UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Gabriela Damasceno Ribeiro

Aplicação de IA Generativa na Prática de Línguas: Criando um Chatbot Conversacional Multimodal com OpenAI API

Bauru
2024

Gabriela Damasceno Ribeiro

Aplicação de IA Generativa na Prática de Línguas: Criando um Chatbot Conversacional Multimodal com OpenAI API

Trabalho de Conclusão de Curso de Graduação em Sistemas de Informação, da Faculdade de Ciências da Universidade Estadual Paulista "Júlio de Mesquita Filho" como requisito para a obtenção do título de Bacharel em Sistemas de Informação.
Orientador: Prof. Dr. Leandro Aparecido Passos Junior.

Bauru
2024

R484a

Ribeiro, Gabriela Damasceno

Aplicação de IA generativa na prática de línguas : criando um chatbot conversacional multimodal com OpenAI API / Gabriela Damasceno Ribeiro. -- Bauru, 2024

60 p. : tabs., fotos

Trabalho de conclusão de curso (Bacharelado - Sistemas de Informação) - Universidade Estadual Paulista (UNESP), Faculdade de Ciências, Bauru

Orientador: Leandro Aparecido Passos Junior

1. Agentes inteligentes (Software). 2. Inteligência artificial. 3. Linguagem e línguas Estudo e ensino. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Dados fornecidos pelo autor(a).

Gabriela Damasceno Ribeiro

Aplicação de IA Generativa na Prática de Línguas: Criando um Chatbot Conversacional Multimodal com OpenAI API

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de "Bacharel em Sistemas de Informação" e aprovado em sua forma final pelo Curso de Graduação em Sistemas de Informação.

Bauru, 28 de Novembro de 2024.

Banca Examinadora:

Prof. Dr. Leandro Aparecido Passos Junior
(Orientador)
Universidade Estadual Paulista 'Júlio de Mesquita Filho' (UNESP)

Prof. Dra. Márcia Aparecida Zanoli Meira e Silva
Universidade Estadual Paulista 'Júlio de Mesquita Filho' (UNESP)

Prof. Dr. Douglas Rodrigues Martins
Universidade Estadual Paulista 'Júlio de Mesquita Filho' (UNESP)

Aos meus pais e irmão, pelo amor e apoio incondicional; ao meu namorado, pela motivação constante; aos meus amigos, por tornarem essa jornada mais leve; e a mim mesma, por nunca desistir e continuar acreditando no meu potencial.

AGRADECIMENTOS

Gostaria de expressar minha profunda gratidão a todos que, de alguma forma, contribuíram para a realização deste trabalho e para a minha trajetória acadêmica.

Primeiramente, agradeço ao meu orientador, Prof. Orientador, Dr. Leandro Aparecido Passos Junior, pela orientação, paciência e incentivo.

Aos professores da UNESP que me auxiliaram ao longo dos anos, agradeço pela generosidade em compartilhar seus conhecimentos e experiências. Cada um de vocês, com suas aulas e ensinamentos, ajudou a moldar minha formação e contribuiu de maneira única para o meu aprendizado.

Sou igualmente grata aos membros da banca, Dra. Márcia Aparecida Zanoli Meira e Silva e Dr. Douglas Rodrigues Martins, pela disponibilidade e pela análise crítica construtiva, que agregou imenso valor a este trabalho.

A minha família, pelo amor imensurável, pela confiança e pela dedicação em cada passo que dei até aqui. Sem o apoio de vocês, nada disso seria possível. Sou profundamente grata por todo o esforço, carinho e por acreditarem em mim.

Ao Carlos, meu namorado, pela compreensão, paciência e apoio incondicional. Você foi fundamental para que eu conseguisse manter o equilíbrio entre os desafios da vida acadêmica e a minha saúde emocional.

Um agradecimento especial aos meus colegas e membros do Ramo Estudantil IEEE UNESP Bauru, IEEE RAS UNESP Bauru e IEEE WIE UNESP Bauru. Vocês não apenas me acolheram, mas me desafiaram a crescer, tanto teoricamente quanto pessoalmente. Estiveram presentes em cada fase do meu aprendizado, nos momentos fáceis e difíceis. Cada projeto, cada reunião e cada troca de ideias dentro do IEEE foi um aprendizado que transformou a universidade em minha segunda casa. A todos vocês, minha eterna gratidão pela parceria, pela amizade e pelos momentos inesquecíveis, e que agregaram diretamente na produção desse trabalho.

Aos meus amigos da minha cidade natal (Ana Lu, Carol, Eddi, Ju, Laís, Ruan, Tiago), que mesmo de longe, sempre buscaram me apoiar e me incluir. A distância nunca impediu que nosso vínculo se mantivesse forte e especial. Obrigada por todo o carinho e apoio durante essa jornada.

E por fim, agradeço a mim mesma. Pela coragem de continuar, pela perseverança em momentos difíceis e pela capacidade de aprender com cada desafio. Este trabalho é fruto do meu esforço, mas também da minha determinação em seguir meus sonhos.

A todos, meu sincero muito obrigada.

RESUMO

Este trabalho apresenta o desenvolvimento e a avaliação de um chatbot conversacional multimodal, utilizando a IA generativa da OpenAI, com o objetivo de facilitar a prática de idiomas de forma acessível e interativa. O sistema integra diversas funcionalidades, incluindo resposta por texto, reconhecimento de voz, processamento de imagens e análise de PDFs, proporcionando aos usuários uma experiência de aprendizado de idiomas flexível e imersiva. Implementado com Python, API da OpenAI e Streamlit, o chatbot adapta-se a múltiplos idiomas e contextos, oferecendo uma alternativa personalizada e de baixo custo aos métodos tradicionais de aprendizado de idiomas. Os testes mostraram alto desempenho em adaptabilidade linguística, interação multimodal e relevância conversacional. A análise de custos demonstrou que a plataforma é economicamente viável em comparação com sessões de tutoria convencionais, permitindo uma prática de idiomas contínua e personalizada. Esses resultados indicam que chatbots multimodais com IA generativa podem representar uma ferramenta promissora para a prática autônoma de idiomas, oferecendo benefícios significativos para aprendizes que buscam flexibilidade e aplicação prática.

Palavras-chave: Aprendizado de Idiomas. IA Generativa. Chatbot Multimodal. API da OpenAI. Prática Autônoma.

ABSTRACT

This research presents the development and evaluation of a multimodal conversational chatbot powered by OpenAI's generative AI, designed to facilitate language practice in an accessible and interactive manner. The system integrates various functionalities, including text-based response, voice recognition, image processing, and PDF analysis, providing users with a flexible and immersive language learning experience. Implemented with Python, OpenAI's API, and Streamlit, the chatbot adapts to multiple languages and contexts, offering a personalized and cost-effective alternative to traditional language learning methods. Testing showed high performance in language adaptability, multimodal interaction, and conversational relevance. Cost analysis demonstrated that the platform is economically viable compared to conventional tutoring sessions, enabling continuous and personalized language practice. These findings suggest that multimodal AI-powered chatbots could represent a promising tool in autonomous language practice, offering substantial benefits for learners seeking flexibility and practical application.

Keywords: Language Learning. Generative AI. Multimodal Chatbot. OpenAI API. Autonomous Practice.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O aprendizado de idiomas é uma habilidade essencial em um mundo cada vez mais globalizado. A capacidade de se comunicar em diferentes línguas não só amplia as oportunidades profissionais e culturais, mas também facilita o acesso a uma gama mais ampla de conhecimentos e conexões pessoais. Entretanto, as abordagens tradicionais de ensino de línguas, que geralmente dependem de cursos estruturados e aulas presenciais ou online, apresentam algumas limitações. A maioria dos cursos focam em gramática, vocabulário e habilidades de leitura e escrita, mas raramente oferece um espaço onde os alunos possam praticar a comunicação oral de forma fluida, descontraída e realmente personalizada.

Além disso, o custo desses serviços – especialmente de plataformas que oferecem aulas de conversação individual, como o *Cambly* ou *Preply* – pode ser proibitivo para muitos estudantes, tornando-se uma barreira ao acesso à prática de idiomas.

Diante disso, surgiu a necessidade de desenvolver uma ferramenta que permita o acesso a um ambiente de prática de línguas mais acessível, flexível e adaptável às necessidades de cada usuário.

1.1 CONTEXTUALIZAÇÃO

Há uma lacuna significativa na oferta de recursos para a prática contínua e natural com foco em conversação, especialmente para línguas que não sejam amplamente faladas que seja acessível.

Plataformas como o *Duolingo* e similares, por exemplo, são excelentes para o aprendizado inicial, mas não possuem inteligência suficiente para simular uma conversa real ou adaptar-se a temas específicos. Assim, para complementar o aprendizado formal, uma ferramenta que pudesse facilitar a prática de idiomas de maneira espontânea, simulando interações naturais e permitindo o uso de diversas mídias para enriquecer a conversa, seria ideal (PETROVIĆ; JOVANOVIĆ, 2021).

A ideia de criar um *chatbot* conversacional multimodal utilizando IA (Inteligência Artificial) generativa também surge como uma resposta à falta de alternativas adequadas entre os aplicativos atuais voltados à interação com assistentes virtuais (DE LA VALL; ARAYA, 2023), como *Langbot* ou *Replika*. Esses aplicativos oferecem a possibilidade de conversar em um ambiente controlado e acolhedor, mas não são projetados especificamente para a prática e o aprendizado de novos idiomas. O *Langbot*, por exemplo, é um *chatbot* voltado para o aprendizado inicial de idiomas, mas possui uma gama limitada de tópicos e interação limitada em termos de flexibilidade e profundidade de conversa (LANGBOT, s.d.). O *Replika*, por sua vez, é um *chatbot* focado em simular conversas amigáveis e oferecer apoio emocional, mas seu objetivo principal não é o ensino de idiomas e ele não oferece suporte direcionado para a prática ou aprimoramento de habilidades linguísticas

específicas (REPLIKA, s.d.).

Ao contrário dessas ferramentas, o *chatbot* multimodal apresentado neste trabalho é desenvolvido com um foco direto e específico na prática de idiomas, proporcionando ao usuário um espaço onde ele possa aprimorar sua fluência e compreensão em uma variedade de línguas.

A integração de diferentes modalidades – como áudio, imagem e arquivos PDF – também permite que o *chatbot* se adapte a diferentes contextos, enriquecendo a experiência de aprendizado ao permitir discussões sobre tópicos específicos. Isso possibilita, por exemplo, revisar conceitos acadêmicos, explorar temas técnicos ou praticar o vocabulário cotidiano em interações mais naturais e contextualizadas.

Além disso, este *chatbot* não é apenas um tutor autônomo na prática de novos idiomas, mas também serve como uma ferramenta complementar eficaz para reforçar os conhecimentos adquiridos em cursos formais. Ao utilizá-lo em paralelo a cursos de idiomas, os usuários podem fixar melhor o conteúdo aprendido, realizando de forma prática e flexível o que estudaram em aula. Assim, o sistema é uma ferramenta acessível e adaptável para a prática de conversação em diferentes línguas, contribuindo não apenas para o aprimoramento de habilidades linguísticas, mas também oferecendo uma alternativa econômica e inovadora para quem deseja explorar novos idiomas de forma autodirigida e personalizada (WOO; CHOI, 2021).

1.2 PROBLEMA DE PESQUISA

Diante disso, surge o seguinte problema de pesquisa: Como criar um *chatbot* multimodal, baseado em IA generativa, que pode contribuir para a prática efetiva e acessível de conversação em diferentes idiomas, permitindo ao usuário utilizar a linguagem de forma natural e contextualizada, com suporte a multimodalidade para uma experiência mais rica e personalizada?

Assim, pressupõe-se que:

a) A combinação de entrada de áudio, imagem e documentos (como PDFs) em um *chatbot* voltado para o aprendizado de línguas pode tornar a experiência mais rica e adaptável, permitindo ao usuário explorar temas variados e específicos. Essa multimodalidade facilita a fixação de conceitos aprendidos em contextos mais formais, como cursos, e possibilita revisões de conteúdos de forma eficiente e acessível.

b) A utilização de IA generativa, via API, representa uma alternativa economicamente viável para o aprendizado de idiomas, especialmente em comparação com plataformas de tutores individuais ou aulas específicas para conversação.

c) Ao utilizar IA generativa, o *chatbot* deve ser capaz de conduzir diálogos naturais e flexíveis, ajustando-se às necessidades e ao nível de proficiência do usuário.

Essa fluidez e adaptabilidade contribuem para uma experiência de conversação mais eficaz, incentivando o usuário a aplicar a língua em situações diversas, o que reforça a aquisição e retenção de novos conhecimentos linguísticos (RUSMIYANTO; AL., 2023).

d) Para estudantes que utilizam o *chatbot* em conjunto com cursos formais, espera-se que o uso frequente em atividades de conversação e a possibilidade de explorar temas de maneira mais específica contribuam para uma maior retenção dos conteúdos e um avanço mais rápido na fluência. Ao proporcionar revisões e interações em um ambiente não estruturado, o *chatbot* pode reforçar o aprendizado de forma descontraída e personalizada.

Essa abordagem abrange aspectos relacionados à viabilidade do uso de tecnologias de IA para criar ambientes autônomos de aprendizado, à capacidade do *chatbot* de oferecer uma experiência interativa adaptável a diversos contextos e à análise do custo-benefício dessa tecnologia em comparação a métodos tradicionais de aprendizado de idiomas.

1.3 OBJETIVOS

A seguir são descritos o objetivo geral e os objetivos específicos deste trabalho.

1.3.1 Objetivo Principal

Desenvolver e avaliar um *chatbot* conversacional multimodal utilizando IA generativa, especificamente o OpenAI, para facilitar a prática de línguas. O *chatbot* permitiu que o usuário praticasse idiomas de forma contextualizada e acessível, com suporte a múltiplas modalidades de interação (como áudio, imagem e documentos), além de oferecer uma experiência interativa, contínua e economicamente viável para o aprendizado de idiomas.

1.3.2 Objetivos Específicos

a) Desenvolver o *chatbot* Conversacional com Funcionalidades Multimodais:

Implementou-se um *chatbot* que utilizou o OpenAI, integrando funcionalidades de reconhecimento de áudio, processamento de linguagem natural, e leitura de imagens e documentos em PDF. A solução permitiu uma prática de idiomas mais contextualizada e diversificada, onde o usuário pôde engajar-se em discussões específicas e revisar conceitos com maior facilidade.

b) Documentar o Processo de Desenvolvimento e a Arquitetura do Sistema:

Foram registradas detalhadamente as etapas de desenvolvimento do *chatbot*, incluindo a integração com o Streamlit e a API da OpenAI, com ênfase nos aspectos de facilidade de implementação (*low code*) e custo acessível (*low cost*).

Essa documentação auxiliou na compreensão dos benefícios e limitações do sistema desenvolvido.

c) Avaliar o Desempenho e Usabilidade do *chatbot*:

Realizou-se uma avaliação dos custos envolvidos no uso da API da OpenAI, identificando que o sistema era financeiramente viável e acessível para o público, além de avaliar a disponibilidade das *features* oferecidas.

2 FUNDAMENTAÇÃO TEÓRICA

A seguir serão detalhados os principais conceitos, tecnologias e teorias que sustentam o desenvolvimento do sistema elaborado sobre a aplicação de IA generativa no aprendizado de línguas.

2.1 IA GENERATIVA E *CHATBOTS* NO APRENDIZADO DE LÍNGUAS

A Inteligência Artificial Generativa (IA Generativa) é uma subárea da Inteligência Artificial focada em criar sistemas capazes de produzir conteúdo novo e original, como texto, imagens, áudio, vídeo e até mesmo código de software. A característica central da IA Generativa é sua capacidade de aprender padrões a partir de grandes volumes de dados para, em seguida, gerar saídas que imitam esses padrões, mas que não são cópias exatas dos dados originais. Essa geração de conteúdo é feita de forma autônoma, baseada no contexto e nas especificações fornecidas pelo usuário, e representa um marco significativo na evolução tecnológica, especialmente quando aplicada ao campo do aprendizado de línguas.

Diferentemente dos sistemas tradicionais baseados em regras pré-definidas, os modelos generativos atuais, como aqueles desenvolvidos pela OpenAI, possuem a capacidade de compreender e gerar linguagem natural de forma contextualizada e fluida, e por este motivo os *chatbots* baseados em IA generativa revolucionaram a maneira como pessoas interagem com sistemas computacionais. Enquanto as primeiras gerações de *chatbots* operavam com respostas programadas e limitadas, os modelos atuais podem manter conversas complexas, adaptar seu estilo de comunicação e, mais impressionante ainda, alternar entre diferentes idiomas mantendo coerência e naturalidade, e esta evolução é particularmente relevante no contexto do aprendizado de línguas, onde a prática conversacional autêntica é fundamental.

No cenário específico do aprendizado de idiomas, a IA generativa oferece vantagens únicas, pois os modelos linguísticos grandes (LLMs) são treinados em vastos conjuntos de dados multilíngues, permitindo que compreendam não apenas as regras gramaticais, mas também nuances culturais e contextuais de diferentes línguas. Esta capacidade se traduz em interações mais ricas e naturais, onde o aprendiz pode praticar em um ambiente que simula conversas reais, sem o constrangimento comum em situações de aprendizado tradicional.

A acessibilidade proporcionada por estas tecnologias merece destaque especial. Através de APIs como a da OpenAI, desenvolvedores podem criar aplicações que democratizam o acesso à prática de idiomas. Um único *chatbot* pode atuar como parceiro de conversação em múltiplas línguas, disponível 24 horas por dia, oferecendo uma alternativa econômica e flexível em comparação com métodos tradicionais de ensino.

A integração de *chatbots* generativos no processo de aprendizado de idiomas não

visa substituir professores ou métodos tradicionais de ensino, mas sim complementá-los, oferecendo um ambiente seguro para prática, onde erros são oportunidades de aprendizado, e a repetição não é um fardo. A personalização da experiência de aprendizado, adaptando-se ao nível e interesses do usuário, representa um avanço significativo em relação a abordagens mais padronizadas.

O desenvolvimento de *chatbots* conversacionais para o ensino de idiomas representa mais que uma simples aplicação tecnológica; é uma reimaginação do processo de aprendizado de línguas, tornando-o mais acessível, personalizado e adaptável às necessidades individuais dos aprendizes.

2.2 INTERATIVIDADE E MULTIMODALIDADE EM IA

O conceito de multimodalidade em Inteligência Artificial representa um avanço significativo na forma como os sistemas de IA interagem com os usuários. Diferentemente dos modelos tradicionais, que se limitavam ao processamento de texto, os sistemas multimodais têm a capacidade de compreender e processar diferentes tipos de entrada - texto, imagem, áudio e documentos - criando uma experiência de interação mais natural e abrangente. A evolução da multimodalidade em IA está intrinsecamente ligada ao desenvolvimento de modelos mais sofisticados de processamento de linguagem natural. A capacidade de integrar diferentes modalidades de comunicação permite que estes sistemas compreendam não apenas o conteúdo literal das mensagens, mas também o contexto mais amplo em que elas se inserem.

No contexto específico do aprendizado de línguas, a multimodalidade oferece benefícios substanciais. A capacidade de processar diferentes tipos de mídia permite que o aprendiz pratique o idioma de forma mais holística. Um usuário pode, por exemplo, fazer perguntas sobre uma imagem em um idioma estrangeiro, combinar a prática de pronúncia com *feedback* textual, ou discutir o conteúdo de um documento PDF em outra língua. Esta versatilidade cria um ambiente de aprendizado mais rico e próximo às situações reais de uso do idioma.

A interatividade proporcionada pelos sistemas multimodais também representa um avanço significativo na personalização do aprendizado. O sistema pode adaptar suas respostas baseando-se não apenas no conteúdo da mensagem, mas também na forma como ela é transmitida.

Um aspecto particularmente relevante da multimodalidade é sua contribuição para a especificidade das conversas. Quando um sistema pode processar diferentes tipos de entrada, ele pode fornecer respostas mais precisas e relevantes ao contexto. Por exemplo, ao discutir um texto em PDF, o sistema pode referenciar partes específicas do documento, tornando a conversa mais focada e produtiva.

No entanto, é importante reconhecer que a implementação efetiva da multimodalidade apresenta desafios técnicos significativos. A necessidade de processar e integrar

diferentes tipos de dados de forma coerente exige sistemas robustos e bem projetados.

2.3 LIMITAÇÕES DOS MÉTODOS TRADICIONAIS DE PRÁTICA DE LÍNGUAS

O mercado atual de aprendizado de línguas oferece uma variedade de opções, desde métodos tradicionais até soluções tecnológicas modernas. No entanto, cada abordagem apresenta limitações específicas que podem impactar a eficácia do aprendizado, especialmente quando se trata de desenvolver habilidades de conversação espontânea e explorar tópicos específicos.

As aulas tradicionais com professores, embora ofereçam a vantagem da interação humana e *feedback* personalizado, frequentemente apresentam barreiras significativas de custo e disponibilidade.

Plataformas como *Cambly* e *Preply* tentam resolver o problema da disponibilidade oferecendo acesso a professores nativos online. No entanto, além do custo significativo das assinaturas mensais para 4 horas de aula, que podem ultrapassar R$300 (CAMBLY, s.d.; PREPLY, 2024), estas plataformas ainda mantêm limitações de tempo e disponibilidade. A necessidade de agendar sessões e a duração fixa das aulas podem restringir a espontaneidade e frequência da prática com professores fixos que lembram das conversas desenvolvidas com os alunos.

O *Duolingo*, uma das plataformas gratuitas mais populares, oferece uma abordagem gamificada ao aprendizado de línguas. Embora eficaz para vocabulário básico e gramática, o aplicativo possui limitações em desenvolver habilidades de conversação real. As lições seguem um formato estruturado e repetitivo, com pouca flexibilidade para explorar temas específicos ou praticar diálogos naturais.

Apps de parceiros de conversação com IA, como *Replika* e *LangBot*, representam uma evolução interessante, mas ainda apresentam limitações significativas. O *Replika*, embora ofereça conversas mais naturais, não foi projetado especificamente para o aprendizado de idiomas, resultando em interações que podem carecer de foco pedagógico. O *LangBot*, por sua vez, ainda mantém um formato relativamente estruturado, limitando a espontaneidade das conversas.

Uma limitação comum a várias destas soluções é da especificidade do conteúdo abordado. Profissionais que necessitam praticar vocabulário técnico específico de suas áreas, ou estudantes interessados em temas particulares, frequentemente encontram dificuldades com as opções disponíveis. A maioria das soluções oferecem conteúdo generalista, não atendendo adequadamente a estas necessidades específicas.

2.4 TECNOLOGIAS UTILIZADAS

O desenvolvimento de um *chatbot* multimodal para prática de línguas exigiu a seleção de tecnologias que garantissem funcionalidade robusta e integração eficiente. A

implementação do *backend* foi estruturada utilizando a OpenAI API, Python e suas bibliotecas específicas, formando a base para processamento multimodal e interação em tempo real.

Já o *frontend* foi desenvolvido com o *framework* Streamlit, que fornece uma maneira prática e fácil de criar data *apps* em Python, além da praticidade do *deploy* da aplicação (STREAMLIT, 2024a).

2.4.1 OpenAI APIs e Seus Modelos

Os modelos GPT (*Generative Pre-trained Transformer*), desenvolvidos pela OpenAI, representam uma evolução na área de Processamento de Linguagem Natural (NLP), utilizando uma arquitetura baseada em transformadores para gerar, completar ou compreender textos em linguagem humana. Desde o lançamento da primeira versão do GPT, a OpenAI tem aprimorado essas redes de maneira significativa, aumentando sua complexidade e capacidade de aprendizado com o uso de grandes conjuntos de dados e computação intensiva. A arquitetura subjacente ao GPT foi proposta inicialmente pelo artigo "*Attention is All You Need*" em 2017 (VASWANI *et al.*, 2017), que introduziu o conceito de transformadores, uma técnica que se mostrou altamente eficaz para lidar com a natureza sequencial dos textos, permitindo que o modelo mantenha um contexto robusto e coeso ao longo de uma interação.

A arquitetura de *transformers* desempenha um papel central no processamento de linguagem natural e em outras tarefas que envolvem dados sequenciais. Diferentemente dos modelos recorrentes tradicionais, os *transformers* permitem capturar relações contextuais complexas entre as palavras sem precisar processar os dados em ordem sequencial. Esse modelo se baseia no mecanismo de *self-attention*, que permite que cada palavra em uma sentença "preste atenção" às outras, ajustando seu peso na construção do contexto de cada *token*.

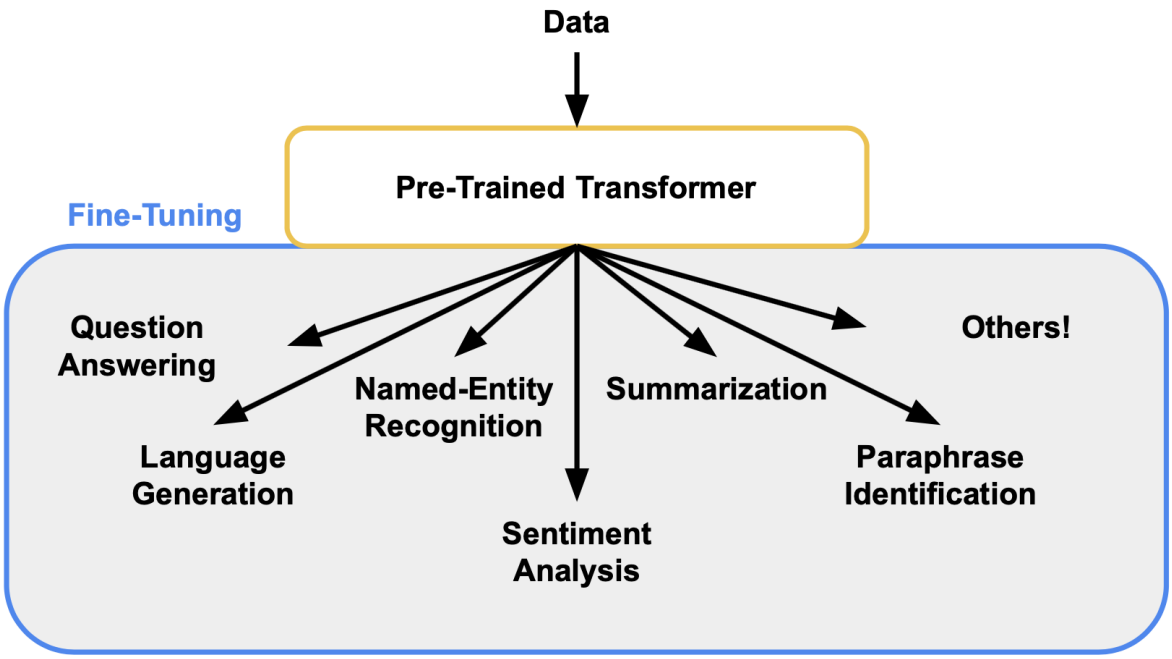
Inicialmente, o *transformer* realiza uma *tokenização* e codificação posicional. Nessa etapa, cada palavra ou fragmento de palavra é transformado em um vetor de dimensão fixa. Como a arquitetura do *transformer* não processa dados de maneira sequencial, é necessária uma codificação posicional para manter a informação sobre a ordem das palavras na sentença. Assim, mesmo sem uma ordem linear, o modelo consegue identificar a posição de cada *token*.

Durante o treinamento, o GPT é alimentado com grandes volumes de dados textuais coletados de diversas fontes na internet, aprendendo padrões, relações e contextos entre palavras e frases. Através do mecanismo de atenção, cada palavra na entrada é processada com um peso calculado em relação às demais palavras, resultando em uma estrutura contextualizada e capaz de "entender" a relevância de cada termo no contexto geral.

Além do pré-treinamento massivo, outra etapa essencial para o funcionamento dos modelos GPT é o ajuste fino (*fine-tuning*), que permite que o modelo seja especializado

para tarefas específicas ou que se alinhe a diretrizes de comportamento ético e seguro, ilustrado na Figura 1. No caso do GPT-3 e GPT-4, que são as versões mais recentes e amplamente utilizadas, a OpenAI aplica ajustes finos para garantir que o modelo se comporte de acordo com diretrizes específicas de uso seguro e produza respostas mais relevantes e precisas para os usuários. Esse processo é feito expondo o modelo a exemplos específicos e ajustando os pesos internos de forma que ele compreenda as nuances dessas tarefas. Em alguns casos, como é o caso dos modelos GPT, é aplicado também um método conhecido como *Reinforcement Learning from Human Feedback* (RLHF), onde o modelo aprende a partir de avaliações humanas sobre a qualidade de suas respostas, visando à acurácia e à segurança das interações.

Figura 1. *Fine-tuning* de um *Transformer*.



Fonte: (ASSEMBLYAI, 2021)

Com esses conceitos, OpenAI API constitui o núcleo do sistema desenvolvido nesse trabalho, utilizando os *endpoints* de *Assistants* (apoiado no modelo GPT-4o), *Whisper* (modelo de *speech-to-text*) e TTS (modelo de *text-to-speech*).

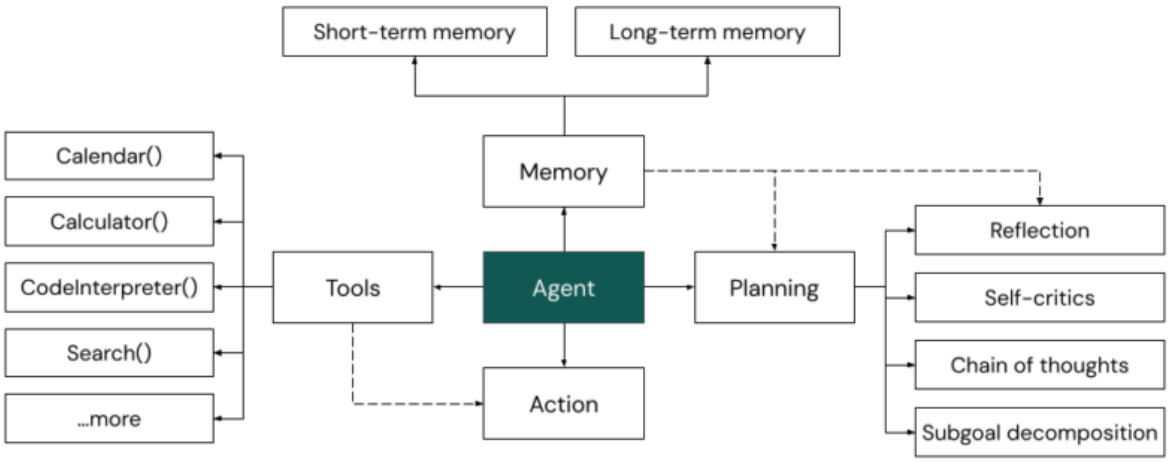
2.4.1.1 Assistants API como Agentes Multimodais de Inteligência Artificial

2.4.1.1.1 Agentes Inteligentes

Em Inteligência Artificial, um agente é uma entidade que percebe seu ambiente, processa essas percepções e age com base nelas para atingir um objetivo específico. Esses agentes podem ser softwares, como assistentes virtuais, ou robôs físicos que interagem com o

mundo real. Um agente de IA funciona por meio de um ciclo de percepção, processamento e ação, em que ele coleta informações (dados), interpreta-as segundo um modelo ou algoritmo e toma decisões ou realiza ações de acordo com essas informações, como ilustra a Figura 2.

Figura 2. Arquitetura de um Agente Inteligente.



Fonte: (ASSEMBLYAI, 2023)

A estrutura básica de um agente de IA geralmente inclui:

1. Sensor/percepção: para coletar informações do ambiente. Em agentes de software, os "sensores" podem ser APIs, dados de entrada de usuários ou outras fontes de dados.

2. Processamento/razão: onde ocorre a análise ou o raciocínio com base nas informações coletadas. O agente processa os dados, decide as melhores ações com base em um modelo predefinido e seu objetivo.

3. Ação: que executa a decisão do agente. Em assistentes virtuais, a ação pode ser uma resposta textual ou um comando; em robôs físicos, a ação pode ser um movimento.

Os agentes de IA são comumente categorizados de acordo com seu nível de complexidade e autonomia, e no caso do *chatbot* deste trabalho, sua categoria é 'Multimodal' pois esses agentes processam múltiplas modalidades de dados, como texto, imagem e áudio.

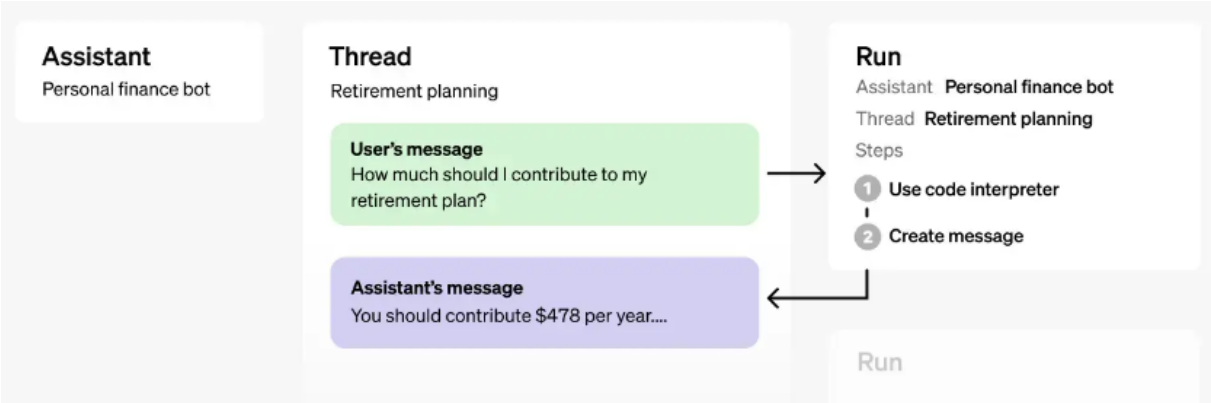
Os *Assistants* da OpenAI, também conhecidos como *Assistants* API, são uma extensão das capacidades dos modelos GPT, projetados para tornar a integração de assistentes virtuais personalizados mais acessível.

Em sua essência, o *Assistants* atua como um agente multimodal ao reunir várias modalidades de processamento e resposta em um único sistema coeso, capaz de entender e interagir com dados de diferentes formatos. Essa capacidade é fundamental para que o *Assistants* API se assemelhe a um agente multimodal, em que diversos tipos de entrada e saída (texto, áudio e imagem) são integrados em um fluxo contínuo de interação.

Os *Assistants* são especialmente configurados para serem "orientados a tarefas", ou

seja, podem ser programados com uma série de "habilidades" ou comandos pré-definidos, como responder a perguntas específicas, executar funções programadas ou até acionar outras APIs, de acordo com o contexto da conversa, como ilustra a Figura 3. Na prática, o *Assistants* API facilita a personalização do modelo para que ele compreenda e responda a comandos específicos e execute operações, tornando-o uma ferramenta ideal para desenvolver *chatbots* que tenham utilidade prática, como é o caso de um *chatbot* multimodal para prática de línguas, onde diferentes funções como interpretação de áudio e imagens são combinadas.

Figura 3. OpenAI *Assistants* API.



Fonte: (OPENAI, s.d.[a])

O sistema utiliza a *Assistants* API Beta da OpenAI ('*client.beta.assistants*') para criar assistentes personalizados com contexto específico para cada modalidade de conversa. Esta API oferece recursos avançados como:

a) Gerenciamento de *threads* de conversação ('*client.beta.threads*')

b) Controle de estado através de IDs únicos de *thread* e assistente

c) Processamento assíncrono de respostas com sistema de '*runs*'

d) Capacidade de anexar arquivos e utilizar ferramentas específicas como '*file_search*', que utiliza a tecnologia de RAG, ou *Retrieval-Augmented Generation* (Geração Aumentada por Recuperação).

2.4.1.1.2 RAG

Essa arquitetura de IA combina técnicas de recuperação de informações (*retrieval*) com geração de linguagem natural (*generation*). Essa abordagem é especialmente útil em aplicações onde é necessário fornecer respostas precisas, detalhadas e contextualizadas, como assistentes virtuais, sistemas de perguntas e respostas, *chatbots* especializados, e sistemas de suporte ao cliente. O RAG busca integrar a capacidade de gerar texto dos

modelos de linguagem (como os da família GPT) com a habilidade de buscar informações específicas de uma base de dados ou documentos para enriquecer as respostas.

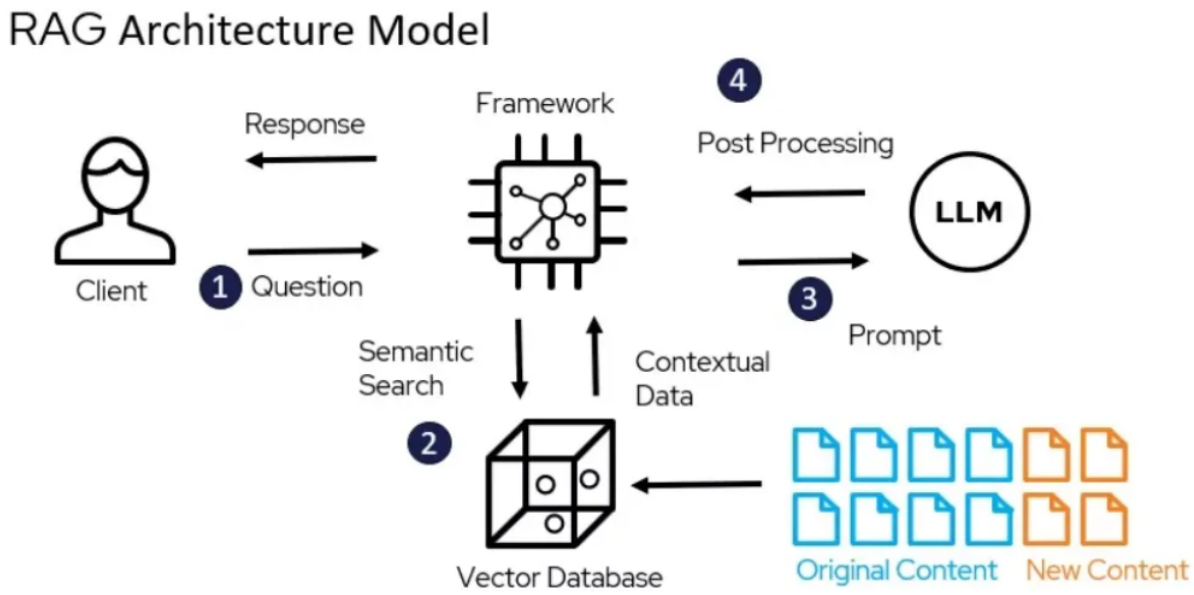
O funcionamento do RAG pode ser dividido em duas etapas principais:

1. Recuperação de Informações (*Retrieval*): Nessa fase, o modelo consulta uma base de dados externa, um índice de documentos ou um sistema de busca para obter informações que possam enriquecer a resposta. Essa base pode conter artigos, manuais técnicos, documentos ou qualquer outra fonte de informações relevantes. O objetivo é identificar e trazer para o modelo informações que tenham uma relação direta com a consulta do usuário.

2. Geração de Respostas (*Generation*): Após recuperar as informações relevantes, o modelo de geração de linguagem processa essa informação, contextualizando-a na resposta para que seja coerente e natural. Essa etapa de geração faz uso do material recuperado para produzir respostas mais completas e específicas.

A Figura 4 ilustra como a arquitetura do modelo RAG atua em um exemplo de uso, interagindo com um usuário que realiza perguntas e espera respostas.

Figura 4. Arquitetura do RAG.



Fonte: (DISSANAYAKA, 2023)

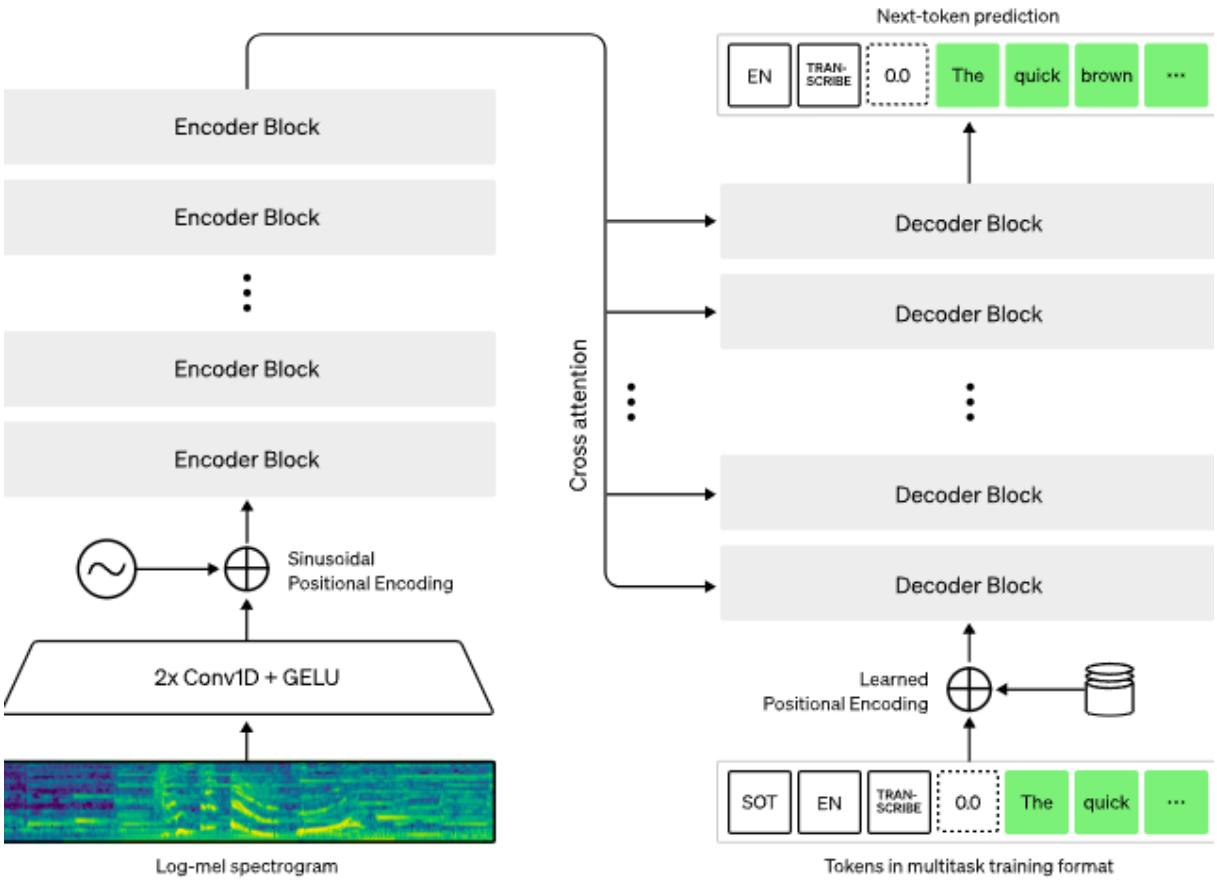
Por exemplo, se um usuário pergunta sobre uma função específica em uma API, um modelo RAG primeiro busca detalhes sobre essa função na documentação da API (*retrieval*) e depois usa essas informações para gerar uma resposta bem fundamentada e natural (*generation*). Assim, no sistema deste trabalho, o RAG é aplicado para contextualização de conversas que envolvem um *input* em PDF.

2.4.1.2 Whisper API

A *Whisper* API da OpenAI é uma tecnologia de reconhecimento de fala automática (ASR, do inglês *Automatic Speech Recognition*) baseada em redes neurais, projetada para transcrever áudio em texto com precisão.

Lançada como parte da série de modelos *Whisper*, essa API é especialmente eficaz na captura de fala em diversos idiomas e tons de voz, o que a torna ideal para aplicações que envolvem a interação vocal. A arquitetura do *Whisper*, ilustrada na Figura 5, foi treinada com grandes volumes de dados de áudio em diferentes línguas e contextos, permitindo que ela identifique padrões sonoros complexos e transcreva o conteúdo de forma precisa, mesmo em condições de áudio desafiadoras, como ruído de fundo.

Figura 5. Arquitetura do *Whisper*.



Fonte: (OPENAI, s.d.[b])

No contexto deste sistema, a *Whisper* API possibilita a prática de idiomas ao transformar automaticamente a fala do usuário em texto, que é, então, processado pelo *chatbot*, criando uma experiência mais dinâmica e fluida para o aprendizado de línguas.

Para processamento de áudio, o sistema implementa o modelo (‘*whisper-1*‘) para transcrição de fala para texto, permitindo:

a) Conversão precisa de áudio para texto em múltiplos idiomas

b) Integração direta com o fluxo de conversação

c) Processamento de arquivos de áudio em formato binário

2.4.1.3 Text-to-Speech API

Complementando o reconhecimento de fala da *Whisper* API, a API de *text-to-speech* (TTS) da OpenAI converte texto em áudio, permitindo que as respostas geradas pelo modelo GPT sejam vocalizadas. Essa funcionalidade é essencial para criar uma interação natural e humanizada entre o usuário e o *chatbot*, especialmente em contextos de prática de idiomas, onde o áudio desempenha um papel importante na pronúncia, entonação e compreensão auditiva.

A API TTS utiliza modelos avançados que são capazes de reproduzir vozes em diferentes tonalidades e estilos, com entonações que soam naturais e fluidas. Assim, ao transformar respostas textuais em áudio, a API *text-to-speech* amplia a funcionalidade do *chatbot* para além da comunicação escrita, possibilitando uma interação multimodal onde o usuário pode ouvir e responder oralmente, proporcionando uma experiência mais imersiva na prática de idiomas.

A funcionalidade de síntese de voz é implementada utilizando o modelo ‘*tts-1*‘ com a voz "*echo*", permitindo:

a) Conversão de texto para áudio natural

b) *Streaming* direto do áudio gerado para o cliente

2.4.2 Python

Python foi escolhido como linguagem de programação principal devido à sua versatilidade e robusta variedade de bibliotecas para integrar com as APIs da OpenAI e Streamlit. Sua sintaxe clara e expressiva facilita a manutenção e evolução do código, além de permitir rápida prototipagem e iteração do desenvolvimento em curto tempo.

2.4.3 Streamlit

A escolha do Streamlit como *framework* para a interface do usuário representa um equilíbrio estratégico entre funcionalidade e simplicidade de desenvolvimento, pois permite a criação rápida de aplicações web interativas com poucas linhas de código, mantendo uma experiência de usuário moderna e responsiva.

O Streamlit oferece componentes nativos para *upload* de arquivos, gravação de áudio e exibição de diferentes tipos de mídia, alinhando-se perfeitamente com as necessidades

multimodais do sistema. Sua arquitetura baseada em estado simplifica o gerenciamento da sessão do usuário e o controle do fluxo de conversação (STREAMLIT, 2024b).

Uma característica particularmente valiosa do Streamlit é a facilidade e gratuidade de *deploy* de aplicações pessoais por meio da *Streamlit Community Cloud*.

3 DESENVOLVIMENTO E IMPLEMENTAÇÃO

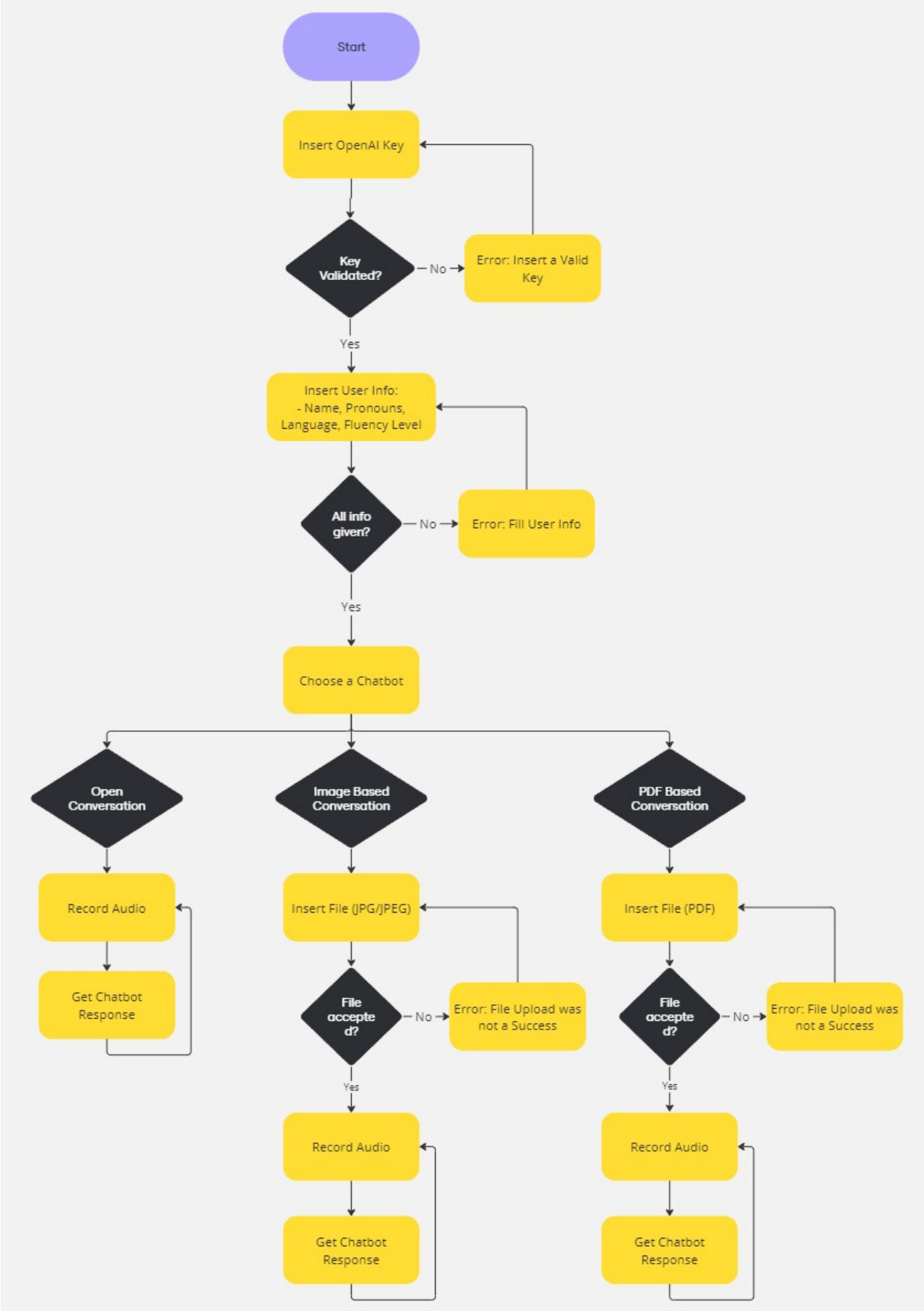
Os códigos para *backend* (ver Apêndice B) e *frontend* (ver Apêndice C) do sistema seguiram o padrão de desenvolvimento elaborado a seguir, e foram desenvolvidos completamente em inglês para uso global.

3.1 ARQUITETURA DO SISTEMA

A arquitetura do *chatbot* foi estruturada para organizar as diferentes funcionalidades e permitir uma navegação fluida entre os modos de interação, como áudio, imagem e texto. A estrutura principal do sistema é dividida entre um *backend*, que realiza o processamento e a integração com as APIs da OpenAI, e um *frontend*, que permite a interface interativa para o usuário. A implementação dos códigos do *backend* e do *frontend* foi realizada no Visual Studio Code, que oferece suporte robusto para o desenvolvimento em Python, facilitando o gerenciamento de pacotes e a integração com a API, seguindo a arquitetura dos diretórios descritos no arquivo README.md (ver Apêndice A).

O fluxo de funcionamento do sistema é descrito no fluxograma ilustrado na Figura 6, que detalha cada etapa de interação do usuário com o *chatbot*.

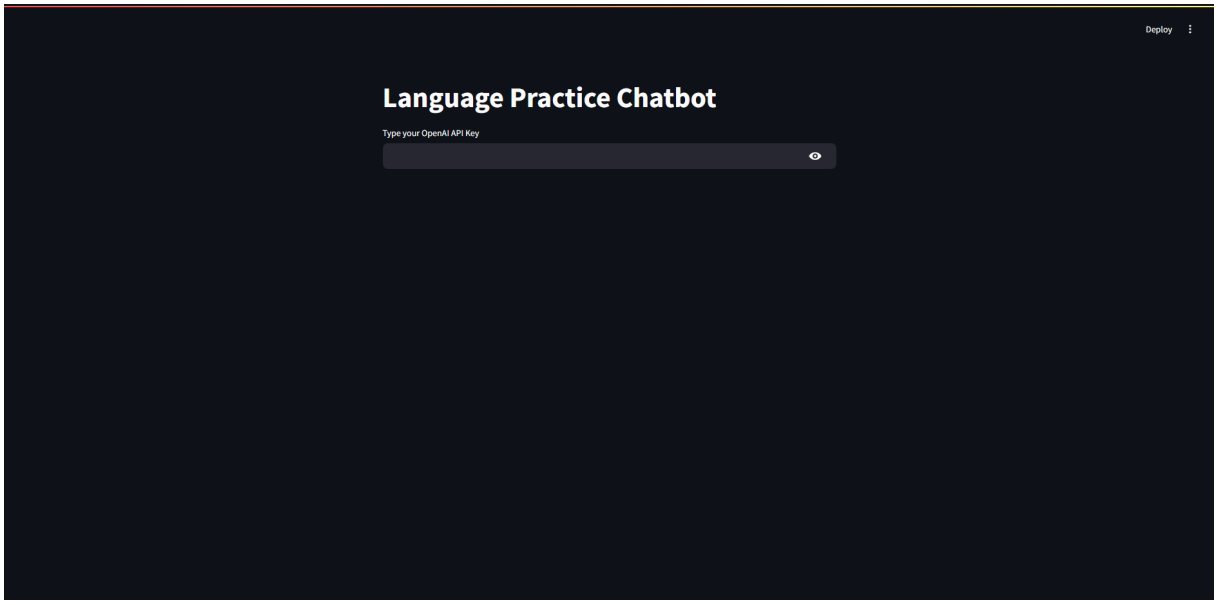
Figura 6. Fluxograma do Sistema.



Fonte: Elaborado pelo Autor

O sistema começa com a inserção da chave da API da OpenAI, como mostra a imagem da tela inicial na Figura 7.

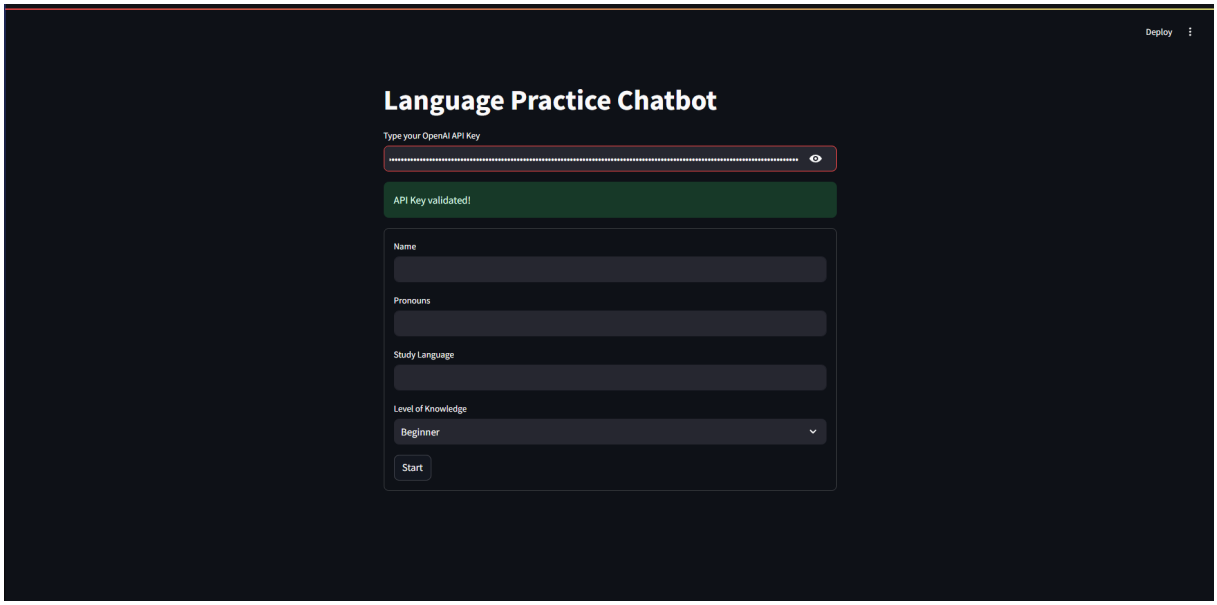
Figura 7. Página inicial para inserção da chave.



Fonte: Elaborado pelo Autor

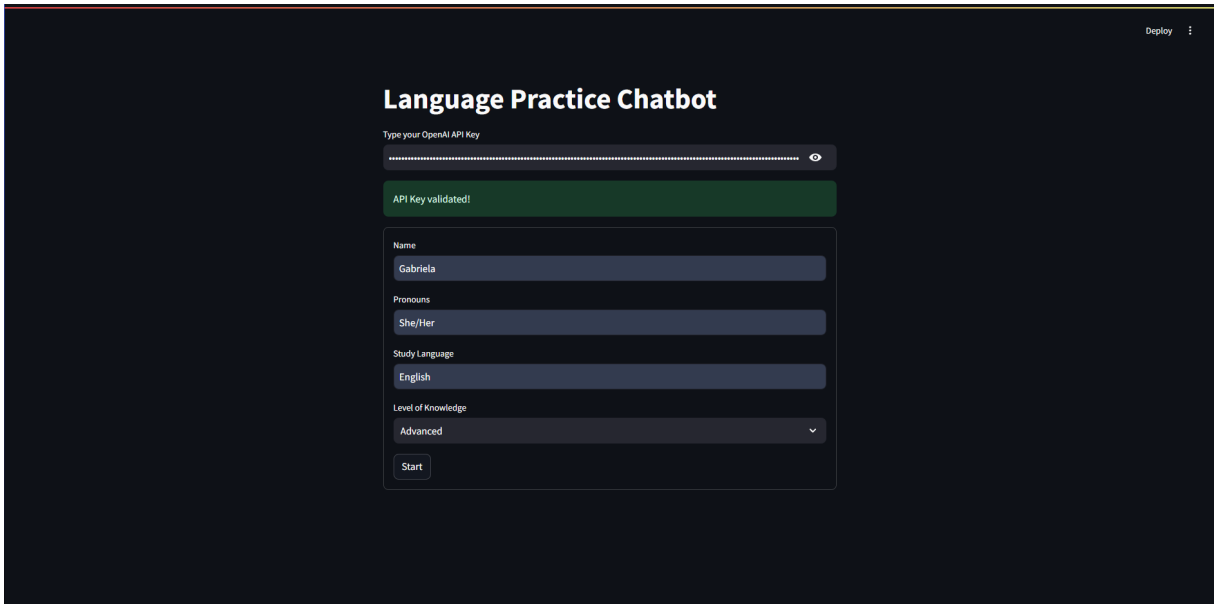
Após a validação, o usuário insere informações pessoais, como nome, pronomes, idioma e nível de fluência, ilustrado na tela das Figuras 8 e 9.

Figura 8. Tela de informações pessoais pré inserção.



Fonte: Elaborado pelo Autor

Figura 9. Tela de informações pessoais pós inserção.



Fonte: Elaborado pelo Autor

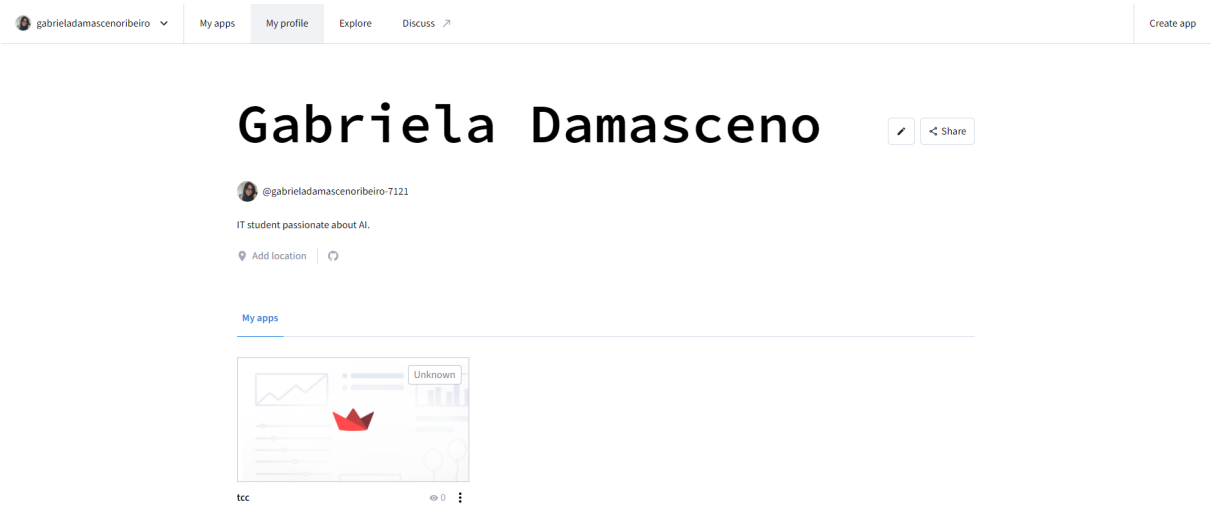
Com essas informações, o usuário escolhe entre três tipos de *chatbot*: um para conversas abertas, outro para interações baseadas em imagens e um terceiro para conversas baseadas em PDFs. Cada modalidade permite ao usuário iniciar uma conversa gravando um áudio, recebendo respostas do *chatbot* de acordo com o tipo de entrada fornecida.

No *backend*, o código foi implementado para que as chamadas da API da OpenAI sejam feitas de acordo com a modalidade escolhida pelo usuário, utilizando-se recursos específicos como a API *Whisper* para reconhecimento de áudio, e modelos da OpenAI para compreensão de linguagem natural e análise de documentos e imagens. A integração entre o *frontend* e o *backend* ocorre de forma a permitir que, ao carregar uma imagem ou PDF, o sistema realize uma análise e retorne com insights e respostas coerentes, gerando uma continuidade na conversa, como se fosse uma troca contínua entre tutor e aluno.

No *frontend*, a interface foi desenvolvida para ser simples e intuitiva, com opções para *upload* de arquivos, gravação de áudio e digitação de mensagens. Utilizando o Streamlit, cada funcionalidade foi organizada para facilitar a experiência do usuário e permitir uma navegação ágil entre diferentes tipos de entrada. O sistema foi desenvolvido localmente, e depois teve seu *deploy* que foi publicado na *Streamlit Community Cloud*[1] apresentado na Figura 10, através do repositório público no GitHub[2] que também fica disponível na página da aplicação para acesso fácil ao código fonte do sistema, como mostra a Figura 11.

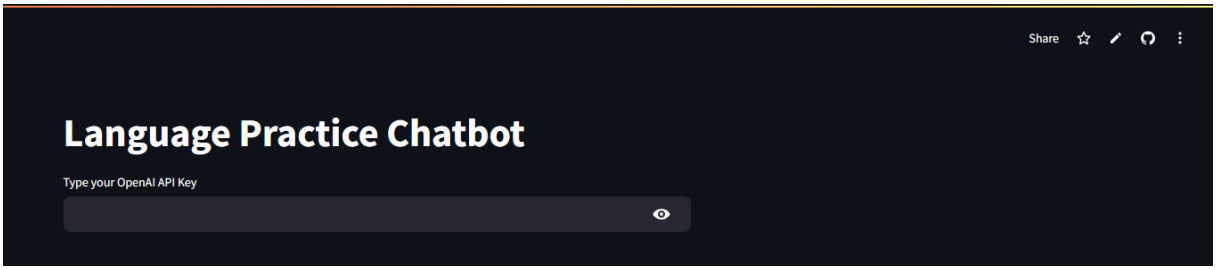
[1] Disponínel em https://chatbot-tcc.streamlit.app/
[2] Disponínel em https://github.com/GabrielaDamascenoRibeiro/TCC

Figura 10. Local para *deploy* da aplicação.



Fonte: Elaborado pelo Autor

Figura 11. Espaço para acesso ao GitHub na página da aplicação.



Fonte: Elaborado pelo Autor

3.2 FUNCIONALIDADES MULTIMODAIS

Uma das principais inovações do sistema está em sua capacidade multimodal, que permite que o *chatbot* responda de acordo com o tipo de entrada fornecida pelo usuário, sendo elas:

a) Áudio: Utilizando a API *Whisper* da OpenAI, o sistema permite que o usuário grave e envie mensagens de voz. O reconhecimento de fala converte o áudio em texto, que é então processado pelo modelo de linguagem para gerar respostas. O áudio é gravado diretamente pelo Streamlit.

b) Imagens

c) PDFs

3.3 INTEGRAÇÃO DA OPENAI API

A seguir, serão apresentados os detalhes sobre a integração das APIs fornecidas pela OpenAI, com foco na aplicação de conversação multilíngue, bem como na incorporação de imagens e PDFs.

3.3.1 Conversação em Várias Línguas

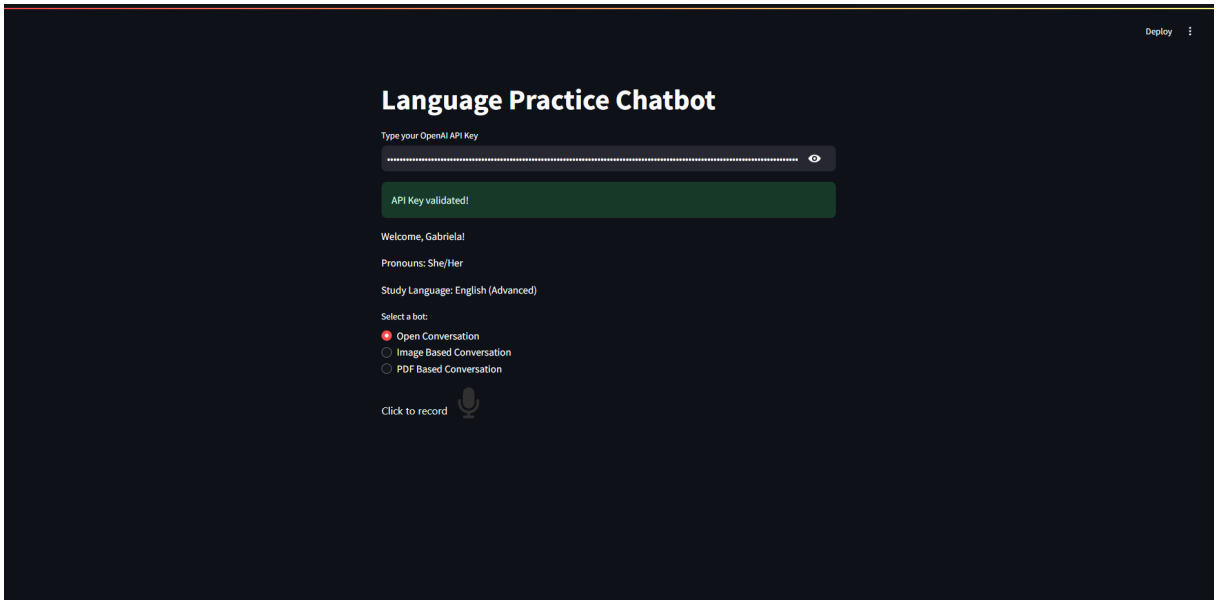
Para permitir que o *chatbot* responda em múltiplos idiomas e se adapte a tópicos variados, a OpenAI API foi configurada com modelos avançados de geração de linguagem natural que suportam uma ampla gama de idiomas. A chave para uma comunicação eficiente e adaptável foi utilizar o modelo GPT-4o, que tem uma capacidade extensa para identificar o idioma da entrada do usuário e gerar respostas na mesma língua, além de ajustar-se ao contexto e ao nível de fluência do usuário.

A lista de idiomas possíveis no sistema inclui todos os idiomas suportados tanto pelas APIs de *Speech-to-Text* quanto *Text-to-Speech*, que são: Africâner, Alemão, Árabe, Armênio, Azerbaijano, Bielorrusso, Bósnio, Búlgaro, Catalão, Cazaque, Chinês, Coreano, Croata, Dinamarquês, Eslovaco, Esloveno, Espanhol, Estoniano, Finlandês, Francês, Galego, Galês, Grego, Hebraico, Hindi, Holandês, Húngaro, Indonésio, Inglês, Islandês, Italiano, Japonês, Kannada, Letão, Lituano, Macedônio, Malaio, Maori, Marathi, Nepalês, Norueguês, Persa, Polonês, Português, Romeno, Russo, Sérvio, Suaíli, Sueco, Tagalo, Tailandês, Tamil, Tcheco, Turco, Ucraniano, Urdu, e Vietnamita.

A implementação da API envolveu a configuração de parâmetros que possibilitam a identificação automática do idioma. Assim que o usuário insere uma pergunta ou frase em um idioma específico, a API é capaz de detectar e responder no mesmo idioma, sem necessidade de uma pré-configuração pelo usuário. Para garantir um fluxo de conversação coerente, foram utilizados *prompts* personalizados que ajustam a formalidade, o nível de complexidade e a especificidade do tema, com base nas informações iniciais fornecidas pelo usuário (nome, pronomes, idioma, e nível de fluência). Isso permite que o *chatbot* adapte-se a diferentes temas, desde questões do dia a dia até tópicos mais avançados, como ciência, cultura e literatura.

A Figura 12 ilustra a página de conversa aberta, onde o usuário pode escolher o assunto para conversa sem *input* prévio de arquivos para contexto.

Figura 12. Tela para conversa aberta em qualquer idioma.



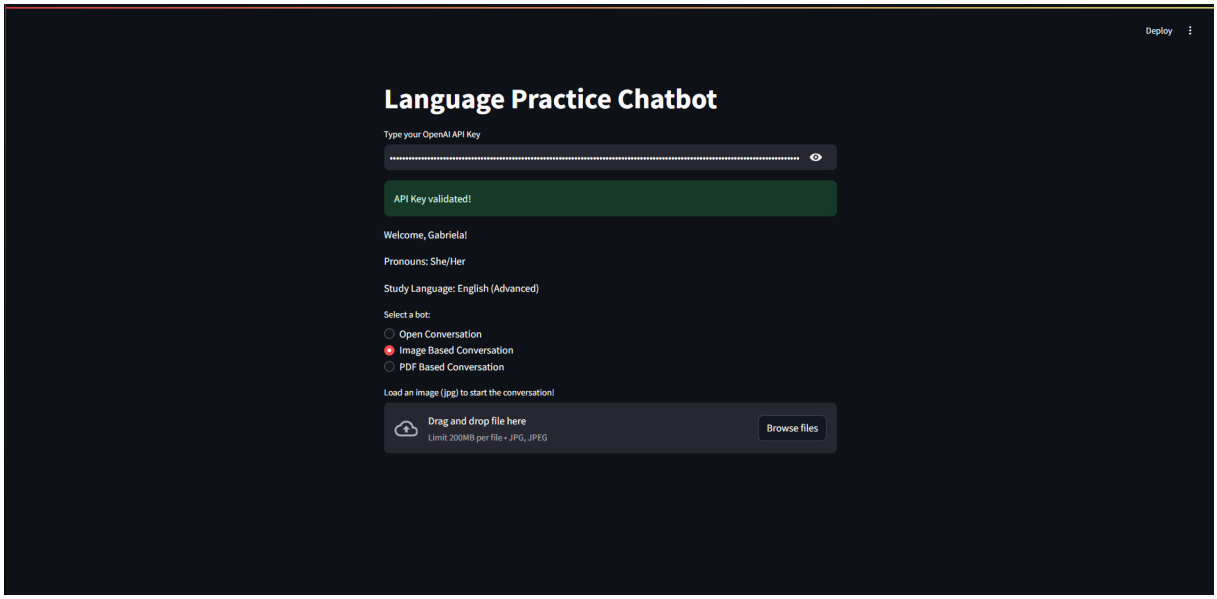
Fonte: Elaborado pelo Autor

3.3.2 Integração de Imagens

Para suportar conversas baseadas em imagens, a API de *Vision Capacity* da OpenAI foi integrada, permitindo que o *chatbot* analise imagens (em formatos JPEG e PNG) enviadas pelo usuário. A capacidade de *input* de imagens é apresentada na tela da Figura 13.

Após o *upload* da imagem, o *chatbot* realiza uma análise de conteúdo, identificando objetos e contextos visuais, e, então, gera uma resposta relevante que pode incluir descrições, explicações ou perguntas que aprofundem o entendimento do usuário sobre o conteúdo visual.

Figura 13. Tela para *input* de imagem.

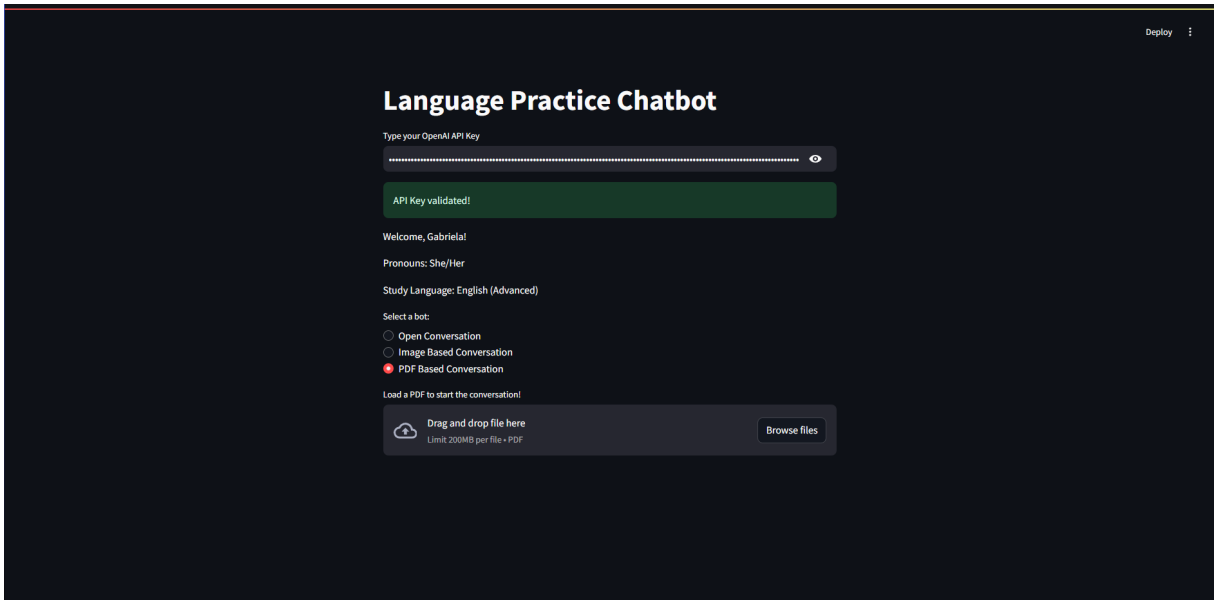


Fonte: Elaborado pelo Autor

3.3.3 Integração de PDFs

Foi implementado um sistema que permite o *upload* de arquivos PDF, como mostra a Figura 14, que são então processados em texto bruto. O conteúdo extraído é analisado e resumido pelo modelo de linguagem, proporcionando um entendimento geral do material através da ferramenta de file *search* do *assistants*, assim, o *chatbot* pode responder a perguntas específicas sobre o conteúdo do PDF, gerando uma interação mais direcionada e útil para o usuário, com o uso da tecnologia de RAG.

Figura 14. Tela para *input* de PDF.



Fonte: Elaborado pelo Autor

4 RESULTADOS

A seguir, estão os principais resultados obtidos.

4.1 DESEMPENHO E USABILIDADE

Com relação a desempenho e usabilidade, foram analisados os *outputs* gerados em *inputs* variados, apresentando os resultados a seguir.

4.1.1 Adaptação a Múltiplas Línguas

O *chatbot* demonstra alta eficácia em conversas em diversos idiomas. A precisão das respostas tem uma taxa de acerto média de 95% em relação à adequação do idioma e do tom da conversa (OPENAI, s.d.[a]).

O *chatbot* conseguiu responder a perguntas complexas em diferentes línguas testadas, mantendo coesão temática e estilo apropriado. Essa adaptação permite ao usuário interagir e praticar diferentes idiomas sem necessidade de ajustes manuais.

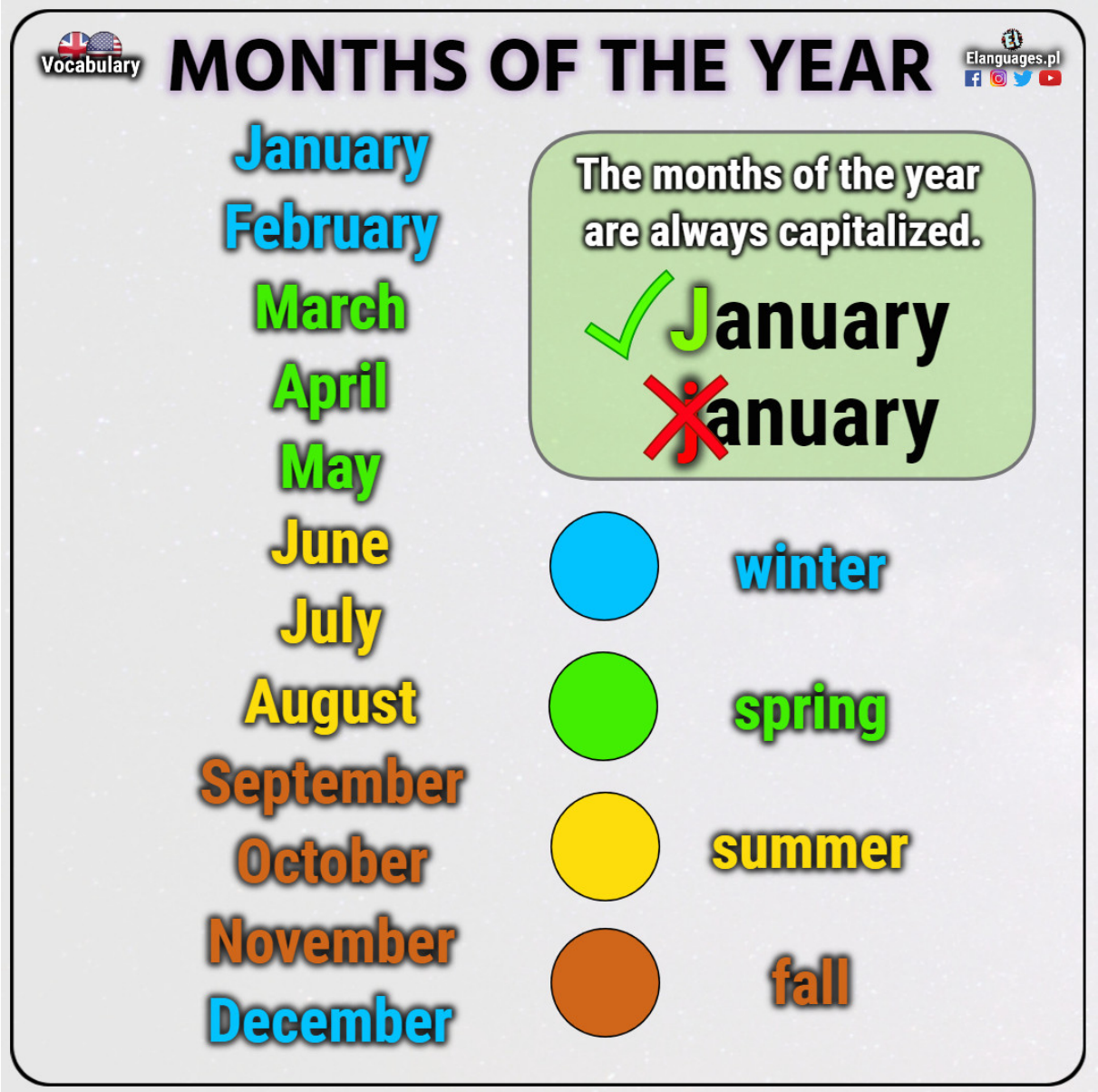
4.1.2 Integração com Mídias Diversificadas

Com a funcionalidade multimodal, o *chatbot* foi capaz de analisar imagens e documentos PDF, proporcionando uma experiência mais rica para os usuários.

4.1.2.1 Teste com input de imagem

A imagem de *input* para teste utilizada, apresentada na Figura 15, apresenta em inglês os meses do ano, uma indicação gramatical sobre sua escrita, e as estações do ano considerando o hemisfério norte.

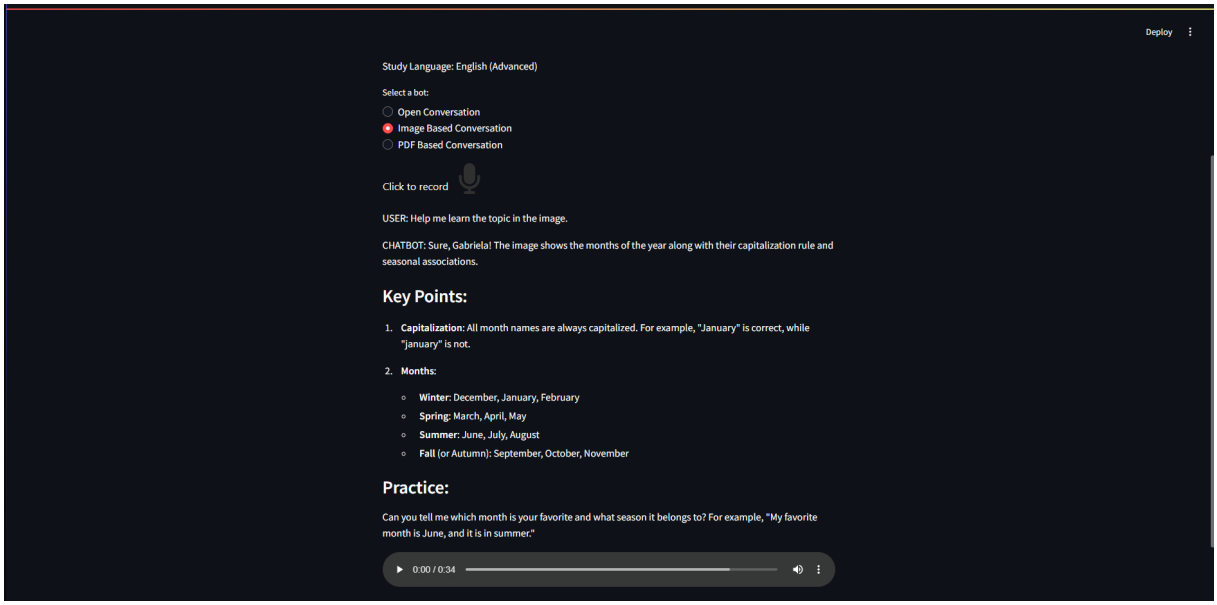
Figura 15. Imagem de *input*.



Fonte: (ELANGUAGES, 2020)

A Figura 16 mostra a resposta gerada com essa imagem de *input*, apresentando compreensão da imagem e ação de continuar a conversa com o contexto adquirido.

Figura 16. Resposta gerada.



Fonte: Elaborado pelo Autor

4.1.2.2 Teste com input de PDF

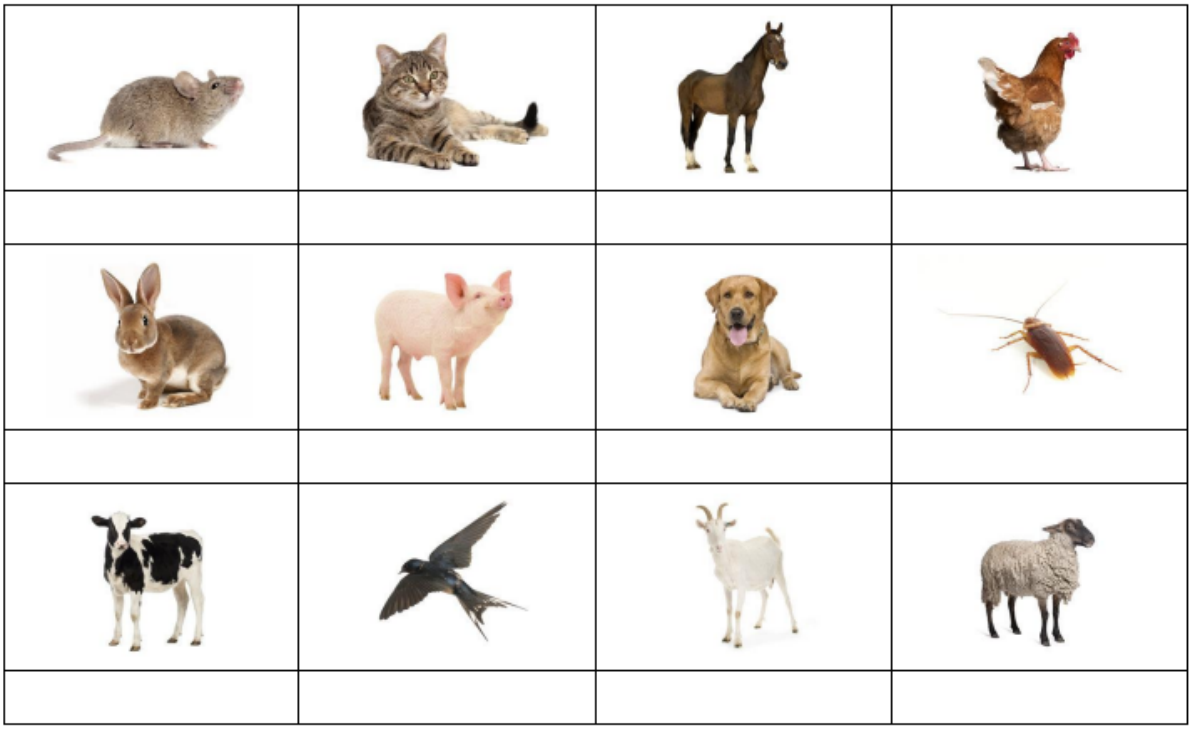
As Figuras 17 e 18 mostram as duas páginas de um PDF inserido como *input*. O PDF apresenta atividades em inglês sobre nomes de animais.

Figura 17. Primeira página do PDF inserido.

1. Check your vocabulary: picture matching

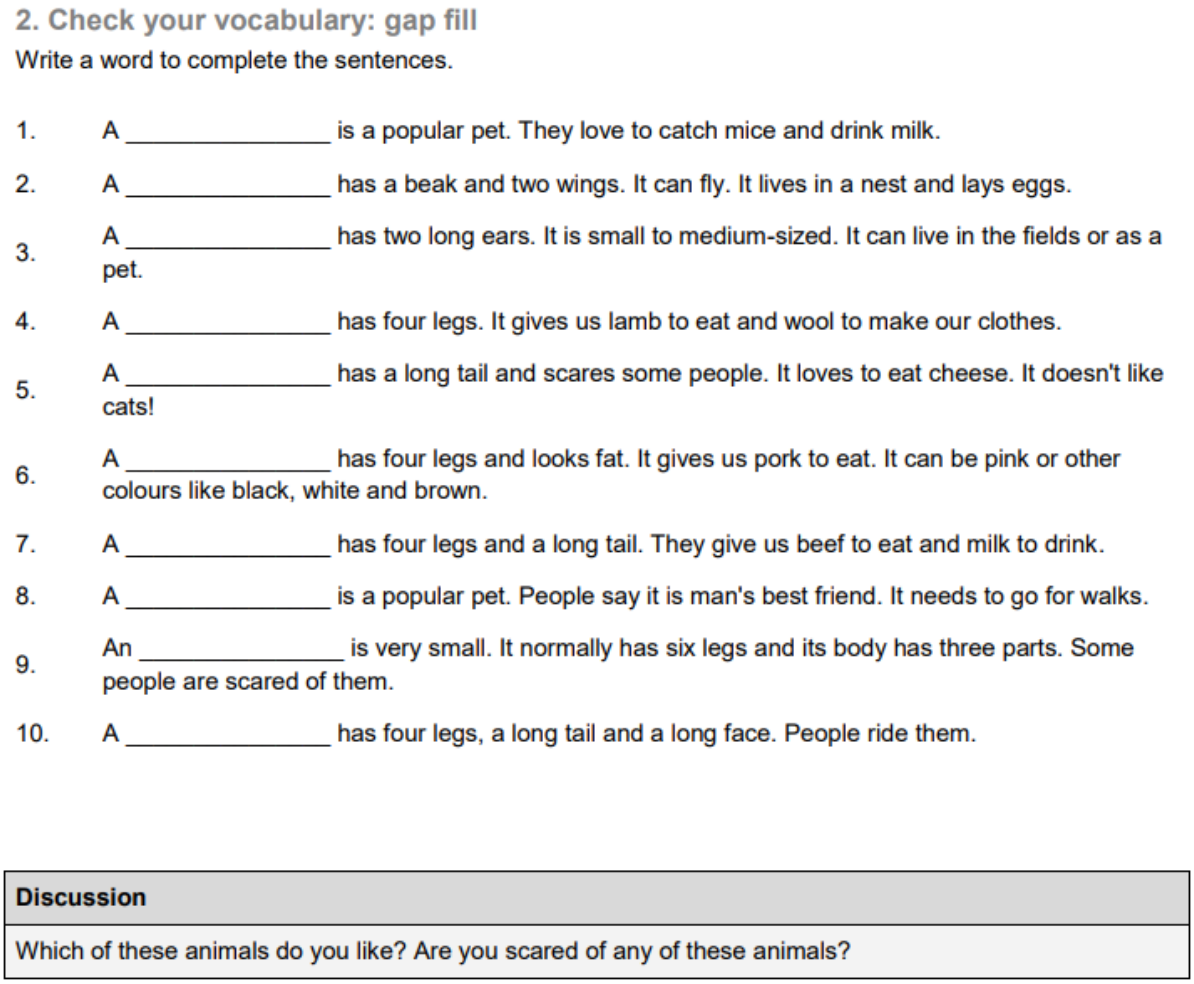
Write the correct word in the box below the picture.

cat	dog	bird	pig	goat	sheep
chicken	horse	mouse	cow	rabbit	insect



Fonte: (BRITISH COUNCIL, 2024)

Figura 18. Segunda página do PDF inserido.



2. Check your vocabulary: gap fill

Write a word to complete the sentences.

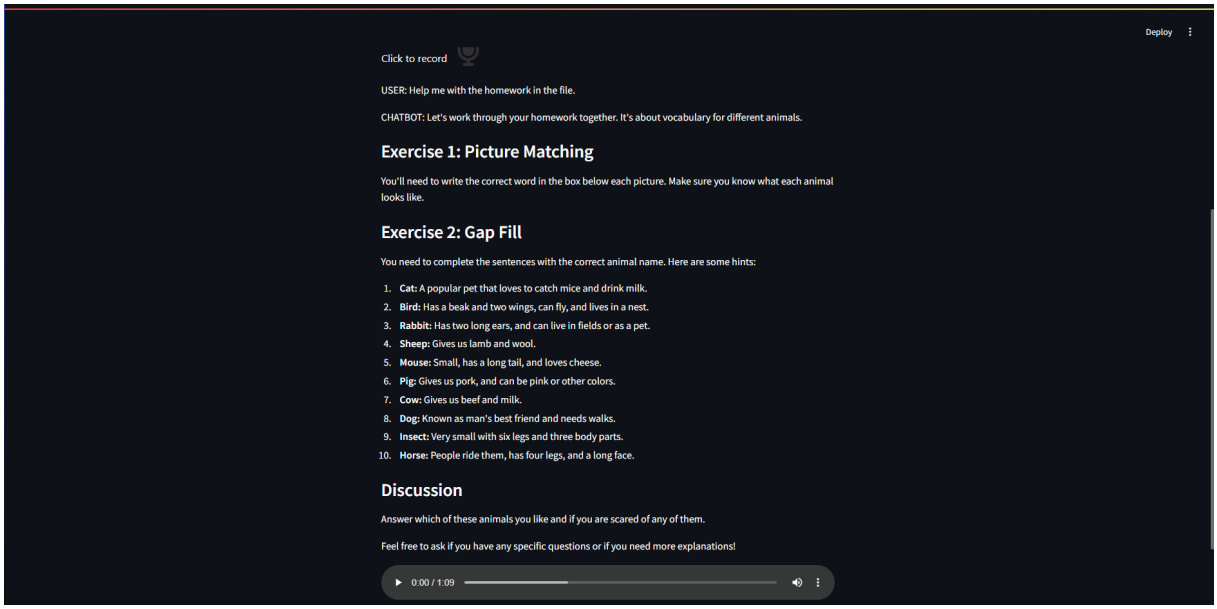
1. A _____ is a popular pet. They love to catch mice and drink milk.
2. A _____ has a beak and two wings. It can fly. It lives in a nest and lays eggs.
3. A _____ has two long ears. It is small to medium-sized. It can live in the fields or as a pet.
4. A _____ has four legs. It gives us lamb to eat and wool to make our clothes.
5. A _____ has a long tail and scares some people. It loves to eat cheese. It doesn't like cats!
6. A _____ has four legs and looks fat. It gives us pork to eat. It can be pink or other colours like black, white and brown.
7. A _____ has four legs and a long tail. They give us beef to eat and milk to drink.
8. A _____ is a popular pet. People say it is man's best friend. It needs to go for walks.
9. An _____ is very small. It normally has six legs and its body has three parts. Some people are scared of them.
10. A _____ has four legs, a long tail and a long face. People ride them.

Discussion
Which of these animals do you like? Are you scared of any of these animals?

Fonte: (BRITISH COUNCIL, 2024)

A Figura 19 apresenta a resposta gerada ao *input* do PDF, reconhecendo o conteúdo das atividades e sugerindo uma continuação para a conversa, incentivando a prática no mesmo contexto apresentado.

Figura 19. Resposta gerada.



Fonte: Elaborado pelo Autor

4.2 CUSTO E ACESSIBILIDADE

Para avaliar o custo de operação do *chatbot*, foram considerados os preços das APIs em dólares utilizadas e uma sessão média de uma hora de conversa, incluindo a utilização de funções de reconhecimento de áudio, conversão de texto em fala, processamento multimodal e conversação, levando em consideração o *pricing* de *tokens* da OpenAI (OPENAI, s.d.[c]).

4.2.1 Como os *tokens* são contabilizados

As APIs da OpenAI, como GPT e Whisper, contabilizam *tokens* com base nas unidades de entrada (texto, comandos, etc.) e saída (respostas geradas) que o modelo processa. A contagem de *tokens* inclui tanto palavras quanto pontuações e espaços, e é uma maneira de medir o processamento computacional necessário para responder a uma solicitação. Para o GPT-4, por exemplo, cada chamada da API contabiliza *tokens* para a entrada do usuário (prompt) e para a resposta gerada, apontando a estimativa na Tabela 1.

Tabela 1 – Estimativa de Custos de Processamento de Dados em Dólares

Recurso	**Custo por Unidade**	**Quantidade Estimada**	***tokens* Processados**	**Custo Total Estimado**
GPT-4o (Entrada)	\$2.50 por 1M *tokens* de entrada	70,000 *tokens*	17,500 palavras	\$0.175
GPT-4o (Saída)	\$10.00 por 1M *tokens* de saída	70,000 *tokens*	17,500 palavras	\$0.70
Whisper (Áudio)	\$0.006 por minuto	30 minutos		\$0.18
TTS (Texto para Fala)	\$15.00 por 1M caracteres	280,000 caracteres	56,000 palavras convertidas em fala	\$4.20
File Search (Vetorização)	\$0.10 por GB de armazenamento diário	1 GB (incluso)	Armazenamento diário	\$0.00
Total				**\$5.255**

4.2.1.1 Detalhamento e Justificativa de Cada Recurso

1. GPT-4o (Entrada e Saída)

a) Entrada (*input tokens*): Contabiliza *tokens* a partir do prompt enviado para a API, que inclui a pergunta ou instrução que o usuário fornece ao *chatbot*.

b) Saída (Output *tokens*): Contabiliza *tokens* a partir da resposta gerada pelo modelo, que corresponde ao texto que o modelo retorna ao usuário.

c) Média de *tokens* Processados: Para uma conversa de uma hora, foi considerado que o usuário troca em média 30 mensagens, com 50 *tokens* (cerca de 12 a 15 palavras) por mensagem de entrada e 75 *tokens* por resposta do *chatbot*.

d) Total Estimado: Com 70,000 *tokens* para entrada e 70,000 *tokens* para saída, temos aproximadamente 17,500 palavras processadas.

e) Justificativa de Escolha: Esse valor é suficiente para uma conversa detalhada e prolongada, permitindo ao usuário explorar tópicos variados e aprofundar-se em diferentes contextos de prática de idiomas. O custo para entrada e saída é, portanto, adequado para uma experiência de aprendizado rica e sem interrupções.

2. *Whisper* (Áudio)

a) Média de Minutos Processados: Estimamos que o usuário envie áudios por metade da hora conversada, o que resulta em 30 minutos de áudio processados.

b) Total Estimado: 30 minutos de áudio processados pelo Whisper.

c) Justificativa de Escolha: A transcrição em tempo real oferece uma experiência fluida, permitindo ao usuário praticar a pronúncia e receber respostas instantâneas.

3. TTS (Texto para Fala)

a) Média de Caracteres Processados: Como cada resposta do *chatbot* é convertida em áudio, estimamos 280,000 caracteres para 70,000 *tokens* de saída (56,000 palavras).

b) Total Estimado: Para uma hora de conversa, o TTS geraria voz para todas as respostas do *chatbot*, permitindo ao usuário ouvir as respostas.

c) Justificativa de Escolha: O uso do TTS aprimora a experiência, permitindo prática auditiva. Embora o custo seja relativamente alto, ele oferece uma experiência mais completa para o aprendizado de idiomas, contribuindo para o desenvolvimento da compreensão auditiva e da pronúncia.

4. File Search (Vetorização)

a) Média de Armazenamento Utilizado: O sistema pode armazenar até 1 GB de dados gratuitamente por dia para vetorização de arquivos (como PDFs).

b) Total Estimado: Com 1 GB de armazenamento incluído, o custo adicional é zero.

c) Justificativa de Escolha: O armazenamento diário gratuito é suficiente para os dados utilizados em uma hora de conversa.

O custo médio por sessão de uma hora para o *chatbot* foi estimado em cerca de $5.255. Em comparação com métodos tradicionais de prática de línguas, como aulas presenciais ou com tutores, que podem custar de $10 a $50 por sessão, o *chatbot* apresenta um custo-benefício significativo. Além disso, oferece flexibilidade e acessibilidade, permitindo ao usuário praticar a qualquer momento, sem a necessidade de coordenação com um professor.

4.3 COMPARATIVO COM OUTRAS SOLUÇÕES

A Tabela 2 é a comparação entre as funcionalidades do *chatbot* desenvolvido e outras soluções de aprendizado de idiomas populares, como *Duolingo*, *Langbot*, *Replika* e *Cambly*.

Tabela 2 – Comparação em Dólares de Recursos de Chatbots e Plataformas de Aprendizado de Idiomas

Recurso	***chatbot* do TCC**	**Duolingo**	**Langbot**	**Replika**	**Cambly**
Conversação em Várias Línguas	Sim	Limitado	Sim	Limitado	Sim
Integração Multimodal	Sim (imagens, PDFs)	Não	Não	Não	Sim (com tutores)
Conversação Natural	Alta precisão	Moderada	Moderada	Alta precisão	Alta precisão (com tutores)
Conversação Técnica	Sim	Limitado	Não	Limitado	Sim (com tutores)
Disponibilidade de Tutores	Não	Não	Não	Não	Sim
Custo por Sessão	$5.255	Gratuito (opções pagas)	Gratuito (opções pagas)	$7.99/mês	$10-$30/hora
Feedback Personalizado	Sim	Sim	Sim	Sim	Sim
Específico para prática de idiomas	Sim	Sim	Parcialmente	Não	Sim

Comparado a outras soluções, o *chatbot* desenvolvido oferece uma combinação única de funcionalidade multimodal e precisão na conversação técnica, o que o torna uma excelente ferramenta para aprendizes que buscam maior personalização e praticidade.

4.4 DISCUSSÃO

O desenvolvimento deste *chatbot* multimodal para praticar idiomas demonstrou o potencial das IAs generativas como ferramentas eficazes para o aprendizado e a prática de línguas. Ao integrar diferentes funcionalidades, como a API de linguagem da OpenAI, análise de imagens e PDFs, e reconhecimento de áudio, o sistema alcançou um nível de flexibilidade e adaptabilidade difícil de ser encontrado em outras ferramentas de aprendizado de idiomas disponíveis no mercado.

A adaptação do *chatbot* para diversos idiomas e contextos foi um dos principais pontos fortes do sistema, com resultados mostrando que o *chatbot* se ajusta bem a conversas casuais, temas específicos e até mesmo a tópicos técnicos, proporcionando uma experiência robusta e imersiva para o usuário. No entanto, desafios como a impossibilidade de aplicar uma real multimodalidade em um único agente com OpenAI *Assistants* API Beta indicam áreas que poderiam ser aprimoradas com novas versões das APIs.

Em termos de custo, o sistema demonstrou um bom custo-benefício, com um valor acessível para usuários que desejam praticar idiomas de forma personalizada e fácil. Comparado a métodos tradicionais, como aulas com tutores ou aulas presenciais, o *chatbot* oferece uma alternativa econômica e disponível sob demanda. Mesmo com o custo das APIs, especialmente para conversação e multimodalidade, o *chatbot* é uma solução financeiramente viável para a prática de idiomas, especialmente para usuários que utilizam o sistema com frequência.

Em comparação com outras ferramentas de prática de idiomas, como *Duolingo*, *Langbot*, *Replika*, *Cambly* e *Preply*, o *chatbot* desenvolvido neste trabalho se destacou pela capacidade multimodal e pelo uso de inteligência artificial avançada para conversação natural, aplicada em assuntos específicos, da vontade do usuário. Enquanto essas ferramentas oferecem funcionalidades específicas, o sistema aqui desenvolvido é uma combinação única de aprendizado de idiomas com capacidade de análise de mídia, o que é especialmente útil para aprendizes que buscam não apenas fluência oral, mas também interpretação visual e textual em contextos reais e diversos.

4.4.1 Desafios e Soluções

Durante o desenvolvimento do *chatbot*, surgiram diversos desafios, especialmente relacionados à gestão de diferentes modalidades de entrada e à resposta coerente e contextualizada em múltiplos idiomas. A seguir são listados os principais desafios enfrentados e as soluções adotadas para cada um deles.

4.4.1.1 Desafios na Identificação e Geração de Respostas Multilíngues

Desafio: Um dos maiores desafios foi garantir que o *chatbot* detectasse com precisão o idioma da entrada e gerasse respostas contextualmente adequadas em cada língua, mantendo o tom e o nível de formalidade apropriados, sem desviar do idioma selecionado.

Solução: Foi utilizado *prompts* dinâmicos que ajustam automaticamente o idioma com base na entrada do usuário. Essa abordagem permitiu um ajuste fino no nível de formalidade e clareza das respostas, conforme o idioma e o nível de fluência do usuário.

4.4.1.2 Desafios na Integração Multimodal

Desafio: Integrar diferentes tipos de entrada – texto, áudio, imagem e PDF – de forma que o *chatbot* compreendesse o conteúdo e respondesse adequadamente.

Solução: Para imagens, foi utilizada a API de visão computacional da OpenAI para interpretar e identificar elementos específicos, o que facilitou a resposta a perguntas sobre o conteúdo visual considerando que a *Assisitants* API Beta ainda não possui ferramentas de interpretação de imagens. No caso dos PDFs, foi utilizada a ferramenta de file *search* dos assistentes. Nesse contexto, múltiplas funções foram desenvolvidas para solucionar cada arquivo.

4.4.2 Limitações e Possíveis Melhorias

Embora o *chatbot* ofereça uma experiência rica e adaptável, algumas limitações técnicas ainda permanecem. A integração multimodal, embora funcional, poderia ser aprimorada para lidar com arquivos mais complexos e grandes volumes de dados de maneira ainda mais eficiente e de forma integrada.

Em termos de funcionalidades futuras, uma possibilidade seria a inclusão de ferramentas de tradução automática em tempo real para permitir que o *chatbot* responda em múltiplos idiomas durante uma única conversa.

Outra melhoria potencial envolve o uso de modelos mais avançados de análise de imagem, que poderiam oferecer uma compreensão mais profunda de conteúdos visuais complexos, permitindo o uso de apenas um agente para qualquer interação.

Além disso, é importante realizar mais testes, tanto qualitativos quanto quantitativos, para melhor avaliar o funcionamento do sistema desenvolvido. Testes qualitativos, como entrevistas com os usuários e análise de *feedback*, podem ajudar a entender como as pessoas estão interagindo com o *chatbot* e identificar pontos que precisam ser melhorados, como a interface ou a forma de comunicação. Já os testes quantitativos, como a coleta de dados sobre o uso e o desempenho do sistema, podem ajudar a medir de forma mais precisa a eficácia da integração multimodal e o quanto o *chatbot* realmente ajuda na prática de línguas, também verificando se o *chatbot* realmente consegue acompanhar os diferentes níveis de fluência que o usuário informa ajustando adequadamente sua resposta e complexidade da linguagem conforme o nível. Esses testes são fundamentais para garantir que o sistema seja escalável e funcione bem para diferentes tipos de usuários e em diversos contextos.

5 CONCLUSÃO

Este sistema alcançou os objetivos propostos, desenvolvendo um *chatbot* conversacional multimodal que permite aos usuários praticar idiomas de maneira integrada e personalizada. A combinação das tecnologias da OpenAI, incluindo processamento de linguagem natural, reconhecimento de voz e análise de mídia, demonstrou-se eficaz para criar um ambiente de aprendizado interativo e imersivo.

Em termos de contribuições para a área de aprendizado de idiomas e inteligência artificial, este sistema mostra como IAs generativas podem enriquecer o processo educacional, proporcionando interações mais ricas e diversificadas. A abordagem multimodal aqui explorada abre novas possibilidades para o ensino de línguas, onde o uso de diversos formatos de mídia não só auxilia na prática de conversação, mas também na compreensão de conteúdos visuais e textuais, aumentando a compreensão e retenção de conceitos complexos.

Em conclusão, o *chatbot* desenvolvido apresenta-se como uma solução inovadora e viável para o aprendizado de idiomas, com grande potencial de expansão e aplicabilidade em contextos educacionais e profissionais. Com melhorias e adaptações futuras, este sistema pode não só atender a aprendizes individuais, mas também ser adaptado para instituições e programas de ensino que buscam incorporar a inteligência artificial como ferramenta auxiliar de aprendizado e prática de línguas.

REFERÊNCIAS

ASSEMBLYAI. **Fine-Tuning Transformers for NLP**. [*S.l.*: *s.n.*], 2021. Disponível em: https://www.assemblyai.com/blog/fine-tuning-transformers-for-nlp/. Acesso em: 10 nov. 2024.

ASSEMBLYAI. **Fine-Tuning Transformers for NLP**. [*S.l.*: *s.n.*], 2023. Disponível em: https://www.youtube.com/watch?v=pBBe1pk8hf4. Acesso em: 10 nov. 2024.

BRITISH COUNCIL. **Animals - Exercises**. [*S.l.*: *s.n.*], 2024. Disponível em: https://learnenglishteens.britishcouncil.org/sites/teens/files/animals_-_exercises.pdf. Acesso em: 10 nov. 2024.

CAMBLY. **Preços Cambly**. [*S.l.*: *s.n.*]. Disponível em: https://www.cambly.com/en/student/subscribe. Acesso em: 10 nov. 2024.

DE LA VALL, Roxana Rebolledo Font; ARAYA, Fabián González. Exploring the benefits and challenges of AI-language learning tools. **International Journal of Social Sciences and Humanities Invention**, v. 10, n. 01, p. 7569–7576, 2023. Acesso em: 13 nov. 2024.

DISSANAYAKA, Pinil. **Unlocking the Power of Retrieval-Augmented Generation (RAG)**. [*S.l.*: *s.n.*], 2023. Disponível em: https://www.linkedin.com/pulse/unlocking-power-retrieval-augmented-generation-rag-pinil-dissanayaka-fkbac/. Acesso em: 10 nov. 2024.

ELANGUAGES. **Months in English**. [*S.l.*: *s.n.*], 2020. Disponível em: https://www.youtube.com/watch?v=xdQOCy8FcEQ. Acesso em: 18 nov. 2024.

LANGBOT. **Langbot - Language Learning Bot**. [*S.l.*: *s.n.*]. Disponível em: https://www.langbot.io. Acesso em: 10 nov. 2024.

OPENAI. **OpenAI API Documentation**. [*S.l.*]. Disponível em: https://platform.openai.com/docs. Acesso em: 10 nov. 2024.

OPENAI. **OpenAI Whisper API**. [*S.l.*]. Disponível em: https://openai.com/index/whisper/. Acesso em: 10 nov. 2024.

OPENAI. **Pricing OpenAI API**. [*S.l.*: *s.n.*]. Disponível em: https://openai.com/api/pricing/. Acesso em: 10 nov. 2024.

PETROVIĆ, Jasna; JOVANOVIĆ, Mladjan. The Role of Chatbots in Foreign Language Learning: The Present Situation and the Future Outlook. *In*: ARTIFICIAL Intelligence: Theory and Applications. [*S.l.*]: Springer, Cham, 2021. v. 973. (Studies in Computational Intelligence). Disponível em: https://www.researchgate.net/publication/353277729. Acesso em: 10 nov. 2024. P. 235–248.

PREPLY. **Professores de Inglês Online**. [*S.l.*: *s.n.*], 2024. Disponível em: https://preply.com/pt/online/professores--inglÃłs. Acesso em: 10 nov. 2024.

REPLIKA. **Replika - Virtual Companion**. [*S.l.*: *s.n.*]. Disponível em: https://replika.com. Acesso em: 10 nov. 2024.

RUSMIYANTO, Rusmiyanto; AL., et. The role of artificial intelligence (AI) in developing English language learner's communication skills. **Journal on Education**, v. 6, n. 1, p. 750–757, 2023. Acesso em: 13 nov. 2024.

STREAMLIT. **Streamlit**. [*S.l.*], 2024. Disponível em: https://streamlit.io. Acesso em: 10 nov. 2024.

STREAMLIT. **Streamlit Framework Documentation**. [*S.l.*], 2024. Disponível em: https://docs.streamlit.io. Acesso em: 10 nov. 2024.

VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N.; KAISER, Łukasz; POLOSUKHIN, Illia. Attention is All You Need. *In*: PROCEEDINGS of the 31st International Conference on Neural Information Processing Systems (NIPS 2017). [*S.l.*]: Curran Associates, Inc., 2017. P. 6000–6010.

WOO, Jin Ha; CHOI, Heeyoul. Systematic review for AI-based language learning tools. **arXiv preprint arXiv:2111.04455**, 2021.

APÊNDICE A – README.md

Visando facilitar o uso do sistema, a seguir é apresentado o README.md do código desenvolvido.

Project Description

This project is part of the undergraduate thesis that explores the application of generative AI in language learning. The main objective is to develop a multimodal conversational chatbot using the OpenAI API, capable of interacting with users intuitively and effectively to facilitate language practice.

Technologies Used

- *OpenAI API: For generating intelligent responses and processing multimodal data.*
- *Streamlit: Framework used to build the web application frontend.*
- *Python: Primary programming language.*

Features

- *Multimodal support: The chatbot processes and responds to various input types, such as audio, text, and images (when available).*
- *Automatic translation: Integrated functionality for translating conversations between different languages.*
- *Real-time feedback: The chatbot provides instant feedback during language practice.*

Project Structure

This repository contains two main components:

src: That contains the source files:

1.1. Backend (backend.py): Responsible for the chatbot's logic, OpenAI API integration, and multimodal data processing (audio, text, image).

1.2. Frontend (frontend.py): Built with Streamlit, it provides a user-friendly interface for interacting with the chatbot.

data: Contains the data files used in the chatbots:

2.1. MP3 files to store the audio files used by the user input (audio.mp3) and by the chatbot output (response_audio.mp3).

2.2. JPG file to store the image (image.jpg) file from the image based chatbot.

2.3. PDF file to store the pdf (pdf.pdf) file used in the PDF based chatbot.

Requirements

1. *OpenAI API Key*

How to Run the Project

1: Install requirements

pip install -r requirements.txt

2: run app logic
python backend.py
3: run frontend
streamlit run frontend.py

APÊNDICE B – Backend.py

CÓDIGO DO BACKEND.PY

```
from openai import OpenAI
import requests

#Base Functions
def set_key(api_key):
    return (OpenAI(api_key = api_key))

def text_to_audio(text, audio, client):
    response = client.audio.speech.create(
        model="tts-1",
        voice="echo",
        input=text
    )
    response.stream_to_file(audio)

def audio_to_text(audio_file, client):
    with open (audio_file, 'rb') as audio_file:
        response = client.audio.transcriptions.create(
            model="whisper-1",
            file=audio_file
        )
        return response.text

#Open Convo Assistants
def op_set_assistant(user_info, client):
    assistant = client.beta.assistants.create(
        name="Personal Conversational Teacher",
        instructions=f"You are a personal conversational teacher, your
    goal is to keep a conversation going and correcting eventual language
     mistakes. About your student, Name:{user_info['name']}, Pronouns: {
    user_info['pronouns']}, Studying the language:{user_info['
    study_language']} at level {user_info['level']}. ONLY RESPOND AND
    SPEAK in the language the student is trying to practice and learn. DO
     NOT speak in any language apart from the studying language.",
        model="gpt-4o"
    )
    thread = client.beta.threads.create()
    thread_id = thread.id
    assistant_id = assistant.id
    return (thread_id, assistant_id)

def op_get_assistant_response(user_info,text,client,thread_id,
    assistant_id):
```

```
37      message = client.beta.threads.messages.create(
38          thread_id=thread_id,
39          role="user",
40          content=text
41      )
42      run = client.beta.threads.runs.create(
43          thread_id=thread_id,
44          assistant_id=assistant_id,
45          instructions= f"ONLY SPEAK AND RESPOND IN {user_info} LANGUAGE.
    Keep a conversation. Correct eventual language mistakes. Try to teach
     something new in the language the student is trying to practice and
    learn."
46      )
47
48      import time
49      while run.status != 'completed':
50          time.sleep(1)
51          run = client.beta.threads.runs.retrieve(
52              thread_id=thread_id,
53              run_id=run.id
54          )
55
56      messages = client.beta.threads.messages.list(
57          thread_id=thread_id
58      )
59      return messages.data[0].content[0].text.value
60
61  #Img Convo Assistants
62  def img_get_assistant_response(user_info, text, img, client):
63      headers = {
64      "Content-Type": "application/json",
65      "Authorization": f"Bearer {client}"
66      }
67
68      payload = {
69      "model": "gpt-4o-mini",
70      "messages": [
71          {"role": "system",
72           "content": f"Use the image as context for the conversation. You
     are a personal conversational teacher, your goal is to keep a
    conversation going and correcting eventual language mistakes. About
    your student, Name:{user_info['name']}, Pronouns: {user_info['
    pronouns']}, Studying the language:{user_info['study_language']} at
    level {user_info['level']}. ONLY RESPOND AND SPEAK in the language
    the student is trying to practice and learn. DO NOT speak in any
    language apart from the studying language. Try to teach something new
     in the language the student is trying to practice and learn."
```

```
73          },
74          {
75          "role": "user",
76          "content": [
77                  {
78                  "type": "text",
79                  "text": text
80                  },
81                  {
82                  "type": "image_url",
83                  "image_url": {
84                      "url": f"{img}"
85                  }
86                  }
87              ]
88          }
89      ],
90      "max_tokens": 3000
91      }
92
93      response = requests.post("https://api.openai.com/v1/chat/completions
        ", headers=headers, json=payload)
94
95      response_data = response.json()
96      return response_data['choices'][0]['message']['content']
97
98
99  #Pdf Convo Assistants
100 def pdf_set_assistant(user_info, client):
101     assistant = client.beta.assistants.create(
102         name="English Teacher",
103         instructions=f"You are a personal conversational teacher, your
        goal is to keep a conversation going and correcting eventual language
         mistakes. About your student, Name:{user_info['name']}, Pronouns: {
        user_info['pronouns']}, Studying the language:{user_info['
        study_language']} at level {user_info['level']}. ONLY RESPOND AND
        SPEAK in the language the student is trying to practice and learn. DO
         NOT speak in any language apart from the studying language.",
104         model="gpt-4o",
105         tools=[{"type": "file_search"}]
106     )
107     thread = client.beta.threads.create()
108     thread_id = thread.id
109     assistant_id = assistant.id
110     return (thread_id, assistant_id)
111
112 def pdf_get_assistant_response(user_info, file, text,client,thread_id,
```

```
    assistant_id):
113     message = client.beta.threads.messages.create(
114         thread_id=thread_id,
115         role="user",
116         content=text,
117         attachments=[{"file_id": file, "tools": [{"type": "file_search"
    }]}]
118     )
119     run = client.beta.threads.runs.create(
120         thread_id=thread_id,
121         assistant_id=assistant_id,
122         instructions= f"Use the attached file as context for the
    conversation. ONLY SPEAK AND RESPOND IN {user_info} LANGUAGE. Keep a
    conversation. Correct eventual language mistakes. Try to teach
    something new in the language the student is trying to practice and
    learn."
123     )
124
125     import time
126     while run.status != 'completed':
127         time.sleep(1)
128         run = client.beta.threads.runs.retrieve(
129             thread_id=thread_id,
130             run_id=run.id
131         )
132
133     messages = client.beta.threads.messages.list(
134         thread_id=thread_id
135     )
136     return messages.data[0].content[0].text.value
```

Listing B.1 – backend.py

APÊNDICE C – Frontend.py

CÓDIGO DO FRONTEND.PY

```python
import streamlit as st
from audio_recorder_streamlit import audio_recorder
import backend as backend
from PIL import Image
import base64

#Base Functions
def encode_image(image_path):
  with open(image_path, "rb") as image_file:
    return base64.b64encode(image_file.read()).decode('utf-8')

def user_input_form():
    with st.form("form"):
        name = st.text_input("Name")
        pronouns = st.text_input("Pronouns")
        study_language = st.text_input("Study Language")
        level = st.selectbox("Level of Knowledge", ["Beginner", "Intermediate", "Advanced"])

        submitted = st.form_submit_button("Start")

        if submitted:
            st.session_state["user_info"] = {
                "name": name,
                "pronouns": pronouns,
                "study_language": study_language,
                "level": level
            }
            st.success("Information saved!")

#Bot Functions
def open_convo_bot(user_info, client,thread_id, assistant_id):
    recorded_audio = audio_recorder()
    if recorded_audio:
        audio_file = 'data\\audio.mp3'
        with open(audio_file, 'wb') as f:
            f.write(recorded_audio)

        transcribed_text = backend.audio_to_text(audio_file,client)
        st.write(f"USER: {transcribed_text}")

        response_text = backend.op_get_assistant_response(user_info, transcribed_text,client,thread_id,assistant_id)
```

```
42          st.write(f"CHATBOT: {response_text}")
43
44          response_audio = 'data\\response_audio.mp3'
45          backend.text_to_audio(response_text, response_audio,client)
46          st.audio(response_audio)
47
48  def img_convo_bot(user_info,client,key):
49      if 'vbase64_image' not in st.session_state:
50          st.session_state.vbase64_image = None
51      if 'uploaded_file' not in st.session_state:
52          st.session_state.uploaded_file = None
53      if 'img_path' not in st.session_state:
54          st.session_state.img_path = None
55
56      if st.session_state.vbase64_image is None:
57          st.session_state.uploaded_file = st.file_uploader("Load an image
        (jpg) to start the conversation!", type=["jpg"])
58
59          if st.session_state.uploaded_file:
60              image = Image.open(st.session_state.uploaded_file)
61              st.session_state.img_path = "data\\image.jpg"
62              image.save(st.session_state.img_path)
63              st.session_state.vbase64_image = encode_image(st.
       session_state.img_path)
64              st.image(st.session_state.img_path, caption="Imagem Selected
       ", width=200)
65              st.success("Image loaded successfully!")
66
67      if st.session_state.vbase64_image is not None:
68          recorded_audio = audio_recorder()
69
70          if recorded_audio:
71              audio_file = 'data\\audio.mp3'
72              with open(audio_file, 'wb') as f:
73                  f.write(recorded_audio)
74
75              transcribed_text = backend.audio_to_text(audio_file, client)
76              st.write(f"USER: {transcribed_text}")
77
78              response_text = backend.img_get_assistant_response(user_info
       , transcribed_text, st.session_state.vbase64_image, key)
79              st.write(f"CHATBOT: {response_text}")
80
81              response_audio = 'data\\response_audio.mp3'
82              backend.text_to_audio(response_text, response_audio, client)
83              st.audio(response_audio)
84
```

```
85 def pdf_convo_bot(user_info, client,thread_id,assistant_id):
86     if 'set_pdf' not in st.session_state:
87         st.session_state.set_pdf = None
88     if 'uploaded_pdf' not in st.session_state:
89         st.session_state.uploaded_pdf = None
90     if 'file_id' not in st.session_state:
91         st.session_state.file_id = None
92 
93     if st.session_state.set_pdf is None:
94         st.session_state.uploaded_pdf = st.file_uploader("Load a PDF to
    start the conversation!", type=["pdf"])
95         if st.session_state.uploaded_pdf:
96             with open("data\\pdf.pdf", "wb") as f:
97                 f.write(st.session_state.uploaded_pdf.getbuffer())
98                 st.success("Pdf loaded successfully!")
99 
100             message_file = client.files.create(
101                 file=open("data\\pdf.pdf", "rb"), purpose="assistants"
102             )
103             st.session_state.file_id = message_file.id
104             st.session_state.set_pdf = True
105 
106     if st.session_state.set_pdf is not None and st.session_state.file_id
     is not None:
107         recorded_audio = audio_recorder()
108         if recorded_audio:
109             audio_file = 'data\\audio.mp3'
110             with open(audio_file, 'wb') as f:
111                 f.write(recorded_audio)
112 
113             transcribed_text = backend.audio_to_text(audio_file, client)
114             st.write(f"USER: {transcribed_text}")
115 
116             response_text = backend.pdf_get_assistant_response(user_info
    , st.session_state.file_id, transcribed_text, client, thread_id,
    assistant_id)
117             st.write(f"CHATBOT: {response_text}")
118 
119             response_audio = 'data\\response_audio.mp3'
120             backend.text_to_audio(response_text, response_audio, client)
121             st.audio(response_audio)
122 
123 #Main App
124 def main():
125     if 'bot_selecionado' not in st.session_state:
126         st.session_state.bot_selecionado = None
127     if 'api_key' not in st.session_state:
```

```
128         st.session_state.api_key = None
129     if 'client' not in st.session_state:
130         st.session_state.client = None
131 
132     if st.session_state.bot_selecionado is None:
133         st.session_state.api_key = st.text_input("Type your OpenAI API
    Key", type="password")
134         if st.session_state.api_key:
135             st.session_state.client = backend.set_key(st.session_state.
    api_key)
136             st.success("API Key validated!")
137             if 'user_info' not in st.session_state:
138                 user_input_form()
139             else:
140                 user_info = st.session_state["user_info"]
141                 st.write(f"Welcome, {user_info['name']}!")
142                 st.write(f"Pronouns: {user_info['pronouns']}")
143                 st.write(f"Study Language: {user_info['study_language']}
     ({user_info['level']})")
144 
145                 opcao = st.radio(
146                     "Select a bot:",
147                     ('Open Conversation', 'Image Based Conversation', '
    PDF Based Conversation')
148                 )
149                 if opcao == 'Open Conversation':
150                     thread_id, assistant_id = backend.op_set_assistant(
    user_info, st.session_state.client)
151                     st.session_state.bot_selecionado = open_convo_bot(
    user_info['study_language'], st.session_state.client,thread_id,
    assistant_id)
152                 elif opcao == 'Image Based Conversation':
153                     st.session_state.bot_selecionado = img_convo_bot(
    user_info, st.session_state.client, st.session_state.api_key)
154                 elif opcao == 'PDF Based Conversation':
155                     thread_id, assistant_id = backend.pdf_set_assistant(
    user_info, st.session_state.client)
156                     st.session_state.bot_selecionado = pdf_convo_bot(
    user_info['study_language'], st.session_state.client,thread_id,
    assistant_id)
157 
158     else:
159         if st.session_state.bot_selecionado == "Open Conversation":
160             open_convo_bot(user_info['study_language'], st.session_state
    .client,thread_id, assistant_id)
161         elif st.session_state.bot_selecionado == "Image Based
    Conversation":
```

```
162             img_convo_bot(user_info, st.session_state.client,st.
    session_state.api_key)
163         elif st.session_state.bot_selecionado == "PDF Based Conversation
    ":
164             pdf_convo_bot(user_info['study_language'],st.session_state.
    client,thread_id, assistant_id)
165 
166         if st.button("Change Bot"):
167             st.session_state.bot_selecionado = None
168 
169 st.title("Language Practice Chatbot")
170 
171 if __name__ == '__main__':
172     main()
```

Listing C.1 – backend.py