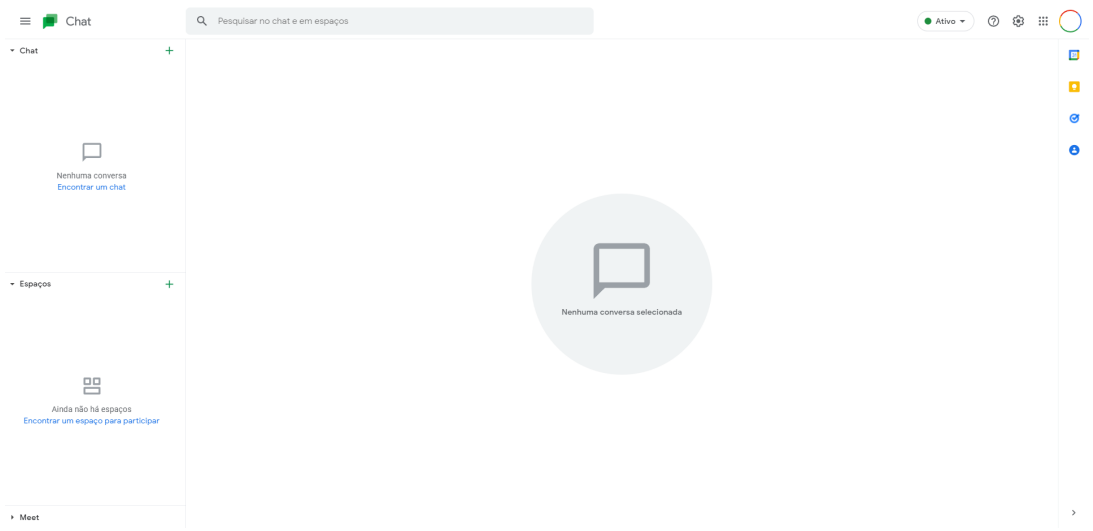


Figura 4.2 Interface do Google Chat sem a proposta do trabalho

Com as funcionalidades definidas é possível esquematizar um design inicial do chat-bot, primeiramente analisando o design atual do Google chat mostrado na figura 3.1. Neste processo, foi identificada que o google chat possui uma funcionalidade que permite procurar e adicionar chat-bots ao ambiente de trabalho o que já facilita a idealização. Além disso, o sistema deve utilizar do próprio google chat para armazenamento das mensagens, comandos e códigos enviados ou recebidos para o chat-bot, de modo a evitar perda de informações. O sistema não deverá consumir muito armazenamento, visto que pelas pesquisas foi detectado a preferência por um chatbot como ferramenta, dessa forma todo processamento de humor, emoção, análise de sentimentos e contexto não será necessário para a proposta, focando o processamento em na análise de linguagem natural e na busca de informações nos bancos de dados do chat-bot. Deverá haver, ao selecionar o chatbot, uma divisão de tela em que seja possível visualizar o campo de preenchimento de comandos e outro de código, assim como a possibilidade de importar e exportar comandos, como arquivos de texto ou armazenamento para caso de teste, entre outras características que englobe as funcionalidades listadas anteriormente.

De modo que a proposta idealizada quando integrada ao sistema do google chat possua a seguinte interface:

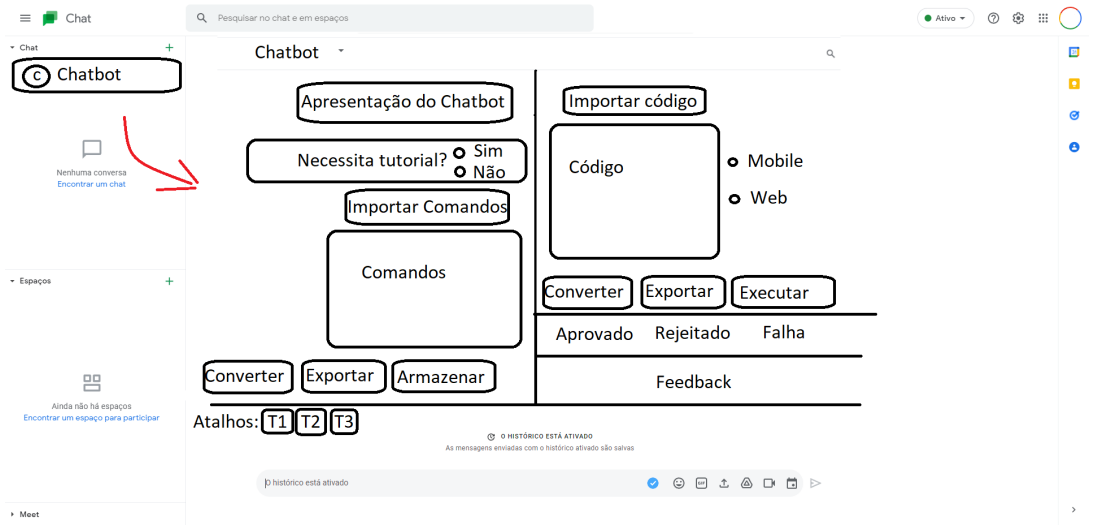


Figura 4.3 Interface do Google Chat com a proposta do trabalho.

A interface é explicada com mais detalhes pelo exemplo de caso de teste abaixo e pela máquina de estados a seguir:

Inicia o Caso de teste

1. O usuário clica no chatbot.
2. O chat-bot se apresenta, informa ser um sistema de criação e execução de testes no-code
3. O usuário pressiona “Sim” na opção “Necessita de tutorial?”
4. O chat-bot informa a funcionalidade das áreas e dos botões
5. O usuário escolhe não importar um arquivo com os comandos do teste
6. O usuário escreve o seguinte caso de teste

Abre o menu
Pressiona e segura o Wifi
Seleciona a opção Rede
Preenche a senha com 123

7. O usuário pressiona exportar que baixa os comandos em um arquivo de texto .txt
8. O usuário pressiona a primeira e segunda linha e então pressiona armazenar, criando o atalho T1

9. O usuário pressiona converter, onde é possível visualizar no lado direito da tela o código correspondente aos comandos gerado pela inteligência artificial utilizando de informações in-

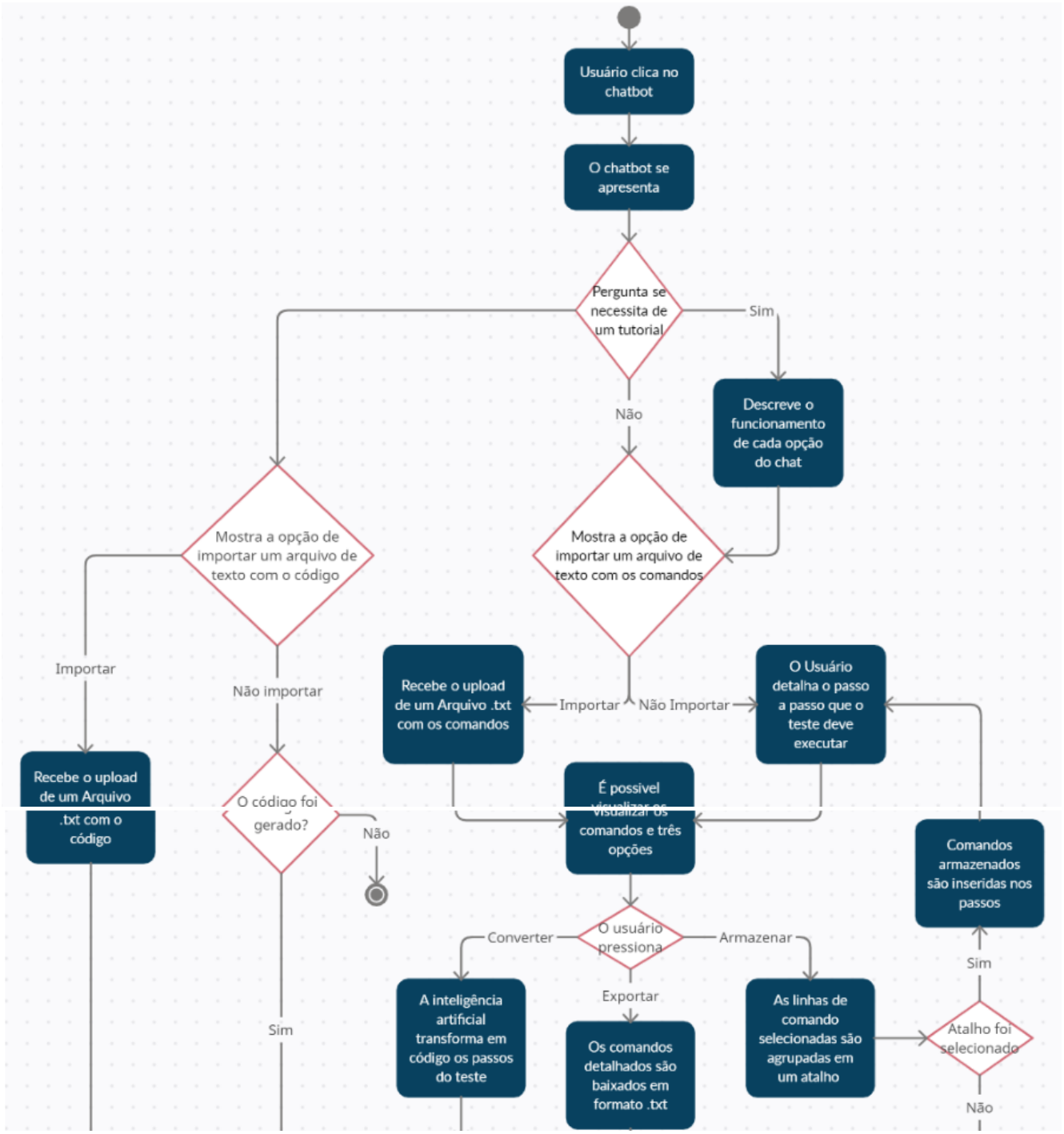
T1
Abre o menu
Pressiona e segura o Wifi

ternas, assim como adquiridas online.
10. O usuário pressiona Exportar, baixando o arquivo de texto .txt da versão em código dos comandos inseridos
11. O usuário pressiona “Executar”, com o dispositivo móvel conectado ao computador onde está funcionando a extensão com o chat-bot. Ele consegue observar os passos inseridos pelo comando sendo executados no celular.
12. O teste retorna falha, e o usuário percebe que a execução não foi como desejada.
13. O usuário informa pelo feedback o seguinte:

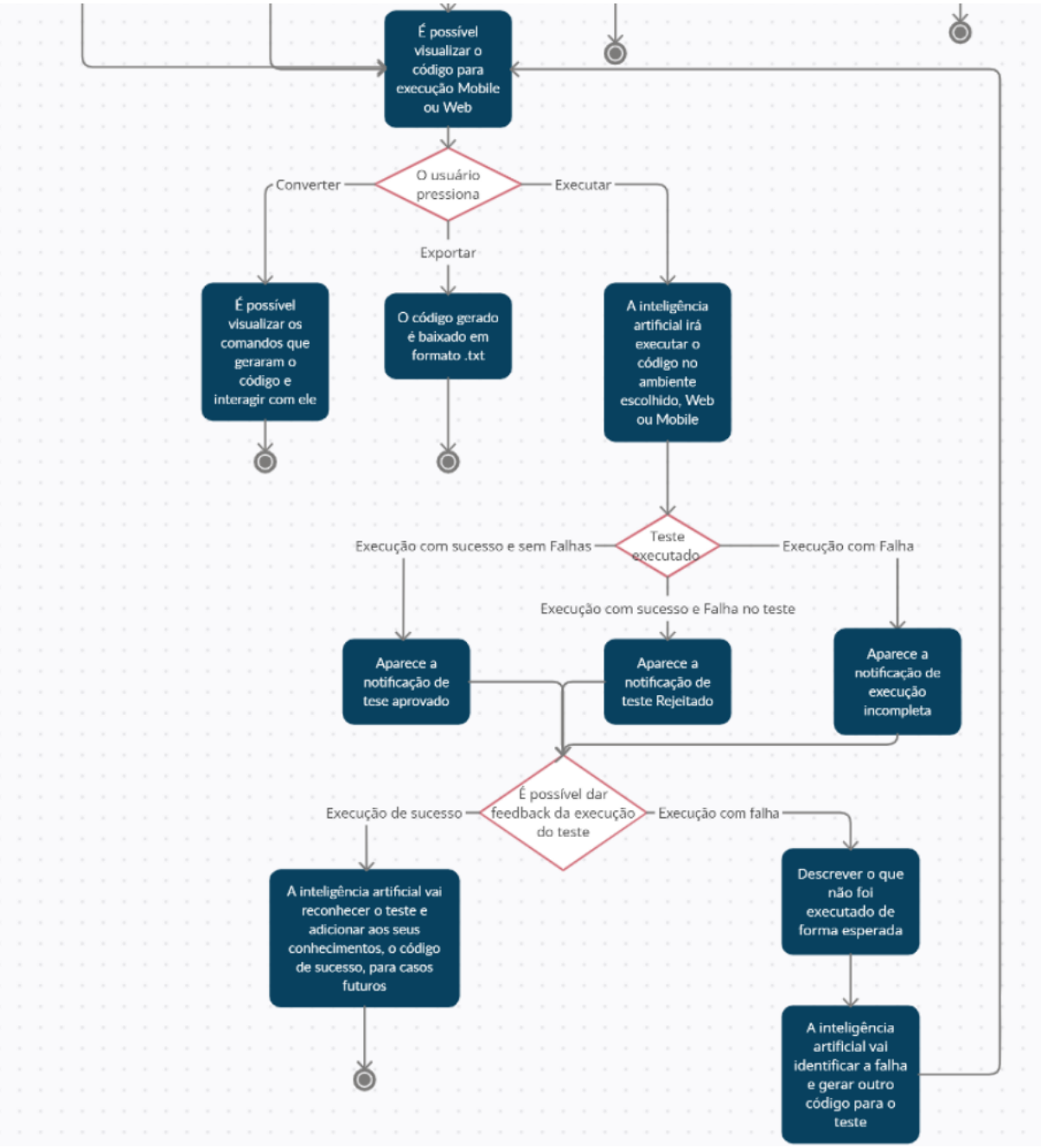
O menu foi aberto
A opção Wifi foi ligada, mas não aberta

15. A inteligência artificial identifica o problema no código gerado anteriormente, onde estava “Press Wifi” é agora alterado para “Press and Hold Wifi”
16. O usuário executa o novo código gerado, que funciona como esperado
17.O usuário dá o feedback “Teste com sucesso”
18. A inteligência artificial agora está mais preparada para caso receba o comando“Pressiona e segura”
Finaliza o Caso de teste

Nessa seção foi idealizada a proposta de um chat-bot que permite a engenheiros de testes criar e executar testes sem a necessidade de possuir conhecimentos na metodologia tradicional de programação. Foram sintetizadas as pesquisas com propostas semelhantes e analisadas as necessidades dos profissionais, de modo a refinar a ideia inicial, em uma versão satisfatória. Foi demonstrado também o funcionamento e método de concepção, para auxiliar a quem interessar



implementar a proposta.



4.5 Validação da proposta com os engenheiros de testes

Está seção abordará os resultados da pesquisa de validação da proposta para criação da sua versão final, composta pela absorção de ideias de trabalhos relacionados e da pesquisa com os engenheiros. Essa validação diferente da primeira, com objetivo de exploração de ideias, foi realizada apenas com engenheiros de teste, visto que com os resultados da primeira pesquisa o público alvo foi melhor delimitado. Visando analisar e validar a proposta, a pesquisa atual

foi realizada quantitativamente, no formato onde os engenheiros de testes, avaliam o valor agregado das características da proposta, variando de "Valor inexistente, pouco valor, neutro, valor satisfatório e muito valor" agregado. Com as informações dessa segunda pesquisa é esperado identificar pontos de melhoria ou alterações na proposta, para atingir melhor o ponto de dor dos profissionais pesquisados.

O primeiro ponto a ser validado é a integração com o google chat, fator identificado pela pesquisa de ferramentas mais utilizadas pelos profissionais e pela vontade de utilização de um mesmo sistema para comunicação e para o chat-bot proposto. Foi levantado que a ferramenta utilizada seria o google chat, e que a maioria preferia integra-lo ao invés de utilizar separadamente.

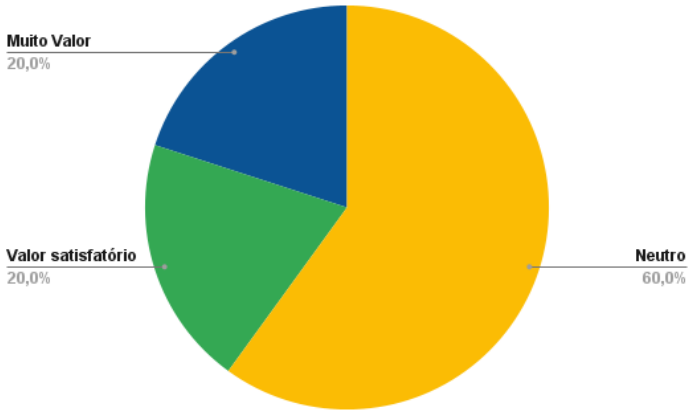


Figura 4.4 Proporção de valor agregado para integração do chat-bot no google chat.

Da validação foi obtida a informação que mesmo somando o quantitativo de valor satisfatório e muito valor, ainda assim o sentimento neutro para o valor agregado por essa característica se sobressai, logo considerando *neutro* como o grupo sem reclamações, essa funcionalidade possui mais valor positivo que negativo, o que é um bom sinal, mas que ao ser comparada com as outras características analisadas, não apresenta uma proporção tão favorável, então caso haja a implementação da proposta desse trabalho, o ponto mais suscetível a alteração é justamente utilizar o chat-bot integrado a outra plataforma, ou em um programa independente.

Seja implementada no google chat ou em um programa dedicado, foi validada pelo gráfico abaixo que um tutorial breve e simples, que mencione como utilizar cada funcionalidade da proposta é de fato uma característica importante para os engenheiros de testes, demonstrando que mesmo integrados a área de testes, ao utilizar um novo ambiente de criação e execução de testes, uma explicação do ambiente agrega valor.

Em seguida foi validada a possibilidade de importar e exportar arquivos de textos com os comandos utilizados para gerar os testes, ou seja, os passos escritos convertidos pela inteligência artificial para o código de teste, essa funcionalidade foi abstraída de uma sugestão durante a primeira pesquisa, onde foi identificado o desejo por compartilhamento de testes entre os engenheiros de testes. Também foi analisada a característica semelhante e inversa, em que acontece a conversão do código gerado para os comandos geradores do código correspondente.

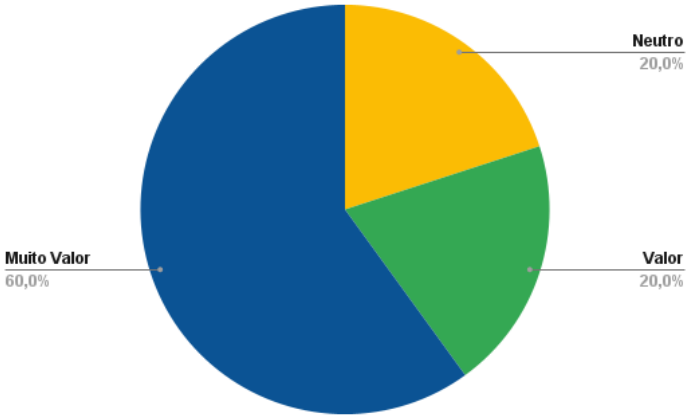


Figura 4.5 Proporção de valor agregado para a presença de um tutorial.

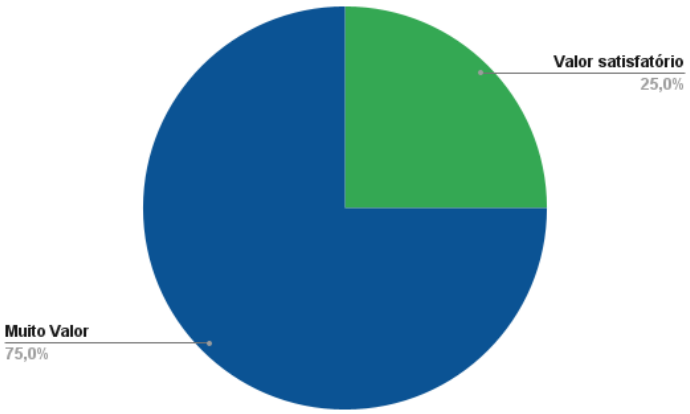


Figura 4.6 Proporção de valor agregado para a funcionalidade de importar e exportar comandos.

É então observada que, embora pequena, uma maior relevância para compartilhamento de comando sobre compartilhamento de código, isso somada a característica fundamental da proposta, que converte os comandos de entrada, dos engenheiros de testes, para os códigos que serão executados é possível identificar que o usuário só precisaria compartilhar os comandos do teste, visto que quem receber os comandos compartilhados só precisariam converter para código, logo a funcionalidade de importar e exportar código no que se refere a compartilhamento de informações, poderia ser removida, caso desejada, mas no que se refere a possibilidade de extração de código para executar em outro ambiente de testes, geraria uma deficiência para proposta.
Analisando todo o cenário, e visto que ambas as características possuem alta proporção de valor para os usuários, o correto deve ser manter ambas as funcionalidades, mesmo que em algum momento sejam redundantes.

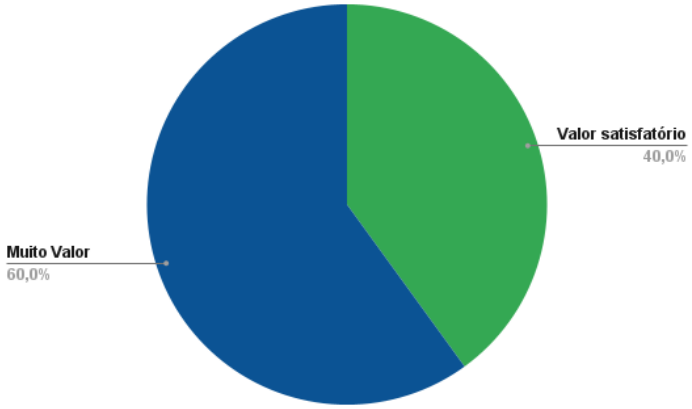


Figura 4.7 Proporção de valor agregado para a funcionalidade de importar e exportar código. Assim como para funcionalidade de converter código nos comandos que os geraram.

Também na área de comando e código, foi validado o sentimento favorável dos engenheiros de testes para a funcionalidade que possibilita converter o código gerado nos comandos que os gerou, permitindo realizar alterações nos testes, de modo mais fácil, ou seja, mudando passos de execução ao invés de linhas de código, o nível de valor para essa funcionalidade possui a mesma proporção que a funcionalidade de importação e exportação de código.

Considerando que o público alvo da pesquisa são engenheiros de testes, e que a proposta busca não necessitar de conhecimentos em programação para criar testes, é importante manter tal função, visto que modificações no teste, pelos engenheiros serão mais facilmente realizadas alterando os passos do teste, se comparado a editar o código a ser executado.

Sobre a criação dos testes, a possibilidade de armazenar porções de comandos para serem reutilizados na criação de outros testes, também possui uma validação positiva, essa foi uma boa reação visto que não foi identificada pela pesquisa com os engenheiros, mas sim absorvida no estudo dos trabalhos relacionados, o gráfico que demonstra a proporção de oitenta por cento favorável é demonstrado a seguir:

O gráfico acima, que demonstra a proporção de valor agregado para a funcionalidade de armazenar comandos, é de forma não relacionada, idêntica à funcionalidade onde o chat-bot realiza o julgamento automático dos testes, ou seja, se aprova ou reprova a execução. Considerando que essa funcionalidade é um ponto extremamente sensível, visto que os testes são criados e executados justamente para se ter conhecimento do seu nível de aceitação, e que houveram uma porcentagem da pesquisa que não se sentiu no mínimo satisfatório com o julgamento automático pela inteligência artificial, é recomendado que a implementação dessa característica seja ou não implementada a depender do nível de acurácia da inteligência artificial disponível, disponibilizando a opção em que o engenheiro de testes possa validar manualmente o resultado da execução do teste.

Por fim, a melhor validação da proposta pelos engenheiros de testes, foi sobre a funcionalidade de feedback sobre a execução do teste, de modo que entradas positivas indicam para a inteligência artificial, que deve guardar a última execução como aprendizado para execuções

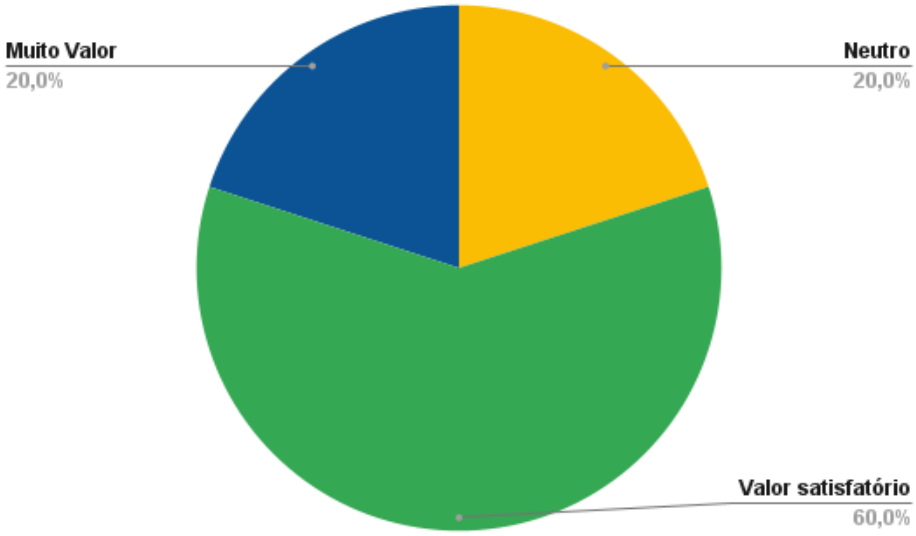


Figura 4.8 Proporção de valor agregado para a funcionalidade de armazenar comandos para criação de outros casos de teste.

futuras, enquanto feedbacks negativos, fazem com que a inteligência artificial procure onde cometeu uma falha na escrita do código, e o corrija o reescrevendo em uma versão melhorada. Considerando que está é a melhor validação da pesquisa, não devem ser realizadas alterações nessa funcionalidade.

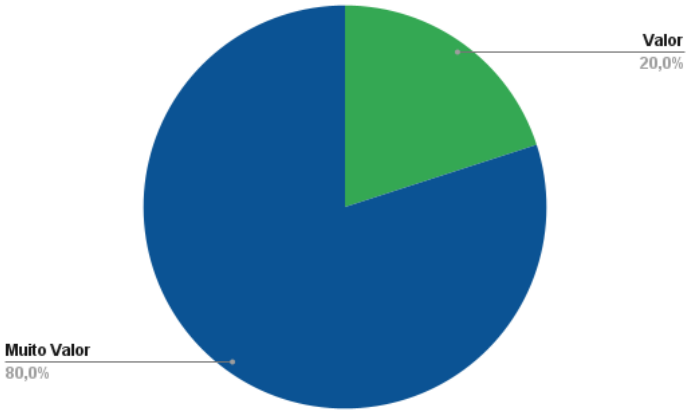


Figura 4.9 Proporção de valor agregado para a funcionalidade feedback de execução de testes.

No fim da pesquisa de validação, foram analisadas os níveis de relevância das funcionalidades da proposta e com isso sintetizadas procedimentos a serem realizadas sobre elas, que

podem ser resumidas a seguir:

1. A integração com o google chat, não possui uma proporção negativa, mas a menor se comparada com as outras, com isso fica a disposição de quem irá implementar a proposta desse trabalho, integrar ou não a plataforma da google, podendo criar seu próprio programa independente.

2. É recomendado a permanência da funcionalidade que adiciona um tutorial que demonstra as funcionalidades do sistema.

3. A funcionalidade de importação e exportação de comandos e também de códigos, possuem proporções em que todos os pesquisados sentiram um valor agregado satisfatório, ou seja, a característica foi julgada valida e deverá ser mantida

4. Para a capacidade de conversão de código novamente em comandos, assim como a funcionalidade de conversão de comandos em código possuem níveis totais de valor agregado positivos, e expandem a funcionalidade do sistema de criação, execução e julgamento de código agora para edição, de modo que os engenheiros podem alterar os comandos, de forma simples, se comparada com editação de código de programação. Por permitir essa atividade extra, a característica deve ser mantida.

5. A funcionalidade de armazenamento de códigos para escrita de casos de testes, não possui uma proporção totalmente a favor da característica, e considerando que a funcionalidade não é intrínseca do funcionamento pleno da proposta, se caso desejado durante a implementação, a funcionalidade pode ser removida, estando ciente que a falta dela vai deixar de agregar valor para parte dos engenheiros de testes.

6. Para validar se a inteligência artificial deve julgar automaticamente os testes executados, foi obtido o mesmo resultado que a funcionalidade de armazenamento, por outro lado, é uma funcionalidade extremamente importante e que por não possuir uma proporção mais favorável a essa característica, a implementação do sistema deve julgar por si próprio o nível da inteligência artificial sendo implementada e as consequências que erros de julgamento dos testes acarretariam, para decidir se a implementação dessa característica deve ser realizada ou se o julgamento ficará de responsabilidade dos engenheiros de testes.

7. Para a última funcionalidade, a possibilidade de enviar feedback da execução para a inteligência artificial, foi obtida a melhor resposta positiva por parte dos engenheiros de testes pesquisados, logo é uma funcionalidade que deve ser mantida durante implementações.

Essas foram as possíveis alternativas validadas para a proposta do trabalho, em que se recomenda a permanência ou possíveis alterações nas características levantadas para o trabalho, identificadas por pesquisas com trabalhos relacionados e pesquisas com engenheiros de testes. A seguir será demonstrada a versão finalizada da idealização.

4.6 Versão final

Com as funcionalidades da proposta validadas e sintetizadas, e buscando facilitar o processo de implementação do sistema para os desenvolvedores, está seção trabalhará a versão final da proposta, que aborda todas as pesquisas realizadas e visa o máximo de satisfação para os engenheiros de testes. Considerando a permanência do chat-bot integrado ao google chat, um novo design de interface para a proposta será necessário, de modo a retratar melhor o funcionamento da versão final, que pode ser observada abaixo:

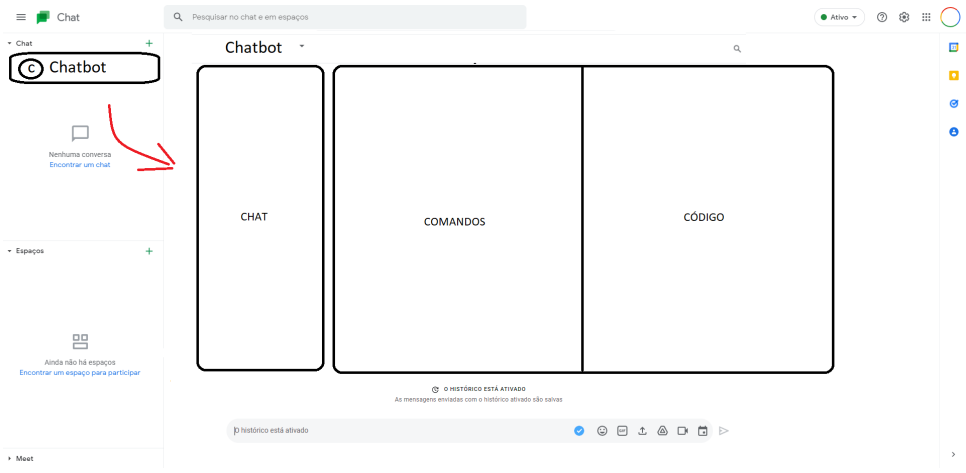


Figura 4.10 Interface do Google Chat com a proposta final do trabalho

Como foi identificada que a funcionalidade de tutorial é relevante para os engenheiros de testes pesquisados, ela não será mais realizada opcionalmente, agora após apresentação do sistema, será mostrado o conjunto de comandos que executarão as funcionalidades do sistema, anteriormente executadas por botões, essa lista de comandos é a seguinte:

Importar: O sistema mostra na interface de comandos e testes, as entradas obtidas pelo último arquivo de texto enviado ao chat-bot, pela funcionalidade nativa do google chat, e sua respectiva conversão para a outra versão, ou seja, ao ser enviado um arquivo de texto com o código do teste, aparece o código e sua versão em comandos nas interfaces correspondentes.

Exportar: O sistema faz o download automático de um arquivo de texto com os comandos e outro com os códigos presentes nas interfaces correspondentes.

Escrever: Nesse momento o chatbot entende que as próximas entradas no chat serão comandos de teste, até que seja recebido o comando Finalizar.

Editar: O chat-bot retorna os comandos da interface para o chat, que podem ser reescritos, até que seja recebido o comando Finalizar.

Executar: O chat-bot irá executar o código de teste gerado e enviará uma mensagem no chat informando o status do resultado

Revisar: O chat-bot entenderá que as próximas entradas no chat será a avaliação do último teste executado, até que seja recebido o comando Finalizar.

Finalizar: O chat-bot retorna ao sistema normal de chat e se houver alteração de comandos. realiza a conversão automática para versão de código do teste.

Armazenar A1 x-y: O chat-bot irá guardar os comandos do último teste presente no chat, da linha x até a linha y em um atalho nomeado “A1”

Atalho: O chat-bot envia ao chat a lista de atalhos e os comandos internos a cada atalho.

Excluir atalho A1: O chat-bot irá deixar de armazenar o atalho com nome “A1”

Já para a característica que possibilitava a escolha de versão de código mobile ou web, essa foi removida da versão final por ser uma funcionalidade deslocada da proposta do trabalho, deixando a decisão de criar um conversor entre versões para aqueles que implementarem o sistema e desejar utilizar tal característica.
A nova abordagem facilita muito o desenvolvimento do sistema e não perde o objetivo do trabalho, de permitir criação e execução de testes sem conhecimentos em linguagens de programação, os acréscimos em palavras chaves para utilização do sistema, não é visto como uma categoria de programação, pois são correspondentes a funcionalidades comuns, por exemplo, a conversão do clique no botão "executar"para digitar a palavra "Executar"no chat, não interfere na criação dos testes, visto que o engenheiro de testes ainda vai poder escrever o passo a passo do teste a ser executado de maneira livre, sem utilizações de palavras chaves, um exemplo de caso de teste para nova interface e seu conjunto de funcionalidades são demonstradas abaixo, e logo a seguir a nova máquina de estados para a versão final da proposta.

Inicia o caso de teste
1. O usuário clica no chat-bot.
2. O usuário consegue visualizar no chat a mensagem do sistema se apresentando e mostrando o conjunto de comandos disponíveis para serem utilizados e o que cada comando realiza.
3. O usuário envia um arquivo de texto pelo upload do google chat contendo os seguintes comandos para o teste.

Abre o menu
Pressiona e segura o Wifi
Seleciona a opção Rede
Preenche a senha com 123

4. O usuário digita *Importar* no chat.
5. Os comandos e sua versão em código aparecem nas respectivas interfaces.
6. O usuário digita *Editar*
7. Os comandos do teste aparecem no google chat.
8. O usuário pressiona a funcionalidade nativa do google chat de reescrever mensagens, e edita os comandos para:

Abre o menu
Pressiona e segura o Wifi
Seleciona a opção Rede
Preenche a senha com 5555

9. O Usuário digita *Finalizar* e pode visualizar os novos comandos e sua versão em código nas interfaces correspondentes.
10. O usuário digita *Armazenar PrimeiroAtalho 3-4*, e o chat-bot armazena no atalho "PrimeiroAtalho" os comandos correspondentes a linha 3 e 4 do último teste no chat.
11. O usuário digita *Atalhos* e o chat-bot envia uma mensagem contendo a lista de atalhos, onde o usuário observa o "PrimeiroAtalho" e em seguida as linhas de comando armazenadas.

Seleciona a opção Rede
Preenche a senha com 5555

12. O usuário digita *Excluir PrimeiroAtalho*
13. O usuário digita *Atalhos* e o chat-bot envia uma mensagem contendo a lista de atalhos, atualmente vazia.
14. O usuário digita *Executar*, onde o chat-bot envia uma mensagem ao chat informando que o teste está em execução, bloqueando o chat até o fim da execução, e por fim enviando o status do resultado do teste também no chat.
15. O usuário consegue visualizar a mensagem de "Teste executado com sucesso".
16. O usuário digita *Revisar*, e em seguida entra com o feedback do último teste executado, o chat-bot armazena a informação, melhorando a inteligência artificial para casos semelhantes.
17. O usuário digita *Exportar*, e o sistema faz download automático de dois arquivos de texto, um contendo os comandos e outro o código do último teste executado.
Finaliza o caso de teste

Esse capitulo abordou o desenvolvimento da proposta do trabalho em suas etapas de construção e no próximo capítulo será abordado a recomendação para uma possível implementação da proposta idealizada.

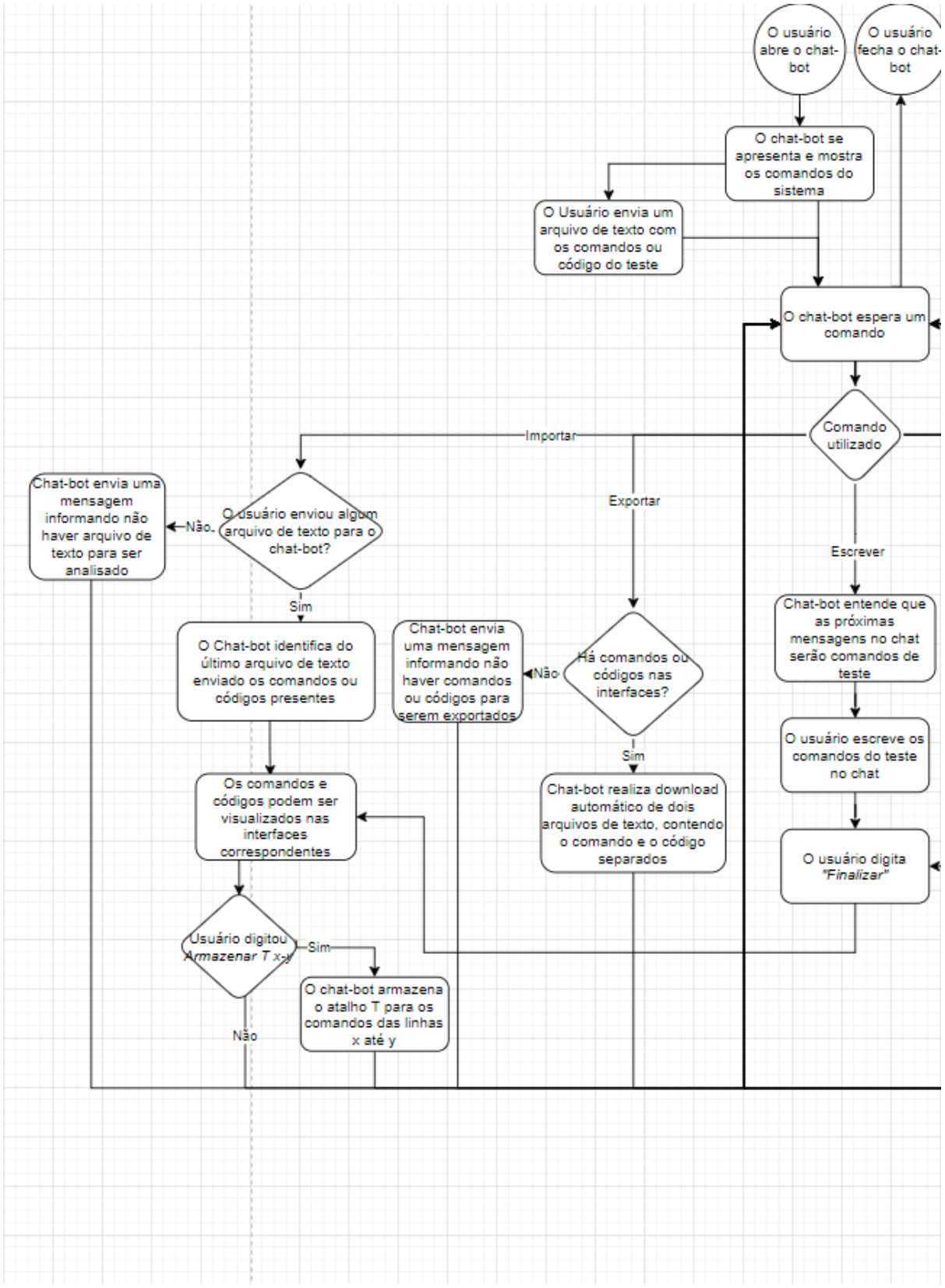


Figura 4.11 Lado esquerdo da máquina de estados da versão final da proposta

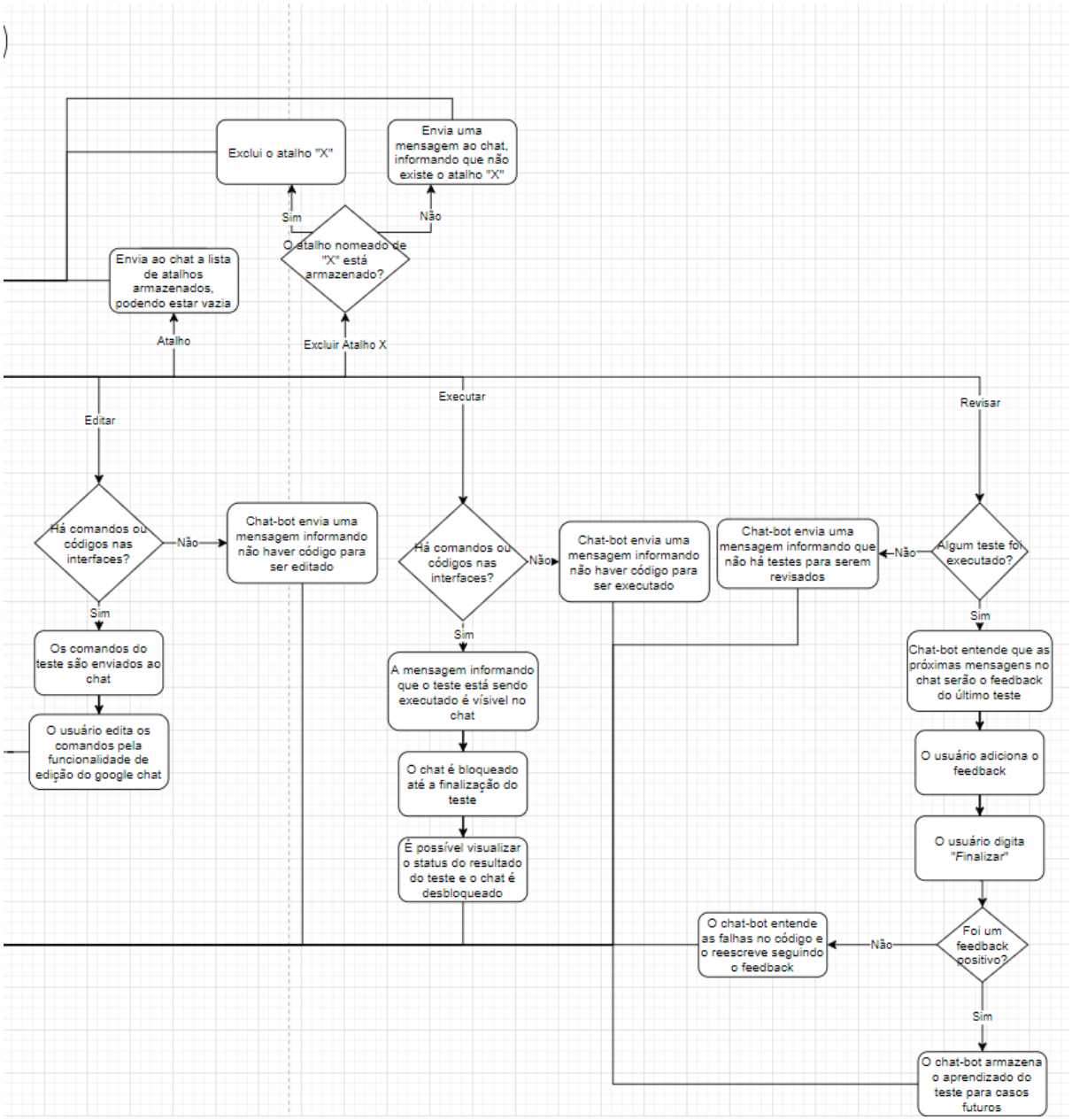


Figura 4.12 Lado direito da máquina de estados da versão final da propostao

CAPÍTULO 5

Implementação

A proposta desenvolvida nesse trabalho está livre para ser implementada da forma preferida ao desenvolvedor, para auxiliar, este capitulo abordará um possível processo de implementação, em que são recomendados um método de implementação simples e princípios de design de criação do chat-bot que visa melhorar a usabilidade para o usuário final. Foram utilizadas como referências para esse capitulo a documentação do Google Chat for Developers .

5.1 Recomendações de Implementação do Chatbot

5.1.1 Google Chat e Apps Script

O google chat possui uma vasta capacidade de compatibilidade com bots. Porém, é recomendado a utilização do Google Apps Script ou do Google App Engine. O segundo possui mais baixo nível, necessitando da análise do corpo das mensagens transmitidas entre o google chat e o bot implementado, para determinar o tipo da mensagem e seu conteúdo, para só então determinar qual a categoria de mensagem se encaixa e então tomar a ação apropriada. Por outro lado, implementar um bot no Google Apps Script permite uma integração facilitada ao google chat, com menor número de linhas de desenvolvimento, a possibilidade de portar para cloud ou node.js, além de permitir chamadas a API do Google da mesma forma realizadas as chamadas para as funções internas do código, integrando mais facilmente as funcionalidades do Google slides, documents, calendar, entre outros serviços Google. Além disso, não é necessário se preocupar com a execução, o gerenciamento do servidor, manutenção ou custo operacional e configuração de ambiente, já que o sistema hospeda a aplicação desenvolvida, essas são algumas das vantagens que baseiam a recomendação de utilização do Google Apps Script para o desenvolvimento da proposta desse trabalho.

Embora a API do google providencie maneiras de integração entre o bot e o google chat, ela não obriga uma metodologia específica de implementação da lógica do bot, podendo-se utilizar desde a lógica baseada em comandos a qualquer processamento de linguagem e serviços de inteligência artificiais desejados. De forma que a relação entre as partes funciona da seguinte forma: O google chat, onde o bot está integrado recebe a mensagem do usuário para então contactar o aplicativo do desenvolvedor, isso é, o local onde se processa todo o conteúdo da proposta do trabalho: a conversão de comandos em códigos, e execução de testes. Para então o aplicativo responder ao google chat, que irá renderizar a resposta de volta ao usuário no formato desejado, com isso é possível perceber que a aplicação desenvolvida funciona de forma similar a uma API, recebendo os pedidos de execução do google chat.

A execução do google chat funciona por interações em formatos de eventos, as mensagens

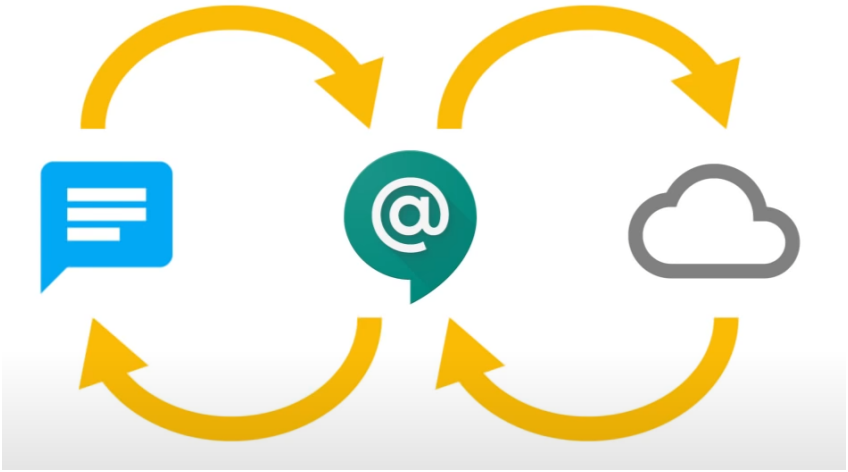


Figura 5.1 Google chat recebe as mensagens dos usuários, direciona ao bot desenvolvido, e quando recebe a resposta converte para ser lida pelo usuário

que partem para o bot, constituem de um objeto no formato JSON contendo as informações do usuário, o conteúdo da mensagem e diferentemente do Google Engine, para o Google Apps scripts também é enviado o tipo da mensagem, que facilita a implementação do código de controle para cada evento. Já o caminho inverso, onde os objetos JSON são enviados do bot para o google chat são convertidas no Google Apps Scripts em mensagens de textos comuns, ou cartões interativos que podem conter imagens e botões. Para lidar com esses eventos gerados pelo google chat, o Apps Script implementa quatro funções, que gerenciam cada caso e devem estar implementados com o mesmos nomes listados abaixo:

Evento *ADDED TO SPACE* ocorre quando o bot é adicionado a uma sala de chat, é gerenciado pela função onAddToSpace(event).

Evento *REMOVED FROM SPACE* ocorre quando um bot é removido de uma sala de chat, e consequentemente não envia resposta de volta ao google chat. No Apps scripts é controlado pela função onRemoveFromSpace(event).

Evento *MESSAGE* engloba todas as mensagens enviadas diretamente ao bot ou mencionadas a ele pelo comando @nomedobot. É definida pela função onMessage(event) no Apps script e irá gerenciar qual mensagem deverá ser retornada ao google chat.

Evento *CARD CLICKED* ocorre quando o usuário clica em um botão em um cartão interativo, é apresentada pela função onCardClick(event) que deve executar as ações correspondentes do botão clicado.

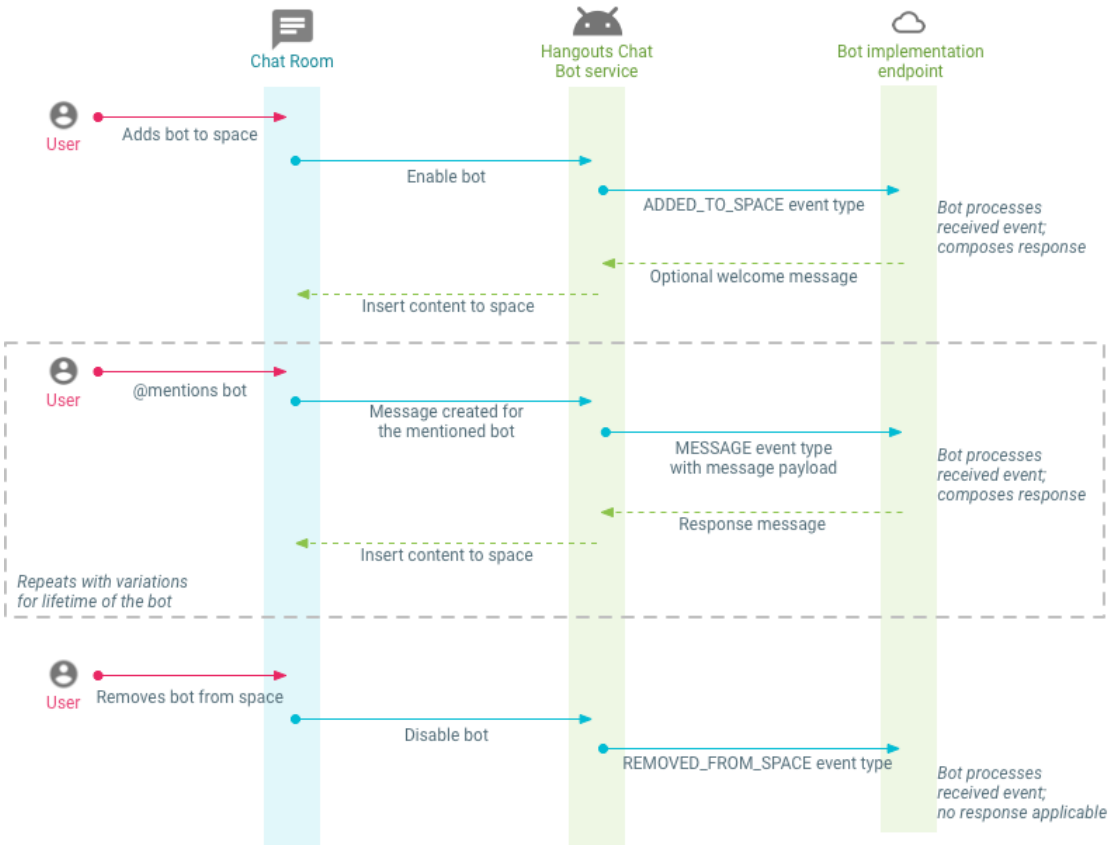


Figura 5.2 Fluxo das ações e eventos entre o usuário, o google chat e aplicação contendo a lógica do bot.

5.1.2 Implementação e arquitetura

Para implementar o bot alguns fatores devem estar presentes na lógica interna, como um analisador de comando, principalmente para bots baseados em regras, que extraem os comandos e seus parâmetros da mensagem para mapear cada comando para sua função correspondente. Outro fator é o processamento de linguagem natural, que irá determinar o que o usuário está pedindo ao bot, o google chat é compatível com diversas implementações de NLP, mas é recomendado usar o Dialogflow, por permitir a criação de agentes inteligentes por utilização de um modelo de implementação que interliga intenções com ações correspondentes. Já para instanciar o código interno do bot, com o google chat permitindo a interação entre as partes, é recomendado pelo próprio Google a utilização do protocolo de comunicação HTTP, já que o google chat é configurado para integrar com serviços remotos via HTTP.

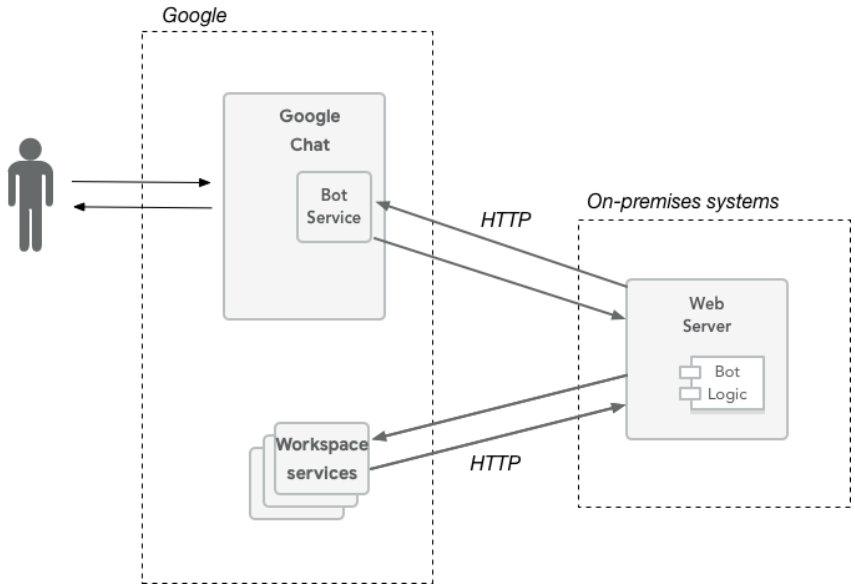


Figura 5.3 Fluxo de comunicação entre o usuário, o google chat e a aplicação, conectados pelo protocolo HTTP.

Para realizar uma implementação da lógica do bot com sucesso é importante saber a arquitetura desejada, considerando diversos fatores como o público alvo, que neste caso de trabalho é o grupo de engenheiros de testes, quais recursos o bot deve acessar, e se precisa de outros serviços do google ou outra API. Assim como considerar quais dados do usuário, vão ser precisos a solicitação de autorização de uso, visto que o google Apps script gerencia o sistema de autorização, será necessário ao usuário autorizar a utilização dos recursos apenas uma vez, como, por exemplo, no primeiro upload de um arquivo do diretório local, o google chat vai solicitar o acesso aos arquivos do computador. Além disso, é importante ter em mente qual o padrão de resposta irá ser implementado, podem ser mensagens síncronas, com uma mensagem e uma resposta, ou assíncrona, com uma mensagem recebendo uma ou mais respostas por parte do bot, que não necessariamente são seguidas uma das outras, para a abordagem do trabalho é recomendado utilizar do método síncrono, em que cada ação desejada pelo usuário será executado em sequência e respondido com tempo limite de trinta segundos, como abordado na idealização, onde durante a execução do teste, o chat ficará bloqueado, evitando que aconteça sobreposição de comandos, facilitando a implementação.

Um fator facilitador para implementação com o google Apps script é que não é necessário qualquer download ou configuração local, já que pode ser acessado por um browser, requerendo apenas uma conta de trabalho do google, com acesso ao google chat. No Apps Script é necessário escolher a opção Complemento durante a implantação, com isso o projeto será alternado para plataforma de nuvem da google, para habilitar a funcionalidade anteriormente mencionada, em que o google hosteia o seu bot, retirando a necessidade de uma máquina virtual executando o sistema, nessa mesma plataforma será necessário ativar a API do google chat, que se encontra facilmente pesquisando o termo “Google Chat API” na barra de pesquisas, com

esses passos realizados, será possível logo mais visualizar o bot no google chat para ser selecionado, utilizando-se de uma conta de trabalho, e clicando na funcionalidade intrínseca do google chat: “encontrar um bot”, ao lado do chat. Outras configurações necessárias serem habilitadas na plataforma de nuvem do google é a que permite que o bot funcione com mensagens diretas, além das menções em salas de chat, e configurar a conexão com o Apps scripts, com isso resta apenas especificar as pessoas e grupos que poderão ter acesso ao bot, que no caso desse trabalho é esperado um grupo de trabalho.

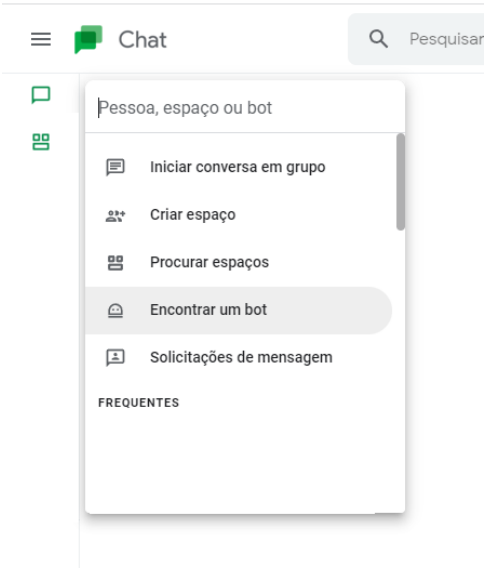


Figura 5.4 Funcionalidade intrínseca do google chat que permite a busca por bots disponiveis.

5.2 Recomendações de design para o Chatbot

Um chat-bot deve conseguir cumprir suas funções, interagindo com um ser humano de forma clara e objetiva, as possibilidades de interação no goggle chat estão disponíveis em duas opções, as mensagens diretas, meio mais natural, e os cartões interativos, que demonstram mais informações e incluem formatos mais avançados como imagens e botões. Independentemente da forma escolhida é recomendado seguir alguns padrões, o bot deve conter características como um nome, ícone e identidade do desenvolvedor e ao ser introduzido a um usuário deve ser apresentada uma mensagem de apresentação com o nome do bot e a sua utilidade. Além disso, é recomendado apresentar ao usuário um comando slash, como, por exemplo, @nomedobot help, que irá explicitar os outros comandos utilizados pelo bot, por exemplo, importar, exportar e executar, esses que não devem aparecer primariamente, para não saturar o usuário com informações excessivas.

Após a apresentação, o bot deve seguir alguns princípios básicos de conversação, para melhorar a usabilidade, utiliza-se da quebra de longos textos em mensagens menores, cada uma

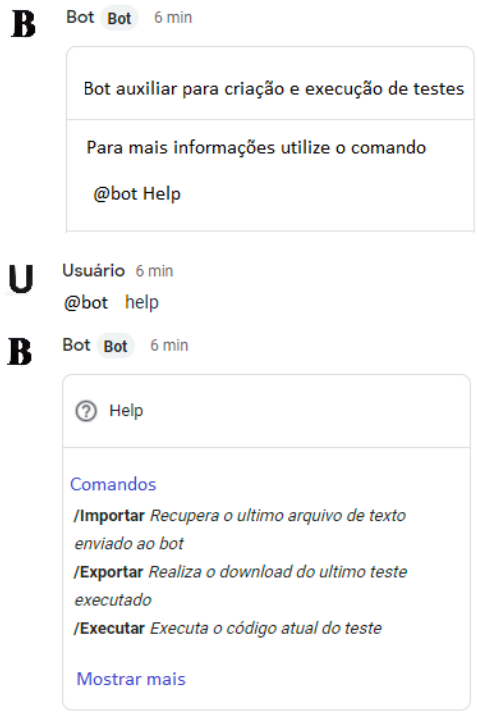


Figura 5.5 Comando Slash que auxilia o usuário a utilizar o bot.

possuindo um proposito, evitando informações desnecessárias. É importante informar ao usuário o que bot pode e não pode fazer, tornando o usuário consciente da execução do bot seja para os casos de sucesso, informando que o bot compreendeu o pedido ou de falha de funcionamento, relatando não compreensão e demonstrando outras opções de ações. Outro princípio é o fluxo, é recomendado enviar mensagens intercaladas que gerem um sentimento de continuação, como, por exemplo, “o que fazer agora”, ou quando o usuário estiver perdido enviar uma mensagem de ajuda, como “Para mais informações use o comando @bot help”, isso interligado as mensagens de confirmação deixa o usuário ciente do funcionamento do bot. Para a abordagem da proposta, que se trata de um ambiente de trabalho é recomendado que o funcionamento do bot seja de modo mais eficiente, e não coloquial de forma que as mensagens sejam transmitidas de forma direta e correta. Por outro lado, a funcionalidade de contexto no que se refere a análise de humor e empatia, não é um princípio que será abordado na proposta, visto que durante as pesquisas, os engenheiros preferiram que o bot funcionasse como ferramenta, descartando a necessidade de simular humor ou emoções, e com isso gerando uma maior efetividade e rapidez do bot, diminuindo processamento e interações desnecessárias.

Como o google chat é compatível com diversas ferramentas, o desenvolvedor poder implementar o bot do modo desejado. Para auxiliar na implementação esse capítulo demonstrou recomendações de implementação, utilizando do Google Apps Scripts e protocolos de comunicação HTTP, assim como indicações de design para melhorar a qualidade do bot desenvolvido.

CAPÍTULO 6

Conclusão

Nesse trabalho foi idealizado um chat-bot para auxiliar na criação e execução de testes de forma automatizada, buscando auxiliar no trabalho dos engenheiros de testes. Nas pesquisas realizadas com os profissionais, foi possível destacar que nem todos possuem afinidade com linguagens de programação, por isso o bot irá utilizar de inteligência artificial e processamento de linguagem natural para converter comandos comuns de ações em códigos de testes, um método no-code de escrita de programas. Auxiliar na construção e execução de testes é importante para área de testes que engloba de quarenta a cinquenta porcento dos recursos do desenvolvimento de programas (JAN e SHAH, 2016), é esperado que a proposta quando em uso atenda a demanda por automação de testes, aumentando a quantidade de casos de testes automatizados e seu nível de acerto. Para alcançar um grau satisfatório, a proposta idealizada passou por diversas fases, desde a concepção, a pesquisa com o público alvo e pesquisa com trabalhos relacionados onde foram projetadas e reunidas as características do sistema pensando nas necessidades dos engenheiros de testes, entendendo suas necessidades, e ouvindo suas requisições. Com essas informações disponíveis, uma versão mais complexa da proposta inicial foi desenvolvida e validada com os engenheiros, para então focar nas características que mais geraram valor aos seus olhos, sintetizando a versão final da proposta, com design mais simples e mais fácil implementação.

Considerando que a proposta do trabalho é a idealização de um bot e durante as pesquisas foi levantado um maior uso do google chat como ferramenta de comunicação durante o trabalho dos engenheiros de testes, também foi abordado recomendações de implementação do sistema, utilizando do Google Apps Script, plataforma recomendada pelo próprio Google Chat, por possuir simples implementação, compatibilidade com outros serviços Google e hospedagem do sistema, facilitando assim todo o processo de manutenção e custos de operação. Também foram retratados alguns princípios básicos de design, com função de auxiliar na construção de um chat-bot intuitivo e com boa usabilidade para os usuários.

Para trabalhos futuros, é relevante colocar em prática a idealização do trabalho. Implementando a lógica do bot, a sua integração com o google chat e realizando testes práticos com profissionais da área. Melhorando a qualidade do trabalho desses profissionais enquanto encontra pontos de melhorias da proposta, validando quantitativamente a melhora em questão de tempo gastos nas atividades de testes, e qualitativamente o sentimento de satisfação na realização do trabalho se comparado o método tradicional e a implantação da proposta.

Referências Bibliográficas

[ADK+20] Rachel Ai, Natasha Thussu Dhar, Disha Kohli, Lee Maina, Evelyn Manelski, Sarah Ramos, and Cory Brzycki. Programming an educational chatbot to support virtual learning. 2020.

[AM20] Eleni Adamopoulou and Lefteris Moussiades. An overview of chatbot technology. In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pages 373–383. Springer, 2020.

[AMO15] DFB de AMORIM. Softwares de sistemas e de aplicações livres: benefícios e limitações no uso dessas tecnologias nos negócios. *Revista Científica Semana Acadêmica, Fortaleza*, 1(69):1–19, 2015.

[Ber11] Paulo Cheque Bernardo. *Padrões de testes automatizados*. PhD thesis, Universidade de São Paulo, 2011.

[BK08] Paulo Cheque Bernardo and Fabio Kon. A importância dos testes automatizados. *Engenharia de Software Magazine*, 1(3):54–57, 2008.

[Dev] Google Developers. Developing bots for hangouts chat. Disponível em: <https://developers.googleblog.com/2018/05/developing-bots-for-hangouts-chat.html>, Acesso em: 13/11/2021.

[dSME+18] Henrique Neves da Silva, Guilherme Ricken Mattiello, Andre Takeshi Endo, Érica Ferreira de Souza, and Simone do Rocio Senger de Souza. Evaluating the impact of different testers on model-based testing. In *Proceedings of the III Brazilian Symposium on Systematic and Automated Software Testing*, pages 57–66, 2018.

[DTZ20] Siti Hawa Mad Daud, Noor Hasimah Ibrahim Teo, and Nurul Hidayah Mat Zain. Ejava chatbot for learning programming language: A post-pandemic alternative virtual tutor. *International Journal*, 8(7):3290–3298, 2020.

[fd] Google Chat for developers. Connect your service to google chat. Disponível em: <https://developers.google.com/chat>, Acesso em: 13/11/2021.

[GGM20] Larisa Gota, Dan Gota, and Liviu Miclea. Continuous integration in automation testing. In *2020 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, pages 1–6. IEEE, 2020.

[Gor21] Jeramy Gordon. 7 most popular jobs in 2021 - accurate. Disponível em:<https://www.accurate.com/blog/7-most-popular-jobs-in-2021/>, Acesso em: 19/10/2021 2021.

[Hob19] Sebastian Hobert. Say hello to 'coding tutor'! design and evaluation of a chatbot-based learning system supporting students to learn to program. *International Conference on Information Systems, ICIS, Munich, Germany*, 2019.

[Iza14] Leonardo Roxo Pessanha Izabel. *Testes automatizados no processo de desenvolvimento de softwares*. PhD thesis, Universidade Federal do Rio de Janeiro, 2014.

[JSJ+16] Syed Roohullah Jan, Syed Tauhid Ullah Shah, Zia Ullah Johar, Yasin Shah, and Fazlullah Khan. An innovative approach to investigate various software testing techniques and strategies. *International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET), Print ISSN*, pages 2395–1990, 2016.

[Käp21] Satu Käppi. How to make an online store without coding. 2021.

[KMS20] Faezeh Khorram, Jean-Marie Mottu, and Gerson Sunyé. Challenges & opportunities in low-code testing. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, pages 1–10, 2020.

[McC07] John McCarthy. What is artificial intelligence? 2007. `http://www-formal.stanford.edu/jmc/whatisai/whatisai.html` (Accessed October 20, 2021).

[Moe19] Myint Myint Moe. Comparative study of test-driven development (tdd), behavior-driven development (bdd) and acceptance test–driven development (atdd). *International Journal of Trend in Scientific Research and Development*, pages 231–234, 2019.

[RRMS21] Achmad Ramaditiya, Suci Rahmatia, Aris Munawar, and Octarina Nur Samijayani. Implementation chatbot whatsapp using python programming for broadcast and reply message automatically. In *2021 International Symposium on Electronics and Smart Devices (ISESD)*, pages 1–4. IEEE, 2021.

[SBC12] Abhijit A Sawant, Pranit H Bari, and PM Chawan. Software testing techniques and strategies. *International Journal of Engineering Research and Applications (IJERA)*, 2(3):980–986, 2012.

[Tem17] Daniel Temkin. Language without code: intentionally unusable, uncomputable, or conceptual programming languages. *Journal of Science and Technology of the Arts*, 9(3):83–91, Sep. 2017.

[VP18] Matthew Verleger and James Pembridge. A pilot study integrating an ai-driven chatbot in an introductory programming course. In *2018 IEEE Frontiers in Education Conference (FIE)*, pages 1–4. IEEE, 2018.

[WYF18] Chen Wei, Zhichen Yu, and Simon Fong. How to build a chatbot: chatbot framework and its capabilities. In *Proceedings of the 2018 10th International Conference on Machine Learning and Computing*, pages 369–373, 2018.

[YGM19] Aditya Ankur Yadav, Ishan Garg, and Pratistha Mathur. Pact-programming assistant chatbot. In *2019 2nd International Conference on Intelligent Communication and Computational Techniques (ICCT)*, pages 131–136. IEEE, 2019.