

Wander Victor Verçosa Mares

**Desenvolvimento de Assistente Virtual em
Linguagem Python Para Filtro de Cliente no
Mercado Imobiliário**

Uberlândia, MG

2024

Wander Victor Verçosa Mares

Desenvolvimento de Assistente Virtual em Linguagem Python Para Filtro de Cliente no Mercado Imobiliário

Trabalho de Conclusão de Curso da Engenharia de Controle e Automação da Universidade Federal de Uberlândia - UFU - Câmpus Santa Mônica, como requisito para a obtenção do título de Graduação em Engenharia de Controle e Automação.

Universidade Federal de Uberlândia - UFU
Faculdade de Engenharia Elétrica - FEELT

Orientador Prof. Dr. Josué Silva de Moraes

Uberlândia, MG

2024

Wander Victor Verçosa Mares

Desenvolvimento de Assistente Virtual em Linguagem Python Para Filtro de Cliente no Mercado Imobiliário/ Wander Victor Verçosa Mares. – Uberlândia, MG, 2024-

31 p. : il. (algumas color.) ; 30 cm.

Orientador Prof. Dr. Josué Silva de Moraes

Trabalho de Conclusão de Curso – Universidade Federal de Uberlândia - UFU
Faculdade de Engenharia Elétrica - FEELT, 2024.

1. Assistente Virtual. 2. Python. I. Prof. Dr. Josué Silva de Moraes. II. Universidade Federal de Uberlândia. III. Faculdade de Engenharia Elétrica. IV. Desenvolvimento de Assistente Virtual em Linguagem Python Para Filtro de Cliente no Mercado Imobiliário

CDU 02:141:005.7

Wander Victor Verçosa Mares

Desenvolvimento de Assistente Virtual em Linguagem Python Para Filtro de Cliente no Mercado Imobiliário

Trabalho de Conclusão de Curso da Engenharia de Controle e Automação da Universidade Federal de Uberlândia - UFU - Câmpus Santa Mônica, como requisito para a obtenção do título de Graduação em Engenharia de Controle e Automação.

Trabalho aprovado em 18 de Novembro de 2024.

COMISSÃO EXAMINADORA

Prof. Dr. Josué Silva de Moraes
Orientador

Prof. Dr. Aniel Silva de Moraes
Membro Avaliador

Prof. Me. Éder Alves de Moura
Membro Avaliador

Uberlândia, MG
2024

Agradecimentos

Agradeço a Deus que me instruiu em todos os caminhos que levaram o meu sucesso até aqui, como também agradeço a minha família por sempre ser o meu ponto de referência no quesito trabalho duro e confiança. Quero agradecer também a todos os meus amigos e companheiros de curso que acompanharam a minha trajetória durante o curso, foi longe de ser fácil, mas valeu a pena cada minuto, cada aula, cada noite mal dormida e cada história compartilhada.

Para quem está lendo esse trecho, acredito que está trilhando um caminho semelhante, dessa forma, deixo a seguinte mensagem: acredite cem por cento em você! Não há força maior no universo do que a fé que você coloca em si mesmo, essa simples ação te transforma em um instrumento na mão de Deus e te faz alcançar lugares que você nunca sonhou.

“Na adversidade, uns desistem, enquanto outros batem recordes.”
(Ayrton Senna)

Resumo

A jornada de compra de um cliente no mercado imobiliário abrange desde a intenção de compra até a aquisição do imóvel, com uma média de duração de cerca de 6 meses. Um dos principais obstáculos nesse processo é a falta de informações sobre a quantidade e tipologia dos imóveis disponíveis na região de interesse do comprador. Este artigo propõe o desenvolvimento de uma ferramenta utilizando a linguagem de programação Python, integrando-a ao aplicativo Telegram e a um banco de dados de imóveis catalogados. Por meio dessa solução, os potenciais clientes podem responder a perguntas simples, permitindo a criação de um perfil de necessidades imobiliárias, resultando em opções de imóveis filtradas de acordo com seus critérios. No dinâmico cenário do mercado imobiliário, a demanda por soluções inovadoras é constante, e a implementação de um assistente virtual em linguagem Python surge como uma resposta eficaz às necessidades do setor. Esse assistente não apenas simplifica, mas revoluciona a maneira como os clientes são filtrados, oferecendo uma abordagem inteligente e automatizada para identificar potenciais compradores. Ao utilizar tecnologia de ponta, como a integração entre Python, Telegram e banco de dados, esse assistente virtual se torna uma ferramenta indispensável para corretores e empresas imobiliárias em Uberlândia e além. A automação proporcionada pela linguagem Python não apenas agiliza o processo de filtragem de clientes, mas também eleva a eficiência operacional, permitindo que corretores e empresas lidem com a crescente demanda e oferta de imóveis na região. A capacidade de adaptar-se às necessidades específicas de um mercado em constante evolução é uma das vantagens-chave desse assistente virtual, tornando-o uma solução flexível e escalável para os desafios do setor imobiliário. Ao reduzir o tempo gasto na busca por informações e na identificação de opções disponíveis, essa ferramenta visa facilitar a jornada de compra para os clientes em potencial, tornando o processo mais eficiente e satisfatório.

Palavras-chaves: automação, python, banco de dados, telegram, ferramenta, cliente.

Abstract

A customer's buying journey in the real estate market ranges from the intention to purchase to the acquisition of the property, with an average duration of around 6 months. One of the main obstacles in this process is the lack of information about the quantity and type of properties available in the region of interest to the buyer. This article proposes the development of a tool using the Python programming language, integrating it with the Telegram application and a database of cataloged properties. Through this solution, potential customers can answer simple questions, allowing the creation of a profile of real estate needs, resulting in filtered property options according to their criteria. In the dynamic scenario of the real estate market, the demand for innovative solutions is constant, and the implementation of a virtual assistant in Python language emerges as an effective response to the needs of the sector. This assistant not only simplifies, but revolutionizes the way in which customers are filtered, offering an intelligent and automated approach to identify potential buyers. By using cutting-edge technology, such as the integration between Python, Telegram and database, this virtual assistant becomes an indispensable tool for brokers and real estate companies in Uberlândia and beyond. The automation provided by Python not only streamlines the client screening process, but also increases operational efficiency, allowing brokers and companies to cope with the growing demand and supply of properties in the region. The ability to adapt to the specific needs of an ever-evolving market is one of the key advantages of this virtual assistant, making it a flexible and scalable solution to the challenges of the real estate industry. By reducing the time spent searching for information and identifying available options, this tool aims to facilitate the purchasing journey for potential clients, making the process more efficient and satisfying.

Key-words: automation, python, database, telegram, tool, client.

Lista de ilustrações

Figura 1 – Porcentagem em tempo médio para busca de imóvel por uma pessoa comum no Brasil.	11
Figura 2 – Bairro Granja Marileusa, símbolo de desenvolvimento tecnológico e imobiliário em Uberlândia-MG.	12
Figura 3 – Criação do bot via botfather.	20
Figura 4 – Terminal de saída com dados captados pelo GPT-4 mediante a resposta do usuário.	21
Figura 5 – ChatBot no telegram com requisição de informação pelo usuário. . . .	22
Figura 6 – Tela de login fornecida pela aplicação Flesk.	23
Figura 7 – Menu da interface web.	24
Figura 8 – Tela de registro de imóvel.	24
Figura 9 – Resposta do sistema para gerar aplicação Flask com interface web. . .	25

Lista de abreviaturas e siglas

API	Interface de Programação de Aplicativos (em inglês, Application Programming Interface), que define métodos de comunicação entre diferentes componentes de software
BD	Banco de Dados, referindo-se a um sistema para armazenamento e recuperação de informações
ML	Machine Learning
NLP	Natural Language Processing
IA	Inteligência Artificial

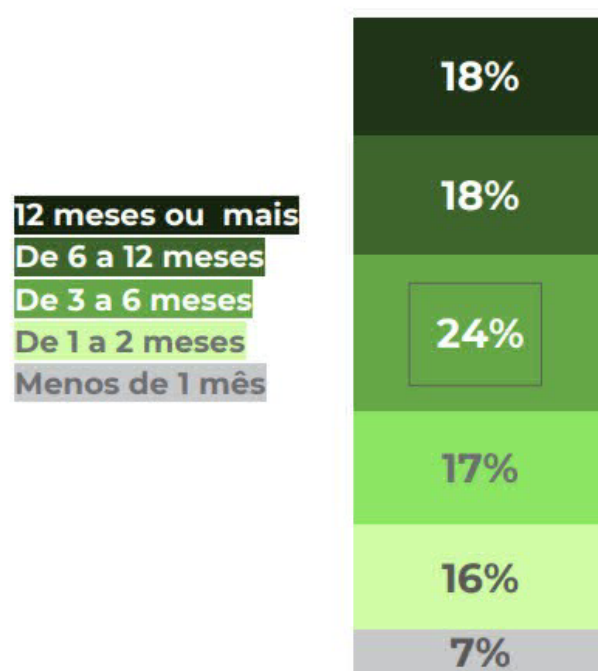
Sumário

1	INTRODUÇÃO	11
1.1	Justificativas	12
1.2	Objetivos	12
1.2.1	Telegram e python-telegram-bot	13
1.2.2	Conexão com bancos de dados	13
1.2.3	Funcionamento do Assistente Virtual	13
1.2.4	Benefícios	14
2	REFERENCIAIS TEÓRICOS	15
2.1	A OpenAI e a utilização de Machine Learning no ChatGPT	15
2.1.1	Fundamentos do Machine Learning no ChatGPT	15
2.1.2	Utilização da API do Telegram e GPT-4 em Python	16
2.1.3	A API do GPT-4 da OpenAI	16
2.2	Modelagem de banco de dados com SQLite	17
2.3	Utilização de ChatBots no Telegram	18
3	METODOLOGIA	19
3.0.1	OpenAI e Integração com Python para Machine Learning	19
3.0.2	Criação de Bot no Telegram com BotFather	19
3.0.3	Configuração Flask para InterfaceWeb e tratamento do Banco de Dados	20
3.0.4	Configuração Bot Telegram para aquisição de resposta	20
3.0.5	Registro de imóveis para alimentação do Banco de Dados	21
4	RESULTADOS E DISCUSSÕES	25
5	CONCLUSÃO E TRABALHOS FUTUROS	26
	REFERÊNCIAS BIBLIOGRÁFICAS	27
A	APÊNDICE A - CÓDIGO APP.PY	28
B	APÊNDICE B - CÓDIGO BOT.PY	30

1 Introdução

A jornada de compra de um cliente no mercado imobiliário é uma jornada complexa que envolve várias etapas, desde o momento em que surge a intenção de adquirir um imóvel até a conclusão da compra, podendo durar em média cerca de 6 meses, conforme explícito na Figura 2. Durante esse período, o cliente passa por diversas fases, incluindo a pesquisa de mercado, a definição de critérios de busca, a análise de opções disponíveis, as visitas aos imóveis, as negociações com os vendedores e, finalmente, a assinatura do contrato de compra e venda. No entanto, um dos principais desafios enfrentados pelos clientes ao longo dessa jornada é a escassez de informações precisas e atualizadas sobre os imóveis disponíveis na região de seu interesse (ZAP+, 2022). Muitas vezes, os clientes se deparam com dificuldades para encontrar informações detalhadas sobre a quantidade, localização, características e preço dos imóveis que atendam às suas necessidades e preferências específicas. Essa falta de transparência e acessibilidade às informações pode resultar em um processo de busca prolongado e frustrante para os clientes, que precisam investir tempo e esforço consideráveis na identificação e avaliação de opções de imóveis. Além disso, a falta de clareza sobre as características e disponibilidade dos imóveis pode dificultar a comparação entre diferentes opções e a tomada de decisão por parte dos clientes.

Figura 1 – Porcentagem em tempo médio para busca de imóvel por uma pessoa comum no Brasil.



Fonte: Datazap (2022).

1.1 Justificativas

Uberlândia tem se destacado como um centro de crescimento populacional e imobiliário expressivo no Brasil. De acordo com o censo do IBGE, entre 2010 e 2022, a cidade aumentou sua população em 18,08%, alcançando 713.224 habitantes, e registrou um impressionante crescimento de 53,74% no número de domicílios. Esse aumento populacional impulsionou o mercado imobiliário local, tornando a cidade um dos maiores polos de vendas de imóveis no país (MARILEUSA, 2024). O Sindicato das Indústrias da Construção Civil do Triângulo Mineiro e Alto Paranaíba destaca que Uberlândia lidera as vendas imobiliárias no interior do Brasil. Esse crescimento significativo atraiu investimentos em infraestrutura e desenvolvimento urbano, consolidando a cidade como um ambiente atrativo para morar, trabalhar e empreender. A combinação de alta demanda por imóveis e a oferta diversificada cria um mercado dinâmico e competitivo.

Figura 2 – Bairro Granja Marileusa, símbolo de desenvolvimento tecnológico e imobiliário em Uberlândia-MG.



Fonte: Autoria própria (2024).

1.2 Objetivos

Nesse contexto, a implementação de um assistente virtual em linguagem Python para filtrar clientes no mercado imobiliário surge como uma solução inovadora e necessária. Esse assistente visa otimizar a identificação de potenciais compradores, facilitando o trabalho de corretores e empresas imobiliárias ao lidar com a alta demanda e a crescente oferta de imóveis na região. A automatização e a eficiência trazidas pela tecnologia de assistentes virtuais podem melhorar significativamente a experiência dos clientes e a produtividade dos profissionais do setor, adaptando-se às necessidades específicas de um mercado em rápida evolução como o de Uberlândia. A integração entre Python, Telegram e banco de dados pode ser feita de forma eficiente utilizando bibliotecas específicas, e essa

combinação oferece uma base poderosa para a criação de um assistente virtual no mercado imobiliário.

1.2.1 Telegram e python-telegram-bot

O Telegram, uma plataforma de mensagens amplamente utilizada, fornece uma API robusta que permite a criação de bots interativos. A biblioteca `python-telegram-bot` facilita o desenvolvimento desses bots em Python, fornecendo uma interface simples e poderosa para interagir com a API do Telegram (DOCUMENTATION, 2022a). Com esta biblioteca, é possível programar bots que colem informações dos usuários, como preferências de imóveis, localização desejada e faixa de preço. Os bots podem ser configurados para responder automaticamente a perguntas frequentes, enviar notificações sobre novos imóveis e até agendar visitas.

1.2.2 Conexão com bancos de dados

Para armazenar e gerenciar os dados coletados, Python pode se conectar a diversos sistemas de gerenciamento de banco de dados. Bibliotecas como `SQLAlchemy` e `pymongo` (DOCUMENTATION, 2022b) são essenciais nesse processo: `SQLAlchemy`: É uma biblioteca SQL toolkit e ORM (Object-Relational Mapping) para Python, que facilita a interação com bancos de dados relacionais como MySQL e PostgreSQL. Ele permite a definição de modelos de dados em Python, que são automaticamente mapeados para tabelas no banco de dados. Isso simplifica operações como consultas, inserções, atualizações e exclusões de registros; `pymongo`: É uma biblioteca que permite a conexão e interação com MongoDB, um banco de dados NoSQL. MongoDB é conhecido por sua flexibilidade e capacidade de lidar com grandes volumes de dados não estruturados. Usando `pymongo`, é possível armazenar documentos JSON-like, que podem incluir informações detalhadas sobre clientes e imóveis.

1.2.3 Funcionamento do Assistente Virtual

o Coleta de Dados: O bot do Telegram interage com os usuários, coletando informações sobre suas preferências imobiliárias; o Armazenamento de Dados: Os dados coletados são armazenados em um banco de dados adequado (relacional ou NoSQL) usando `SQLAlchemy` ou `pymongo`; o Análise e Filtragem: O assistente virtual analisa os dados armazenados para identificar potenciais clientes, utilizando critérios predefinidos como faixa de preço, localização e tipo de imóvel; o Interação Automatizada: O bot do Telegram pode enviar automaticamente listas de imóveis que correspondem às preferências dos clientes, notificando-os sobre novas oportunidades.

1.2.4 Benefícios

A automação de tarefas repetitivas proporciona maior eficiência, pois libera tempo para que os corretores possam se concentrar em negociações e oferecer um atendimento mais personalizado. Além disso, a análise de dados precisa possibilita a identificação dos melhores clientes em potencial, contribuindo para o aumento da taxa de conversão de vendas. Por sua vez, respostas rápidas e precisas aprimoram a experiência dos usuários, tornando o processo de busca por imóveis mais agradável e satisfatório. Dessa forma, a integração entre Python, Telegram e bancos de dados não apenas eleva a eficiência operacional das empresas imobiliárias, como também proporciona uma experiência superior aos clientes, atendendo às demandas do mercado dinâmico de Uberlândia.

2 Referenciais Teóricos

2.1 A OpenAI e a utilização de Machine Learning no ChatGPT

A OpenAI é uma organização de pesquisa em inteligência artificial cujo foco é desenvolver sistemas de IA avançados que beneficiem a sociedade. Entre suas principais inovações está o ChatGPT, um modelo de linguagem baseado na arquitetura Transformer, que exemplifica a aplicação de técnicas avançadas de Machine Learning no campo do Processamento de Linguagem Natural (NLP) (ALURA, 2023).

2.1.1 Fundamentos do Machine Learning no ChatGPT

O ChatGPT é um exemplo de modelo de linguagem baseado em aprendizado profundo (Deep Learning), uma subárea de Machine Learning. Ele utiliza redes neurais profundas treinadas em grandes conjuntos de dados textuais para realizar a tarefa de modelagem de linguagem. O objetivo central é prever a próxima palavra em uma sequência, dado o contexto anterior, mas isso é feito de maneira extremamente sofisticada graças à arquitetura Transformer. A base do ChatGPT é o modelo Transformer, introduzido no artigo seminal “Attention is All You Need” (VASWANI et al., 2017). Essa arquitetura revolucionou o campo do Processamento de Linguagem Natural (NLP) ao implementar o mecanismo de atenção, que permite ao modelo atribuir diferentes pesos às partes de uma sequência de entrada. Isso possibilita a identificação das palavras ou frases mais relevantes para a tarefa em questão. No ChatGPT, esse mecanismo é fundamental para compreender e gerar texto de forma contextualizada, mesmo em diálogos complexos. O treinamento do ChatGPT ocorre em duas etapas principais: pré-treinamento e fine-tuning. Na fase de pré-treinamento, o modelo é exposto a grandes volumes de texto provenientes de diversas fontes, como livros, artigos científicos, sites da web e fóruns. Durante essa etapa, o modelo aprende padrões gerais da linguagem, incluindo gramática, relações semânticas e estruturas de frases, com o objetivo de adquirir um conhecimento amplo sobre o uso da linguagem. Após o pré-treinamento, o modelo passa pela etapa de fine-tuning, onde é ajustado para tarefas específicas utilizando conjuntos de dados mais focados, frequentemente rotulados manualmente. Essa etapa é essencial para treinar o modelo a responder perguntas, participar de diálogos e fornecer informações úteis, garantindo coerência e relevância no contexto. Além disso, uma técnica fundamental no treinamento do ChatGPT é o aprendizado por reforço com feedback humano (RLHF). Nessa etapa, humanos interagem com o modelo e avaliam suas respostas, destacando quais são mais apropriadas para diferentes cenários. As avaliações humanas são então utilizadas para treinar um modelo de recompensa, que orienta o ajuste do modelo principal, priorizando respostas mais úteis, educadas e informativas.

Esse processo aprimora ainda mais a qualidade das interações geradas pelo ChatGPT.

2.1.2 Utilização da API do Telegram e GPT-4 em Python

A utilização de APIs (Interface de Programação de Aplicações) permite a comunicação e interação entre diferentes sistemas e plataformas. No caso do desenvolvimento de chatbots e automação de tarefas em plataformas de mensageria, como o Telegram, as APIs desempenham um papel fundamental ao possibilitar a comunicação entre o servidor e o usuário, permitindo ações como o envio de mensagens, recepção de comandos e respostas automatizadas. A API do Telegram oferece recursos poderosos para a criação de bots que interagem de forma automatizada com os usuários da plataforma. Esses bots podem ser programados para realizar uma série de funções, como enviar mensagens, gerenciar grupos, responder a comandos e até executar tarefas mais complexas, como análise de texto e integração com outros sistemas. A API do Telegram facilita a criação de soluções personalizadas para diversas necessidades de comunicação. Ela permite que desenvolvedores criem bots que interagem diretamente com os usuários por meio de mensagens. A API é baseada em um modelo de requisições HTTP, o que a torna altamente compatível com diversas tecnologias e linguagens de programação, incluindo Python. Para integrar um bot com a plataforma, é necessário obter um token único, que identifica o bot, e a partir daí os desenvolvedores podem enviar e receber mensagens em tempo real. Para facilitar o processo de desenvolvimento, existem bibliotecas em Python, como a `python-telegram-bot`, que abstraem boa parte das complexidades da API, tornando a implementação mais simples e eficiente. A comunicação entre o bot e o usuário ocorre por meio de mensagens enviadas e recebidas em tempo real. O bot pode responder às mensagens dos usuários ou até iniciar uma interação, dependendo da programação. O Telegram também oferece funcionalidades como comandos ("/comando"), bots interativos e botões de escolha rápida, o que torna a interação ainda mais dinâmica. No caso do seu projeto, o bot receberá as mensagens enviadas pelo usuário, as enviará para a API do GPT-4 para processamento, e a resposta gerada pelo modelo será então devolvida ao usuário via Telegram.

2.1.3 A API do GPT-4 da OpenAI

A GPT-4 é uma das versões mais avançadas dos modelos de linguagem desenvolvidos pela OpenAI. Essa API foi projetada para compreender e gerar texto de forma altamente precisa, com base no treinamento em grandes volumes de dados textuais. A integração da API do GPT-4 com o Telegram é crucial para criar uma experiência interativa, permitindo que o conteúdo enviado pelo usuário seja processado de maneira eficiente, com uma resposta gerada com base no contexto fornecido. Além disso, a GPT-4 é uma rede neural do tipo Transformer, altamente otimizada para tarefas de Processamento de Linguagem Natural (NLP). Sua principal funcionalidade é gerar respostas textuais de forma contextualizada, levando em consideração o histórico da conversa e a intenção do usuário. A API do

GPT-4 é acessada por meio de chamadas HTTP, onde o texto de entrada, conhecido como prompt, é enviado, e a API devolve uma resposta gerada pelo modelo. Uma das grandes vantagens do GPT-4 é sua capacidade de fornecer respostas mais naturais, relevantes e precisas, em comparação com modelos anteriores. Isso é especialmente importante em aplicações como a que está sendo desenvolvida, onde as respostas precisam ser coerentes e contextuais, baseadas nas interações anteriores. A integração entre a API do Telegram e o GPT-4 pode ser feita de forma simples e eficaz utilizando Python. O processo típico envolve capturar a mensagem enviada pelo usuário no Telegram, enviá-la para a API do GPT-4 para processamento, gerar a resposta e, por fim, enviar a resposta de volta ao usuário através do Telegram. Essa integração pode ser realizada por meio de requisições HTTP utilizando bibliotecas como requests em Python, ou por meio de frameworks como o python-telegram-bot, que facilita a interação com a API do Telegram e o envio de mensagens de forma assíncrona. Para se comunicar com a API do GPT-4, pode-se usar a biblioteca openai, que permite o envio de prompts e a obtenção de respostas de maneira simples e direta.

2.2 Modelagem de banco de dados com SQLite

A modelagem de bancos de dados é uma etapa essencial no desenvolvimento de sistemas que requerem armazenamento e gerenciamento de dados de maneira eficiente e estruturada. No caso de aplicações leves ou sistemas embarcados, o SQLite se destaca como uma solução de banco de dados relacional que oferece uma abordagem simplificada e de fácil implementação. O SQLite é um banco de dados embutido, ou seja, ele armazena dados em um único arquivo de disco e não requer a instalação de um servidor de banco de dados separado, o que o torna ideal para aplicações de pequeno a médio porte, sistemas embarcados e dispositivos móveis. O SQLite possui um conjunto completo de recursos SQL e oferece uma abordagem transacional para garantir a consistência dos dados, mesmo em cenários de falhas. Ele segue o modelo relacional, no qual os dados são organizados em tabelas, com linhas representando registros e colunas representando atributos dos dados. A normalização, uma técnica importante na modelagem de dados, visa organizar as informações de maneira a evitar redundância e inconsistência, sendo a 3ª Forma Normal (3NF) a mais comum, pois elimina dependências transitivas e redundâncias. Além disso, no SQLite, é possível estabelecer relacionamentos entre tabelas utilizando chaves primárias e estrangeiras. A chave primária é um identificador único para cada registro em uma tabela, enquanto a chave estrangeira cria vínculos entre tabelas, referenciando a chave primária de outra. Embora o SQLite suporte integridade referencial, ele não valida automaticamente as chaves estrangeiras, permitindo que essa funcionalidade seja configurada manualmente. O SQLite também oferece suporte a diversos tipos de dados, como INTEGER, REAL, TEXT e BLOB, proporcionando flexibilidade no armazenamento, embora seja necessário

garantir a consistência dos dados no design do banco. No que diz respeito ao desempenho, o SQLite permite a criação de índices para acelerar consultas em grandes volumes de dados. No entanto, a criação excessiva de índices pode afetar negativamente o desempenho em operações de inserção, atualização ou exclusão, o que exige um uso cuidadoso dessa técnica. O SQLite também suporta transações, garantindo a atomicidade das operações de banco de dados, ou seja, ou todas as operações são realizadas com sucesso, ou nenhuma delas é aplicada, o que assegura a integridade dos dados. Embora o SQLite seja uma solução simples e eficiente, ele apresenta algumas limitações, como o suporte limitado a operações de concorrência. Como é um banco de dados embutido, ele não é ideal para ambientes de produção com alta carga de leitura e escrita simultânea. Além disso, ele não oferece suporte completo a algumas funcionalidades avançadas de bancos de dados relacionais, como triggers complexas ou replicação e clustering, tornando-o menos adequado para sistemas de grande porte.

2.3 Utilização de ChatBots no Telegram

Chatbots são programas desenvolvidos para simular interações humanas, permitindo conversas automatizadas com usuários. Esses sistemas são amplamente utilizados em áreas como suporte ao cliente, educação, entretenimento e execução de tarefas automatizadas. Com os avanços em Machine Learning e Processamento de Linguagem Natural (NLP), os chatbots evoluíram significativamente, tornando-se capazes de compreender o contexto das conversas, gerar respostas relevantes e até aprender com as interações realizadas (ALURA, 2023). Os chatbots podem ser classificados em dois tipos principais: baseados em regras e baseados em aprendizado de máquina. Os chatbots baseados em regras utilizam scripts pré-definidos e fluxos de decisão para responder a perguntas específicas. Embora sejam eficientes para interações simples e perguntas diretas, possuem limitações em cenários mais complexos. Já os chatbots baseados em aprendizado de máquina e NLP empregam algoritmos avançados para compreender intenções e gerar respostas dinâmicas. Modelos de linguagem, como os desenvolvidos pela OpenAI, são amplamente utilizados nesse contexto, permitindo respostas mais naturais e adaptadas ao contexto da interação. No que diz respeito à implementação de chatbots no Telegram, a plataforma oferece uma API completa que possibilita a criação e o gerenciamento de bots. Essa API é compatível com diversas linguagens de programação, sendo Python uma das mais populares devido à simplicidade e à vasta disponibilidade de bibliotecas. Entre essas, a biblioteca `python-telegram-bot` destaca-se por facilitar a integração com a API, permitindo a criação de bots que podem receber e responder a mensagens de maneira eficiente e personalizada (COMPOSE, 2023).

3 Metodologia

3.0.1 OpenAI e Integração com Python para Machine Learning

A OpenAI oferece modelos de Machine Learning avançados, como o GPT (Generative Pre-trained Transformer), que são amplamente utilizados para aplicações que demandam interpretação de linguagem, resposta a perguntas, tradução e criação de conteúdo (OPENAI, 2023). Esses modelos destacam-se pela sua capacidade de compreender a linguagem natural e gerar texto de maneira contextualizada e precisa. Para integrar os serviços da OpenAI em um projeto Python, é possível utilizar a OpenAI API, que oferece acesso direto aos seus modelos avançados. O primeiro passo envolve a configuração da API, que exige a criação de uma conta na plataforma OpenAI e a obtenção de uma chave de API. Essa chave é essencial para realizar chamadas aos modelos da OpenAI. Em Python, a biblioteca `openai` simplifica a interação com a API e pode ser instalada facilmente com um comando apropriado. Após a instalação do SDK e a configuração da chave de API, é possível realizar chamadas para os modelos, permitindo a execução de diversas tarefas de Processamento de Linguagem Natural (NLP). Com as configurações devidamente ajustadas, o desenvolvedor pode enviar entradas de texto para a API e receber respostas processadas, utilizando-as para tarefas como geração de texto, análise de sentimentos e respostas a perguntas. Além disso, a OpenAI API é uma ferramenta poderosa para aplicações avançadas de NLP e Machine Learning. Por exemplo, para gerar texto com base em uma entrada específica, o desenvolvedor pode usar códigos simples que enviam o texto para a API e retornam uma resposta gerada pelo modelo. Isso permite construir soluções eficientes para elaboração de resumos, criação de conteúdo automatizado e outras funcionalidades. Por fim, a OpenAI oferece modelos adicionais que vão além da geração de texto, incluindo reconhecimento de imagem e outras tarefas relacionadas a dados. Isso possibilita a criação de projetos integrados que combinam visão computacional e processamento de linguagem. A combinação do GPT com outros modelos permite o desenvolvimento de sistemas capazes de aprender com interações, ajustar respostas automaticamente e realizar análises preditivas, ampliando ainda mais o potencial de inovação em projetos. As configuração da API junto a configuração das respostas geradas pela mesma, podem ser visualizadas no Apêndice B entre as linhas 39 e 88 do código fonte referente ao "bot.py"

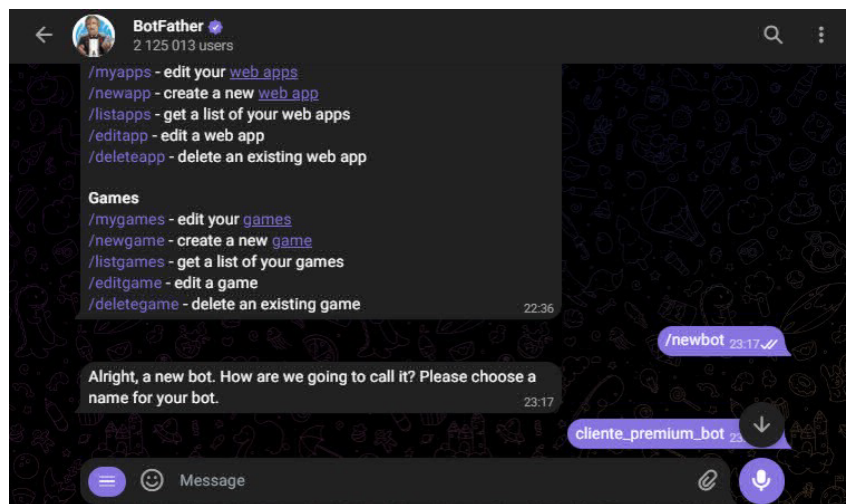
3.0.2 Criação de Bot no Telegram com BotFather

Inicialmente, foi necessária a criação de um bot para simular um cliente real que buscasse informações sobre imóveis. Através desse chatbot e da API do Telegram, tornou-se possível utilizar um token exclusivo para captar as informações geradas nas conversas

realizadas no chat.

Após a criação do bot Figura 3, é fornecido um token que será utilizado posteriormente no código-fonte da aplicação. Esse token está vinculado à conta do Telegram associada ao usuário, sendo, portanto, de caráter pessoal e intransferível devido a questões de segurança e direitos autorais. Para aplicações em larga escala ou com finalidades profissionais, recomenda-se o uso de uma conta específica no Telegram, dedicada exclusivamente a essa funcionalidade, garantindo maior organização e confiabilidade no sistema.

Figura 3 – Criação do bot via botfather.



Fonte: Autoria própria (2024).

3.0.3 Configuração Flask para InterfaceWeb e tratamento do Banco de Dados

Para iniciar o código, é necessário importar as bibliotecas e todas as APIs utilizadas, junto aos seus respectivos Tokens. Nessa parte do código, também são definidas as funções de usuários e senhas para acesso a interface web, junto a aplicação Flask que executará a aplicação.

Definido esse primeiro passo, é necessário estruturar o banco de dados da aplicação, isso será realizado via SQLite, utilizando o SQLAlchemy:

Até aqui, todas as opções referentes ao perfil do imóvel e perfil do cliente foram definidos em um banco de dados junto a sua estrutura. Dessa forma, a interação que for resultante do ChatBot, já terá uma “planilha” organizada para preenchimento de requerimento, o que posteriormente facilita a obtenção de resposta pelo usuário.

3.0.4 Configuração Bot Telegram para aquisição de resposta

Neste ponto, são configuradas as APIs do Telegram e OpenAI para utilização tanto do chatbot quanto do ChatGPT para o processo de Machine Learning. Estas APIs, utilizam um Token referente as contas do Telegram e da OpenAI do usuário. A codificação

referente a esta parte, pode ser visualizada no Apêndice B entre as linhas 36 e 88 do código fonte referente ao "bot.py". O resultado deste processo no código, é retirar do campo de texto digitado pelo usuário as palavras chaves para obtenção do imóvel ideal para o cliente, itens como: número de quartos, se o cliente em questão tem filhos e zonas de interesse na cidade são alguns dos exemplos utilizados, conforme ilustrado na Figura 4.

Figura 4 – Terminal de saída com dados captados pelo GPT-4 mediante a resposta do usuário.

```
"filhos": "",
"pagamento": "a vista"
}
{'imovel': 'apartamento', 'numero_de_quartos': '', 'obra': '', 'urgencia': '',
'referencia': ['laranjeiras'], 'filhos': '', 'pagamento': 'a vista'}
['laranjeiras']
imoveis
[<Propriedade 2>]
busca por zona sul
Todas as propriedades:
[<Propriedade 1>, <Propriedade 2>]
resposta
█
```

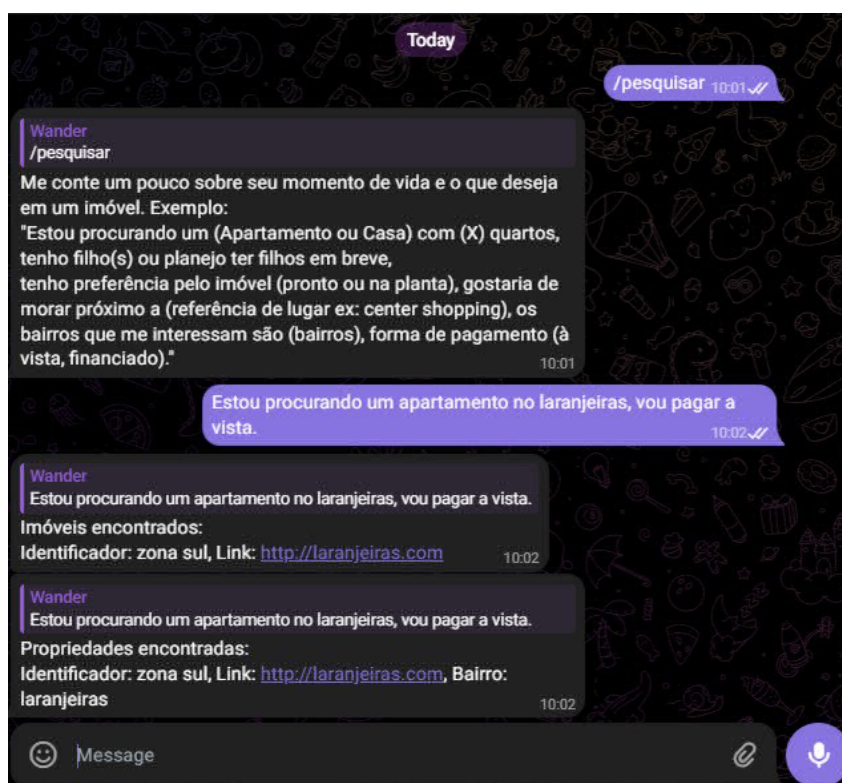
Fonte: Autoria própria (2024).

Caso o banco de dados não possua imóveis catalogados que irão atender o cliente, algumas possibilidades de respostas são fornecidas, como também a possibilidade do usuário ter digitado algo incomum, o que faz o Bot pedir uma nova informação ou consulta, conforme ilustrado na Figura 5

3.0.5 Registro de imóveis para alimentação do Banco de Dados

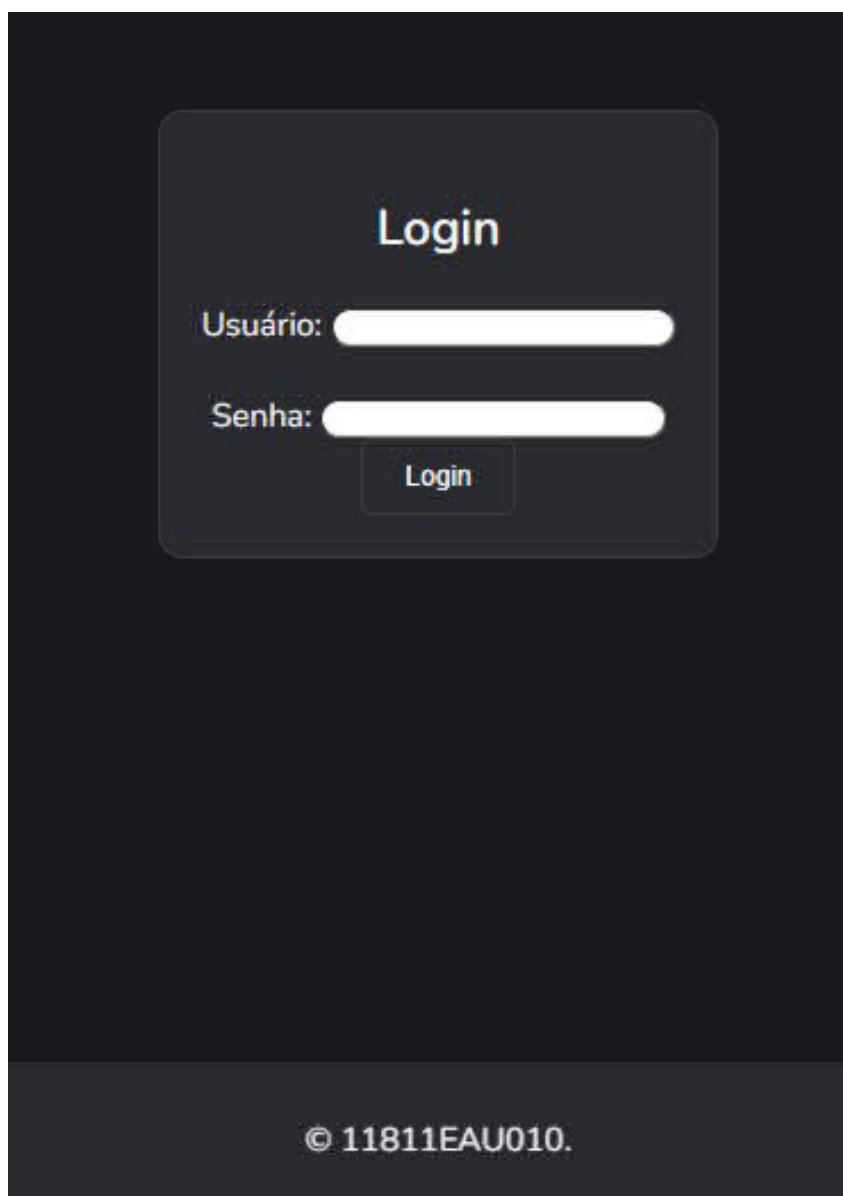
Todos os imóveis que serão utilizados para retorno de resposta ao cliente, são catalogados em um banco de dados com auxílio de uma interface web desenvolvida via Flask. A codificação referente a esta parte, pode ser visualizada no Apêndice A entre as linhas 145 e 179 do código fonte referente ao "app.py". Como resultado, posteriormente, é liberada uma aplicação html, que pode ser acessada via link fornecido no terminal, para consulta dos imóveis cadastrados, por lá é possível registrar, alterar ou excluir um imóvel do banco de dados. Esta interface é representada pelas Figuras 6, 7 e 8.

Figura 5 – ChatBot no telegram com requisição de informação pelo usuário.



Fonte: Autoria própria (2024).

Figura 6 – Tela de login fornecida pela aplicação Flesk.

The image shows a login screen for the Flesk application. It features a dark background with a central, slightly lighter gray rounded rectangle containing the login form. The form has the title "Login" at the top. Below it are two input fields: "Usuário:" and "Senha:", each followed by a white rounded rectangular input box. A "Login" button is positioned below the password field. At the bottom of the screen, there is a dark gray bar containing the copyright notice "© 11811EAU010." in white text.

Login

Usuário:

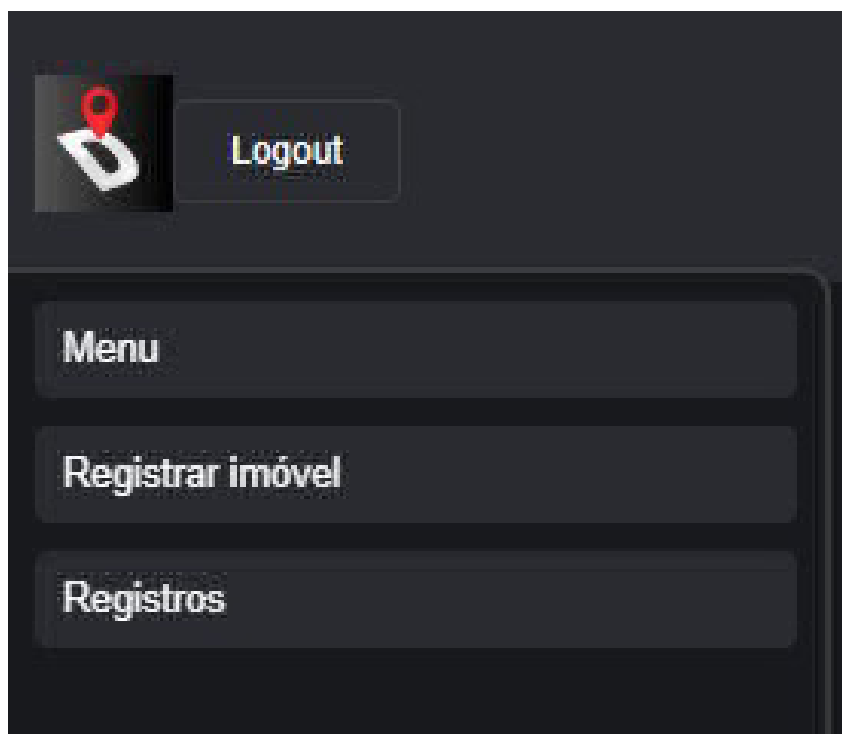
Senha:

Login

© 11811EAU010.

Fonte: Autoria própria (2024).

Figura 7 – Menu da interface web.



Fonte: Autoria própria (2024).

Figura 8 – Tela de registro de imóvel.

A tela de registro de imóvel possui um formulário com o título 'Registrar Imóvel' em amarelo. Os campos de entrada são: 'Identificador do Imóvel' com o valor 'zona leste', 'Categoria' com o valor 'Casa' e uma seta para baixo, 'Número de Quartos' com o valor '3', 'Obra' com o valor 'Pronta' e uma seta para baixo, 'Bairro' com o valor 'santa mônica', 'Áreas de Lazer em Comum' com o valor 'Sim' e uma seta para baixo, e 'Link do Imóvel' com o valor 'http://santamonica.com'. Um botão 'Registrar Imóvel' em amarelo está localizado abaixo dos campos.

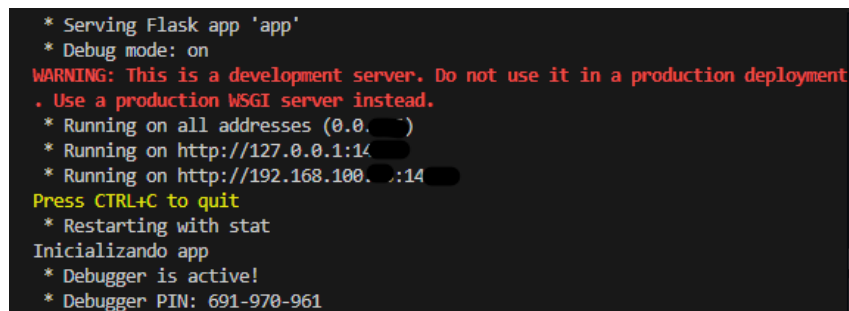
Fonte: Autoria própria (2024).

4 Resultados e Discussões

Este trabalho teve como objetivo o desenvolvimento de um assistente virtual em linguagem python para filtro de cliente no mercado imobiliário. O sistema foi projetado para coletar dados dos usuários via chata e responder à consultas de forma interativa e intuitiva, utilizando modelos de linguagem natural da OpenAI para interpretação e solicitação via SQLAlchemy para acessar e retornar dados precisos do banco de imóveis catalogados. O experimento demonstrou que o chatbot é capaz de responder a uma ampla variedade de perguntas, desde detalhes específicos sobre imóveis até informações de localização, faixa de preço e tipo de propriedade. O uso da API da OpenAI trouxe precisão e naturalidade nas respostas, oferecendo uma experiência de conversação envolvente e adaptável a cada interação. A estrutura do banco de dados SQLAlchemy garantiu um armazenamento eficiente e uma recuperação de dados rápida, fundamental para atender a demanda de informações em tempo real.

O sistema foi testado em um ambiente de produção simulado e obteve desempenho robusto e confiável, com baixo tempo de resposta, uma vantagem da aplicação Flask, como representada pela Figura 9 que exibe a resposta rápida do sistema. As consultas foram respondidas com um tempo médio de menos de dois segundos, demonstrando a capacidade de atender a diversas solicitações simultâneas sem comprometer a qualidade do atendimento, como resultado explícito na Figura 4.

Figura 9 – Resposta do sistema para gerar aplicação Flask com interface web.

A terminal window with a dark background and light-colored text. The text shows the startup of a Flask application in debug mode. It includes a warning about using a development server in production, the IP addresses the server is running on (0.0.0.0 and 127.0.0.1), and instructions to press CTRL+C to quit. It also shows the app restarting with 'stat' and the debugger being active with a PIN.

```
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment
. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:14
* Running on http://192.168.100.1:14
Press CTRL+C to quit
* Restarting with stat
Iniciando app
* Debugger is active!
* Debugger PIN: 691-970-961
```

Fonte: Autoria própria (2024).

5 Conclusão e Trabalhos Futuros

O presente trabalho foi desenvolvido com base nas necessidades e aprendizados adquiridos durante uma experiência de trabalho. Ele abrangeu fundamentações teóricas e aplicações de indicadores de desempenho, além de resposta ao sistema de aplicações robustas. Através da coleta de informações e dos devidos tratamentos realizados, foi possível obter uma compreensão mais aprofundada do funcionamento de uma IA vinculada ao atendimento ao público. Com esse trabalho, fica evidente que a automação de processos de dados desempenha um papel crucial no sucesso das empresas modernas. Ademais, trata-se de uma ferramenta poderosa para construir indicadores relevantes e obter informações valiosas que impulsionam o aprimoramento contínuo e a tomada de decisões estratégicas. Em termos de potencial de mercado, a aplicação deste chatbot no setor imobiliário de Uberlândia representa uma inovação significativa, uma vez que facilita a comunicação entre clientes e imobiliárias, otimizando o processo de busca por imóveis. Contudo, para um projeto de maior alcance, é interessante salientar o investimento em APIs pagas, como WhatsApp, aplicativo mais difundido e comumente utilizado pelos usuários. Este projeto não apenas confirmou a viabilidade técnica da integração entre Telegram, IA e banco de dados em Python, mas, também abre portas para expansões futuras, como o uso de algoritmos de recomendação e análises de perfil dos usuários para uma experiência ainda mais personalizada. Em conclusão, os resultados obtidos indicam que a solução proposta é eficaz, escalável e apresenta grande potencial para transformar a maneira como o mercado imobiliário lida com atendimento e informações.

Referências Bibliográficas

- ALURA. O que é pln? 2023. Disponível em: <<https://www.alura.com.br/artigos/o-que-e-pln?srsltid=AfmBOoqRjgU1uIeAkJt2hl2aAu5n8fqj7rUEVs9EzKkbkvR4NWTtgZO>>. Acesso em: 10 de outubro de 2024. Citado 2 vezes nas páginas 15 e 18.
- COMPOSE, C. Bot de telegram. 2023. Disponível em: <<https://www.chatcompose.com/PT/telegram.html>>. Acesso em: 11 de outubro de 2024. Citado na página 18.
- DOCUMENTATION, P. Python telegram bot. 2022. Disponível em: <<https://docs.python-telegram-bot.org/en/v21.3/>>. Acesso em: 02 de junho de 2024. Citado na página 13.
- DOCUMENTATION, S. Sqlalchemy. 2022. Disponível em: <<https://docs.sqlalchemy.org/en/20/>>. Acesso em: 02 de junho de 2024. Citado na página 13.
- MARILEUSA, G. Uberlândia é destaque em desenvolvimento e crescimento populacional e imobiliário. 2024. Disponível em: <<https://www.granjamarileusa.com.br/blog/uberlandia-e-destaque-em-desenvolvimento-e-crescimento-populacional-e-imobiliario-registra-numeros#:~:text=O%20censo%20do%20IBGE%20apontou,habitantes%2C%20entre%202010%20e%202022>>. Acesso em: 02 de junho de 2024. Citado na página 12.
- OPENAI. Chatgpt. 2023. Disponível em: <<https://openai.com/index/chatgpt/>>. Acesso em: 10 de outubro de 2024. Citado na página 19.
- VASWANI, A. et al. Attention is all you need. *Advances in Neural Information Processing Systems*, v. 30, 2017. Disponível em: <<https://arxiv.org/abs/1706.03762>>. Citado na página 15.
- ZAP+. Jornada do consumidor de imóveis. 2022. Disponível em: <https://www.datazap.com.br/wp-content/uploads/2022/03/202108_jornada_do_consumidor_compra_final.pdf>. Acesso em: 02 de junho de 2024. Citado na página 11.

A Apêndice A - Código app.py

O código Python utilizado no Aplicativo está anexado a seguir.

B Apêndice B - Código bot.py

O código Python utilizado para Bot está anexado a seguir.

[illegible]