Desenvolvimento de um Chatbot para Agendamentos utilizando Dialogflow Integrado a API do Google Calendar

Paulo André Machado Maracci, André Flores dos Santos
Curso de Ciência da Computação
UFN - Universidade Franciscana
Santa Maria - RS
paulo.andre@ufn.edu.br, andre.flores@ufn.edu.br

***Resumo*—Este projeto teve como objetivo desenvolver um *chatbot* para auxiliar o usuário, possibilitando o agendamento de compromissos através do Dialogflow e API do Google Calendar. A proposta foi criar uma solução inovadora que atenda às demandas do mercado, utilizando inteligência artificial para aprimorar a eficiência e a personalização. Foi utilizada a metodologia XP (*Extreme Programming*) para garantir flexibilidade e melhoria contínua. O *framework* Flutter e a linguagem de programação Dart foram usados para o desenvolvimento do sistema. O Dialogflow para incorporar inteligência artificial e API do Google Calendar para gerenciar eventos e tarefas. O projeto resultou no desenvolvimento de um *chatbot* funcional, capaz de realizar o agendamento de eventos no Google Calendar, por meio da integração com a API do Google Calendar e do treinamento de um agente e suas *intents* no Dialogflow.**

Palavras-chave: Software; Tecnologia; Sociedade; Chatbot; Inteligência Artificial;

I. Introdução

Diante da incessante evolução tecnológica e pela crescente demanda por meios que simplifiquem o cotidiano das pessoas, a necessidade de criar soluções inovadoras é mais evidente do que nunca. É nesse contexto dinâmico e desafiador que surge o desenvolvimento de um *chatbot*, representando uma resposta relevante às demandas emergentes da sociedade contemporânea. Este projeto automatiza a criação de eventos no Google *Calendar* através do *chatbot* e ferramentas selecionadas [1]. A essência deste projeto reside na compreensão da grande importância desses serviços para o bem-estar e a qualidade de vida dos cidadãos.

A integração da IA (Inteligência Artificial) nesse *chatbot* representa um marco significativo para potencializar sua eficácia. Por meio de algoritmos de inteligência artificial, como o Processamento de Linguagem Natural, o *chatbot* pode automatizar o processo de agendamento de eventos para o usuário. A capacidade da IA de analisar grandes volumes de dados em tempo real promove uma gestão inteligente e dinâmica das agendas, resultando na otimização do tempo e na redução da necessidade de intervenção manual [2].

A grande importância desses serviços para o bem-estar e a qualidade de vida dos cidadãos, combinada com a necessidade premente de adaptação às tecnologias para a otimização das tarefas diárias, serviu como alicerces sólidos para desenvolver este trabalho. Em um mundo onde a eficiência e a conveniência são valorizadas como nunca, o desenvolvimento de soluções inovadoras que atendam às exigências do mercado atual é essencial para proporcionar uma experiência satisfatória aos usuários [3].

A. Justificativa

A automação de tarefas manuais por meio de ferramentas digitais tem se tornado indispensável diante da crescente necessidade de otimizar o tempo e aumentar a eficiência em atividades cotidianas. Nesse contexto, o desenvolvimento de soluções para facilitar o agendamento de eventos, como o proposto neste trabalho, representa uma resposta prática e alinhada às demandas da sociedade contemporânea [4].

A adaptação às tecnologias emergentes é essencial para acompanhar o ritmo das transformações atuais e melhorar a gestão de tarefas rotineiras. Além disso, a evolução tecnológica tem redefinido a forma como as pessoas interagem com compromissos, destacando a relevância de ferramentas automatizadas que promovam maior praticidade e eficiência [5].

A evolução constante da tecnologia tem impulsionado mudanças significativas na forma como as pessoas interagem com os serviços ao seu redor. Espera-se que esses serviços ofereçam conveniência, eficiência e uma experiência satisfatória para os usuários.

B. Objetivo

O objetivo é oferecer uma solução inovadora e relevante às demandas do mercado atual, com o desenvolvimento de um *chatbot* que auxiliará o usuário, proporcionando a conveniência de criar diferentes eventos e tarefas de forma eficiente por meio da conexão com *Dialogflow* e a *API* do Google *Calendar*.

Os objetivos específicos deste trabalho são:

- Pesquisar e implementar uma interface para usuário;
- Fazer a configuração e o treinamento do agente do *chatbot Dialogflow*;
- Projetar funcionalidades de agendamento;
- Implementar e configurar as ferramentas necessárias do Google *Cloud Console*;
- Integrar todos os serviços do usuário, *chatbot* e serviços do Google;
- Testar o funcionamento geral da ferramenta;

II. Referencial Teórico

Nesta seção, são enfatizados os principais conceitos e tecnologias que foram utilizadas para o desenvolvimento do trabalho proposto.

A. *Inteligência Artificial*

A inteligência artificial é um campo focado no desenvolvimento de sistemas que podem realizar tarefas que normalmente exigem inteligência humana. Isso inclui uma ampla variedade de capacidades, como aprendizagem, raciocínio, percepção, compreensão da linguagem natural e interação com o ambiente [6].

As principais áreas de pesquisa e aplicação da Inteligência Artificial incluem o aprendizado de máquina (*Machine Learning* - ML), que se concentra na construção de algoritmos que permitem que as máquinas aprendam a partir de dados [7]. Dentro do aprendizado de máquina, uma abordagem avançada é o *deep learning*, que utiliza redes neurais artificiais profundas para aprender padrões complexos e extrair representações significativas dos dados [8]. Essas redes neurais, inspiradas no funcionamento do cérebro humano, são compostas por múltiplas camadas de unidades interconectadas, permitindo uma aprendizagem hierárquica e uma capacidade de generalização poderosa.

Além do aprendizado de máquina, a Inteligência Artificial abrange o processamento de linguagem natural (*Natural Language Processing* - NLP), que capacita as máquinas a entenderem e responder à linguagem humana [9]. As técnicas de *deep learning* têm sido particularmente eficazes em tarefas como tradução automática, sumarização de textos e análise de sentimentos.

Outra área crucial é a visão computacional, que permite que as máquinas interpretem e compreendam o mundo visual ao seu redor [10]. O *deep learning* desempenha um papel fundamental nesse campo, possibilitando avanços significativos em reconhecimento de imagens, detecção de objetos e segmentação de cena, entre outras aplicações.

Por fim, a robótica também se beneficia da Inteligência Artificial, integrando sistemas de aprendizado de máquina e visão computacional em robôs físicos para realizar tarefas que envolvem interação com o ambiente físico de forma autônoma e inteligente [11]. Essas áreas de pesquisa e aplicação demonstram como a Inteligência Artificial, incluindo o *deep learning* e redes neurais, está transformando diversos setores e impulsionando avanços significativos na tecnologia.

Os benefícios da Inteligência Artificial são vastos e incluem a automação de tarefas repetitivas e perigosas, melhorias em diagnósticos médicos e tratamentos personalizados, eficiência em processos industriais e logística, e avanços em serviços ao cliente através de *chatbots* e assistentes virtuais [12].

No entanto, também existem desafios significativos, como preocupações éticas relacionadas a vieses algorítmicos e impacto no emprego, privacidade e segurança dos dados, necessidade de regulamentação e governança, e os riscos associados ao desenvolvimento de Inteligência Artificial superinteligente [13].

O futuro da Inteligência Artificial promete transformar vários setores, incluindo saúde, transporte, educação e entretenimento. É crucial abordar as questões éticas e regulatórias para garantir que a Inteligência Artificial seja desenvolvida e utilizada de maneira responsável e benéfica para a sociedade [14], [15].

B. *Flutter*

O *Flutter* é um *framework* de desenvolvimento de aplicativos móveis criado pelo Google, lançado inicialmente em maio de 2017 [16]. Ele permite a criação de aplicações nativas para Android e iOS utilizando uma única base de código, o que economiza tempo e recursos para os desenvolvedores. O *Flutter* é baseado na linguagem de programação *Dart*, também desenvolvida pelo Google, e se destaca por seu desempenho rápido e alto nível de personalização.

Uma das principais vantagens do *Flutter* é o recurso de "*hot reload*", que permite aos desenvolvedores ver as alterações no código instantaneamente, sem a necessidade de recompilar toda a aplicação. Isso acelera significativamente o processo de desenvolvimento e depuração. Além disso, o *Flutter* oferece uma vasta biblioteca de *widgets* personalizáveis, permitindo a criação de interfaces de usuário atraentes e responsivas que se comportam de maneira consistente em diferentes plataformas.

Outra característica importante do *Flutter* é seu motor de renderização próprio, o *Skia*, que garante alta performance e um controle preciso sobre cada pixel na tela. Isso possibilita a criação de animações suaves e experiências de usuário altamente interativas. O *Flutter* também suporta integração com APIs (*Application Programming Interface*) e serviços de *back-end*, tornando-o uma solução robusta para aplicações complexas.

Com o crescimento da comunidade e o suporte contínuo do Google, o *Flutter* tem se tornado cada vez mais popular entre desenvolvedores e empresas. Ele é utilizado por grandes marcas como Alibaba, BMW, e Tencent, demonstrando sua capacidade de atender a demandas de alta escala e qualidade. Além disso, o *Flutter* está se expandindo para além do mobile, com suporte para desenvolvimento *web* e *desktop*, aumentando ainda mais sua versatilidade [17], [18], [19], [20].

C. *Dart*

Dart é uma linguagem de programação desenvolvida pela Google, lançada em 2011 [21]. Projetada para ser produtiva, rápida e escalável, é usada principalmente no desenvolvimento de aplicativos móveis, *web* e de servidor [17], [22]. Seu propósito principal inclui o desenvolvimento de aplicativos móveis, sendo a linguagem principal do *framework Flutter*, amplamente adotado para criar aplicativos para iOS e Android com um único código-base [23]. Além disso, *Dart* pode ser compilado para *JavaScript*, permitindo a criação de aplicativos *web* modernos e dinâmicos. Embora menos comum, também pode ser usado para desenvolvimento de *back-end* e *scripts* de linha de comando.

A sintaxe de *Dart* é familiar para desenvolvedores acostumados com linguagens como *JavaScript*, *Java* e *C#*, facilitando a curva de aprendizado. Ele suporta tipagem estática e dinâmica, oferecendo flexibilidade para os desenvolvedores escolherem o estilo que melhor se adapta às suas necessidades. *Dart* pode ser compilado para código nativo *ARM* e x64, proporcionando um desempenho próximo ao de linguagens como *C++* em dispositivos móveis. Em conjunto com o *Flutter*, *Dart* suporta *hot reload*, permitindo que os desenvolvedores vejam as mudanças no código imediatamente sem reiniciar o aplicativo.

O ecossistema de *Dart* inclui o *Dart SDK*, que traz o compilador, *runtime*, gerenciador de pacotes e diversas ferramentas de desenvolvimento. O *Flutter*, um *framework UI* de código aberto baseado em *Dart*, permite a criação de interfaces de usuário bonitas e rápidas. Além disso, *Dart* possui um gerenciador de pacotes robusto (pub.dev) [24], com uma vasta coleção de bibliotecas e *plugins* disponíveis para facilitar o desenvolvimento [25].

D. Dialogflow

Dialogflow, uma plataforma do Google, facilita a criação de interfaces de conversação utilizando NLP avançado [26].

Ele oferece uma interface intuitiva para criar e gerenciar agentes, definir intenções e entidades, manter contextos de conversa e integrar-se com serviços de *back-end* através do *fulfillment, que* representa a capacidade da plataforma de executar tarefas e interagir com sistemas externos para completar as ações solicitadas pelos usuários.

Além disso, o *Dialogflow* suporta integração com várias plataformas de comunicação, como Facebook Messenger, Slack e Google Assistant, e permite o treinamento contínuo do agente para melhorar a precisão das respostas.

E. Google Calendar API

A *API* do Google *Calendar* permite que os desenvolvedores interajam com as funcionalidades de edição do calendário, facilitando a integração de recursos de em aplicativos web e mobile [27].

Essa *API* possibilita a criação de tarefas e eventos no calendário através de um *chatbot*, tendo uma sincronização e atualização em tempo real.

A partir de comandos por textos, o usuário consegue através do *chatbot* criar, listar ou excluir uma tarefa, escolhendo o título, data e o horário, sendo sincronizado diretamente com a conta escolhida do Google *Calendar*.

F. Componentes do Google Cloud Console e Google Developers Playground

A Google disponibiliza vários componentes para o desenvolvimento de softwares, com o intuito de aumentar a eficiência dos desenvolvedores que os utilizam. O Google *Cloud Console* e o Google *Developers Playground* estão entre esses componentes, oferecendo uma ampla gama de possibilidades, desde *APIs* até *Machine Learning* [28], [29]. Esses componentes serão mais detalhados na seção de Metodologia.

G. Trabalhos correlatos

1) AstroBot: Um chatbot com inteligência artificial para auxiliar no processo de ensino e aprendizagem de física: No trabalho de Adilmar Coelho Dantas [30], é apresentado o desenvolvimento de um chatbot utilizando o processamento de linguagem natural e inteligência artificial para suportar o ensino de física. O chatbot foi integrado em aplicativos como Telegram e WhatsApp, visando tornar uma utilização mais acessível para os alunos. O desenvolvimento desse chatbot teve como tecnologias utilizadas a linguagem de programação Python e uma arquitetura baseada em nuvem. Foram aplicadas técnicas de processamento de linguagem natural e inteligencia artificial, com o uso do Dialogflow para construção de intenções e fluxos de diálogo. A avaliação de desempenho demonstrou que esse chatbot consegue lidar eficientemente com um grande número de requisições simultâneas.

2) HelpCare: Um Protótipo de ChatBot para o Auxílio do Tratamento de Doenças Crônicas: No trabalho de Natália Oliveira [1], foi desenvolvido um protótipo de *chatbot* denominado HelpCare. O objetivo desse protótipo é auxiliar pacientes com doenças crônicas, como colesterol alto, hipertensão e diabetes, oferecendo cuidados personalizados, reduzindo o tempo de espera por atendimento. Com essa interação, o *chatbot* pode informar o usuário sobre suas condições de saúde, dicas de tratamento e monitoramento dos sintomas. Esse *chatbot* teve como tecnologia em seu desenvolvimento a utilização de processamento de linguagem natural, permitindo com que o *chatbot* compreenda e responda os usuários de forma mais adequada e natural.

3) Increasing customer service efficiency through artificial intelligence chatbot: No trabalho de Ivan Martins de Andrade [2], a pesquisa examina como a inovação de inteligência artificial, especialmente através de *chatbots* torna-se mais eficiente no atendimento ao cliente em um banco comercial. O estudo demonstra um impacto positivo, reduzindo filas em centros de atendimento, permitindo que atendentes humanos se encarreguem de questões mais complexas.

4) Considerações sobre os trabalhos correlatos: Os três trabalhos citados anteriormente demonstram similaridades com o trabalho proposto, como implementação de inteligência artificial a partir de um *chatbot* para aumentar a eficiência de seus processos. Também é citado o uso de processamento de linguagem natural, que é uma técnica de inteligência artificial responsável por tornar o *chatbot* mais responsível à linguagem humana.

III. Metodologia

A abordagem metodológica adotada neste trabalho é estruturada em uma pesquisa bibliográfica abrangente e detalhada, cujo propósito é servir de base sólida para o desenvolvimento do projeto em questão.

Para o desenvolvimento desse trabalho, optou-se pela metodologia *Extreme Programming* (XP) [31]. Trata-se de uma metodologia de desenvolvimento ágil de software que enfatiza a flexibilidade e a capacidade de resposta às mudanças. Essa metodologia incentiva a comunicação constante entre desenvolvedores e clientes, atendendo as necessidades dos clientes e evitando complexidades desnecessárias.

A. Flutter

O *Flutter* ofereceu a vantagem de compilar diretamente para código nativo, garantindo um ótimo desempenho para plataforma Web. Sua característica de *Hot Reload*, que permite atualizações em tempo real, acelerou significativamente o ciclo de desenvolvimento, enquanto sua biblioteca de *widgets* personalizáveis facilitou a criação da interface de usuário simples e intuitiva.

B. Dart

A escolha da linguagem de programação *Dart* complementou de forma ideal o uso do framework *Flutter*. *Dart* possui uma sintaxe limpa e moderna, tornando-a acessível a desenvolvedores de diversos níveis de habilidade e experiência.

Sua capacidade de criar interfaces reativas melhorou ainda mais a experiência do usuário, garantindo uma interação suave e responsiva [32]. Além disso, *Dart* oferece uma robusta biblioteca padrão e ferramentas integradas para testes e depuração, o que contribuiu para a produtividade e eficiência do desenvolvimento.

C. Google Cloud Console

Para o desenvolvimento do trabalho, foi necessário abrir um projeto no Google *Cloud Console* e criar uma conta de serviço, utilizada ao longo de todo o processo de configuração. Em seguida, foi preciso atribuir papéis específicos a essa conta, conforme mostrado na Figura 1. Os papéis atribuídos foram: Administrador da conta de serviço, Administrador da *API Dialogflow*, Agente de serviço do *Dialogflow*, Cliente da *API Dialogflow*, Editor e Proprietário.

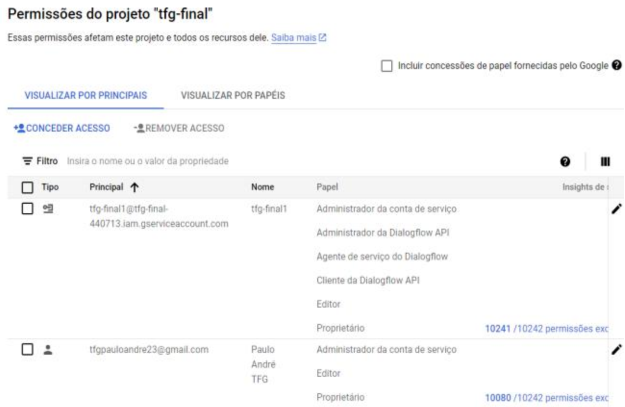

Figura 1. Papéis atribuídos para conta de serviço.

Ainda no Google *Cloud Console*, foi necessário ativar as *APIs* para possibilitar a comunicação com o *Flutter*, conforme mostrado na Figura 2. As *APIs* ativadas foram: *Dialogflow API* e Google *Calendar API*.

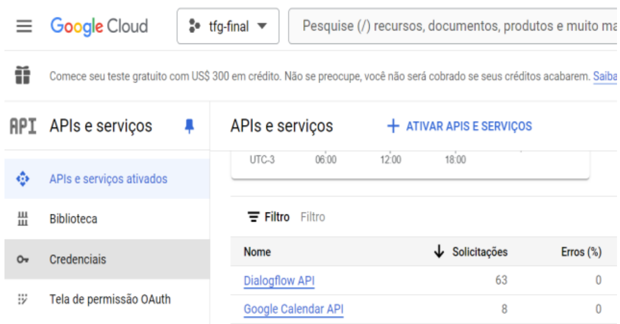

Figura 2. APIs ativadas.

Os últimos passos no Google *Cloud Console* incluíram a criação de uma tela de permissão *OAuth* e a geração de credenciais no formato JSON, conforme ilustrado na Figura 3.

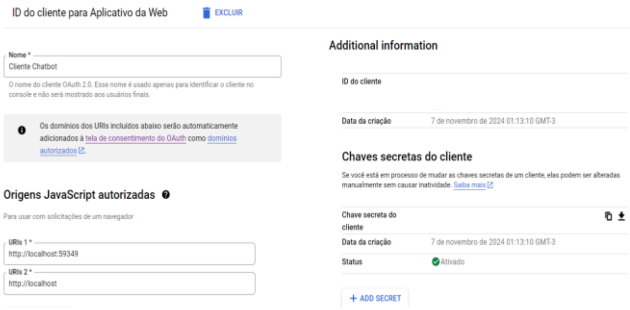

Figura 3. Credenciais do cliente.

D. Google Developers Playground

A configuração no Google *Developers Playground* foi simples e dividida em três etapas: Configuração do *OAuth* 2.0, Selecionar e autorizar *APIs* e Trocar código de autorização por *tokens*, conforme ilustrado na Figura 4. Na primeira etapa, foi necessário utilizar o *ID* do cliente e a chave secreta do cliente, gerados previamente no Google *Cloud Console*. Na segunda etapa, foram selecionadas e autorizadas as *APIs* do *Dialogflow*, Google *Calendar* e Google *OAuth*. Por fim, a terceira etapa realizou a troca dos códigos de autorização por *tokens*, permitindo que as *APIs* se comunicassem com o *Flutter*.

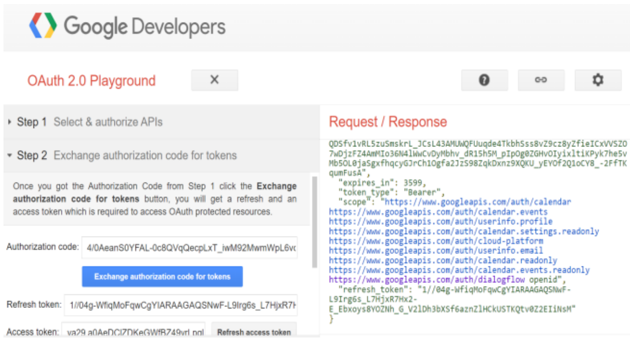

Figura 4. Configurações no Google Developers Playground.

E. Dialogflow

A plataforma *Dialogflow* foi utilizada para criar e treinar o agente, configurando *intents* com diversas frases. O agente foi projetado para se adaptar às interações dos usuários, respondendo de forma cada vez mais precisa. A Figura 5 ilustra algumas das *intents* criadas para o agente.

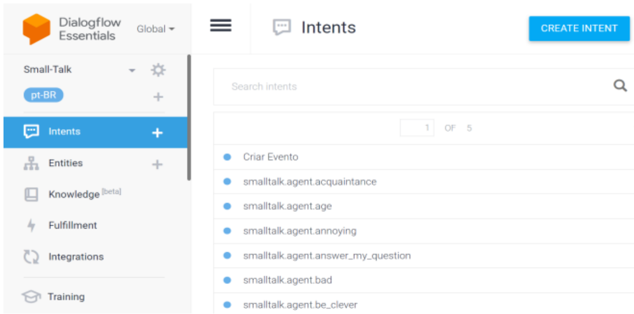

Figura 5. Intents do agente.

A *intent* "Criar Evento" é responsável por fornecer as respostas relacionadas à criação de eventos no Google *Calendar*, que são exibidas no *chatbot*, conforme ilustrado na Figura 6.

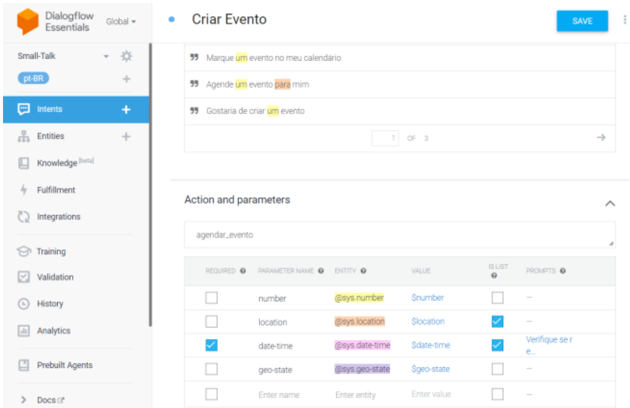

Figura 6. Intent Criar Evento.

F. *Autenticação com Conta Google*

Na Figura 7, pode-se observar o método que inicia o processo de autenticação com o Google. Caso o *login* seja bem-sucedido, o usuário é autenticado e os detalhes, como o nome, podem ser exibidos na interface.

```
Future<void> _handleSignIn() async {
  try {
    await _googleSignIn.signIn();
    print("Usuário logado: ${_googleSignIn.currentUser!.displayName}");
    _messages.add({'data': 0, 'message': 'Usuário logado: ${_googleSignIn.currentUser!.displayName}'});
  } catch (error) {
    print(error);
  }
}
```

Figura 7. Autenticação com conta Google.

G. *Acesso a Google Calendar API*

Na Figura 8, pode-se observar a função que cria um cliente autenticado para acessar a API do Google Calendar. É essencial para permitir ações como criar eventos.

```
Future<void> _initializeCalendarApi() async {
  final authClient = await _googleSignIn.authenticatedClient();
  if (authClient == null) {
    print("Erro ao obter o cliente autenticado.");
    return;
  }
  _calendarApi = calendar.CalendarApi(authClient);
}
```

Figura 8. Acessar API do Google Calendar.

H. *Diagrama de Atividade*

Na Figura 9, pode-se observar o diagrama de atividade, que ilustra o funcionamento do sistema. O usuário realiza o *login*, passando pela autenticação. Após a autenticação, o usuário pode interagir com o *chatbot* e receber respostas. Caso o usuário solicite a criação de um evento, o *chatbot* pede a data e hora desejadas, e, após o usuário informar e confirmar, o evento é criado no Google *Calendar*.

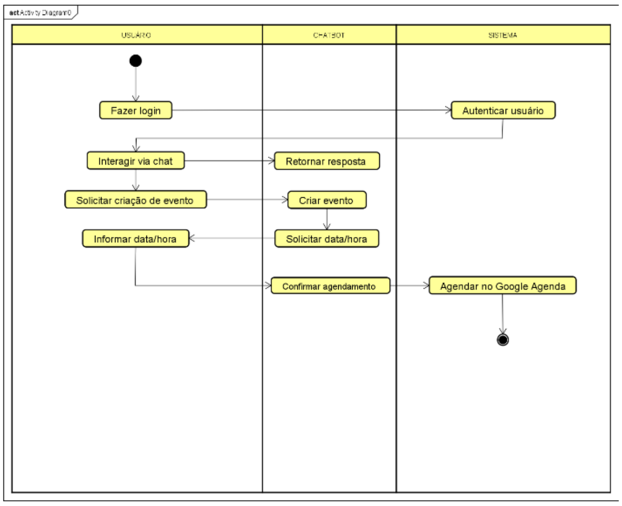

Figura 9. Diagrama de atividade (Fonte: Autor).

I. *Diagrama de Caso de Uso*

Na Figura 10, pode-se observar o diagrama de caso de uso, que mostra o usuário realizando a autenticação (*login* com conta do Google). Também estão representadas as opções de interagir com o *chatbot*, criar um evento com o *chatbot* e confirmar o agendamento.

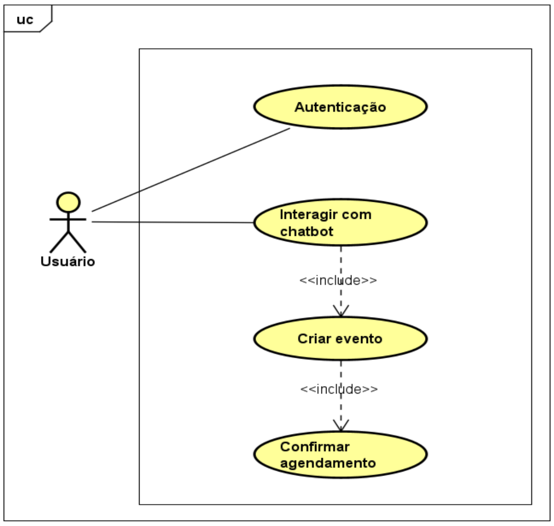

Figura 10. Diagrama de caso de uso (Fonte: Autor).

IV. TESTES

Testes de *intents* e *APIs* foram realizados com o objetivo de aprimorar o presente trabalho. O treinamento das *intents* do agente no *Dialogflow* demonstrou que, ao fornecer algumas perguntas e respostas, o sistema consegue reconhecer a *intent* e retornar a resposta adequada, conforme ilustrado na Figura 11.

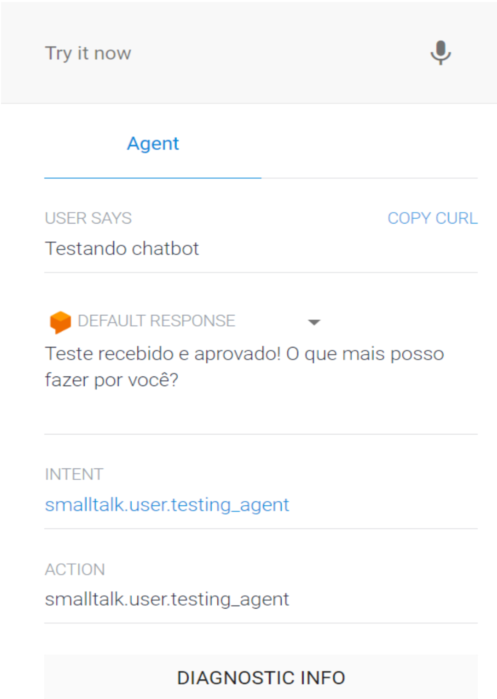

Figura 11. Teste de treinamento de intent.

O componente Google *Developers Playground* foi utilizado para realizar testes, a fim de verificar se as *APIs* selecionadas são chamadas sem erros.

V. RESULTADOS

Com o desenvolvimento concluído do *chatbot*, utilizando as ferramentas *Dialogflow* e Google *Calendar*, obtiveram-se resultados positivos. O projeto está disponível no repositório do GitHub [33], onde é possível acessar instruções detalhadas sobre sua configuração e utilização. Além disso, o repositório inclui links para as APIs e ferramentas necessárias para a implementação.

Uma função importante implementada foi a opção de realizar *login* com uma conta Google na interface do *chatbot*, conforme ilustrado na Figura 12. Isso permitiu a interação entre o *chatbot* e o calendário do usuário logado, possibilitando a criação de eventos no Google *Calendar*.

A interface do *chatbot* é simples e minimalista, contendo um botão de *login* no canto superior direito, que, após o *login* do usuário, se transforma em um botão de *logout*. Na parte inferior, encontra-se um campo de texto (*textfield*) para digitar mensagens, acompanhado de um botão para enviar a mensagem.

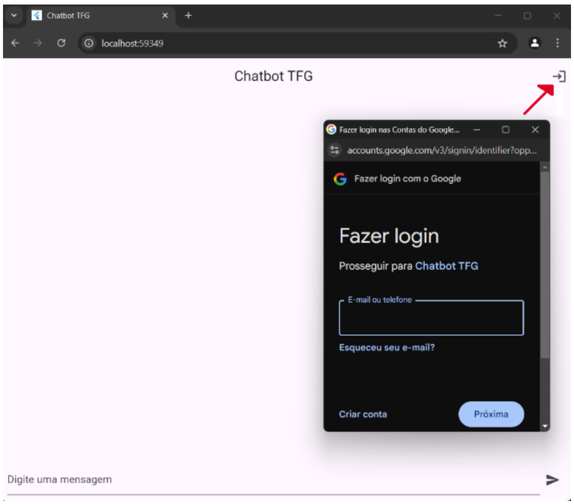

Figura 12. Interface inicial e login com conta Google.

Na Figura 13, ao enviar uma mensagem, o *chatbot* responde com base em seu treinamento. Se o usuário enviar as mensagens "Criar evento" ou "Agendar evento", novos campos aparecem na interface do *chatbot* logo abaixo do campo de digitação. As opções "Título do evento" e um ícone de calendário são exibidos, permitindo que o usuário selecione facilmente a data e hora, criando rapidamente um evento em seu Google *Calendar*.

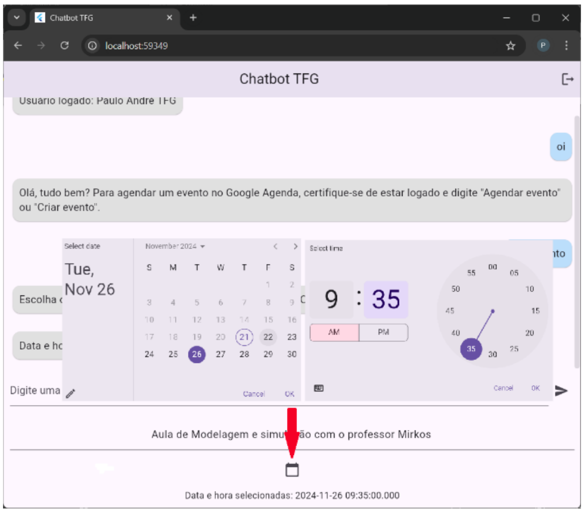

Figura 13. Novas opções ao digitar "Criar evento".

Com o usuário logado, após definir o título do evento e selecionar a data e hora, o evento é criado no Google *Calendar*. Além disso, um link para acessar o evento é gerado diretamente no *chatbot*, conforme ilustrado na Figura 14.

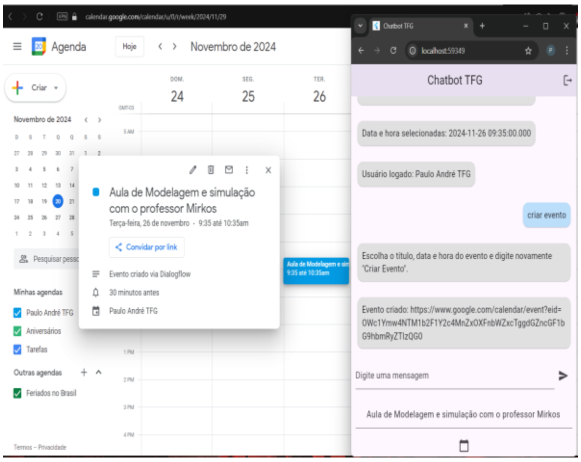

Figura 14. Evento criado e exibido no Google Calendar.

VI. Conclusão

Diante da crescente demanda por soluções tecnológicas que simplifiquem tarefas do dia a dia, este trabalho apresentou o desenvolvimento de um *chatbot* focado exclusivamente na automatização do agendamento de eventos no Google *Calendar*. O objetivo principal, prometido no início do projeto, foi plenamente alcançado, oferecendo uma solução funcional, eficiente e integrada para criação de eventos por meio de um *chatbot*.

A integração da inteligência artificial, por meio de Processamento de Linguagem Natural e do *Dialogflow*, permitiu ao *chatbot* compreender as solicitações dos usuários e realizar agendamentos de forma prática e intuitiva. O uso de tecnologias como o *framework Flutter* e a linguagem Dart foi essencial para criar uma interface simples, minimalista e funcional, demonstrando a viabilidade de unir ferramentas modernas em um único projeto.

Este trabalho representa um marco para aprendizado, ampliando o conhecimento sobre desenvolvimento de aplicações com inteligência artificial, integração de *APIs* e utilização de tecnologias modernas. Além disso, deixa um legado valioso para outros pesquisadores interessados em expandir ou aprofundar estudos sobre *chatbots* voltados para serviços automatizados.

Apesar do escopo proposto ter sido cumprido, o projeto está longe de se encerrar. Futuras implementações estão previstas, como a ampliação do *chatbot* para atender serviços empresariais, abrangendo processos mais completos de agendamento. Essas melhorias permitirão que o sistema alcance um público mais amplo e atenda a demandas cada vez mais diversificadas.

Assim, este trabalho não apenas cumpre seu propósito inicial, mas também estabelece uma base sólida para futuras inovações, contribuindo para o avanço tecnológico na área de *chatbots* e sistemas automatizados.

Referências

[1] N. Oliveira, A. Costa, D. Araújo, and C. Portela, “HelpCare: Um Protótipo de ChatBot para o Auxílio do Tratamento de Doenças Crônicas.”

[2] I. M. De Andrade and C. Tumelero, “Increasing customer service efficiency through artificial intelligence chatbot,” *Revista de Gestao*, vol. 29, no. 3, pp. 238–251, Jul. 2022, doi: 10.1108/REGE-07-2021-0120.

[3] E. D. Boudreaux, M. E. Waring, R. B. Hayes, R. S. Sadasivam, S. Mullen, and S. Pagoto, “Evaluating and selecting mobile health apps: strategies for healthcare providers and healthcare organizations,” Dec. 01, 2014, *Springer Science and Business Media, LLC*. doi: 10.1007/s13142-014-0293-9.

[4] D. R. T. Knight *et al.*, “Artificial intelligence for patient scheduling in the real-world health care setting: A metanarrative review,” Dec. 01, 2023, *Elsevier B.V.* doi: 10.1016/j.hlpt.2023.100824.

[5] M. Huang, H. Huang, I. Chen, K. Chen, and A. Wang, “Artificial intelligence aided course scheduling system,” in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Feb. 2021. doi: 10.1088/1742-6596/1792/1/012063.

[6] M. Pikhart, “Intelligent information processing for language education: The use of artificial intelligence in language learning apps,” in *Procedia Computer Science*, Elsevier B.V., 2020, pp. 1412–1419. doi: 10.1016/j.procs.2020.09.151.

[7] X. Dai, I. Spasic, S. Chapman, and B. Meyer, “The State of the Art in Implementing Machine Learning for Mobile Apps: A Survey,” in *Conference Proceedings - IEEE SOUTHEASTCON*, Institute of Electrical and Electronics Engineers Inc., Mar. 2020. doi: 10.1109/SoutheastCon44009.2020.9249652.

[8] S. Rezaei, B. Kroencke, and X. Liu, “Large-Scale Mobile App Identification Using Deep Learning,” *IEEE Access*, vol. 8, pp. 348–362, 2020, doi: 10.1109/ACCESS.2019.2962018.

[9] P. Story *et al.*, “Natural Language Processing for Mobile App Privacy Compliance.” [Online]. Available: https://data.usableprivacy.org.

[10] L. A. P. Neves, H. Vieira Neto, and A. Gonzaga, *Avanços em Visão Computacional*. Omnipax Editora Ltda, 2012. doi: 10.7436/2012.avc.0.

[11] S. Sarker, L. Jamal, S. F. Ahmed, and N. Irtisam, “Robotics and artificial intelligence in healthcare during COVID-19 pandemic: A systematic review,” Dec. 01, 2021, *Elsevier B.V.* doi: 10.1016/j.robot.2021.103902.

[12] H. Kim, S. Jung, and G. Ryu, “A Study on the Restaurant Recommendation Service App Based on AI Chatbot Using Personalization Information,” *International Journal of Advanced Culture Technology*, vol. 8, pp. 263–270, 2020, doi: 10.17703/IJACT.2020.8.4.263.

[13] J. S. Sichman, “Inteligência Artificial e sociedade: avanços e riscos,” *Estudos Avancados*, vol. 35, no. 101, pp. 37–49, Jan. 2021, doi: 10.1590/s0103-4014.2021.35101.004.

[14] C. Nebeker, J. Torous, and R. J. Bartlett Ellis, “Building the case for actionable ethics in digital health research supported by artificial intelligence,” Jul. 17, 2019, *BioMed Central Ltd.* doi: 10.1186/s12916-019-1377-7.

[15] G. Rubeis, “iHealth: The ethics of artificial intelligence and big data in mental healthcare,” *Internet Interv*, vol. 28, Apr. 2022, doi: 10.1016/j.invent.2022.100518.
[16] Flutter Documentation, “Flutter Documentation,” https://docs.flutter.dev/.
[17] G. Idan Arb and K. Al-Majdi, “A Freights Status Management System Based on Dart and Flutter Programming Language,” in *Journal of Physics: Conference Series*, Institute of Physics Publishing, May 2020. doi: 10.1088/1742-6596/1530/1/012020.
[18] C. DE Eduardo Oliveira Bueno, “PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS ESCOLA DE CIÊNCIAS EXATAS E DA COMPUTAÇÃO GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO DESENVOLVIMENTO DE UM APLICATIVO UTILIZANDO O FRAMEWORK FLUTTER E ARQUITETURA LIMPA,” 2021. Accessed: Jun. 18, 2024. [Online]. Available: https://repositorio.pucgoias.edu.br/jspui/handle/123456789/1861
[19] M. Maião Franklin and R. Aparecido Samuel Filho, “Desenvolvimento de um Sistema de Gestão Escolar com o uso da Linguagem Dart com Framework Flutter Development of a School Management System Using the Dart Language with Framewok Flutter.” Accessed: Jun. 18, 2024. [Online]. Available: https://ric.cps.sp.gov.br/handle/123456789/7070
[20] S. Riyadi and T. Cahyono, “Information System for Providing Food Services Based on Mobile Application Using Flutter Framework,” 2021. doi: 10.2991/aer.k.210204.031.
[21] Dart Documentation, “Dart Documentation,” https://dart.dev/guides.
[22] B. A. P. Candra and R. Ahmad, “Design and Development of Automotive Workshop Application Based on Android and IOS Using Dart Programming Language,” in *Journal of Physics: Conference Series*, Institute of Physics Publishing, Jul. 2020. doi: 10.1088/1742-6596/1539/1/012016.
[23] S. Y. Ameen and D. Y. Mohammed, “Developing Cross-Platform Library Using Flutter,” *European Journal of Engineering and Technology Research*, vol. 7, no. 2, pp. 18–21, Mar. 2022, doi: 10.24018/ejeng.2022.7.2.2740.
[24] pub.dev, “The official package repository for Dart and Flutter apps.” Accessed: Jun. 18, 2024. [Online]. Available: https://pub.dev/
[25] U. Alias and S. Swathiga, “AN INTERPRETATION OF DART PROGRAMMING LANGUAGE,” 2021. [Online]. Available: https://www.researchgate.net/publication/358661479
[26] Dialogflow Documentation, “Dialogflow Documentation.” Accessed: Jun. 18, 2024. [Online]. Available: https://cloud.google.com/dialogflow/docs
[27] S. L. Xuan, U. Tunku, and A. Rahman, “DIGITAL ASSISTANT FOR WORKSPACE APPS A REPORT SUBMITTED TO,” 2022.
[28] Google Cloud Documentation, “Google Cloud Console Documentation,” https://cloud.google.com/compute/docs/console?hl=en.
[29] Google Developers Playground, “Google Developers Playground,” https://cloud.google.com/appengine/docs/admin-api/trying-the-api?hl=en.
[30] A. C. Dantas *et al.*, “AstroBot: Um chatbot com inteligência artificial para auxiliar no processo de ensino e aprendizagem de física,” Sociedade Brasileira de Computacao - SB, Nov. 2019, p. 1196. doi: 10.5753/cbie.wcbie.2019.1196.
[31] A. Shrivastava, I. Jaggi, N. Katoch, D. Gupta, and S. Gupta, “A Systematic Review on Extreme Programming,” in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Jul. 2021. doi: 10.1088/1742-6596/1969/1/012046.
[32] G. Idan Arb and K. Al-Majdi, “A Freights Status Management System Based on Dart and Flutter Programming Language,” in *Journal of Physics: Conference Series*, Institute of Physics Publishing, May 2020. doi: 10.1088/1742-6596/1530/1/012020.
[33] P. A. M. Maracci, "TFG_CHATBOT," GitHub, [Online]. Disponível em: https://github.com/PauloMM23/TFG_CHATBOT. [Acessado em: 29 nov. 2024].