COIMBRA BUSINESS SCHOOL

iscac Politécnico de Coimbra

100 ANOS

since 1921

Chatbots baseados em IA: Um protótipo para uma IES

Bingnan Zheng

ISCAC | 2024

COIMBRA BUSINESS SCHOOL

iscac Politécnico de Coimbra

100 ANOS

Bingnan Zheng

Chatbots baseados em Inteligência Artificial: Um protótipo para uma instituição de ensino superior

Coimbra, julho de 2024

COIMBRA BUSINESS SCHOOL

iscac 100 ANOS
Politécnico de Coimbra

Bingnan Zheng

Chatbots baseados em Inteligência Artificial: Um protótipo para uma instituição de ensino superior

Trabalho de projeto submetido ao Instituto Superior de Contabilidade e Administração de Coimbra para cumprimento dos requisitos necessários à obtenção do grau de **Mestre em Análise de Dados e Sistemas de Apoio à Decisão**, realizado sob a orientação do Professor Doutor Fernando Paulo dos Santos Rodrigues Belfo e coorientação do Professor Doutor António Rui Trigo Ribeiro.

Coimbra, julho de 2024

TERMO DE RESPONSABILIDADE

Declaro ser a autora deste projeto, que constitui um trabalho original e inédito, que nunca foi submetido a outra Instituição de ensino superior para obtenção de um grau académico ou outra habilitação. Atesto ainda que todas as citações estão devidamente identificadas e que tenho consciência de que o plágio constitui uma grave falta de ética, que poderá resultar na anulação do presente projeto.

AGRADECIMENTOS

Quero agradecer toda as pessoas que me apoiaram na realização deste projeto.

Primeiro, agradeço ao Professor Belfo por aceitar a minha proposta e ser o meu orientador. Sem ele, este projeto não existiria.

Ao Professor Trigo, agradeço pela ajuda técnica, guiando-me na criação do *chatbot* e na resolução dos problemas que surgiram. Sem a sua ajuda, o projeto não teria sido concluído tão rapidamente e com sucesso.

À Professora Joana, agradeço pelo incentivo e conselhos, que me ajudaram a seguir o caminho que escolhi.

Agradeço à minha mãe pelo apoio em várias frentes.

Por fim, agradeço a todos os que me apoiaram e me deram força para concluir este estudo com sucesso.

RESUMO

Os *websites* institucionais das universidades e outras instituições de ensino superior contêm uma vasta gama de informações essenciais para alunos, docentes e outras partes interessadas. No entanto, a navegação e a procura eficientes por informações específicas podem ser desafiadoras devido à complexidade e ao volume de conteúdo. Isso frequentemente resulta em frustração dos utilizadores e um aumento no tempo necessário para encontrar informações relevantes. Esta também foi a realidade identificada relativa ao *website* institucional do Instituto Superior de Contabilidade e Administração de Coimbra, tendo aí sido detetada uma clara necessidade de uma solução que pudesse simplificar a interação com os utilizadores e fornecer-lhes respostas rápidas e precisas às suas consultas.

Surgiu assim a ideia de desenvolver um *chatbot* inteligente para facilitar a consulta ao website institucional do ISCAC. Este *chatbot* integrou três tecnologias: 1) Messenger do *Facebook*, uma plataforma com ampla popularidade e acessibilidade, que permite que os utilizadores interajam com o *chatbot* de forma fácil e conveniente diretamente através de mensagens; 2) ChatGPT da *OpenAI*, um grande modelo de linguagem que permite o processamento de língua natural, que permite tanto compreender as perguntas dos utilizadores como o conteúdo do website respondendo de forma precisa às questões dos utilizadores; 3) um *middleware*, contruído em *Python* que permite a comunicação entre o Messenger e o ChatGPT. O desenvolvimento da solução passou pelas fases tradicionais de desenvolvimento de software, nomeadamente: análise de requisitos; conceção do *chatbot*; implementação, integração dos diferentes componentes e testes, com dados reais extraídos do *website*.

A implementação do *chatbot* poderá gerar vários benefícios para a instituição e para os seus utilizadores, designadamente: acesso rápido às informações; facilidade de utilização; redução da frustração do utilizador; e disponibilidade (capacidade de operar 24/7). O desenvolvimento do *chatbot* utilizando a API do Messenger do *Facebook* e o ChatGPT da *OpenAI* demonstrou ser uma solução eficaz para facilitar a consulta ao website institucional.

Palavras-chave: Apoio ao Cliente; Inteligência Artificial; *Chatbot*; *Facebook*; *OpenAI*.

ABSTRACT

The institutional websites of universities and other higher education institutions contain a wide range of essential information for students, lecturers and other stakeholders. However, navigating and efficiently searching for specific information can be challenging due to the complexity and volume of content. This often results in user frustration and an increase in the time needed to find relevant information. This was also the reality identified in relation to the institutional website of the Instituto Superior de Contabilidade e Administração de Coimbra, where a clear need was detected for a solution that could simplify interaction with users and provide them with quick and accurate answers to their queries.

This gave rise to the idea of developing an intelligent chatbot to facilitate consultation of ISCAC's institutional website. This chatbot integrated three technologies: 1) *Facebook* Messenger, a platform with widespread popularity and accessibility, which allows users to interact with the chatbot easily and conveniently directly through messages; 2) *OpenAI*'s ChatGPT, a large language model that enables natural language processing, which allows both understanding of user questions and website content, responding accurately to user queries; 3) a middleware, built in *Python* that allows communication between Messenger and ChatGPT. The development of the solution went through the traditional phases of software development, namely: requirements analysis; chatbot design; implementation, integration of different components; and testing, with real data extracted from the website.

The implementation of the chatbot can generate several benefits for the institution and its users, namely: quick access to information; ease of use; reduction of user frustration; and availability (ability to operate 24/7). The development of the chatbot using the *Facebook* Messenger API and *OpenAI*'s ChatGPT proved to be an effective solution for facilitating consultation of the institutional website.

Keywords: Customer Support; Artificial Intelligence; Chatbot; *Facebook*; *OpenAI*

ÍNDICE GERAL

ÍNDICE DE FIGURAS

Lista de abreviaturas, acrónimos e siglas

AIML ..... *Artificial Intelligence Markup Language*

ANN..... *Artificial Neuronal Network*

API..... *Application Programmable Interfaces*

B2C ..... *Business to Customer*

CNN..... *Convolutional Neuronal Network*

DINA ..... *Dinus Intelligent Assistance*

EANN ..... *Event Adversarial Neuronal Network*

GPT..... *Generative Pre-trained Transformer*

GPU ..... *Graphics Processing Unit*

IES ..... Instituição de Ensino Superior

ISCAC..... Instituto Superior de Contabilidade e Administração de Coimbra

LLM..... *Large Language Model*

NLP..... *Natural Language Processing*

NLU ..... *Natural Language Understanding*

NLG ..... *Natural Language Generation*

O2O..... *Online to Offline*

SGA ..... Serviço de Gestão Académica

INTRODUÇÃO

O *chatbot*, como produto da evolução dos sistemas de informação, tem conquistado grande aplicação em diversos domínios. Nos últimos anos, o seu crescimento tem sido notável, com cada vez mais setores a adotar essa ferramenta para otimizar a eficiência e reduzir a necessidade de mão de obra humana. Um dos seus mais notáveis exemplos é o ChatGPT, um *chatbot* desenvolvido pela OpenAI e lançado em 30 de novembro de 2022, utilizado nas mais diferentes áreas, como por exemplo, a contabilidade (Dias et al., 2023), a área nuclear da escola em que este trabalho foi desenvolvido, o Instituto Superior de Contabilidade e Administração de Coimbra (ISCAC). Na área académica, a investigação sobre a aplicação de *chatbots* em faculdades e universidades tem se tornado cada vez mais frequente. Eles são estudados em várias funções, como em ensino e aprendizagem, administração, assessoria e pesquisa e desenvolvimento (Okonkwo & Ade-Ibijola, 2021).

A motivação que impulsionou a pretender fazer o presente projeto prende-se aos múltiplos benefícios que este poderá conseguir trazer a uma organização como uma instituição de ensino superior (IES). De entre estes benefícios destacam-se uma eventual melhor gestão dos recursos humanos, pela poupança que poderá existir com o tempo que estes deixam de gastar em respostas tradicionais, a maior atração de potenciais estudantes ou pela satisfação e integração dos estudantes atuais, através de uma mais rápida e completa resposta a um conjunto de questões habituais. Um *chatbot* pode melhorar a eficiência dos serviços académicos, proporcionando respostas rápidas e precisas aos utilizadores, otimizando o processo de atendimento, libertando os colaboradores de tarefas repetidas para atividades mais estratégicas e complexas, contribuindo para a eficiência e produtividade no trabalho.

O presente projeto tem como objetivos principais a elaboração de uma revisão de literatura sobre *chatbots* baseados em IA, a apresentação de algumas das atuais soluções e abordagens no mercado e a construção de um protótipo de *chatbot* baseado em IA para ISCAC. O objetivo deste protótipo de *chatbot* é atuar como assistente do gabinete dos serviços académicos, realizando tarefas de prestação de informações aos utilizadores. Este projeto visa assim desenvolver um *chatbot* que seja capaz de responder a um

conjunto de perguntas dos utilizadores relacionadas com questões habitualmente respondidas através da consulta do *website* institucional do ISCAC.

O presente relatório está estruturado em nove capítulos.

Após uma introdução, o documento apresenta, no seu primeiro capítulo, o enquadramento teórico, oferecendo uma revisão dos principais trabalhos relacionados com a área de estudo, respetivamente, sobre IA, *chatbots* e técnicas e tecnologias de funcionamento.

Em seguida, no segundo capítulo, é apresentada a metodologia a ser aplicada no projeto, detalhando-se os procedimentos e abordagens a serem adotados para alcançar os objetivos propostos. A metodologia utilizada é a investigação-ação proposta por Baskerville (1999). Neste capítulo são descritas as cinco etapas da metodologia: diagnóstico, planeamento de ações; execução de ações; avaliação; e especificação da aprendizagem. Cada uma destas etapas corresponderá a um capítulo autónomo que detalhará a abordagem realizada nessa fase.

O terceiro capítulo apresenta o diagnóstico realizado para interpretar o âmbito do projeto, explorar os problemas identificados e procurar perceções para possíveis soluções. Nele, são apresentados a entidade, sua missão, visão e valores, além dos problemas e oportunidades na resposta ao cliente.

O quarto capítulo abordará o planeamento das ações e aí serão apresentados os planos de execução, considerando diferentes abordagens, escolha de ferramentas e direções para o projeto, tais como a preparação dos dados, a escolha do modelo LLM, da forma de implementação e da linguagem de programação.

O quinto capítulo refere-se à execução das ações. Este capítulo detalha o processo de construção da aplicação, apresentando-se um guia passo a passo para sua implementação. Os passos incluem a recolha dos dados, o treino do modelo, os testes de ligação às *Application Programmable Interfaces* (API) e o seu funcionamento.

O sexto capítulo apresenta a avaliação do projeto. Nele são apresentados alguns testes ao protótipo, avaliando-se o desempenho do *chatbot* quando este responde a perguntas com informação presente na base de dados, sem informação ou sem relacionamento. Este

apresenta a monitorização dos resultados obtidos e compara-os com os objetivos iniciais do projeto.

A especificação da aprendizagem é apresentada no capítulo sétimo. Neste capítulo apresenta-se uma reflecção sobre os conhecimentos adquiridos ao longo da aplicação da metodologia, destacando os conhecimentos e capacidades adquiridas durante o processo.

O último capítulo apresenta a conclusão do projeto. Aí está enfatizada a integração bem-sucedida do protótipo de *chatbot* com a API do *OpenAI* no *Facebook*, realçando o seu potencial para melhorar o atendimento ao cliente e a eficiência operacional da Escola.

1 ENQUADRAMENTO TEÓRICO

Neste capítulo serão abordados alguns conceitos de Inteligência Artificial (AI) juntamente com alguns relacionados com os *chatbots*. Será apresentado um conceito mais amplo dos *chatbots* baseados em IA, detalhando-se as suas técnicas de funcionamento e como essas tecnologias se podem integrar para fornecer soluções eficazes no atendimento ao cliente e outras aplicações relevantes.

1.1 Inteligência Artificial

A inteligência artificial está a tornar-se cada vez mais onipresente nas nossas vidas quotidianas. Os investimentos globais estimados nesta área até 2021 rondam os $52,2 bilhões (Adadi & Berrada, 2018).

1.1.1 Definição de IA

O que é Inteligência Artificial (IA)? Para compreender esse conceito, é necessário entender o que significa "inteligência" e "artificial". Ao longo do século passado, a discussão em torno da definição de “inteligência” nunca cessou. De forma mais genérica, podemos defini-la como a capacidade de atingir certos objetivos em uma ampla variedade de ambientes através de atividades mentais, tais como a capacidade de aprender, raciocinar, compreender e interagir com informações provenientes de experiências, bem como a capacidade de lidar com a incerteza (Legg & Hutter, 2007). O termo “artificial” faz menção a algo que é feito por seres humanos, que poderá ser uma reprodução ou réplica de algo natural. Portanto, a IA é compreendida como “a capacidade de um sistema identificar, interpretar, fazer inferências e aprender com os dados para atingir objetivos organizacionais e sociais predeterminados” (Mikalef & Gupta, 2021).

Uma segunda perspetiva define a IA como uma ferramenta que resolve tarefas específicas que podem ser impossíveis ou extremamente demoradas para um ser humano realizar (Enholm et al., 2022). A definição exata da IA ainda é desafiadora, devido à falta de um consenso sobre o significado de “inteligência” (Kaplan & Haenlein, 2019), cada um definindo-a como o seu trabalho precisar. No âmbito do presente projeto, optou-se pela segunda perspetiva.

1.1.2 História e evolução da IA

De acordo com Lu (2019), a história da inteligência artificial (IA) pode ser dividida em três principais fases de desenvolvimento.

A fase inicial ocorreu do início do ano de 1956 até ao ano de 1980. Em 1956, na Universidade de Dartmouth, nos Estados Unidos, alguns jovens e promissores cientistas, como McCarthy e Minsky, participaram em uma conferência para estudar e explorar o uso de simulação de inteligência através de máquinas. Nesta reunião, eles tentaram descobrir como fazer a máquina pensar como um humano, como permitir que a máquina comunique em língua natural, como fazer com que a máquina atinja um certo grau de inteligência e, pela primeira vez, fornecer IA. A conferência de Dartmouth também é conhecida como a origem de IA.

A fase de industrialização da IA correspondeu ao período de 1980 até 2000 e teve por objetivo criar máquinas que suportassem o diálogo homem-máquina, a tradução e o reconhecimento de imagens. Posteriormente, alguns países desenvolvidos da Europa e os Estados Unidos também responderam e começaram a fornecer financiamento substancial para a investigação em IA. Nesta fase, o "processamento de conhecimento" é o foco da investigação em IA.

A fase da explosão da IA corresponde ao período desde o ano 2000 até ao presente. Esta fase caracterizou-se pelo desenvolvimento da Internet, do *big data* e da unidade de processamento gráfico (GPU, do inglês *Graphics Processing Unit*) e das tecnologias de IA, como o reconhecimento de voz e o reconhecimento de imagens, as quais já são atualmente aplicadas à vida real das pessoas comuns em diversas situações.

A IA tornou-se um dos principais impulsionadores do desenvolvimento industrial e um fator crítico na integração de tecnologias emergentes, como unidades de processamento gráfico, Internet das Coisas, computação em nuvem e *blockchain*, na nova geração de *big data* e Indústria 4.0.

1.1.3 Processamento de língua natural

O processamento de língua natural, do inglês *Natural Language Processing* (NLP), é um campo da inteligência artificial que se foca no processamento e na análise da língua humana, utilizando algoritmos e métodos diversos para interpretar e manipular dados. Envolve o desenvolvimento de ferramentas para enfrentar os desafios do processamento de língua natural, como aplicações de tradução automática, reconhecimento de entidades nomeadas e análise de sentimento. Recentes avanços no NLP têm sido impulsionados por técnicas como redes neurais, aprendizagem de máquina e transformadores, tornando-o crucial para diversas aplicações práticas (Khurana et al., 2023).

Nas primeiras décadas do desenvolvimento do NLP, os investigadores tentaram codificar manualmente os vocabulários e regras das línguas humanas, mas essa abordagem mostrou-se difícil devido à variabilidade, ambiguidade e dependência do contexto das linguagens. A partir do ano de 1980 e, em especial, na década de 1990, o NLP foi transformado pela adoção de modelos estatísticos e baseados em *corpus*, aproveitando a crescente disponibilidade de dados linguísticos digitais. Isso permitiu o desenvolvimento de ferramentas de alto desempenho para identificar informações sintáticas, semânticas e discursivas, como o Stanford CoreNLP. Avanços subsequentes em reconhecimento automático de voz, sistemas de diálogo falado e mineração de mídias sociais impulsionaram ainda mais o progresso do NLP. No entanto, o campo ainda enfrenta desafios significativos, como a falta de recursos para muitas linguagens de baixo recurso e a dificuldade dos sistemas de diálogo falado em lidar com interações de domínio aberto (Hirschberg & Manning, 2015).

1.1.4 Redes neurais e aprendizagem profunda

As redes neurais artificiais, do inglês *Artificial Neuronal Networks* (ANN), de acordo com Xin Yao (1999), são sistemas computacionais inspirados no funcionamento do cérebro humano, compostos por unidades de processamento interconectadas, chamadas neurónios. Algumas características-chave das ANN incluem:

- Arquitetura: Compostas por uma estrutura de nós interconectados, organizados em camadas *feedforward* ou recorrentes.

- Aprendizagem: Aprendem a partir de exemplos, ajustando iterativamente os pesos das conexões entre os neurónios, utilizando algoritmos de aprendizagem supervisionada, não supervisionada ou por reforço.
- Adaptabilidade: Podem-se adaptar a ambientes dinâmicos, combinando aprendizagem automática e evolução, formando as chamadas Redes Neurais Adversariais para Eventos, do inglês *Event Adversarial Neuronal Networks* (EANN).

A aprendizagem profunda é uma forma de aprendizagem de representação com múltiplos níveis de abstração, que tem demonstrado avanços significativos em diversas áreas, como reconhecimento de imagem, reconhecimento de voz, descoberta de drogas, genómica e compreensão de língua natural. Funciona através de uma arquitetura de múltiplas camadas de módulos simples, a maioria dos quais sujeitas a aprendizagem, transformando as suas entradas para aumentar tanto a seletividade quanto a invariância da representação.

Essa arquitetura é treinada usando o algoritmo de *backpropagation*, que calcula o gradiente do erro em relação aos pesos da rede neuronal, permitindo ajustá-los para minimizar o erro. Além disso, a aprendizagem profunda tem a capacidade de aprender representações distribuídas, como vetores de palavras, que capturam relações semânticas entre os símbolos de entrada e saída. Isso confere à aprendizagem profunda vantagens exponenciais em relação aos algoritmos clássicos de aprendizagem de máquina, tornando-a numa ferramenta poderosa em diversas aplicações de inteligência artificial (LeCun et al., 2015).

1.1.5 Aplicações práticas de IA

A IA tem revolucionado diversas áreas. Uma dessas áreas é a da medicina. O Processamento da Língua Natural facilita a comunicação com pacientes através de *chatbots* e assistentes virtuais. Técnicas de aprendizagem profunda (*deep learning*) são utilizadas para classificar e analisar dados médicos, como imagens de diagnóstico, sem intervenção humana. As Redes Neurais Convulsionais, do inglês *Convolutional Neuronal Network* (CNN) desempenham um papel crucial na análise de imagens médicas, permitindo a deteção de lesões e a geração automatizada de relatórios. Na

gastroenterologia, a IA melhora o diagnóstico de doenças, previsão de prognósticos e resposta a tratamentos. A IA também oferece assistência valiosa em endoscopia, com a deteção de pólipos intestinais e cancro gástrico. Nos exames de cápsula endoscópica, a IA identifica anomalias no intestino delgado e cancro gástrico. As redes neurais artificiais, do inglês *Artificial Neuronal Network* (ANN), são usadas para diferenciar entre pancreatite crónica e cancro de pâncreas, melhorando a precisão diagnóstica (Kaul et al., 2020).

A IA também tem sido amplamente aplicada em projetos ligados à sustentabilidade em diversos setores-chave. No âmbito das cidades inteligentes, a IA resolve problemas de agendamento, previsão e monitoramento, ajudando a otimizar o consumo de energia e a gerir sistemas de transporte público. Nos sistemas de energia, a IA modela a procura e oferta de energia, além de prever e monitorar padrões de consumo. Na cadeia de fornecimento, embora a adoção da IA ainda seja limitada, existem propostas para integrar IA com tecnologias como *blockchain* e Internet das Coisas, visando melhorar a rastreabilidade e a eficiência. Essas aplicações demonstram o potencial da IA para promover a sustentabilidade através de soluções mais eficientes e inovadoras (WU et al., 2022).

A IA pode ser aplicada na educação de três maneiras principais: como um novo sujeito no sistema educacional, onde a IA é projetada para tomar decisões e intervir diretamente no ensino e aprendizagem; como mediador direto, onde a IA auxilia professores e alunos a atingirem os seus objetivos educacionais, considerando as necessidades e a aceitação dos utilizadores finais; e como assistente suplementar, influenciando as relações professor-aluno e aluno-aluno, através de ferramentas de avaliação automatizada, modelos de aprendizagem automática e sistemas de recomendação personalizados (Xu & Ouyang, 2022).

1.2 *Chatbots*

O *chatbot* é um software que simula uma interação de conversa entre o utilizador e um computador. Ao longo do tempo, esses programas ganharam muita fama e têm sido amplamente utilizados em diversos domínios, como comércio, saúde, banca, entre outros

(S. Yang & Evans, 2019). Muitos estudos destacaram a potencial aplicação dessas tecnologias numa vasta gama de domínios, sendo que estes podem realizar tarefas de mão de obra intensiva a custos mais baixos, tais como o serviço ao cliente, acompanhamento com crianças, cuidados a idosos, assistência pessoal em cenários como escolas, instituições bancárias, empresas e estabelecimentos de saúde (Nagarhalli et al., 2020).

1.2.1 História e evolução dos *Chatbots*

O conceito básico de *chatbot* foi inicialmente proposto por Alan Turing em 1950 e, em 1966, o ELIZA imitou o diálogo humano, embora com limitações. Em 1988, o Jabberwacky iniciou a era dos sistemas baseados em voz a partir de sistemas baseados em texto, sendo uma das primeiras tentativas de criar IA por meio da interação humana. Em 1995, o ELIZA evoluiu para um padrão de diálogo usando a linguagem de marcação de IA, do inglês *Artificial Intelligence Markup Language* (AIML), mas as suas limitações ainda eram evidentes. Recentemente, os *chatbots* incorporaram tecnologia de aprendizagem profunda, permitindo que entendam o contexto e respondam adequadamente a perguntas ou comandos em língua natural. Além disso, com o avanço da aprendizagem profunda, os *chatbots* têm sido cada vez mais integrados em serviços públicos, empresas, modelos "*Online to Offline*" (O2O) e sistemas de assistente pessoal inteligente (Park et al., 2022).

1.2.2 Tipos de *Chatbots*

Há vários tipos de *chatbots*. Uma das classificações mais comuns distinguem os *chatbots* em dois tipos: *chatbots* baseados em regras e *chatbots* baseados em inteligência artificial (B. Luo et al., 2022).

1.2.2.1 Chatbots baseados em regras

O *chatbot* baseado em regras é desenvolvido com o uso da Linguagem de Marcação de Inteligência Artificial (AIML) para estabelecer regras para o sistema (Thorat & Jadhav, 2020). A AIML é uma linguagem de código aberto baseada em XML, na qual o *chatbot* responde aos utilizadores ao detetar a correspondência com perguntas previamente definidas nos dados (Adamopoulou & Moussiades, 2020).

1.2.2.2 Chatbots baseados em IA

Os *chatbots* baseados em IA aplicam algoritmos mais sofisticados, que se baseiam em técnicas como *Natural Language Processing* (NLP), *Natural Language Understanding* (NLU) ou *Natural Language Generation* (NLG) (Maroengsit et al., 2019). Esses *chatbots* são mais flexíveis nas suas respostas e aproveitam as técnicas de aprendizagem profunda (*deep learning*) para otimizar as suas funções (Kumar & Ali, 2020).

Os *chatbots* baseados em IA podem ser divididos em dois tipos:

- *Chatbot* baseado em recuperação de informação
- *Chatbot* baseado em geração

Esses dois tipos desempenham papéis diferentes. O modelo de recuperação foca-se em áreas que requerem respostas mais precisas e credíveis, enquanto o modelo generativo é capaz de fornecer respostas em uma área mais abrangente, porém com menor qualidade e credibilidade. Atualmente, o modelo generativo tem-se desenvolvido mais rapidamente que o modelo de recuperação de informação, uma vez que os dados de treino são mais gerais, ao contrário do modelo de recuperação, que exige treino personalizado para cada domínio específico (Caldarini et al., 2022).

1.2.3 Aplicações práticas de *Chatbot*

A adoção de *chatbots* está a tornar-se cada vez mais popular em diversas áreas, como por exemplo o comércio eletrónico e o atendimento ao cliente. Estas áreas são das que apresentam maior adoção de *chatbots*. As previsões da Gartner apontavam para um aumento no uso de *chatbots* e assistentes virtuais no atendimento ao cliente de 2% em 2017 para 25% em 2020. O setor financeiro também tem uma quantidade significativa de estudos sobre a adoção de *chatbots*, impulsionada pelo crescimento das *fintechs* e aplicações como *robo-advisors*, automação de investimentos e pedidos de empréstimos online. No campo do marketing, a tecnologia de *chatbot* é amplamente utilizada para publicidade, marketing móvel e construção de marca. O setor da saúde também beneficia dos *chatbots* para promover a mudança de comportamentos, suporte ao tratamento e monitorização do estado de saúde. Em contrapartida, há menos estudos sobre a adoção de

chatbots em áreas como recrutamento, marcas de luxo, gestão do conhecimento, veterinária, retalho, política, conteúdo de mídia e imobiliário. No geral, a adoção de *chatbots* está em crescimento em áreas como comércio eletrónico, atendimento ao cliente, finanças, marketing e saúde, refletindo a procura e a aplicação desta tecnologia em diversos setores (Alsharhan et al., 2023).

Os *chatbots* de IA estão a ser cada vez mais utilizados como agentes de atendimento de primeira linha, com o potencial de lidar com até 80% das consultas padrão dos clientes e auxiliar em várias tarefas ao longo da jornada do cliente (Chong et al., 2021).

Por exemplo, *chatbots* do tipo *Dinus Intelligent Assistance* (DINA) são aplicados em ambientes educacionais, onde a instituição de ensino tem como missão a satisfação dos candidatos e estudantes, oferecendo-lhes o serviço de consulta e informação (Agus Santoso et al., 2018).

Num estudo no setor alimentar desenvolvido por Klein & Martinez (2023), os resultados confirmaram a associação positiva entre características antropomórficas e a satisfação do cliente, mediada pelo prazer, atitude e confiança. O resultado da adoção de tecnologias inovadoras e a integração de elementos de design antropomórfico nos aplicativos de *chatbots* pelos retalhistas de alimentos visaram manter a competitividade da empresa.

1.2.4 Forças e fraquezas na utilização de *chatbots*

Os *chatbots* baseados em IA oferecem diversos benefícios para os negócios, principalmente por meio da automatização do atendimento ao cliente. Eles facilitam a comunicação das empresas com os clientes, garantindo uma qualidade estável no atendimento, uma vez que as máquinas não possuem emoções negativas ou cansaço. Além disso, os *chatbots* podem lidar facilmente com um grande volume de comunicações (X. Luo et al., 2019).

Por outro lado, os *chatbots* também apresentam algumas fraquezas. Eles podem fornecer respostas incorretas ou não ter uma resposta adequada para certas perguntas, o que pode afetar negativamente a experiência do utilizador. Além disso, a implementação e manutenção de *chatbots* requerem investimentos em recursos de tecnologia da informação, o que pode representar um custo significativo para as empresas. Outra

fragilidade é a possibilidade de riscos de imagem e reputação quando fornecem respostas inadequadas ou inapropriadas para os utilizadores. Isso pode levar a opiniões negativas e danos à reputação da empresa (Zumstein & Hundertmark, 2017).

1.3 Modelos de linguagem de grande escala

1.3.1 Em que consistem os modelos de linguagem de grande escala

Os modelos de linguagem de grande escala, do inglês *Large Language Models* (LLM), são modelos treinados com extensas quantidades de dados textuais, permitindo-lhes gerar texto de forma fluente e compreender a língua natural.

Os LLM adquirem um vasto conhecimento geral a partir de grandes conjuntos de dados de texto, capacitando-os a realizar uma ampla gama de tarefas de processamento de língua natural, tais como a geração de texto, resposta a perguntas e tradução. Estes modelos demonstram um desempenho excecional em tarefas de processamento de língua natural, evidenciando capacidades avançadas em diversas aplicações linguísticas.

Além disso, os LLM têm a capacidade de generalizar o conhecimento apreendido, permitindo-lhes lidar com novos cenários e tarefas além daqueles vistos durante o treino. No entanto, armazenam o conhecimento de forma implícita nos seus parâmetros, o que dificulta a interpretação e validação das informações que geram. Uma crítica significativa aos LLM é a tendência para a "alucinação", ou seja, a geração de informações factualmente incorretas. Como modelos de caixa-preta, os LLM carecem de interoperabilidade, uma vez que os padrões e funções utilizadas para chegar a previsões ou decisões não são diretamente acessíveis ou explicáveis aos humanos (Pan et al., 2024).

1.3.2 Evolução dos modelos de linguagem de grande escala

A Figura 1-1 apresenta uma árvore evolutiva dos LLM modernos, a qual ilustra o desenvolvimento dos modelos de linguagem de larga escala nos últimos anos e destaca alguns dos modelos mais conhecidos. Modelos no mesmo ramo têm relações mais próximas. Modelos baseados em *Transformer* são mostrados em cores não cinzas, modelos apenas com decodificador no ramo azul, modelos apenas com codificador no

ramo rosa e modelos codificador-decodificador no ramo verde. A posição vertical dos modelos na linha do tempo representa as suas datas de lançamento.

Modelos apenas com decodificador têm dominado gradualmente o desenvolvimento dos LLM. No início, os modelos apenas com decodificador não eram tão populares quanto os modelos apenas com codificador e os modelos codificador-decodificador. Após 2021, com a introdução de LLM revolucionários como o GPT-3, os modelos apenas com decodificador experimentaram um crescimento significativo. Após o crescimento inicial impulsionado pelo BERT, os modelos apenas com codificador começaram a perder relevância.

A *OpenAI* tem mantido consistentemente uma posição de liderança na área dos LLM, prevendo-se que essa posição se mantenha num futuro próximo. Outras empresas e instituições estão a lutar para alcançar a *OpenAI* no desenvolvimento de modelos comparáveis ao GPT-3 e ao atual GPT-4. Essa liderança pode ser atribuída ao compromisso constante da *OpenAI* com o seu caminho técnico, mesmo quando este não era amplamente reconhecido inicialmente (J. Yang et al., 2024).

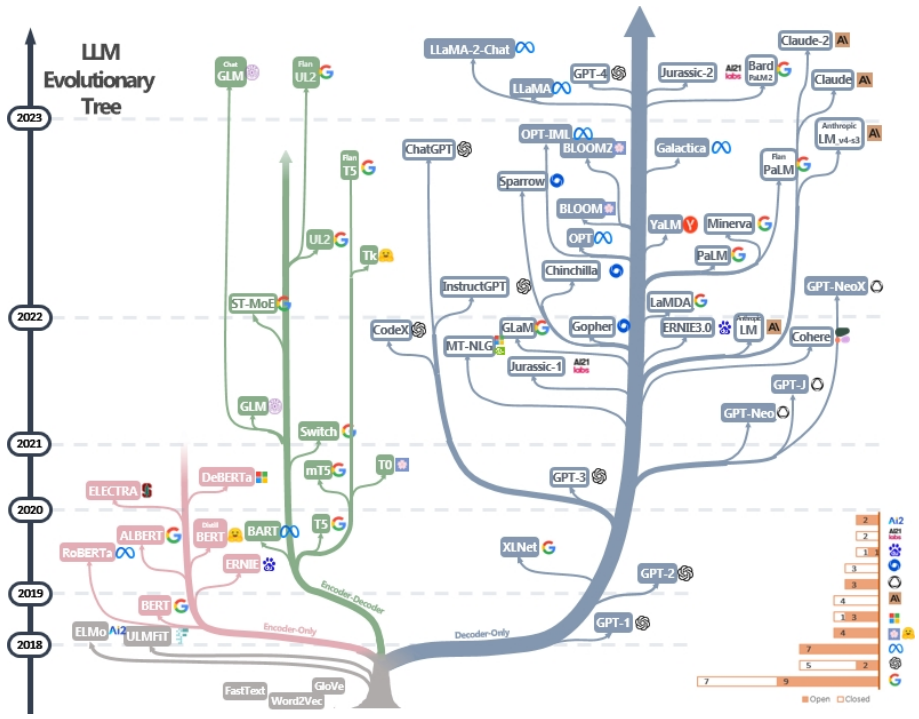

Figura 1-1: Evolução dos modelos de linguagem de grande escala

1.3.3 LLM do *OpenAI*

O ChatGPT é um modelo de língua avançado desenvolvido pela *OpenAI*, projetado para conversas em língua natural. Foi treinado com uma enorme quantidade de dados de texto gerada por humanos, como livros, artigos, publicações de blog e conversas. O objetivo do ChatGPT é fornecer respostas precisas e contextualizadas às perguntas dos utilizadores.

O ChatGPT baseia-se no modelo GPT (*Generative Pre-trained Transformer*) desenvolvido anteriormente pela *OpenAI*. O GPT foi projetado para prever a próxima palavra em um texto gerado por humanos, e o seu modelo foi treinado com uma grande quantidade destes dados.

Além do ChatGPT, a *OpenAI* oferece outros modelos com diferentes capacidades, como o GPT-3.5 para compreensão e geração de língua natural e código, o DALL·E para geração de imagens, o *Whisper* para conversão de áudio em texto, o *Embedding* para representação numérica de texto, o *Codex* para interpretação e geração de código, e o *Moderation* para identificação de conteúdo potencialmente sensível ou inseguro (Roumeliotis & Tselikas, 2023).

1.3.4 Prompt engineering

Um *prompt* é um conjunto de instruções fornecidas a um modelo de linguagem de larga escala que o questiona, personalizando e/ou aperfeiçoando ou refinando as suas capacidades. Um *prompt* pode influenciar interações subsequentes e a saída gerada de um LLM, fornecendo regras e diretrizes específicas para uma conversa com o LLM, definindo o contexto da conversa e informando o LLM quais as informações que são importantes e qual deve ser a forma e o conteúdo desejados da saída.

Prompt engineering é o processo de criar e ajustar *prompts* para obter respostas desejadas de um modelo de linguagem de larga escala. Envolve experimentar diferentes formas de redigir os *prompts*, ajustar a sua complexidade e especificidade, e otimizar a forma como as instruções são apresentadas ao modelo. O objetivo do *prompt engineering* é maximizar a precisão, relevância e utilidade das respostas geradas pelo LLM, garantindo que o

modelo compreenda e responda de forma eficaz às instruções fornecidas (White et al., 2023).

2 METODOLOGIA

Na metodologia deste projeto foi baseada no método investigação-ação. A investigação-ação facilita a integração entre a investigação e a prática e torna a investigação em sistemas de informação mais prática. Além disso, uma das premissas fundamentais da investigação-ação é a filosofia do seu pragmatismo, que enfatiza fazer as perguntas certas e respondê-las empiricamente. O contexto social da investigação-ação também sugere que o comportamento humano é socialmente reflexivo e que o ambiente social molda as visões e comportamentos dos investigadores da ação. Portanto, a importância da investigação-ação reside na sua capacidade de aliar a teoria e a prática, e trazer resultados mais práticos de investigação para o campo dos sistemas de informação por meio da participação social e dos métodos empíricos de pesquisa (Baskerville & Myers, 2004).

De acordo com Baskerville (1999), a investigação-ação é um método de investigação baseado na ação prática, com o objetivo de resolver situações imediatas. A descrição mais comum da investigação-ação requer, em primeiro lugar, o estabelecimento de um ambiente de investigação. Depois são identificadas cinco fases, respetivamente:

1- Diagnóstico

2- Planeamento de ações

3- Execução das ações

4- Avaliação

5- Especificação da aprendizagem

Neste projeto a metodologia investigação-ação foi aplicada com um único ciclo. A adoção desta metodologia abrangeu o detalhe de todos os processos das ações praticadas, tendo em vista o objetivo a atingir. Incluiu o diagnóstico do problema, a proposta e planeamento da resolução do problema, a fase de execução das ações planeadas, a qual exigiu que o plano proposto fosse possível de ser implementado. Após as ações terem sido realizadas foi feita a avaliação para testar se o resultado resolveu o problema que foi diagnosticado. Por fim, foi feita uma reflecção final sobre todo este processo e foram registadas as principais conclusões.

2.1 Diagnóstico

No diagnóstico identificam-se os problemas primários que são as causas subjacentes do desejo de mudança da organização. O diagnóstico envolve a autointerpretação do problema organizacional complexo, não por meio de redução e simplificação, mas sim de forma holística. Esse diagnóstico desenvolve certas suposições teóricas (ou seja, uma hipótese de trabalho) sobre a natureza da organização e seu domínio de problemas.

No âmbito deste projeto foi apresentado o enquadramento da organização, o Instituto Superior de Contabilidade e Administração de Coimbra, dando-se a conhecer melhor a organização que o projeto vai servir, perceber a sua missão, visão e os seus valores, e depois diagnosticando-se a situação que se pretende melhorar.

2.2 Planeamento das ações

No planeamento foi elaborada uma proposta implementável que pretende minimizar os problemas primários identificados, apontando para a sua resolução, mesmo que parcial.

Isso envolveu descrever como o problema será abordado e apresentar as expectativas em relação à solução. Em seguida, foram delineados os passos necessários para preparar e implementar o plano de ação.

Durante a preparação para a execução do plano, foram considerados todos os recursos necessários, incluindo materiais, equipamentos, pessoal e tempo. Foi também explicada a escolha das ferramentas e metodologias. Essa explicação teve o objetivo de garantir que o plano fosse executado de forma eficiente e eficaz, maximizando as hipóteses de sucesso na resolução do problema identificado.

2.3 Execução das ações

No processo de execução, foram feitas as implementações das ações planeadas, com base na seção anterior e com o objetivo explícito no diagnóstico.

Nesta secção, foram descritos todos os atos de construção do projeto, desde a recolha de dados até à configuração final do sistema para resolver o problema proposto. A primeira etapa foi a obtenção de dados relevantes e de qualidade. De seguida, os dados foram

utilizados para treinar o modelo, otimizando os seus parâmetros. Após o treino, o modelo foi integrado numa API e submetido a testes. Simultaneamente, são desenvolvidos os códigos necessários para a implementação e operação do sistema. Por fim, todas as configurações e integrações são realizadas para garantir que todos os componentes funcionem de forma coesa e eficiente.

2.4 Avaliação

Na avaliação foram avaliados os resultados das ações implementadas. Isso incluiu determinar se os efeitos esperados da ação foram alcançados e se esses efeitos aliviaram os problemas identificados. Para avaliar o resultado, foi concebida uma avaliação tridimensional, em que se avaliou as respostas do *chatbot* a perguntas com informação na base de dados, sem informação e não relacionados.

2.5 Especificação da aprendizagem

A especificação da aprendizagem foi a última etapa do ciclo de investigação-ação. Nesta etapa foram refletidos os novos conhecimentos adquiridos durante a pesquisa. Este projeto não incluiu a implementação do *chatbot* em ambiente real. O seu âmbito foi o do desenvolvimento de um protótipo, ou seja, um produto cujo objetivo é ser experimental, que permita recolher conhecimento e perceções antes do eventual desenvolvimento de uma versão definitiva a colocar em ambiente de produção no campo de ação do *site* oficial da organização. Nesta etapa foram refletidos os conhecimentos adquiridos neste ciclo de pesquisa, incluindo na utilização do modelo LLM, do uso de API, do *web scraping*, da automatização de respostas automáticas no *Facebook* e do desenvolvimento de programas de integração.

3 DIAGNÓSTICO

Neste capítulo será feita uma apresentação da entidade, designadamente a sua missão, visão e valores, seguindo-se o diagnóstico dos problemas e oportunidades identificados.

3.1 Apresentação da Coimbra Business School | ISCAC

O Instituto Superior de Contabilidade e Administração de Coimbra (ISCAC), também conhecido como Coimbra Business School | ISCAC, é uma escola superior que pertence ao Instituto Politécnico de Coimbra (IPC). Esta escola de negócios e ciências empresariais tem uma história que remonta a mais de 100 anos, desde a fundação do Instituto Industrial e Comercial de Coimbra em 1921. Este instituto, inicialmente estabelecido para oferecer ensino técnico médio, expandiu o seu âmbito ao longo dos anos, passando a abranger áreas como engenharia, comércio e administração (ISCAC, 2024). Posteriormente, deste instituto derivaram o ISCAC e o Instituto Superior de Engenharia de Coimbra.

O ISCAC oferece uma formação de excelência nos seus cursos de licenciatura, orientada para as necessidades do mercado contemporâneo. Disponibiliza as licenciaturas de Ciência de Dados para a Gestão, Comércio e Relações Económicas Internacionais, Contabilidade e Auditoria, Contabilidade e Gestão Pública, Finanças e Contabilidade, Gestão de Empresas, Informática de Gestão, Marketing e Negócios Internacionais, Secretariado Direção e Administração, e Solicitadoria e Administração. Além disso, oferece diversos programas de mestrado, incluindo Análise de Dados e Sistemas de Apoio à Decisão, Análise Financeira, Auditoria Empresarial e Pública, Contabilidade e Fiscalidade Empresarial, Contabilidade e Gestão Pública, Controlo de Gestão, Gestão de Empresas Agrícolas, Gestão de Recursos Humanos, Gestão Empresarial, Inteligência Logística e Gestão da Cadeia de Abastecimento, Marketing e Negócios Internacionais, Sistemas de Informação de Gestão, e Solicitadoria. Cada programa é desenhado para proporcionar uma base sólida de conhecimentos teóricos e práticos, capacitando os estudantes para enfrentar os desafios profissionais com competência e inovação.

A missão da Coimbra Business School | ISCAC é ultrapassar os limites do ensino técnico e académico, fomentando o pensamento crítico, a liberdade e a responsabilidade cívica e humana. Os seus valores fundamentais, os quais incluem a liberdade, a responsabilidade,

a independência, a honestidade e a integridade, são a base para uma educação holística e ética que prepara os alunos para os desafios do mundo moderno (IPC, 2021).

Como pioneira na integração de tecnologia e educação, a visão da Coimbra Business School | ISCAC é tornar-se uma referência global nas ciências empresariais. Esta escola procura o reconhecimento pela excelência académica, pela pesquisa inovadora e pela criação de soluções práticas para os problemas do mundo real. Almeja ser líder na formação de profissionais altamente qualificados e empreendedores, prontos para enfrentar os desafios do mercado global (IPC, 2021).

A escola promove uma cultura de inovação, colaboração e excelência. Esses valores são fundamentais para o seu compromisso em oferecer uma educação de classe mundial e para a criação de um ambiente inclusivo e inspirador que capacite os alunos a alcançarem o seu pleno potencial (IPC, 2021).

Esses elementos refletem a visão inovadora e a ambição da Coimbra Business School | ISCAC (CBS | ISCAC) em liderar a transformação do ensino superior e preparar os alunos para um futuro de sucesso e impacto positivo no mundo.

3.2 Serviços e âmbito de ação

A CBS | ISCAC oferece uma ampla gama de serviços conforme estabelecido no artigo 68.º dos seus estatutos (ISCAC, 2021). Esses serviços são essenciais para o funcionamento eficiente da instituição e para o suporte adequado às atividades académicas e administrativas. Dos doze serviços oferecidos, o mais relevante é o Serviço de Gestão Académica (SGA), o qual é responsável pelas atividades relacionadas com os estudantes, incluindo a gestão de processos individuais, propinas e matrículas. O SGA também trata dos processos curriculares, de ingresso e de mobilidade dos estudantes. Para além do SGA existem outros serviços no ISCAC, tais como, o Serviço de Administração e Finanças, e Serviço de Recursos Humanos, o Gabinete de Estágios e Saídas Profissionais.

3.3 Problemas e oportunidades na resposta ao cliente

Como em todas as entidades que prestam serviços a terceiros, também na CBS | ISCAC a qualidade e a eficiência na resposta às dúvidas dos clientes desempenham um papel crucial. Além de influenciar a atração de novos clientes, também estão em jogo a satisfação dos clientes existentes e a manutenção de uma reputação positiva no mercado. Uma resposta lenta ou inadequada pode não apenas afastar potenciais clientes, mas também levar à insatisfação e à perda da confiança dos clientes já existentes.

A insatisfação dos clientes em relação à velocidade e eficácia das respostas é uma preocupação crescente. Muitas vezes, os clientes reclamam da demora na obtenção de respostas e da falta de solução para as suas questões. Essa situação pode ser atribuída, em parte, à sobrecarga de trabalho dos funcionários, que muitas vezes estão ocupados com uma variedade de tarefas e têm dificuldade em lidar com o volume crescente de perguntas dos clientes.

É importante reconhecer que a insatisfação dos clientes não apenas afeta a reputação da escola, mas também pode ter consequências financeiras significativas. A perda de potenciais clientes devido a respostas inadequadas ou lentas pode resultar em uma redução na receita e na rentabilidade da instituição. Além disso, a má reputação gerada por comentários negativos pode afetar a capacidade de a escola atrair e reter alunos de qualidade no futuro.

Diante de esses desafios, é essencial adotar uma abordagem proativa para melhorar o atendimento ao cliente. A implementação de um sistema de resposta automática pode ser uma solução eficaz para aliviar a carga de trabalho dos funcionários e melhorar a eficiência na gestão das consultas dos clientes. Paralelamente, é igualmente importante investir em práticas de melhoria contínua para garantir que as necessidades dos clientes sejam atendidas de forma eficaz e satisfatória ao longo do tempo.

Em resumo, a melhoria do atendimento ao cliente é uma prioridade fundamental para a Coimbra Business School | ISCAC. Ao reconhecer os desafios atuais e implementar soluções inovadoras, a escola pode fortalecer a sua posição no mercado e garantir uma experiência positiva para os seus clientes existentes e os potenciais.

4 PLANEAMENTO DE AÇÕES

Neste capítulo é proposta uma resolução do problema detetado anteriormente e as ações planeadas que serão realizadas posteriormente.

4.1 Proposta de uma solução baseada num sistema *Chatbot*

A introdução de um *chatbot* na Coimbra Business School | ISCAC representa uma estratégia inovadora para otimizar o atendimento ao cliente e melhorar a eficiência operacional. Ao incorporar um *chatbot* nas suas operações, a escola poderá distribuir de forma inteligente a carga de trabalho entre os funcionários e o sistema automatizado.

Um dos principais benefícios da implementação de um *chatbot* é a sua capacidade de lidar com tarefas repetitivas e rotineiras de forma rápida e eficiente. Por exemplo, o *chatbot* pode ser implementado de forma que responda a perguntas frequentes dos clientes, forneça informações sobre os cursos oferecidos, horários de funcionamento da escola, procedimentos de matrícula e muito mais. Isso não apenas agiliza o processo de atendimento ao cliente, mas também liberta os funcionários humanos para lidar com consultas mais complexas e exigentes.

Além disso, o *chatbot* pode operar 24 horas por dia, 7 dias por semana, proporcionando aos clientes acesso instantâneo às informações de que precisam, independentemente do horário do dia ou da noite. Isso é especialmente vantajoso para os atuais estudantes e potenciais interessados que podem estar em fusos horários diferentes ou que preferem pesquisar informações fora do horário de atendimento convencional.

Ao implementar um *chatbot*, a Coimbra Business School | ISCAC também estará investindo em eficiência operacional. Os *chatbots* são capazes de lidar com um grande volume de consultas simultaneamente, sem comprometer a qualidade ou a velocidade das respostas. Isso pode ajudar a reduzir os tempos de espera dos clientes, aumentar a produtividade da equipe e melhorar a satisfação geral do cliente.

Além disso, a escolha de um *chatbot,* em vez de aumentar o número de funcionários, representa uma decisão financeiramente responsável. Os custos associados à implementação e manutenção de um *chatbot* são geralmente mais baixos do que os custos

de contratação, formação e retenção de pessoal. Isso permite que a escola maximize os seus recursos financeiros e direcione investimentos para outras áreas prioritárias.

Em resumo, a implementação de um *chatbot* na Coimbra Business School | ISCAC oferece uma solução eficaz para distribuir a carga de trabalho, melhorar a eficiência operacional e proporcionar uma experiência de atendimento ao cliente mais ágil e satisfatória. Ao adotar essa abordagem inovadora, a escola estará a posicionar-se para o sucesso futuro e a demonstrar o seu compromisso com a excelência em serviço e inovação tecnológica.

4.2 Preparação dos dados

Para garantir a eficácia do *chatbot*, foi necessário um cuidadoso processo de preparação de dados, que abordou duas fontes principais: as informações gerais disponibilizadas ao público e as perguntas frequentes recebidas pelos funcionários. Esta abordagem multifacetada visou abranger uma ampla gama de consultas e cenários possíveis, proporcionando ao *chatbot* uma base sólida para oferecer respostas precisas e úteis aos clientes.

O processo envolveu a recolha de informações gerais fornecidas ao público, que geralmente estão disponíveis no site da escola. Estas informações incluem detalhes sobre os programas académicos, histórico da instituição, contatos, políticas de admissão e outros aspetos relevantes para os potenciais alunos e partes interessadas. Esses dados são fundamentais para garantir que o *chatbot* possa responder a consultas básicas de forma precisa e abrangente.

Muitas destas informações estão na base de perguntas frequentes que os funcionários recebem, muitas das quais podem resultar em trabalhos repetitivos.

4.3 Escolha das ferramentas de implementação

Para maximizar a eficácia e minimizar a dependência, a escolha das ferramentas de implementação foi crucial. Diversas ferramentas podem ser consideradas, mas foi essencial selecionar aquelas que melhor se adequam às necessidades específicas deste projeto.

4.3.1 Escolha do modelo de LLM

No início do processo de escolha de ferramentas de implementação, foi dedicado um tempo significativo à pesquisa de *Large Language Model* (LLM) disponíveis no mercado. Para garantir que o *chatbot* fosse capaz de gerar respostas com base em texto, era imperativo escolher um modelo de LLM adequado. Optou-se por esta abordagem, em vez de utilizar um *chatbot* baseado em regras, pois desejava-se oferecer aos utilizadores uma interação mais inteligente e dinâmica.

Após uma análise cuidadosa das opções disponíveis, o LLM da *OpenAI* emergiu como a escolha mais atrativa. Além de apresentar um custo acessível para os clientes, o modelo da *OpenAI* destacou-se pela sua ampla aceitação e pela capacidade computacional robusta, o que o torna capaz de lidar eficientemente com um grande volume de utilizadores. A popularidade e a confiabilidade deste modelo foram fatores determinantes na sua seleção. O modelo da *OpenAI* tem demonstrado consistentemente alto desempenho e precisão em tarefas de geração de texto, superando outros modelos em *benchmarks* relevantes. Estudos e comparações demonstraram que o modelo da *OpenAI* é capaz de produzir respostas mais coerentes e contextualmente apropriadas, o que é fundamental para a qualidade da interação com os utilizadores.

Uma pesquisa da LMSYS Org (*Large Model Systems Organization*), uma organização de pesquisa aberta fundada por estudantes e professores da Universidade da Califórnia, Berkeley, em colaboração com a Universidade da Califórnia, San Diego (UCSD) e a Universidade Carnegie Mellon, revelou perceções significativas sobre o desempenho dos modelos de linguagem de grande escala (LLM). O objetivo desta organização é tornar os modelos de grande escala acessíveis a todos, por meio do desenvolvimento conjunto de modelos abertos, conjuntos de dados, sistemas e ferramentas de avaliação (Large Model Systems Organization, 2023).

De entre os projetos acompanhados pela LMSYS Org, destaca-se também, por exemplo, o Vicuna-13B, um *chatbot* de código aberto treinado com dados de conversas compartilhados pelos utilizadores, provenientes do ShareGPT. Avaliações iniciais indicam que o Vicuna-13B atingiu mais de 90% da qualidade dos modelos *OpenAI*

ChatGPT e Google Bard, quando julgado pelo GPT-4. Além disso, superou outros modelos como o LLaMA e o Stanford Alpaca em mais de 90% das situações (Large Model Systems Organization, 2023).

Outro aspeto decisivo na escolha do LLM da *OpenAI* foi a facilidade de uso oferecida. A sua interface amigável e intuitiva permite que os clientes utilizem os seus próprios dados e também consigam treinar modelos personalizados de maneira simples e eficaz. Essa flexibilidade era essencial para adaptar o *chatbot* às necessidades específicas da escola e proporcionar uma experiência personalizada aos utilizadores.

Dessa forma, a escolha do LLM da *OpenAI* como base para o *chatbot* não apenas reflete uma abordagem estratégica para garantir uma experiência de alta qualidade, mas também demonstra o compromisso em fornecer soluções inovadoras e acessíveis aos utilizadores.

4.3.2 Escolha da forma de implementação

Na seleção da forma de implementação, o processo envolveu uma comparação entre duas abordagens distintas: criar uma janela integrada à página do site da escola ou conectar-se a uma página do *Facebook*.

A primeira opção, a criação de uma janela sobreposta ao site, exigiria um esforço considerável de desenvolvimento, bem como a integração com a infraestrutura existente da página *web*. Esse método permitiria uma personalização total do desenho e funcionalidades do *chat* e alinhar-se com a identidade visual da escola. No entanto, a manutenção dessa janela exigiria recursos contínuos, uma vez que qualquer atualização ou modificação no site poderia afetar a sua funcionalidade.

A segunda opção, optar por integrar o *chatbot* à página do *Facebook*, ofereceria uma série de vantagens significativas. Em primeiro lugar, a implementação foi consideravelmente mais simples e rápida, uma vez que a plataforma do *Facebook* já fornece as ferramentas necessárias para criar e gerir *chatbots*. Além disso, a página do *Facebook* é uma interface familiar para muitos utilizadores, o que pode facilitar a adoção e a interação com o *chatbot*.

Outro benefício importante da segunda opção seria a facilidade de manutenção. Ao conectar o *chatbot* à página do *Facebook*, as atualizações e modificações podem ser facilmente geridas através da própria plataforma, sem a necessidade de intervenção técnica adicional. Isso reduziria significativamente o ónus de manutenção e permitiria uma resposta mais ágil às necessidades em constante evolução da escola e dos seus utilizadores.

Após uma análise cuidadosa dos prós e contras de cada opção, optou-se pela implementação através de uma página do *Facebook*. A combinação de maior alcance potencial, facilidade de implementação e a disponibilização de ferramentas adicionais oferecidas pela plataforma foram fatores decisivos. Esta decisão refletiu um compromisso em proporcionar uma solução acessível, eficiente e de alta qualidade aos utilizadores, maximizando o impacto e a usabilidade do *chatbot*.

A Figura 4-1 apresenta um exemplo do funcionamento do *chatbot* no Messenger do telemóvel. A página é clara e simples e a operação é intuitiva e fácil de usar.

Figura 4-1: Exemplo do funcionamento do Chatbot no Messenger

Ao valorizar tanto a facilidade de implementação, quanto a simplificação da manutenção, a integração do *chatbot* à página do *Facebook* surge como a escolha mais vantajosa e eficaz para garantir uma experiência otimizada para os utilizadores da escola.

4.3.3 Escolha da linguagem de programação

A escolha da linguagem de programação recaiu sobre o *Python*. Esta decisão foi motivada por uma série de razões que evidenciam a sua adequação para o projeto em questão. Em primeiro lugar, o *Python* destaca-se pela sua sintaxe clara e pela sua legibilidade, tornando-o acessível mesmo para programadores iniciantes. A sua facilidade de compreensão é um fator crucial, especialmente em projetos complexos como o desenvolvimento de um *chatbot* para a escola, onde a clareza e a simplicidade são essenciais. Além disso, o *Python* oferece uma ampla variedade de bibliotecas e *frameworks*, abrangendo desde *web scraping* até integrações com API de inteligência artificial, como a *OpenAI*, e plataformas como o *Facebook*. Essa versatilidade simplifica o processo de desenvolvimento, permitindo atender a todas as necessidades do projeto de forma eficaz. Além disso, a comunidade *Python* é extremamente ativa e oferece um amplo suporte e recursos adicionais, o que facilita ainda mais o desenvolvimento e a manutenção do projeto ao longo do tempo. Assim, a escolha do *Python* como linguagem de programação para implementar o projeto foi fundamentada na sua facilidade de compreensão, a sua versatilidade e a sua robusta comunidade de programadores, garantindo assim uma base sólida e eficiente para a criação do *chatbot* da escola.

5 EXECUÇÃO DAS AÇÕES

Neste capítulo são apresentadas detalhadamente as ações que foram executadas para a construção do modelo. Começou-se por decidir quais os dados que se deveriam recolher, de seguida seguiu-se uma série de ações de desenvolvimento dos diversos códigos necessários para a construção do sistema. Por fim, a configuração na *Meta Developer*, a ferramenta fornecida pelo *Facebook* para criação e integração de aplicações. Este capítulo também serve como guia para criação de um *chatbot* personalizado.

5.1 Recolha dos dados

Neste projeto, a recolha de dados consistiu em reunir informações essenciais sobre a escola, os seus cursos e outra informação de carácter mais geral. No âmbito do atual projeto, houve uma concentração exclusiva na recolha de informações básicas existentes no *site* da escola. A recolha da informação associada a perguntas mais frequentes feitas aos funcionários dos serviços académicos é indicada para trabalhos futuros.

A fonte primária dos dados é o *site* oficial da escola, localizado em https://www.iscac.pt/. Utilizando a técnica de *web scraping*, implementada com a linguagem de programação *Python*, desenvolveu-se um conjunto de códigos que foram capazes de extrair informações de cada página do menu do site.

É importante notar que, durante a realização deste projeto, a escola lançou uma nova versão do *site* em 22 de maio de 2024. Consequentemente, os códigos desenvolvidos neste trabalho não são compatíveis com a nova versão do site, devido às diferenças na estrutura do HTML.

Foi possível aceder e extrair os dados de forma automatizada através de *web scraping*, navegando pelas diferentes páginas do site e recolhendo as principais informações aí alojadas. Isso permitiu obter uma ampla gama de dados, desde descrições detalhadas dos programas académicos até informações sobre a história da escola, contactos, eventos e outras informações úteis para os alunos e outras partes interessadas.

Ao concentrar os nossos esforços na recolha das informações básicas, foi possível estabelecer uma base sólida para o desenvolvimento futuro do *chatbot*. Como trabalhos

futuros, propõe-se a expansão do âmbito desta implementação de forma que inclua a recolha e análise das perguntas frequentes dos funcionários dos serviços académicos, utilizando métodos semelhantes de extração de dados.

5.1.1 Desenvolvimento do *web scraping*

No desenvolvimento do *web scraping*, o primeiro passo foi estudar as características do *site* para compreender o que precisa de ser recolhido e como fazer isso. O site da ISCAC utilizado era composto por uma lista de menus, cada um contendo sublistas, em que cada página foi obtida por meio de uma ligação. Portanto, ao recolher todos as ligações deste menu, pode-se aceder a todo o conteúdo das páginas do site.

Em seguida, foi necessário identificar e localizar os conteúdos específicos para obter as informações necessárias. A Figura 5-1 apresenta um exemplo da parte dos dados que é extraída do *site* da Escola. De acordo com o exemplo apresentado, recolheram-se apenas os conteúdos referentes à parte da página que está dentro do retângulo vermelho, excluindo-se os conteúdos que estão acima e ao lado direito deste retângulo, pois essas informações são repetitivas e desnecessárias.

Figura 5-1: Exemplo da parte dos dados que é extraída

Após esta etapa, avançou-se para a criação do código de *web scraping*, o qual fez uso de três bibliotecas do *Python: requests, Selenium e BeautifulSoup*. Uma função foi então

desenvolvida para conter todas as ações necessárias. A Figura 5-2 apresenta a primeira parte do código desenvolvida para a recolha dos dados do *site* ISCAC.pt.

É importante destacar que, devido ao menu da página ser gerado através de Java*Script* e exibir as subclasses somente quando o cursor do rato ser posicionado sobre ele, é essencial utilizar o *webdriver.Chrome* do *Selenium*, o que requer a utilização do navegador Google Chrome para iniciar esse processo.

O código foi responsável por extrair todo o conteúdo da página inicial da ISCAC e analisar os elementos HTML pertinentes, garantindo a obtenção das informações necessárias.

```
import requests

from selenium import webdriver

from bs4 import BeautifulSoup

def iscacScraping(url, output_file):

    driver = webdriver.Chrome()  #Garante que tem google chrome baixado no
computador

    driver.get(url)

    # Procurar todos conteúdos da página

    html_content = driver.page_source

    # Usar biblioteca BeautifulSoup para analisar conteúdos do HTML

    soup = BeautifulSoup(html_content, 'html.parser')
```

Figura 5-2: Passo 1 do código para recolha dos dados do site ISCAC.pt

Avançou-se em seguida para a etapa de extrair as ligações do menu principal da página. No entanto, nem todas as ligações são relevantes para os nossos propósitos. Portanto, foi essencial extrair apenas as ligações que estavam contidas na classe do menu.

A Figura 5-3 apresenta o segundo passo do código para recolha dos dados do site ISCAC.pt. Primeiro, o código encontrou o elemento <ul> com a classe *sf-menu* e o id *example*. Em seguida, o código verificou se esse elemento foi encontrado e, se sim, extraiu todos as ligações (<a>) dentro dele. Para cada ligação encontrada, o código recuperou o atributo 'href', que contém o *URL* da ligação, e imprime-o na tela para fins

de verificação. Isso garantiu que apenas as ligações dentro da estrutura do menu sejam extraídas e consideradas para futuras análises ou ações.

```
# extrair os links do <ul class="sf-menu sf-js-enabled sf-arrows" id="example">
ul_element = soup.find('ul', class_='sf-menu', id='example')
if ul_element:
    links = ul_element.find_all('a', href=True)
    for link in links:
        link_href = link['href']
        print("Link:", link_href)
```

Figura 5-3: Passo 2 do código para recolha dos dados do site ISCAC.pt

Após a extração das ligações do menu principal, o próximo passo foi percorrer a lista de ligações extraídas e simular o comportamento do navegador para obter o conteúdo de cada página. Para isso, utilizou-se a biblioteca *Selenium*, que permitiu automatizar a interação com o navegador.

No trecho de código fornecido Figura 5-4: Passo 3 do código para recolha dos dados do site ISCAC.pt, o *Selenium* foi utilizado para abrir cada *link* em um navegador real. O método *driver.get(link_href)* foi empregue para carregar a ligação em questão no navegador controlado pelo *Selenium*. Em seguida, o conteúdo HTML da página foi capturado utilizando o método *driver.page_source*.

Posteriormente, o *BeautifulSoup* foi utilizado para analisar o conteúdo HTML da página recém-carregada. Esta etapa é fundamental para extrair as informações relevantes da página e processá-las conforme necessário. O *BeautifulSoup* permitiu navegar na estrutura HTML da página de forma eficiente e extrair os elementos desejados para posterior processamento.

Uma vez que o conteúdo da página foi analisado com o *BeautifulSoup*, pôde-se proceder à escrita desses conteúdos em um arquivo TXT, conforme requisitado. Este arquivo pôde então ser utilizado para armazenar ou processar as informações extraídas das páginas.

```
            # Usar Selenium para simular o comportamento do navegador e obter o
conteúdo do link
            driver.get(link_href)
            link_html_content = driver.page_source
            # Usar BeautifulSoup para analisar os conteúdos do link
            link_soup = BeautifulSoup(link_html_content, 'html.parser')
            # Extrair text do <div class="coluna_conteudo">
            coluna_conteudo = link_soup.find('table', class_='content_area')
            if coluna_conteudo:
                content_text = coluna_conteudo.get_text(separator='\n', strip=True)
                print("Content Text:", content_text)
                # excrever o text num ficheiro txt
                with open(output_file, 'a', encoding='utf-8') as file:
                    file.write(f"\n\n=== Link: {link_href} ===\n")
                    file.write(f"\n=== Content Text ===\n{content_text}\n")
                    print(f"Successfully wrote content for link: {link_href}")
    # fechar o navegador
    driver.quit()
```

Figura 5-4: Passo 3 do código para recolha dos dados do site ISCAC.pt

Por fim, para executar o código acima e iniciar o processo de recolha de dados, foi necessário indicar a ligação da escola e o local onde se deseja guardar os dados recolhidos.

A Figura 5-5 apresenta o passo 4 do código para recolha dos dados do site ISCAC.pt. Como pode ser constatado, start_*url* correspondeu à ligação da página inicial da escola, enquanto *output_file* é o caminho para o arquivo onde os dados recolhidos foram guardados. Após configurar essas variáveis com os valores desejados, o programa executou a função *iscacScraping*, que contém toda a lógica para recolher os dados da página da escola e salvar no arquivo especificado. No final da execução, uma mensagem foi impressa indicando que o processo de recolha e armazenamento dos conteúdos foi concluído com sucesso.

```
if __name__ == "__main__":

    start_url = "https://www.iscac.pt/index.php?lang=PT"

    output_file = "D:/OneDrive - ISCAC/MADSAD/Tese Mestrado/output_td_content2.txt"

    iscacScraping(start_url, output_file)

    print("Finished crawling and saving content. Content saved to", output_file)
```

Figura 5-5: Passo 4 do código para recolha dos dados do site ISCAC.pt

Em resumo, este código realizou a recolha de dados da página da escola ISCAC, navegando pelo menu principal, obtendo cada ligação e guardando o conteúdo das páginas em um arquivo do tipo "TXT". Ao executar o código, pôde-se obter uma cópia do conteúdo da página para análise ou processamento posterior.

Ao executar o código completo (ANEXO 1), foi gerado um documento de texto. A Figura 5-6 apresenta um exemplo de parte dos dados recolhidos e armazenados.

```
Ficheiro  Editar  Ver

=== Link: https://www.iscac.pt/index.php?m=3_14&lang=PT ===

=== Content Text ===
Escola
Contactos
ISCAC | COIMBRA BUSINESS SCHOOL
Quinta Agrícola - Bencanta
3045-601 Coimbra
Tel: +351 239 802 000
E-Mail:
presidencia@iscac.pt
URL:
www.iscac.pt
GPS: N 40º 12' 34.50"  W 8º 27' 7.00"


=== Link: https://www.iscac.pt/index.php?m=3_412&lang=PT ===

=== Content Text ===
Escola
Localização
Coordenadas GPS:     N: 40º 12' 34.50"  W: 8º 27' 7.00"
Ver mapa maior
COMO CHEGAR
De carro
Partindo do Porto (Norte) ou de Lisboa (Sul) utilizando a auto-estrada A1:
deve sair em Coimbra-Sul/Alfarelos (a 100 km do Porto e 200 km de Lisboa) e seguir pelo IC2 em direcção a Coimbra. Antes de chegar a Coimbra, sair do IC2 na
saída Bencanta- ISCAC e seguir as indicações "ISCAC".
Vindo de Vilar Formoso (fronteira Portugal/Espanha, perto de Salamanca):
prosseguir na auto-estrada A25 até Aveiro (170 km desde a fronteira). Ao chegar à saída Aveiro-Norte sair para a auto-estrada A1 na direcção Sul (Lisboa). Na
auto-estrada A1, sair em Coimbra-Sul/Alfarelos (50 km depois de Aveiro)e seguir pelo IC2 em direcção a Coimbra. Antes de chegar a Coimbra, sair do IC2 na
saída Bencanta- ISCAC e seguir as indicações "ISCAC".
De autocarro
Consultar os horários da
Ln 1, Col 1   243 293 carateres   120%   Windows (CRLF)   UTF-8
```

Figura 5-6: Exemplo de parte dos dados recolhidos e armazenados

5.2 Treino do modelo

A criação e treino do modelo foram realizados no site da *OpenAI*, onde foi possível criar um assistente e fornecer a ele dados privados para personalizá-lo como um *chatbot* de IA para atender às necessidades específicas deste projeto.

Neste processo, além de fornecer os dados necessários, também foi essencial escrever uma indicação para orientar o modelo sobre sua função. A indicação (*prompt*) para orientar o *chatbot* foi a seguinte:

> *"És um atendimento de cliente da escola ISCAC. Dê respostas precisas e esclarecedoras. Só tens como função responder aos clientes com o objetivo de fornecer informações sobre o ISCAC. Se for fora disso, algo que viole a privacidade, algo não escolar ou não souberes a resposta, responde de seguinte forma:*
>
> *'Pergunta fora do âmbito.*
>
> *Contacta-nos por:*
>
> *Telefone: 239 802 018*
>
> *E-mail: academicos@iscac.pt'"*

Esta orientação garante que o *chatbot* "compreenda" claramente o seu propósito e limite a sua atuação, fornecendo um serviço consistente e útil aos utilizadores. A Figura 5-7 apresenta a configuração das instruções para a aprendizagem do modelo.

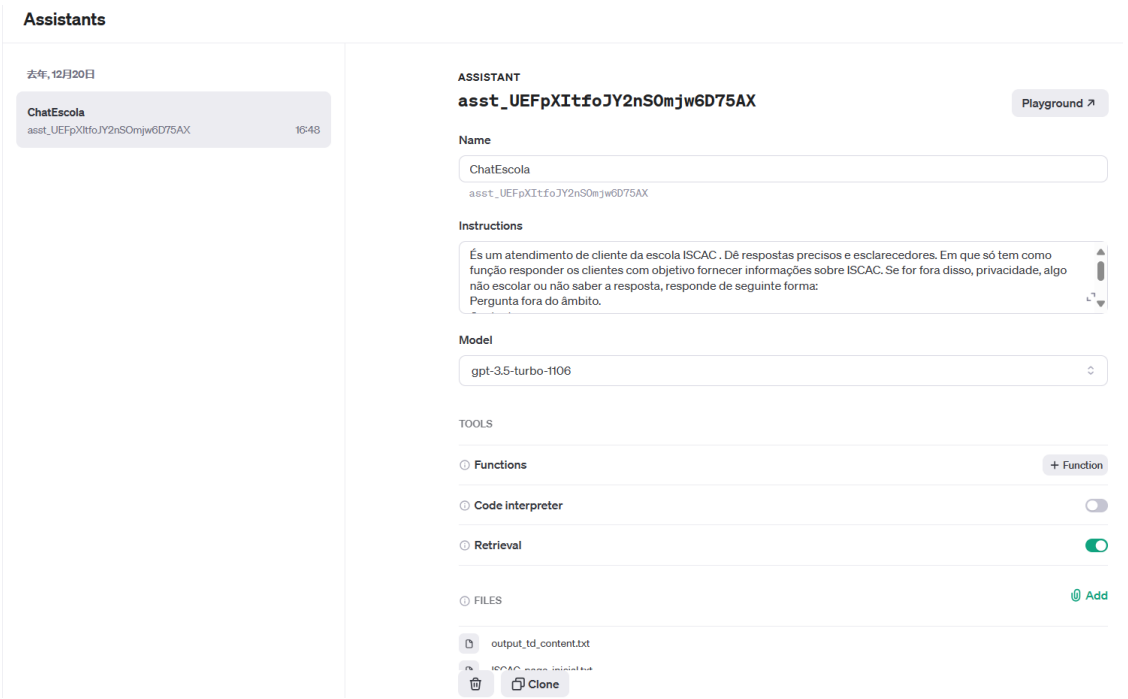

Figura 5-7: Configuração das instruções para a aprendizagem do modelo.

5.3 Utilização da API

A utilização da API ocorre em duas etapas: a conexão com o *OpenAI* para gerar respostas e a conexão com o *Facebook* para disponibilizá-las aos clientes. A Figura 5-8 apresenta um enquadramento do funcionamento do *chatbot*.

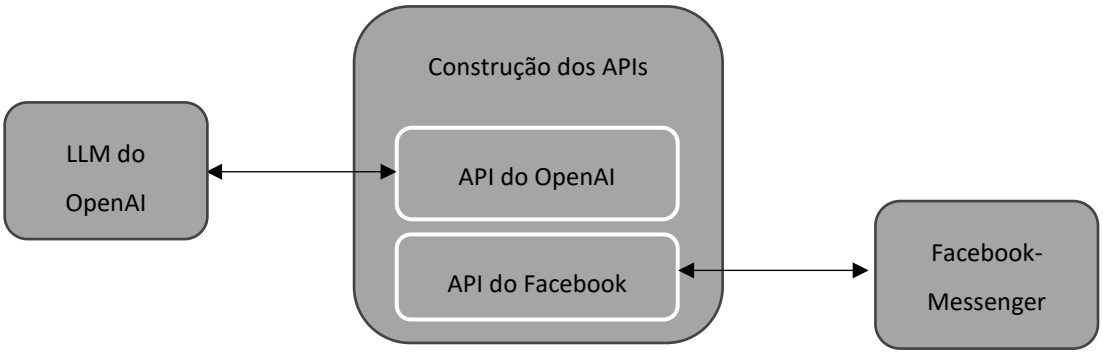

Figura 5-8: Enquadramento do funcionamento do chatbot

No que diz respeito à conexão com a plataforma de desenvolvimento do *Facebook*, o processo ocorre em dois passos: primeiro, é realizada uma conexão de teste que retorna a mensagem enviada pelo cliente; em seguida, ocorre a integração com o *OpenAI* para criar um sistema completo capaz de responder às perguntas dos utilizadores de forma eficaz e abrangente.

Essa abordagem permite que o *chatbot* interaja de forma dinâmica com os clientes, respondendo às suas consultas de maneira precisa e satisfatória.

5.3.1 Teste do API do *OpenAI*

No que diz respeito a testes do API da *OpenAI*, numa primeira etapa, o objetivo foi realizar um teste de conexão com esta API. Isso foi fundamental para garantir que o protótipo pudesse interagir com os serviços fornecidos pela *OpenAI* de forma adequada.

Foi necessário definir dois parâmetros: a chave da API e o ID do assistente, obtidos no site oficial da *OpenAI*. Após obter os parâmetros, foi preciso importar a biblioteca *OpenAI* e configurar esses valores no código. Essa etapa foi crucial para iniciar a integração com a API da *OpenAI*, permitindo que o *chatbot* utilize os recursos disponíveis na plataforma.

Com os parâmetros de conexão configurados, o código ficou pronto para estabelecer a comunicação com a API da *OpenAI*, permitindo que o *chatbot* iniciasse a interação com a plataforma e explorasse suas funcionalidades (Figura 5-9).

```
from OpenAI import OpenAI

chave = ""

assis_id = ""
```

Figura 5-9: Passo 1 do código da ligação ao API do OpenAI

Depois de ter os valores preparados, o próximo passo foi conectar o modelo treinado e realizar a entrada da pergunta e a saída da resposta, seguindo para isso o guia oficial (*OpenAI*, 2024). Para isso, estabeleceu-se a conexão com a *OpenAI* utilizando a chave de API fornecida. Em seguida, criou-se uma *thread* para simular um utilizador a fazer uma pergunta (Figura 5-10).

```
client = OpenAI(api_key=chave)

ask = input()

thread = client.beta.threads.create(messages=[{"role":"user",
                                               "content":ask
                                               }])
```

Figura 5-10: Passo 2 do código da ligação ao API do OpenAI

Após a criação da *thread* para simular a interação do utilizador com o modelo treinado, o próximo passo foi executar essa interação. Isso foi feito através da criação de uma execução da *thread*, utilizando o ID da *thread* e o ID do assistente. Em seguida, o código imprimiu o ID dessa execução, o que foi útil para referência posterior (Figura 5-11).

```
run = client.beta.threads.runs.create(thread_id=thread.id, assistant_id=assis_id)
print("run id:", run.id)
```

Figura 5-11: Passo3 do código da ligação ao API do OpenAI

Em seguida, tal como pode ser observado na Figura 5-12, implementou-se um ciclo "*while*" para verificar o "status" da execução da *thread*. Enquanto o "*status*" não for "*completed"* (completo), o código continuará a verificar o estado da execução. Assim que

o estado se torne "*completed*", o *loop* é encerrado e uma mensagem a indicar que a execução está completa é impressa.

```
hile run.status != "completed":

    run = client.beta.threads.runs.retrieve(thread_id=thread.id, run_id=run.id)

else:

    print("run complete")
```

Figura 5-12: Passo 4 do código da ligação ao API do OpenAI

Após a execução da *thread* ter sido concluída, o próximo passo foi obter as mensagens trocadas durante essa interação. Isso foi feito através da listagem das mensagens associadas à *thread* (Figura 5-13).

```
message_response = client.beta.threads.messages.list(thread_id=thread.id)
messages = message_response.data
```

Figura 5-13: Passo 5 do código da ligação ao API do OpenAI

Por fim, de acordo com a Figura 5-14, foi extraída a pergunta feita pelo utilizador e a resposta gerada pelo modelo. Isso foi feito obtendo o conteúdo da primeira mensagem na lista de mensagens.

```
response = messages[0]

print("Pergunta:\n", ask, "\n")
print("Resposta:\n", response.content[0].text.value)
```

Figura 5-14: Passo 6 do código da ligação ao API do OpenAI

Ao correr o código completo (ANEXO 2), pôde-se imprimir a pergunta feita pelo utilizador e a resposta gerada pelo modelo, permitindo uma inspecção e análise do diálogo entre o utilizador e o modelo.

Na Figura 5-15 apresenta-se um exemplo de uma resposta gerada pelo *chatbot*, em que este respondeu com sucesso à pergunta relacionada ao horário de atendimento da secretaria.

```
···   run id: run_4QgdLeCiyg9U7iUsC606zJwb
      run complete
      Pergunta:
       a que horas posso ir sao secretaria?

      Resposta:
       O horário de atendimento presencial na secretaria da escola é o seguinte:
      - Terça-feira: 9:30 às 13:00
      - Quarta-feira: 9:30 às 13:00
      - Sexta-feira: 14:00 às 19:30 (fora do período letivo encerra às 16:30)

      O atendimento telefônico ocorre nos seguintes horários:
      - Segunda-feira: 14:00 às 17:00
      - Terça, quarta, quinta e sexta-feira: 9:00 às 12:00 e 14:00 às 17:00【9†fonte】.
```

Figura 5-15: Exemplo de uma resposta gerada

5.3.2 Teste da ligação à API do Messenger do *Facebook*

Como o sistema requer duas partes distintas para funcionar corretamente, nesta etapa o objetivo foi garantir a receção e o retorno das mensagens através da API do *Facebook*. Este foi um teste de viabilidade, onde a lógica do código se concentrou em permitir que o sistema reconhecesse a mensagem enviada pelo utilizador e enviasse a mesma mensagem de volta. Esta abordagem simplificada ajudou a verificar se a comunicação básica entre o sistema e o utilizador estava a funcionar corretamente. O código explica-se em 8 passos.

Começou por utilizar o *framework Flask* em *Python* para criar um aplicativo *web*. *Flask* é uma biblioteca popular que permite criar aplicativos *web* de forma eficiente e simples em *Python* (Figura 5-16). Em primeiro lugar, as classes *Flask* e *request* são importadas do módulo *Flask*. A classe *Flask* é usada para criar o aplicativo *web*, enquanto a classe *request* é utilizada para processar solicitações HTTP recebidas pelo aplicativo. Em seguida, os módulos *json* e *requests* foram importados. O módulo *json* é utilizado para trabalhar com dados no formato JSON, enquanto o *requests* é uma biblioteca HTTP que permite enviar solicitações HTTP para recursos na *web*.

```
from Flask import Flask, request

import json

import requests
```

Figura 5-16: Passo 1 do código da ligação ao API do Facebook

Posteriormente, uma instância da classe *Flask* foi criada e atribuída à variável app. A seguir, foram definidas duas variáveis, respetivamente *PAGE_ACCESS_TOKEN* e *LATEST_API_VERSION*. A variável *PAGE_ACCESS_TOKEN* armazena o *token* de acesso à página do *Facebook*. Este *token* foi necessário para autenticar as solicitações feitas à API do *Facebook*. A variável *LATEST_API_VERSION* armazena a versão mais recente da API do *Facebook* a ser utilizada (Figura 5-17).

```
app = Flask(__name__)

PAGE_ACCESS_TOKEN =
"EAAWNYKpX3uE1SLOvcTivnS4MZCxmfNrjXDZCkR3Oqk95KA3590IVRSg6DzqonROW7ZBz083btFZAu4RaA
Mp6siBO305sZD"

LATEST_API_VERSION = "v19.0"
```

Figura 5-17: Passo 2 do código da ligação ao API do Facebook

Posteriormente, foi definida a *URL* da API do *Facebook* que foi utilizada para enviar mensagens. A *URL* é composta pela versão mais recente da API, o *endpoint* */me/messages*, e o *token* de acesso à página (Figura 5-18).

```
API=
"https://graph.Facebook.com/"+LATEST_API_VERSION+"/me/messages?access_token="+PAGE_
ACCESS_TOKEN
```

Figura 5-18: Passo 3 do código da ligação ao API do Facebook

No próximo passo, foi definida uma rota na aplicação *web Flask* usando o *decorator @app.route()*. Esta rota foi obtida através do caminho /hello/ e pode lidar com solicitações HTTP do tipo GET e POST. Quando um cliente faz uma solicitação GET ou POST para a rota */hello/*, a função *hello_world()* é executada. Essa função simplesmente retorna a string "Hello World at localhost:8080", indicando que o servidor está a funcionar corretamente e pode ser alcançado através do endereço *localhost* na porta 8080 (Figura 5-19).

```
@app.route('/hello/', methods = ['GET', 'POST'])

def hello_world():

return "Hello World at localhost:8080"
```

Figura 5-19: Passo 4 do código da ligação ao API do Facebook

De seguida, temos uma função chamada *send_message(sender_id, message_text)*. Esta função é responsável por enviar uma resposta de volta ao utilizador utilizando a API do *Facebook Graph*.

Nesta função, é feita uma solicitação POST para a API do *Facebook* Graph, utilizando a biblioteca *requests*. Os parâmetros da solicitação incluem o *token* de acesso à página (*PAGE_ACCESS_TOKEN*), o cabeçalho com o tipo de conteúdo sendo enviado (*Content-Type*) e os dados da mensagem em formato JSON.

A mensagem é enviada para o destinatário cujo ID é fornecido como *sender_id*, e o texto da mensagem é especificado em *message_text*.

Após o envio da mensagem, o conteúdo da resposta HTTP é impresso na consola. Isso pode ser útil para depurar e verificar se a mensagem foi enviada com sucesso (Figura 5-20).

```
def send_message(sender_id, message_text):

    '''Sending response back to the user using Facebook graph API'''

    #print("sending message to {recipient}: {text}".format(recipient=sender_id,
text=message_text))

    r=requests.post("https://graph.Facebook.com/"+LATEST_API_VERSION+"/me/messages"
, params={"access_token": PAGE_ACCESS_TOKEN}, headers={"Content-Type":
"application/json"},  data=json.dumps({"recipient": {"id": sender_id}, "message":
{"text": message_text} }))

    print(r.text)
```

Figura 5-20: Passo 5 do código da ligação ao API do Facebook

No sexto passo, temos uma função chamada *handle_verification*(), decorada com o *@app.route('/')*. Esta função lida com a verificação de segurança necessária para conectar o aplicativo ao *webhook* do *Facebook Messenger*.

Quando o aplicativo recebe uma solicitação GET na raiz (‘/’), esta função é acionada. Primeiro, verifica se o *token* de verificação recebido (*hub.verify_token*) corresponde ao *token* esperado (“chatescola2024”). Se os *tokens* coincidirem, o servidor retorna o desafio fornecido pelo *Facebook* (*hub.challenge*) como parte da resposta, confirmando a autenticidade do aplicativo. Caso contrário, uma mensagem de erro é retornada indicando que o *token* de validação está incorreto (Figura 5-21).

```
@app.route('/', methods=['GET'])
def handle_verification():
    if (request.args.get('hub.verify_token', '') == "chatescola2024"):
        print("Verified")
        return request.args.get('hub.challenge', '')
    else:
        print("Wrong token")
        return "Error, wrong validation token"
```

Figura 5-21: Passo 6 do código da ligação ao API do Facebook

No sétimo passo, temos uma função chamada *handle_message()*, também decorada com @app.route('/'). Essa função é responsável por lidar com as mensagens recebidas pelo aplicativo do *Facebook* Messenger.

Quando o aplicativo recebe uma solicitação POST na raiz ('/'), esta função é acionada. Primeiro, ela verifica se o objeto recebido é uma página do *Facebook (data["object"] == “page”)*, o que indica que a mensagem recebida é do Messenger. Em seguida, itera sobre as entradas (*entries*) e os eventos de mensagens (*messaging events*) para processar cada mensagem recebida. Se a mensagem contiver texto, o texto da mensagem é extraído e enviado à função send_*message*() para ser enviada de volta ao remetente. Finalmente, a função retorna “ok” para confirmar o recebimento da mensagem (Figura 5-22).

```
@app.route('/', methods=['POST'])
def handle_message():
    '''
    Handle messages sent by Facebook messenger to the applicaiton
    '''
    data = request.get_json()
    print(data)
    if data["object"] == "page":
        for entry in data["entry"]:
            for messaging_event in entry["messaging"]:
                if messaging_event.get("message"):
                    sender_id = messaging_event["sender"]["id"]
                    recipient_id = messaging_event["recipient"]["id"]
                    try:
                        message_text = messaging_event["message"]["text"]
                    except:
                        message_text = "erro"
                    send_message(sender_id, message_text)
    return "ok"
```

Figura 5-22: Passo 7 do código da ligação ao API do Facebook

No passo final, temos uma verificação *if __name__ == '__main__'*: que garante que o servidor *Flask* seja iniciado apenas se o *script* for executado diretamente, e não se for importado por outro *script*.

Se este *script* estiver sendo executado diretamente, o *Flask* iniciará o servidor na máquina *local (host='0.0.0.0')* na porta 8080. Isso permite que o aplicativo *web Flask* esteja acessível através do endereço IP da máquina na porta 8080 (Figura 5-23).

```
if __name__ == '__main__':
    app.run( host = '0.0.0.0', port = 8080 )
```

Figura 5-23: Passo 8 do código da ligação ao API do Facebook

Em resumo, todo este código configura um servidor *Flask* que interage com a API do *Facebook Messenger* para receber mensagens de utilizadores, processá-las e enviar respostas de volta aos utilizadores. Ele inclui funções para verificar a autenticidade do aplicativo, manipular mensagens recebidas e enviar mensagens de resposta.

Ao executar o código completo, que pode ser visto no ANEXO 3, foi possível observar que o sistema foi capaz de realizar a ação conforme apresentado na Figura 5-24, retornando as mensagens enviados pelo utilizador de volta.

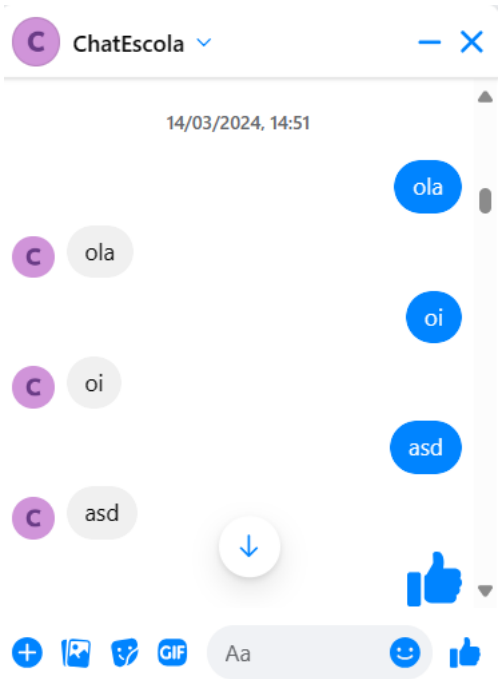

Figura 5-24: Exemplo do funcionamento do API do Facebook

É importante notar que, para o API do Facebook poder funcionar corretamente, é essencial que as configurações indicadas na secção Configurações no Meta Developer sejam feitas adequadamente.

5.3.3 Ligação do *Facebook* com o modelo do *OpenAI*

Após os dois testes bem-sucedidos de ligação, que demonstraram a viabilidade das duas partes individualmente, nesta etapa o objetivo foi integrar essas partes. O objetivo foi permitir que as mensagens recebidas do *Facebook* fossem processadas pelo modelo da *OpenAI* e que as respostas geradas fossem enviadas de volta ao utilizador.

No primeiro passo (Figura 5-25), foi necessário juntar os valores indispensáveis das duas partes, sem empregar técnicas específicas. Inicializaram-se apenas as variáveis essenciais para uso subsequente. Essas variáveis são cruciais para o funcionamento correto da integração entre o *Facebook* e o modelo de *OpenAI*.

```
from Flask import Flask, request

import json

import requests

from OpenAI import OpenAI

app = Flask(__name__)

PAGE_ACCESS_TOKEN =
"EAAWNYKpXnL4BOzS2uZB0ZBXZCU7o0XsjuuqfO7c9oAFJqirWsTOuW8hH64ZBLP7CZAzVfRnrG7SCytAZA
FcWLHsngSh2WlBzBcZBVqHSgWCnQ1XKne3uE1SLOvcTivnS4MZCxmfNrjXDZCkR3Oqk95KA3590IVRSg6Dz
qonROW7ZBz083btFZAu4RaAMp6siBO305sZD"

LATEST_API_VERSION = "v19.0"

API =
"https://graph.Facebook.com/"+LATEST_API_VERSION+"/me/messages?access_token="+PAGE_
ACCESS_TOKEN

chave = "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"

assis_id = "asst_UEFpXItfoJY2nSOmjw6D75AX"

OpenAI_client = OpenAI(api_key=chave)
```

Figura 5-25: Passo 1 do código da ligação do Facebook com o OpenAI

Nesta parte do código, não há nenhuma parte relacionada com o *OpenAI*. As funções *hello_world()* e *handle_verification()* permanecem como antes, lidando com as solicitações GET e POST e verificando a autenticação do aplicativo, respetivamente. A

função *send_message()* também permanece, sendo responsável por enviar respostas de volta aos utilizadores usando a API do *Facebook Graph* (Figura 5-26).

```
@app.route('/hello/', methods=['GET', 'POST'])
def hello_world():
    return "Hello World at localhost:8080"
def send_message(sender_id, message_text):
    '''
    Sending response back to the user using Facebook graph API
    '''
    r =
requests.post("https://graph.Facebook.com/"+LATEST_API_VERSION+"/me/messages",
                      params={"access_token": PAGE_ACCESS_TOKEN},
                      headers={"Content-Type": "application/json"},
                      data=json.dumps({
                          "recipient": {"id": sender_id},
                          "message": {"text": message_text}
                      }))
    print(r.text)
@app.route('/', methods=['GET'])
def handle_verification():
    if (request.args.get('hub.verify_token', '') == "chatescola2024"):
        print("Verified")
        return request.args.get('hub.challenge', '')
    else:
        print("Wrong token")
        return "Error, wrong validation token"
```

Figura 5-26: Passo 2 do código da ligação do Facebook com o OpenAI

No terceiro passo (Figura 5-27), ocorre a junção das duas partes, onde as mensagens recebidas do *Facebook* Messenger são processadas pelo modelo da *OpenAI* e as respostas geradas são enviadas de volta aos utilizadores.

Na função *handle_message(),* quando o servidor *Flask* recebe uma solicitação POST, o código verifica se o objeto recebido é uma página do *Facebook (data["object"] == "page")*. Se for uma página do *Facebook*, ele itera sobre as entradas e os eventos de mensagens para processar cada mensagem recebida.

Para cada mensagem recebida, o código extrai o ID do remetente (*sender_id*) e o texto da mensagem. Em seguida, envia o texto da mensagem para a API da *OpenAI* usando o método *OpenAI_client.beta.threads.create().* Isso cria uma *thread* nova na *OpenAI* com a mensagem do utilizador. Em seguida, é iniciado um processo de execução (*run*) para gerar uma resposta usando o modelo da *OpenAI*.

O código entra em ciclo enquanto o estado do processo de execução não é *completed*. Isso é necessário porque a resposta do modelo da *OpenAI* pode não ser imediata. Uma vez que o processo de execução esteja concluído, a resposta é recuperada da *OpenAI* e enviada de volta ao remetente usando a função *send_message()*. Em caso de erro durante o processamento da mensagem, o erro é capturado e impresso na consola. Por fim, a função retorna "ok" para confirmar o recebimento da mensagem pelo servidor *Flask*.

```
@app.route('/', methods=['POST'])
def handle_message():
    '''    Handle messages sent by Facebook Messenger to the application'''
    data = request.get_json()
    print(data)
    if data["object"] == "page":
        for entry in data["entry"]:
            for messaging_event in entry["messaging"]:
                if messaging_event.get("message"):
                    sender_id = messaging_event["sender"]["id"]
```

```
                    recipient_id = messaging_event["recipient"]["id"]

                    try:

                        message_text = messaging_event["message"]["text"]

                        # Send the received message to OpenAI API

                        thread =
OpenAI_client.beta.threads.create(messages=[{"role": "user","content":
message_text}])

                        run =
OpenAI_client.beta.threads.runs.create(thread_id=thread.id, assistant_id=assis_id)

                        while run.status != "completed":

                            run =
OpenAI_client.beta.threads.runs.retrieve(thread_id=thread.id, run_id=run.id)

                        else:

                            message_response =
OpenAI_client.beta.threads.messages.list(thread_id=thread.id)

                            messages = message_response.data

                            response = messages[0]

                            response_text = response.content[0].text.value

                            send_message(sender_id, response_text)

                    except Exception as e:

                        print("Error:", e)

    return "ok"
```

Figura 5-27: Passo 3 do código da ligação do Facebook com o OpenAI

Por fim, conclui-se o código com a verificação if __name__ == '__main__':, que garante que o servidor *Flask* seja iniciado apenas se o *script* for executado diretamente, e não se for importado por outro *script* (Figura 5-28).

```
if __name__ == '__main__':

    app.run(host='0.0.0.0', port=8080)
```

Figura 5-28: Passo 4 do código da ligação do Facebook com o OpenAI

O ANEXO 4 apresenta o código completo da ligação do Facebook com o OpenAI.

5.3.4 Configurações no *Meta Developer*

Para realizar a conexão com a plataforma do *Facebook*, foi necessário seguir uma série de etapas cuidadosamente delineadas. Em primeiro lugar, foi necessário ter uma página oficial no *Facebook*. Em seguida, foi preciso criar uma conta no *Meta Developer*, que é um serviço fornecido pelo Meta, anteriormente conhecido como *Facebook*, permitindo que utilizadores e empresas criem as suas próprias aplicações para promover os seus negócios.

Para criar uma aplicação que se conecte à página do *Facebook* e possa responder às perguntas recebidas pelo *Messenger*, o processo segue os passos que se seguem.

Passo 1: Ao começar a criar a app, é escolhida a primeira opção: autenticar e solicitar os dados dos utilizadores com login do *Facebook* (Figura 5-29).

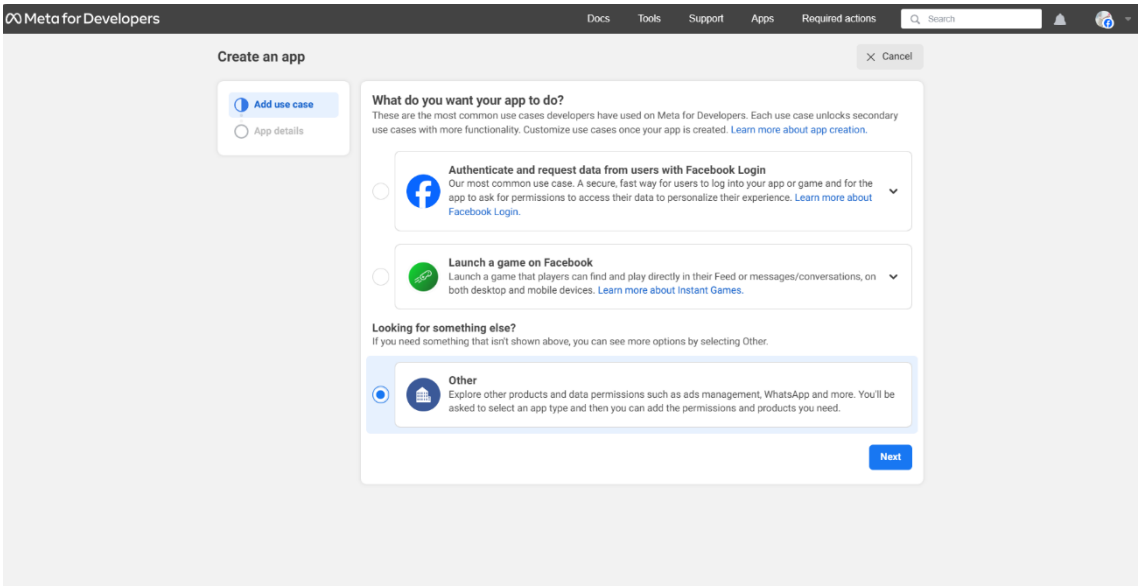

Figura 5-29: Passo 1 da configuração no Meta Developer

Passo 2: Escolher no tipo de aplicação o negócio (Figura 5-30).

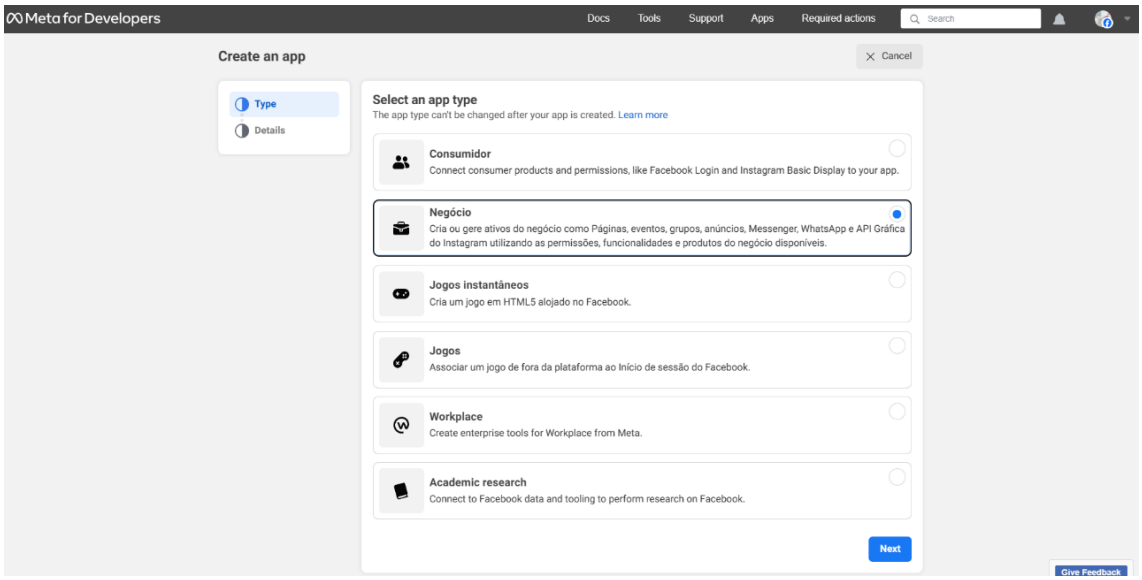

Figura 5-30: Passo 2 da configuração no Meta Developer

Passo 3: Dar um nome à aplicação e associar a um email (Figura 5-31).

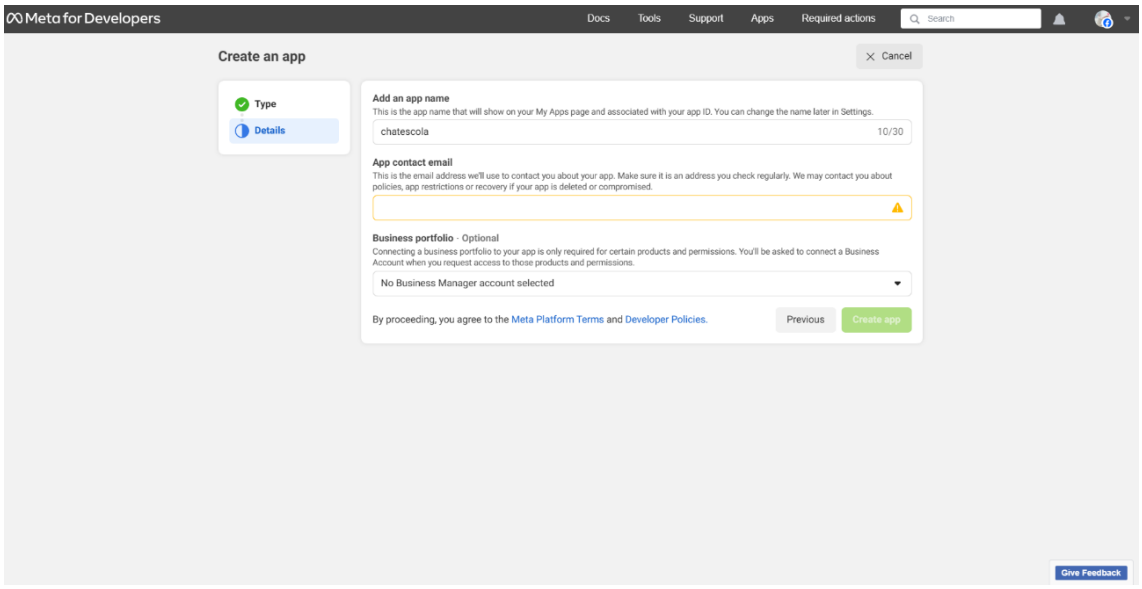

Figura 5-31: Passo 3 da configuração no Meta Developer

Passo 4: Adicionar a aplicação Messenger (Figura 5-32).

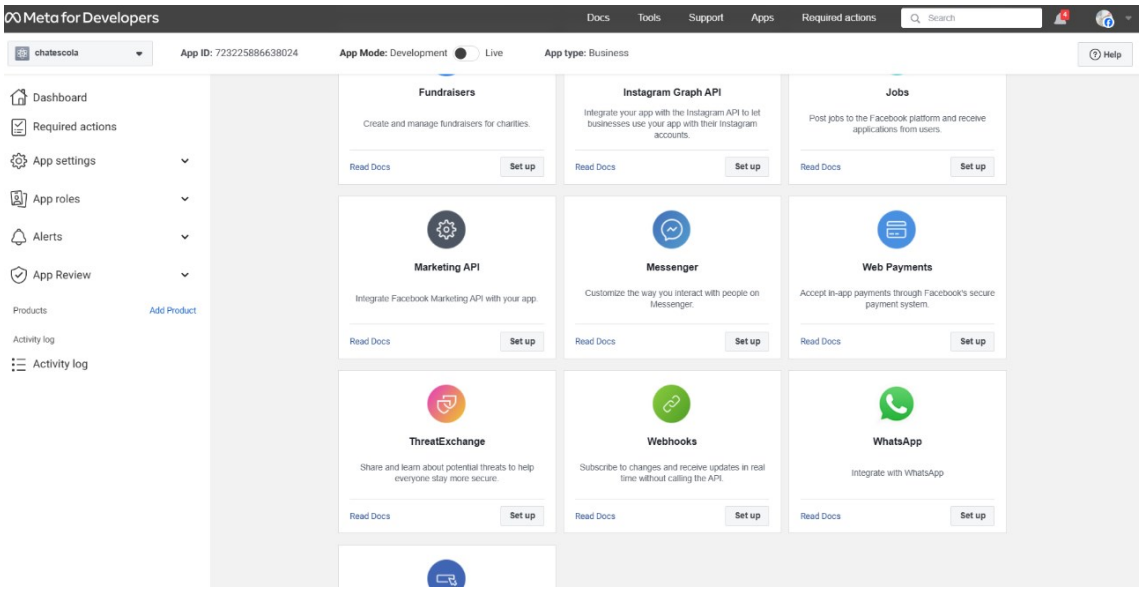

Figura 5-32: Passo 4 da configuração no Meta Developer

Passo 5: Configurar os três passos (Figura 5-33).

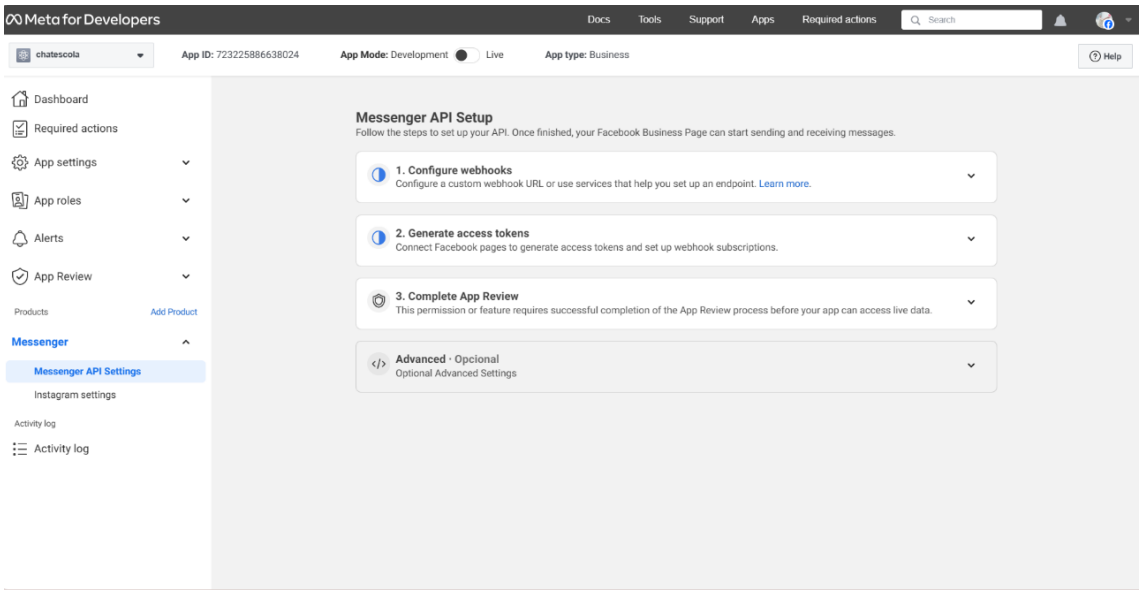

Figura 5-33: Passo 5 da configuração no Meta Developer

Passo 6: Editar um *callback url* para configurar o *webhook*.

Este passo é essencial para fazer a autenticação da transação das mensagens (Figura 5-34).

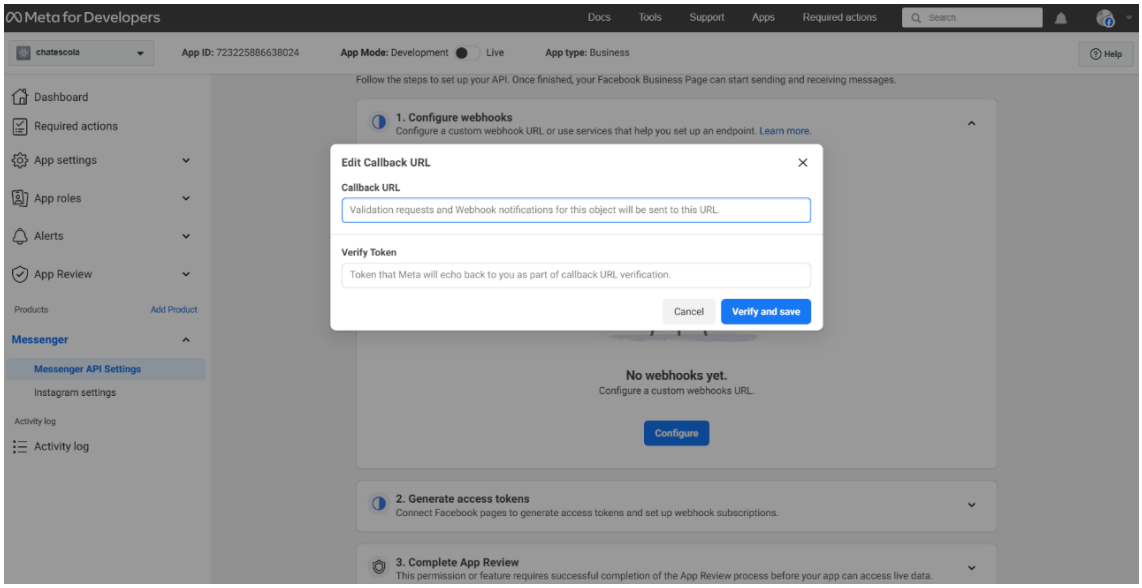

Figura 5-34: Passo 6 da configuração no Meta Developer

Passo 7: Escolher um terceiro servidor do *url* (Figura 5-35).

Optou-se por usar o *Ngrok* para obter o *url* para autenticação do *webhook*. *Ngrok* é uma ferramenta que permite criar um túnel seguro para conectar um servidor local a um servidor remoto, tornando possível expor uma aplicação local para a Internet.

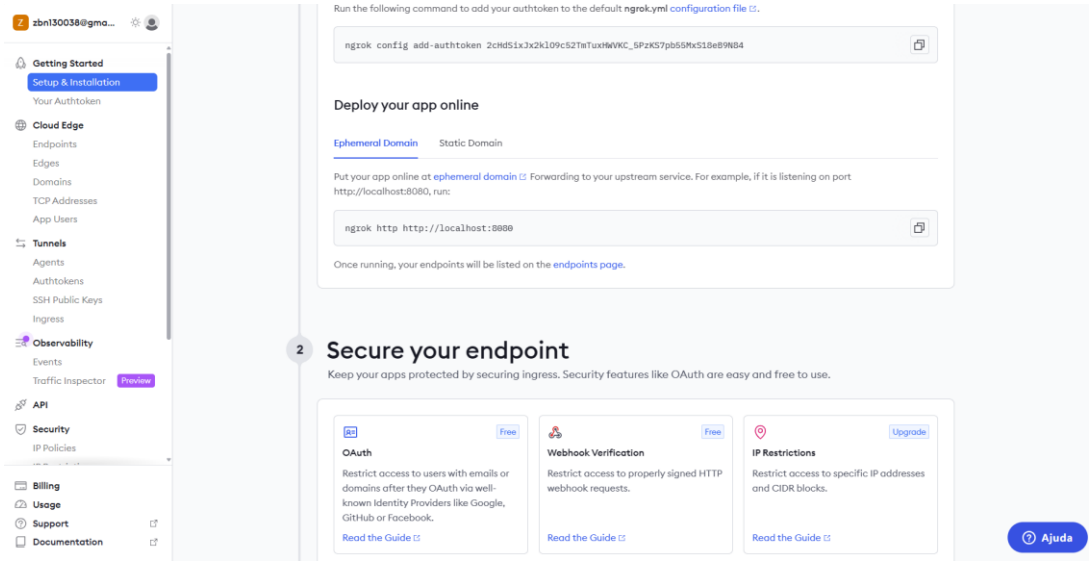

Figura 5-35: Passo 7 da configuração no Meta Developer

Passo 8: Abrir o CMD, baixar o *ngrok*, correr o código oferecido indicado do *ngrok* como "*ngrok http* http://localhost:8080" (Figura 5-36).

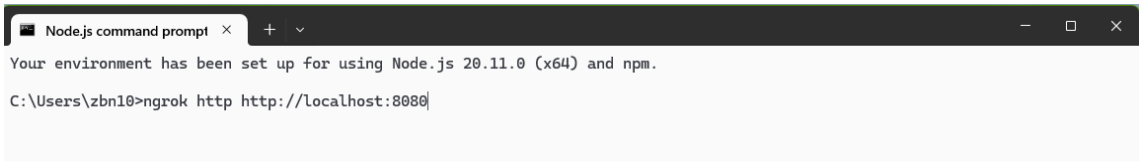

Figura 5-36:Passo 8 da configuração no Meta Developer

Passo 9: Correr o código, sendo logo dado um *link* que possa funcionar como callback *url* (assinalado por linha vermelha) (Figura 5-37).

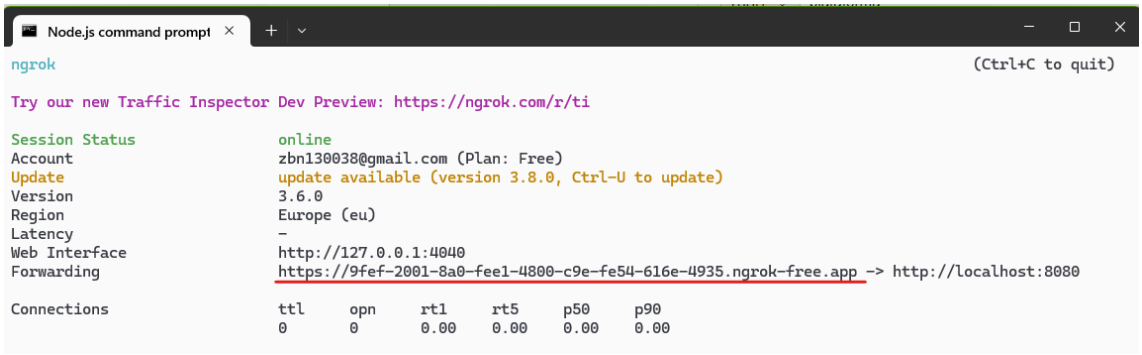

Figura 5-37: Passo 9 da configuração no Meta Developer

Passo 10: Inserir o *url* e criar um verify *token*, mas só sendo possível guardá-lo após ter o código de ligação da Meta Developer estar a correr (Figura 5-38).

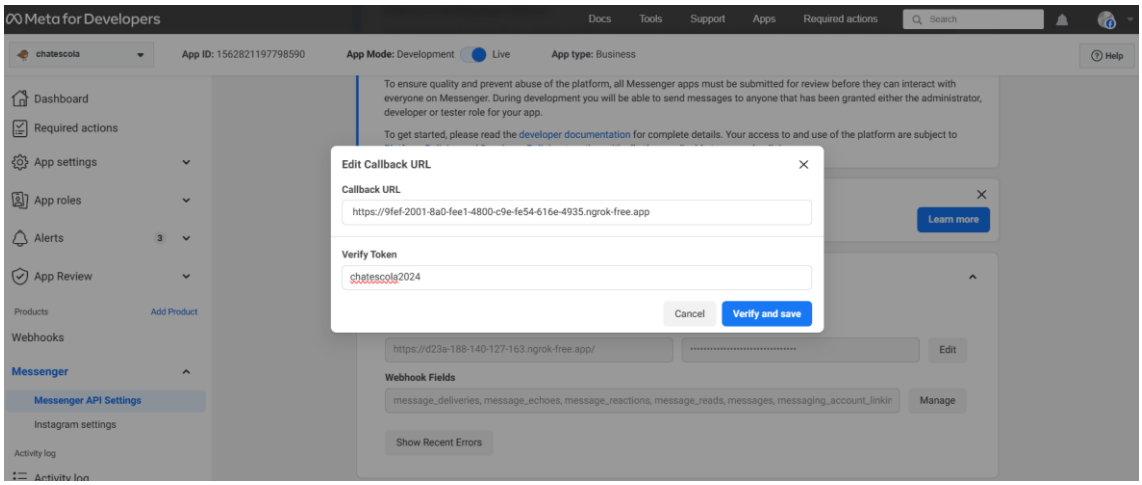

Figura 5-38: Passo 10 da configuração no Meta Developer

Passo 11: A seguir, configurar e conectar a página do *Facebook* e procurar o acess *token*, fundamental para fazer a ligação e a transmissão das mensagens do *Facebook* (Figura 5-39).

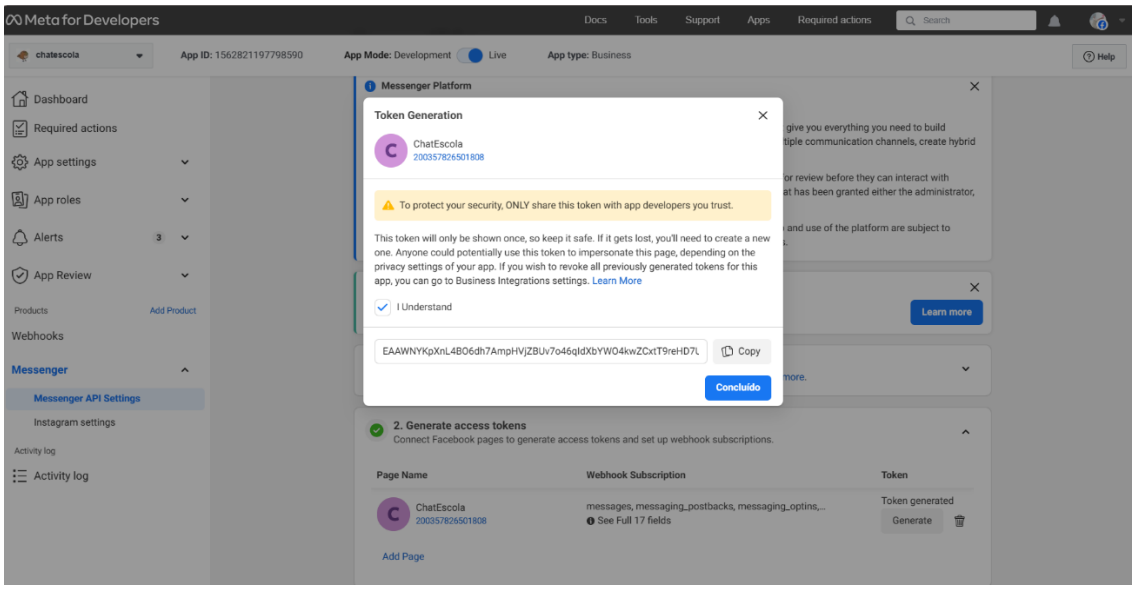

Figura 5-39: Passo 11 da configuração no Meta Developer

Passo 12: Para disponibilizar ao público, tem de se abrir no modelo "*live*", no App *Mode* (Figura 5-40).

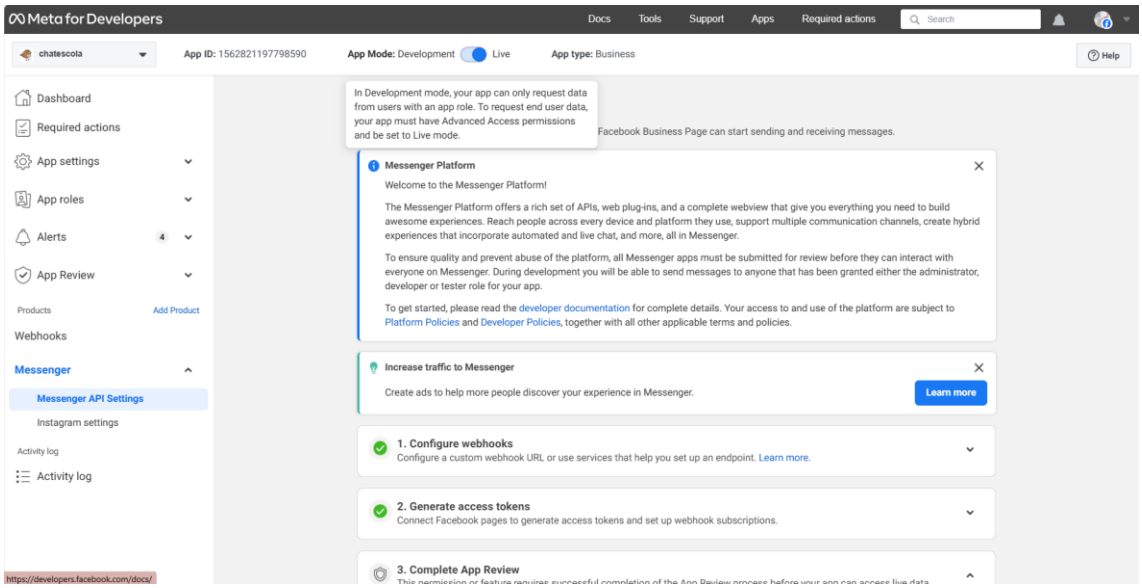

Figura 5-40: Passo 12 da configuração no Meta Developer

Escolha opcional do passo 7:

Como o *Ngrok* funciona ligando-se ao próprio computador, é necessário que o computador pessoal esteja sempre ligado para que o *chatbot* esteja sempre em funcionamento. Assim, optou-se por utilizar o *Glitch*, que permite que o código funcione num servidor na nuvem, assegurando que o sistema esteja em funcionamento contínuo e permanente (Figura 5-41).

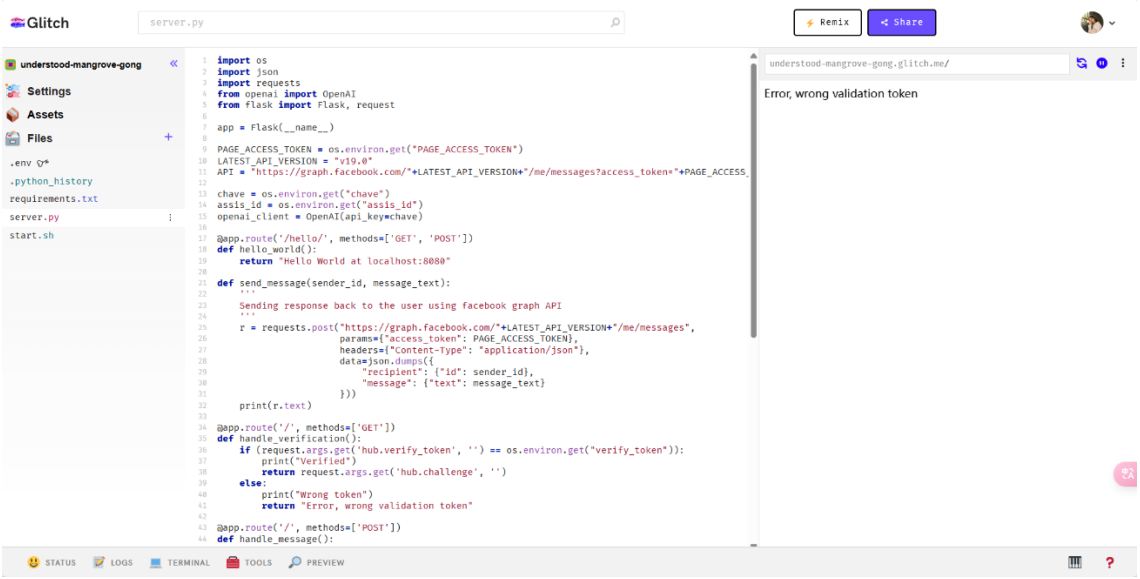

Figura 5-41: Ilustração da utilização do Glitch para correr o programa

6 AVALIAÇÃO

Foi desenvolvido um modelo e *chatbot*, que tem demonstrado um bom desempenho nos testes da resposta, tudo apontando para que seja possível resolver o problema proposto neste projeto. O *chatbot* permite que os clientes recebam respostas diretas ao interagirem através da página do *Facebook*.

Para avaliar o funcionamento do *chatbot*, fizeram-se alguns testes em vários tipos de cenários. Como o *chatbot* só responde com base nos dados fornecidos, avaliação será realizada segundo três perspetivas: perguntas com informações, perguntas sem informações e perguntas sem relacionamento.

6.1 Capacidade de resposta a perguntas com informações

Neste tipo de teste, procuraram-se algumas informações específicas que estejam na base de dados e testou-se a capacidade e a qualidade da resposta do *chatbot*.

Teste 1: Perguntar quem é o diretor de um curso

Este teste pretendeu testar o sistema relativamente ao diretor de um determinado curso. Em particular, o diretor do mestrado em Análise de Dados e Sistemas de Apoio à Decisão (ADSAD). A informação está presente no documento, tal como pode ser observado na Figura 6-1.

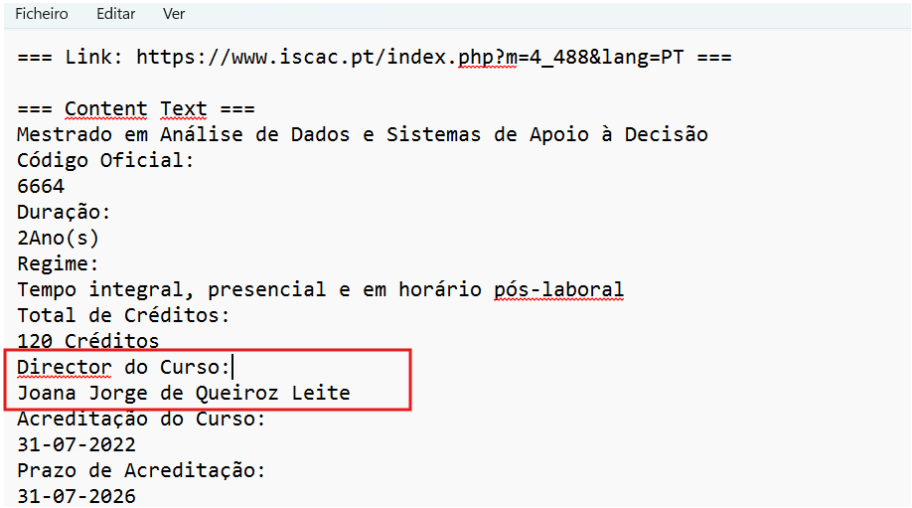

Figura 6-1: Dados existentes sobre o diretor do mestrado em ADSAD

A resposta que o *chatbot* gerou foi correta e bem estruturada (Figura 6-2).

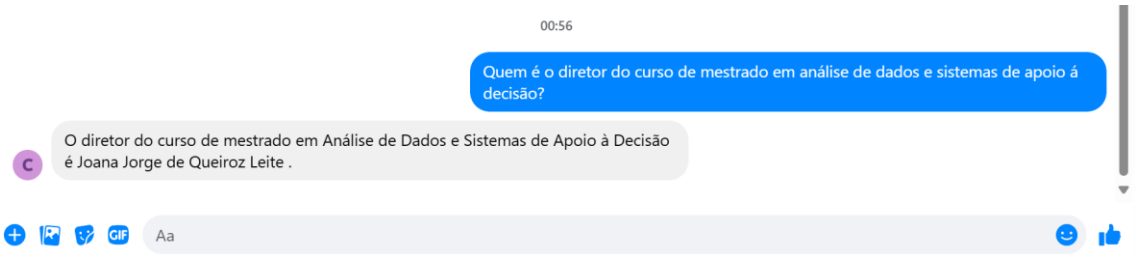

Figura 6-2: Resposta sobre o diretor do mestrado em ADSAD

Teste 2: Perguntar sobre horário de atendimento da biblioteca

A Figura 6-3 apresenta os dados que estão presentes no documento sobre o horário de atendimento da biblioteca.

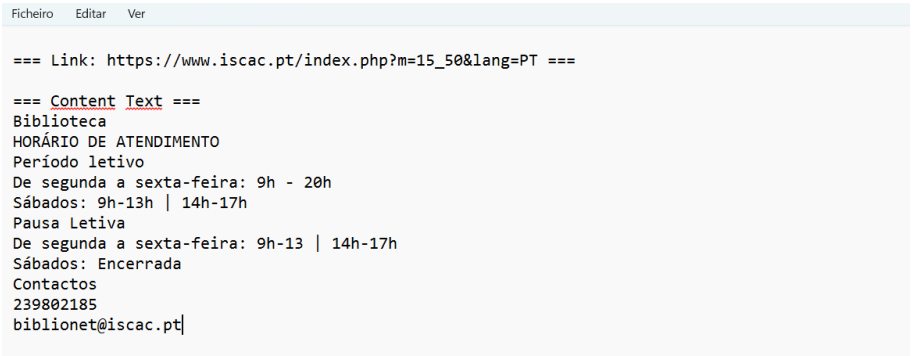

Figura 6-3: Dados sobre o horário de atendimento da biblioteca

A resposta que o *chatbot* gerou foi correta e bem estruturada (Figura 6-4).

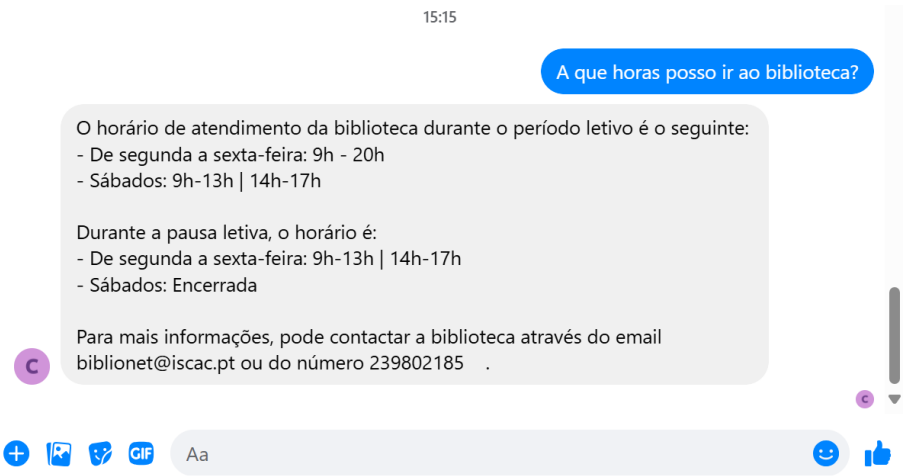

Figura 6-4: Resposta sobre horário de atendimento da biblioteca

6.2 Capacidade de resposta a perguntas sem informações

Teste 1: Perguntar sobre o valor da propina do mestrado

Como pode ser constatado na Figura 6-5, como esta informação não está presente no documento, o *chatbot* não consegue responder e direcionou o utilizador a procurar informações de outra forma.

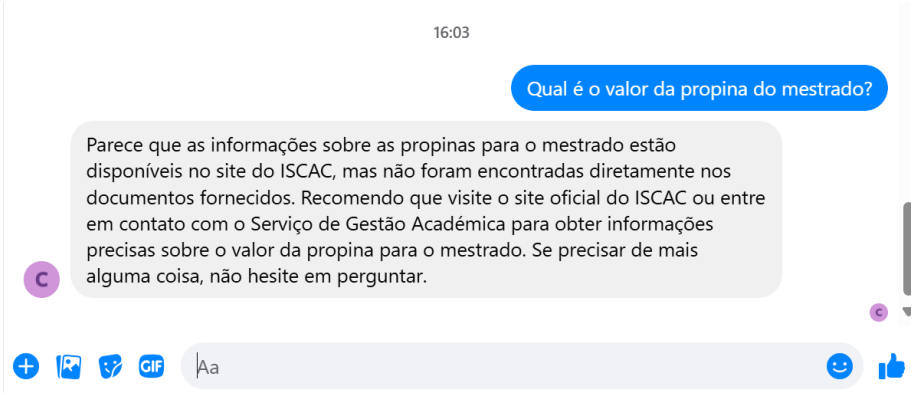

Figura 6-5: Resposta sobre o valor da propina do mestrado

Teste 2: Perguntar sobre a inscrição nos exames de recurso.

O *chatbot* tenta responder ao máximo possível com base nos dados (Figura 6-6).

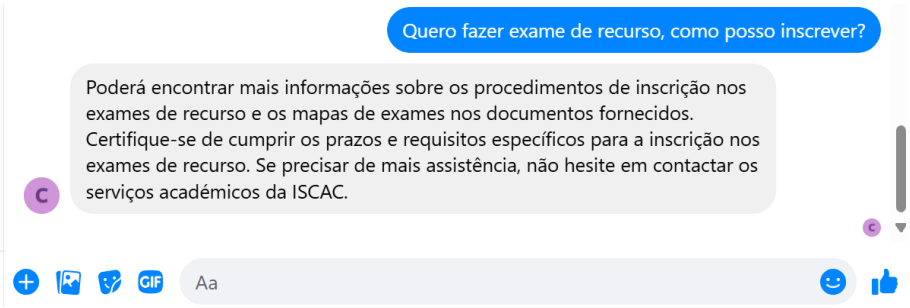

Figura 6-6: Resposta sobre a inscrição nos exames de recurso

6.3 Capacidade de resposta às perguntas sem relacionamento

Teste 1: Perguntar ao *chatbot* como ganhar dinheiro

O *chatbot* deteta que a pergunta não tem haver com o papel dele (atendimento de cliente da escola), e recusa responder (Figura 6-7).

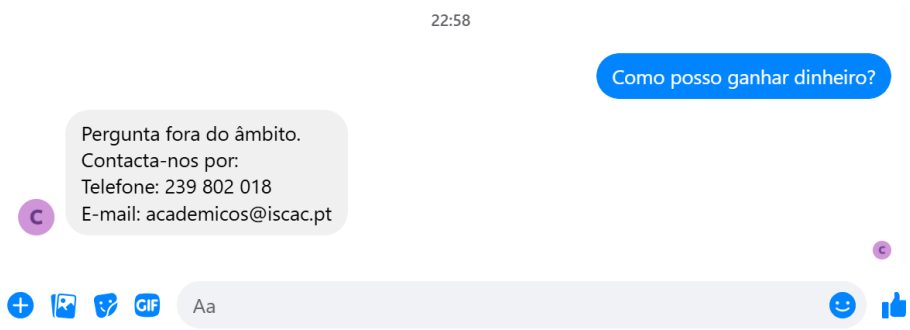

Figura 6-7: Resposta sobre como ganhar dinheiro

Teste 2: Perguntar se acha o utilizado bonito

Para perguntas fora do âmbito, o *chatbot* recusa responder mais para além da resposta apresentada na Figura 6-8.

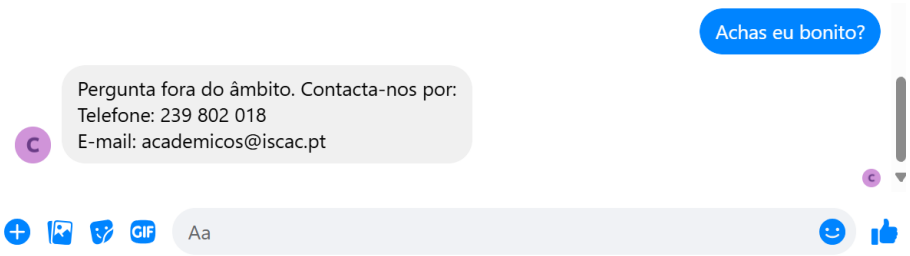

Figura 6-8: Resposta sobre se o utilizador é bonito

Em conclusão, o protótipo desenvolvido consegue resolver o problema detetado no diagnóstico, fornece um bom desempenho para o papel que lhe é atribuído, respondendo eficazmente aos utilizadores com base nos dados disponíveis no *site* da escola e recusando responder às questões que não se relacionam com o âmbito da escola.

A página do Facebook que foi criada para fazer o protótipo pode ser acedida através de https://www.facebook.com/profile.php?id=61555921707333. O código desenvolvido neste projeto pode ser acedido através da página do GitHub que foi criada para este efeito, através de https://github.com/Bingnan99/Chatbot-baseado-em-IA-do-ISCAC.

7 ESPECIFICAÇÃO DA APRENDIZAGEM

O projeto realizado com base no método de investigação-ação tinha como objetivo a implementação de um *chatbot* baseado em IA para o ISCAC que permitisse sistematizar e automatizar a consulta do seu *website*.

A solução implementada teve por base a tecnologia LLM, em concreto o ChatGPT, tecnologia essa que não tinha sido ainda experimentada pelo ISCAC para divulgação da informação existente no web site institucional aos diferentes utilizadores, desde os colaboradores aos estudantes e visitantes. A execução deste projeto de investigação permitiu ao ISCAC não só ganhar conhecimento na tecnologia, mas também, conhecer as potencialidades da mesma para aplicação noutros contextos. A avaliação da solução tecnológica, ainda que não muito extensa e rigorosa, viabilizou a utilidade da sua aplicação no contexto do ISCAC.

Dado que esta solução tecnológica integrou diversas tecnologias para além do ChatGPT, como a utilização da API do Messenger do Facebook, foi possível integrar o Chatbot desenvolvido no Messenger, permitindo a comunicação via mensagens do Facebook. Este recurso, que o ISCAC ainda não possuía, representa um contributo importante, pois possibilita que, no futuro, o ISCAC utilize esta solução tecnológica para outros contextos de resposta automática além do ChatGPT.

Por último, é importante destacar que o ISCAC ganha com este projeto de investigação novas capacidades de integração tecnológica, que poderão ser aplicadas a diversos serviços da organização. Esta nova competência permitirá ao ISCAC otimizar processos, melhorar a eficiência operacional e oferecer soluções inovadoras em diferentes áreas, desde a administração académica até ao apoio ao estudante. Além disso, a capacidade de integrar múltiplas tecnologias abre portas para futuras inovações e adaptações, permitindo ao ISCAC responder de forma mais ágil e eficaz às necessidades emergentes da instituição e dos seus utilizadores.

CONCLUSÃO

Na conclusão deste projeto, realça-se a construção bem-sucedida de um protótipo de *chatbot* baseado em inteligência artificial para uma instituição educacional, o Instituto Superior de Contabilidade e Administração de Coimbra, integrando a API do *OpenAI* a uma página do *Facebook*, aborda eficazmente a necessidade de melhorar o atendimento ao cliente. Este *chatbot* foi projetado especificamente para responder a perguntas sobre a escola e fornecer suporte ao cliente, desempenhando um papel crucial como ferramenta de apoio ao cliente.

Neste projeto, conseguiu-se explorar mais uma possibilidade de aplicação prática dos *chatbots* de IA em contexto do mundo real. Este projeto monstra como as tecnologias de IA podem ser utilizadas para apoiar a disseminação do conhecimento e facilitar a aprendizagem contínua, contribuindo para a inovação no campo dos serviços prestados pela instituição.

A integração do referido *chatbot* com o *Facebook* explora as possibilidades e desafios da interoperabilidade entre diferentes plataformas tecnológicas. Este trabalho contribuiu para a literatura sobre integração de sistemas, destacando aspetos técnicos e operacionais envolvidos na ligação de serviços de IA com redes sociais amplamente utilizadas.

A documentação detalhada do projeto, desenvolvido segundo uma metodologia baseada no método de investigação-ação e que inclui o diagnóstico, o planeamento das ações, as ações executadas, a avaliação e a especificação da aprendizagem, enriquece a literatura técnica e fornece um estudo de caso prático que pode ser referenciado em investigações futuras sobre este tema. Este estudo de caso adiciona valor ao corpo de conhecimento existente e serve como uma referência útil para investigadores e profissionais na área de desenvolvimento de *chatbots* de IA na integração com redes sociais.

O desenvolvimento deste protótipo revela uma solução viável e promissora, com potencial para aumentar a eficiência na resposta aos clientes e reduzir a carga de trabalho dos funcionários do gabinete dos serviços académicos. A tecnologia de IA empregada permite respostas automáticas e imediatas às consultas dos clientes, melhorando a

eficiência e a qualidade do atendimento ao cliente, e libertando os funcionários para se concentrarem em tarefas mais complexas e de maior valor.

Embora o protótipo ainda não tenha sido implementado em um ambiente real e existam áreas que podem ser aperfeiçoadas, a apresentação dos passos explícitos seguidos para a sua implementação foi um contributo significativo. Este protótipo serve como uma base sólida, que pode ser rapidamente adaptada e implementada quando necessário para outras situações, evidencia o compromisso com a inovação e a melhoria contínua dos serviços educacionais.

O desenvolvimento deste protótipo não responde apenas a um problema crítico, mas também propõe uma solução prática e implementável. Com melhorias contínuas, esta solução tem o potencial de transformar significativamente o atendimento ao cliente e a eficiência operacional dentro da referida instituição de ensino.

A maior limitação diz respeito aos dados usados neste protótipo. Os dados utilizados no modelo de *chatbot* foram somente os disponíveis no site da escola, mas a adição de novos dados é um processo simples e pode ser facilmente implementada. Este aperfeiçoamento permitirá aumentar a capacidade do *chatbot* em responder a mais perguntas. A manutenção dos dados é essencial, especialmente em caso de atualizações, e pode ser realizada facilmente através de novo *web scraping*, atualizando-se posteriormente o *site* da *OpenAI* com esses novos dados. A liberdade de manipulação de dados é ampla, permitindo adicionar, corrigir, ou armazenar dados conforme necessário, com custos relativamente baixos. O que ainda também pode ser melhorado é a apresentação, estruturação e agrupamento dos dados, de modo a otimizar a eficiência do modelo na interpretação dos dados e, consequentemente, melhorar a qualidade das respostas fornecidas pelo *chatbot*.

Outra limitação diz respeito à mudança que existiu no *site* oficial da escola no período da realização do presente projeto, o que significa que o código da recolha dos dados deve ser reescrito por motivos da alteração da estrutura do HTML do site.

Apesar do sucesso alcançado no desenvolvimento do *chatbot* para a *OpenAI* responder a perguntas via *Facebook*, há várias áreas onde futuras investigações e melhorias podem ser realizadas, especialmente no que diz respeito à base de dados utilizada pelo modelo.

Assim, a principal sugestão para trabalhos futuros aponta para a expansão e melhoria da base de dados. O desempenho do modelo é diretamente influenciado pela qualidade e abrangência da base de dados utilizada. Uma base de dados mais robusta e abrangente pode permitir que o modelo responda a uma gama mais ampla de perguntas com maior precisão. Para alcançar isso, é essencial continuar a expandir a base de dados, incorporando mais informações relevantes e atualizadas.

Outra direção importante é a construção de uma base de dados proprietária. Desenvolver uma base de dados proprietária pode oferecer um controle mais preciso sobre a qualidade e a relevância das informações. Este processo envolveria a recolha e armazenamento de perguntas, implementando um sistema para recolher e armazenar todas as perguntas recebidas pelo *chatbot* em funcionamento. Essa coleção de dados reais dos utilizadores permitirá uma análise aprofundada das perguntas mais frequentes e das áreas onde o modelo precisa melhorar.

Sugere-se a realização futura de entrevistas com os funcionários da escola para identificar as perguntas mais comuns e recorrentes. Pode-se solicitar acesso aos registos de e-mails recebidos para analisar os tipos de consultas que são frequentemente feitas. Essas informações são valiosas para entender os padrões de consulta e preparar o *chatbot* para lidar com elas de forma eficaz.

Além disso, a análise de dados deve ser realizada periodicamente. Realizar análises periódicas aos dados recolhidos permitirá identificar padrões e tendências nas perguntas dos utilizadores. Essas análises podem informar ajustes na base de dados e no treino do modelo, garantindo uma melhoria contínua na precisão e relevância das respostas fornecidas pelo *chatbot*.

REFERÊNCIAS

Adadi, A., & Berrada, M. (2018). Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, *6*, 52138–52160. https://doi.org/10.1109/ACCESS.2018.2870052

Adamopoulou, E., & Moussiades, L. (2020). Chatbots: History, technology, and applications. *Machine Learning with Applications*, *2*, 100006. https://doi.org/10.1016/j.mlwa.2020.100006

Agus Santoso, H., Anisa Sri Winarsih, N., Mulyanto, E., Wilujeng saraswati, G., Enggar Sukmana, S., Rustad, S., Syaifur Rohman, M., Nugraha, A., & Firdausillah, F. (2018). Dinus Intelligent Assistance (DINA) Chatbot for University Admission Services. *2018 International Seminar on Application for Technology of Information and Communication*, 417–423. https://doi.org/10.1109/ISEMANTIC.2018.8549797

Alsharhan, A., Al-Emran, M., & Shaalan, K. (2023). Chatbot Adoption: A Multiperspective Systematic Review and Future Research Agenda. *IEEE Transactions on Engineering Management*. https://doi.org/10.1109/TEM.2023.3298360

Baskerville, & Myers. (2004). Special Issue on Action Research in Information Systems: Making IS Research Relevant to Practice: Foreword. *MIS Quarterly*, *28*(3), 329. https://doi.org/10.2307/25148642

Baskerville, R. L. (1999). Investigating Information Systems with Action Research. *Communications of the Association for Information Systems*, *2*. https://doi.org/10.17705/1CAIS.00219

Caldarini, G., Jaf, S., & McGarry, K. (2022). A Literature Survey of Recent Advances in Chatbots. *Information (Switzerland)*, *13*(1). https://doi.org/10.3390/info13010041

Chong, T., Yu, T., Keeling, D. I., & de Ruyter, K. (2021). AI-chatbots on the services frontline addressing the challenges and opportunities of agency. *Journal of Retailing and Consumer Services*, *63*, 102735. https://doi.org/10.1016/j.jretconser.2021.102735

Dias, R., Sousa, F., & Trigo, A. (2023). Certified Accountants and ChatGPT. *XIX CONGRESSO INTERNACIONAL DE CONTABILIDADE E AUDITORIA*. https://www.researchgate.net/publication/375162742

Enholm, I. M., Papagiannidis, E., Mikalef, P., & Krogstie, J. (2022). Artificial Intelligence and Business Value: a Literature Review. *Information Systems Frontiers*, *24*(5), 1709–1734. https://doi.org/10.1007/s10796-021-10186-w

Hirschberg, J., & Manning, C. D. (2015). Advances in natural language processing. *Science*, *349*(6245), 261–266. https://doi.org/10.1126/science.aaa8685

IPC. (2021). *Estatutos do Instituto Superior de Contabilidade e Administração de Coimbra*. https://www.ipc.pt/wp-content/uploads/2021/08/Estatutos-ISCAC_DR.pdf

ISCAC, Pub. L. No. 112, Diário da República (2021).

ISCAC. (2024). *Síntese Histórica*. https://www.iscac.pt/index.php?m=3_10&lang=PT

Kaplan, A., & Haenlein, M. (2019). Siri, Siri, in my hand: Who's the fairest in the land? On the interpretations, illustrations, and implications of artificial intelligence. *Business Horizons*, *62*(1), 15–25. https://doi.org/10.1016/j.bushor.2018.08.004

Kaul, V., Enslin, S., & Gross, S. A. (2020). History of artificial intelligence in medicine. *Gastrointestinal Endoscopy*, *92*(4), 807–812. https://doi.org/10.1016/j.gie.2020.06.040

Khurana, D., Koli, A., Khatter, K., & Singh, S. (2023). Natural language processing: state of the art, current trends and challenges. *Multimedia Tools and Applications*, *82*(3), 3713–3744. https://doi.org/10.1007/s11042-022-13428-4

Klein, K., & Martinez, L. F. (2023). The impact of anthropomorphism on customer satisfaction in chatbot commerce: an experimental study in the food sector. *Electronic Commerce Research*, *23*(4), 2789–2825. https://doi.org/10.1007/s10660-022-09562-8

Kumar, R., & Ali, M. M. (2020). A Review on Chatbot Design and Implementation Techniques. *International Research Journal of Engineering and Technology*. www.irjet.net

Large Model Systems Organization. (2023). *Chatbot Arena Leaderboard Updates*. https://lmsys.org/blog/2023-05-25-leaderboard/

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Legg, S., & Hutter, M. (2007). Universal Intelligence: A Definition of Machine Intelligence. *Minds and Machines*, *17*(4), 391–444. https://doi.org/10.1007/s11023-007-9079-x

Lu, Y. (2019). Artificial intelligence: a survey on evolution, models, applications and future trends. *Journal of Management Analytics*, *6*(1), 1–29. https://doi.org/10.1080/23270012.2019.1570365

Luo, B., Lau, R. Y. K., Li, C., & Si, Y. (2022). A critical review of state-of-the-art chatbot designs and applications. *WIREs Data Mining and Knowledge Discovery*, *12*(1). https://doi.org/10.1002/widm.1434

Luo, X., Tong, S., Fang, Z., & Qu, Z. (2019). Frontiers: Machines vs. humans: The impact of artificial intelligence chatbot disclosure on customer purchases. *Marketing Science*, *38*(6), 937–947. https://doi.org/10.1287/mksc.2019.1192

Maroengsit, W., Piyakulpinyo, T., Phonyiam, K., Pongnumkul, S., Chaovalit, P., & Theeramunkong, T. (2019). A survey on evaluation methods for chatbots. *ACM International Conference Proceeding Series*, *Part F148391*, 111–119. https://doi.org/10.1145/3323771.3323824

Mikalef, P., & Gupta, M. (2021). Artificial intelligence capability: Conceptualization, measurement calibration, and empirical study on its impact on organizational creativity and firm performance. *Information & Management*, *58*(3), 103434. https://doi.org/10.1016/j.im.2021.103434

Nagarhalli, T. P., Vaze, V., & Rana, N. K. (2020). A Review of Current Trends in the Development of Chatbot Systems. *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 706–710. https://doi.org/10.1109/ICACCS48705.2020.9074420

Okonkwo, C. W., & Ade-Ibijola, A. (2021). Chatbots applications in education: A systematic review. *Computers and Education: Artificial Intelligence*, *2*, 100033. https://doi.org/10.1016/j.caeai.2021.100033

OpenAI. (2024). *Assistants API* . https://platform.*OpenAI*.com/docs/assistants/overview

Pan, S., Luo, L., Wang, Y., Chen, C., Wang, J., & Wu, X. (2024). Unifying Large Language Models and Knowledge Graphs: A Roadmap. *IEEE Transactions on Knowledge and Data Engineering*, *36*(7), 3580–3599. https://doi.org/10.1109/TKDE.2024.3352100

Park, D. M., Jeong, S. S., & Seo, Y. S. (2022). Systematic Review on Chatbot Techniques and Applications. *Journal of Information Processing Systems*, *18*(1), 26–47. https://doi.org/10.3745/JIPS.04.0232

Roumeliotis, K. I., & Tselikas, N. D. (2023). ChatGPT and Open-AI Models: A Preliminary Review. *Future Internet*, *15*(6), 192. https://doi.org/10.3390/fi15060192

Thorat, S. A., & Jadhav, V. (2020). A Review on Implementation Issues of Rule-based Chatbot Systems. *SSRN Electronic Journal*, *2020*. https://doi.org/10.2139/ssrn.3567047

White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., & Schmidt, D. C. (2023). *A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT*. http://arxiv.org/abs/2302.11382

WU, S. R., Shirkey, G., Celik, I., Shao, C., & Chen, J. (2022). A Review on the Adoption of AI, BC, and IoT in Sustainability Research. *Sustainability*, *14*(13), 7851. https://doi.org/10.3390/su14137851

Xin Yao. (1999). Evolving artificial neuronal networks. *Proceedings of the IEEE*, *87*(9), 1423–1447. https://doi.org/10.1109/5.784219

Xu, W., & Ouyang, F. (2022). A systematic review of AI role in the educational system based on a proposed conceptual framework. *Education and Information Technologies*, *27*(3), 4195–4223. https://doi.org/10.1007/s10639-021-10774-y

Yang, J., Jin, H., Tang, R., Han, X., Feng, Q., Jiang, H., Zhong, S., Yin, B., & Hu, X. (2024). Harnessing the Power of LLM in Practice: A Survey on ChatGPT and Beyond. *ACM Transactions on Knowledge Discovery from Data*, *18*(6), 1–32. https://doi.org/10.1145/3649506

Yang, S., & Evans, C. (2019). Opportunities and challenges in using AI chatbots in higher education. *ACM International Conference Proceeding Series*, 79–83. https://doi.org/10.1145/3371647.3371659

Zumstein, D., & Hundertmark, S. (2017). Chatbots-An Interactive Technology for Personalized Communication, Transactions and Services. *IADIS International Journal on WWW/Internet*, *15*(1), 96–109. https://www.researchgate.net/publication/322855718

ANEXOS

ANEXO 1. Código completo para recolha dos dados do site ISCAC.pt

```
import requests
from selenium import webdriver
from bs4 import BeautifulSoup

def iscacScraping(url, output_file):
    driver = webdriver.Chrome()  #Garante que tem google Chrome baixado no
computador
    driver.get(url)

    # Procurar todos conteúdos da página
    html_content = driver.page_source

    # Usar biblioteca BeautifulSoup para analisar conteúdos do HTML
    soup = BeautifulSoup(html_content, 'html.parser')

    # extrair os links do <ul class="sf-menu sf-js-enabled sf-arrows" id="example">
    ul_element = soup.find('ul', class_='sf-menu', id='example')
    if ul_element:
        links = ul_element.find_all('a', href=True)
        for link in links:
            link_href = link['href']
            print("Link:", link_href)

            # Usar Selenium para simular o comportamento do navegador e obter o
conteúdo do link
            driver.get(link_href)
            link_html_content = driver.page_source

            # Usar BeautifulSoup para analisar os conteúdos do link
            link_soup = BeautifulSoup(link_html_content, 'html.parser')

            # Extrair text do <div class="coluna_conteudo">
            coluna_conteudo = link_soup.find('table', class_='content_area')
            if coluna_conteudo:
                content_text = coluna_conteudo.get_text(separator='\n', strip=True)
                print("Content Text:", content_text)

                # excrever o text num ficheiro txt
                with open(output_file, 'a', encoding='utf-8') as file:
                    file.write(f"\n\n=== Link: {link_href} ===\n")
```

```
                    file.write(f"\n=== Content Text ===\n{content_text}\n")
                    print(f"Successfully wrote content for link: {link_href}")

    # fechar o navegador
    driver.quit()

if __name__ == "__main__":
    start_url = "https://www.iscac.pt/index.php?lang=PT"
    output_file = "D:/OneDrive - ISCAC/MADSAD/Tese Mestrado/output_td_content2.txt"

    iscacScraping(start_url, output_file)
    print("Finished crawling and saving content. Content saved to", output_file)
```

ANEXO 2. Código completo de ligação ao API do *OpenAI*

```
from OpenAI import OpenAI

chave = ""

assis_id = ""


client = OpenAI(api_key=chave)


#input do questao

ask = "a que horas posso ir à secretaria?"

thread = client.beta.threads.create(messages=[{"role":"user",

                                               "content":ask

                                               }])

run = client.beta.threads.runs.create(thread_id=thread.id, assistant_id=assis_id)

print("run id:", run.id)


while run.status != "completed":

    run = client.beta.threads.runs.retrieve(thread_id=thread.id, run_id=run.id)

else:

    print("run complete")


message_response = client.beta.threads.messages.list(thread_id=thread.id)

messages = message_response.data


response = messages[0]

print("Pergunta:\n", ask, "\n")

print("Resposta:\n", response.content[0].text.value)
```

ANEXO 3. Código completo de ligação ao API do *Facebook*-Messenger

```
from Flask import Flask, request
import json
import requests


app = Flask(__name__)


PAGE_ACCESS_TOKEN =
"EAAWNYKpXnL4BOzS2uZB0ZBXZCU7o0XsjuuqfO7c9oAFJqirWsTOuW8hH64ZBLP7CZAzVfRnrG7SCytAZA
FcWLHsngSh2WlBzBcZBVqHSgWCnQ1XKne3uE1SLOvcTivnS4MZCxmfNrjXDZCkR3Oqk95KA3590IVRSg6Dz
qonROW7ZBz083btFZAu4RaAMp6siBO305sZD"


LATEST_API_VERSION = "v19.0"


API =
"https://graph.Facebook.com/"+LATEST_API_VERSION+"/me/messages?access_token="+PAGE_
ACCESS_TOKEN


@app.route('/hello/', methods = ['GET', 'POST'])
def hello_world():
    return "Hello World at localhost:8080"


def send_message(sender_id, message_text):
    '''
    Sending response back to the user using Facebook graph API
    '''
    #print("sending message to {recipient}: {text}".format(recipient=sender_id,
text=message_text))
    r =
requests.post("https://graph.Facebook.com/"+LATEST_API_VERSION+"/me/messages",
        params={"access_token": PAGE_ACCESS_TOKEN},
```

```
        headers={"Content-Type": "application/json"},
        data=json.dumps({
        "recipient": {"id": sender_id},
        "message": {"text": message_text}
    }))

    print(r.text)

@app.route('/', methods=['GET'])
def handle_verification():
    if (request.args.get('hub.verify_token', '') == "chatescola2024"):
        print("Verified")
        return request.args.get('hub.challenge', '')
    else:
        print("Wrong token")
        return "Error, wrong validation token"

@app.route('/', methods=['POST'])
def handle_message():
    '''
    Handle messages sent by Facebook messenger to the applicaiton
    '''
    data = request.get_json()
    print(data)
    if data["object"] == "page":
        for entry in data["entry"]:
            for messaging_event in entry["messaging"]:
                if messaging_event.get("message"):
                    sender_id = messaging_event["sender"]["id"]
                    recipient_id = messaging_event["recipient"]["id"]
                    try:
```

```
                        message_text = messaging_event["message"]["text"]
                    except:
                        message_text = "erro"
                    send_message(sender_id, message_text)
    return "ok"


if __name__ == '__main__':
    app.run( host = '0.0.0.0', port = 8080 )
```

ANEXO 4. Código completo da ligação do *Facebook* com o *OpenAI*

```
from Flask import Flask, request
import json
import requests
from OpenAI import OpenAI


app = Flask(__name__)


PAGE_ACCESS_TOKEN =
"EAAWNYKpXnL4BOzS2uZB0ZBXZCU7o0XsjuuqfO7c9oAFJqirWsTOuW8hH64ZBLP7CZAzVfRnrG7SCytAZA
FcWLHsngSh2WlBzBcZBVqHSgWCnQ1XKne3uE1SLOvcTivnS4MZCxmfNrjXDZCkR3Oqk95KA3590IVRSg6Dz
qonROW7ZBz083btFZAu4RaAMp6siBO305sZD"
LATEST_API_VERSION = "v19.0"
API =
"https://graph.Facebook.com/"+LATEST_API_VERSION+"/me/messages?access_token="+PAGE_
ACCESS_TOKEN


chave = "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"
assis_id = "asst_UEFpXItfoJY2nSOmjw6D75AX"
OpenAI_client = OpenAI(api_key=chave)


@app.route('/hello/', methods=['GET', 'POST'])
def hello_world():
    return "Hello World at localhost:8080"


def send_message(sender_id, message_text):
    '''
    Sending response back to the user using Facebook graph API
    '''
    r =
requests.post("https://graph.Facebook.com/"+LATEST_API_VERSION+"/me/messages",
```

```
                    params={"access_token": PAGE_ACCESS_TOKEN},
                    headers={"Content-Type": "application/json"},
                    data=json.dumps({
                        "recipient": {"id": sender_id},
                        "message": {"text": message_text}
                    }))
    print(r.text)


@app.route('/', methods=['GET'])
def handle_verification():
    if (request.args.get('hub.verify_token', '') == "chatescola2024"):
        print("Verified")
        return request.args.get('hub.challenge', '')
    else:
        print("Wrong token")
        return "Error, wrong validation token"


@app.route('/', methods=['POST'])
def handle_message():
    '''
    Handle messages sent by Facebook Messenger to the application
    '''
    data = request.get_json()
    print(data)
    if data["object"] == "page":
        for entry in data["entry"]:
            for messaging_event in entry["messaging"]:
                if messaging_event.get("message"):
                    sender_id = messaging_event["sender"]["id"]
                    recipient_id = messaging_event["recipient"]["id"]
                    try:
```

```
                        message_text = messaging_event["message"]["text"]
                        # Send the received message to OpenAI API
                        thread = 
OpenAI_client.beta.threads.create(messages=[{"role": "user",
                                                                    "cont
ent": message_text}])
                        run = 
OpenAI_client.beta.threads.runs.create(thread_id=thread.id, assistant_id=assis_id)
                        while run.status != "completed":
                            run = 
OpenAI_client.beta.threads.runs.retrieve(thread_id=thread.id, run_id=run.id)
                        else:
                            message_response = 
OpenAI_client.beta.threads.messages.list(thread_id=thread.id)
                            messages = message_response.data
                            response = messages[0]
                            response_text = response.content[0].text.value
                            send_message(sender_id, response_text)
                    except Exception as e:
                        print("Error:", e)
    return "ok"


if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080)
```