UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Pedro Soethe Chagas

Aplicação de Chatbot em Âmbito Corporativo para a Automação de Processos em uma Empresa de Eletroeletrônicos Industriais

Florianópolis - SC
2024

Pedro Soethe Chagas

Aplicação de Chatbot em Âmbito Corporativo para a Automação de Processos em uma Empresa de Eletroeletrônicos Industriais

Relatório final da disciplina DAS5511 (Projeto de Fim de Curso) como Trabalho de Conclusão do Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Santa Catarina em Florianópolis.
Orientador: Prof. Ricardo José Rabelo, Dr.
Supervisor: Darley Rodrigo Machado

Florianópolis - SC
2024

Ficha catalográfica gerada por meio de sistema automatizado gerenciado pela BU/UFSC.
Dados inseridos pelo próprio autor.

Chagas, Pedro Soethe
Aplicação de Chatbot em Âmbito Corporativo para a Automação de Processos em uma Empresa de Eletroeletrônicos Industriais / Pedro Soethe Chagas ; orientador, Ricardo José Rabelo, 2024.
71 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia de Controle e Automação, Florianópolis, 2024.

Inclui referências.

1. Engenharia de Controle e Automação. 2. Chatbot. 3. Python. 4. Banco de Dados. I. Rabelo, Ricardo José. II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. III. Título.

Pedro Soethe Chagas

Aplicação de Chatbot em Âmbito Corporativo para a Automação de Processos em uma Empresa de Eletroeletrônicos Industriais

Esta monografia foi julgada no contexto da disciplina DAS5511 (Projeto de Fim de Curso) e aprovada em sua forma final pelo Curso de Graduação em Engenharia de Controle e Automação

Florianópolis, 10 de Julho de 2024.

Prof. Marcelo de Lellis Costa de Oliveira, Dr.
Coordenador do Curso

Banca Examinadora:

Documento assinado digitalmente
Ricardo Jose Rabelo
Data: 29/07/2024 11:03:46-0300
CPF: ***.802.119-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Ricardo José Rabelo, Dr.
Orientador
UFSC/CTC/DAS

DARLEY RODRIGO MACHADO:00568314980

Assinado de forma digital por DARLEY RODRIGO MACHADO:00568314980
Dados: 2024.07.29 11:21:25 -03'00'

Darley Rodrigo Machado
Supervisor
Empresa WEG

Prof. Eric Aislan Antonelo, Dr.
Avaliador
Instituição UFSC

Prof. Hector Bessa Silveira, Dr.
Presidente da Banca
UFSC/CTC/DAS

Este trabalho é dedicado aos meus colegas trabalho, de classe e a minha família e professores que sempre me deram apoio e amparo em todos os momentos.

AGRADECIMENTOS

Agradeço à minha família por sempre me darem incentivarem a continuar em frente em todas as fases da minha vida.

Ao Departamento de Automação e Sistemas pelas oportunidades de aprendizado oferecidas ao longo do curso.

Aos meus colegas e colaboradores da WEG que me deram possibilitaram e ajudaram a desenvolver esse projeto.

Agradeço ao meu orientador, Prof. Dr. Ricardo José Rabelo, por todo o suporte e conselhos fornecidos ao longo do desenvolvimento desse projeto.

DECLARAÇÃO DE PUBLICIDADE

Jaraguá do Sul - SC, 01 de Julho de 2024.

Na condição de representante da WEG Equipamentos Elétricos S/A na qual o presente trabalho foi realizado, declaro não haver ressalvas quanto ao aspecto de sigilo ou propriedade intelectual sobre as informações contidas neste documento, que impeçam a sua publicação por parte da Universidade Federal de Santa Catarina (UFSC) para acesso pelo público em geral, incluindo a sua disponibilização *online* no Repositório Institucional da Biblioteca Universitária da UFSC. Além disso, declaro ciência de que o autor, na condição de estudante da UFSC, é obrigado a depositar este documento, por se tratar de um Trabalho de Conclusão de Curso, no referido Repositório Institucional, em atendimento à Resolução Normativa n° 126/2019/CUn.

Por estar de acordo com esses termos, subscrevo-me abaixo.

DARLEY RODRIGO MACHADO:00568314980

Assinado de forma digital por DARLEY RODRIGO MACHADO:00568314980
Dados: 2024.07.29 11:21:43 -03'00'

Darley Rodrigo Machado
WEG Equipamentos Elétricos S/A

RESUMO

No ambiente corporativo, as empresas buscam constantemente melhorar seus níveis de competitividade. Para tal, diversos fatores são considerados. Desde a qualidade dos produtos ou serviços oferecidos, até o atendimento prestado aos clientes. Neste contexto, um dos aspectos a lidar referem-se à agilidade para identificar e resolver problemas, o que inclui mais eficazmente obter as necessárias informações e dados. Entretanto, em um ambiente fabril-corporativo, a quantidade de dados e de sistemas é muito grande, sistemas estes distribuídos e heterogêneos. Um usuário saber como acessar esses dados não é uma tarefa simples e rápida de realizar. Nessa direção, os *chatbot* têm emergido como uma potente tecnologia de informação. Um *chatbot* atua como uma interface humano-máquina mais sofisticada e inteligente, permitindo que os usuários interajam e obtenham dados de vários sistemas de uma forma integrada, utilizando-se de várias formas de interação, incluindo via linguagem natural. No departamento de logística da WEG, vários clientes, nacionais ou internacionais, buscam constantemente por informações sobre seus complexos pedidos, o que têm que ser feito de forma manual. Um dos problemas é que isso não apenas toma um grande tempo dos colaboradores, mas até mesmo não é possível uma resposta rápida por questões de fuso-horário. Dessa maneira, esse projeto tem como objetivo desenvolver um sistema que permita rápidas consultas, de forma intuitiva e facilitada, às informações de pedidos. Junto da empresa, é proposto desenvolver e utilizar de um *chatbots* para realizar esse acompanhamento e localização de pedidos a fim de automatizar os processos de busca das equipes de vendas e logística bem como responder as dúvidas dos clientes de forma automática. Para tal, foi desenvolvido um programa em Python utilizando da biblioteca LangChain para realizar as funções de *chatbots* de interpretação de texto de entrada e formação de textos de saída, e outras para o acesso aos bancos de dados da empresa. O trabalho resultou na concepção de um *chatbots* capaz de realizar buscas no banco de dados da empresa, retornando ao usuário as informações por ele solicitadas.

Palavras-chave: *Chatbot*. *Python*. Banco de Dados.

ABSTRACT

In the corporate environment, companies constantly seek to improve their levels of competitiveness. To do so, several factors are considered. From the quality of the products or services offered to the customer service provided. In this context, one of the aspects to deal with refers to the agility to identify and solve problems, which includes more effectively obtaining the necessary information and data. However, in a corporate-manufacturing environment, the amount of data and systems is very large, these systems being distributed and heterogeneous. For a user to know how to access this data is not a simple and quick task to perform. In this direction, chatbots have emerged as a powerful information technology. A chatbot acts as a more sophisticated and intelligent human-machine interface, allowing users to interact and obtain data from various systems in an integrated manner, using various forms of interaction, including natural language. In the logistics department at WEG, several customers, national or international, constantly seek information about their complex orders, which must be done manually. One of the problems is that this not only takes a lot of time from employees, but it is also not possible to provide a quick response due to time zone issues. Therefore, this project aims to develop a system that allows for quick, intuitive, and facilitated queries to order information. Together with the company, it is proposed to develop and use a chatbot to perform this order tracking and location in order to automate the search processes of the sales and logistics teams as well as to automatically respond to customer inquiries. For this purpose, a program was developed in Python, using the LangChain library to perform the chatbot functions of interpreting input text and forming output texts, and others to access the company's databases. The work resulted in the conception of a *chatbot* capable of searching the company's database, returning to the user the requested information.

Keywords: *Chatbot*. *Python*. Database. *Azure OpenAI*. *Azure AI Services*.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Em um mundo empresarial acelerado e competitivo, eficiência e precisão são cruciais para se manter no topo da sua indústria. Muitos processos manuais frequentemente não atendem a essas demandas (FORBES, 2023). Dessa forma, a busca por eficiência e otimização de processos é uma prioridade constante para as empresas, especialmente aquelas atuantes no setor logístico. A necessidade de responder rapidamente às demandas dos clientes e fornecer informações precisas de forma ágil é crucial para manter a competitividade e a satisfação do cliente. Neste contexto, a automação de processos mediante o uso de tecnologias avançadas, como chatbots, tem se destacado como uma solução promissora. O objetivo principal dos chatbots é permitir que os computadores conduzam conversas em linguagem natural com seres humanos; do ponto de vista humano, essas conversas devem ser o mais parecidas possível com as interações entre pessoas. Portanto, tornar isso possível se tornou um desafio central, com muitos pesquisadores buscando a melhor técnica para permitir que o chatbot converse como um ser humano. Um chatbot bem-sucedido tem a capacidade de entender a mensagem do usuário, cumprir a solicitação, recuperar as informações necessárias com precisão e responder de maneira que o usuário considere a conversa comparável a um diálogo com quase qualquer outra pessoa (MOHAMAD SUHAILI; SALIM; JAMBLI, 2021).

A logística corporativa envolve uma série de operações complexas que requerem coordenação precisa e comunicação eficiente. Frequentemente, os funcionários dedicam grande parte do seu tempo a atividades repetitivas, como o atendimento de consultas sobre status de entregas, disponibilidade de produtos e prazos de entrega. Esse tempo, que poderia ser utilizado em tarefas mais estratégicas, acaba sendo consumido por interações que, embora importantes, são rotineiras e padronizáveis.

A implementação de um chatbot oferece uma solução prática e eficiente para essa problemática. Utilizando inteligência artificial e processamento de linguagem natural, um chatbot pode responder automaticamente a uma ampla gama de consultas dos clientes, reduzindo significativamente a carga de trabalho dos funcionários e permitindo que eles se concentrem em tarefas de maior valor agregado (FORBES, 2024). Além disso, a capacidade de operar 24 horas por dia, 7 dias por semana, garante que os clientes obtenham respostas imediatas, independentemente do horário, melhorando assim a experiência do cliente e a eficiência operacional da empresa.

Este PFC explora os benefícios e desafios da aplicação de chatbots no setor logístico, apresentando a tentativa de aplicação em uma empresa do ramo de eletroeletrônicos industriais que deseja utilizar dessa tecnologia para aprimorar seus processos internos e melhorar a eficiência no atendimento a clientes. A pesquisa inclui uma análise dos impactos da automação no tempo de resposta às solicitações dos clientes, na

produtividade dos funcionários.

1.1 PROBLEMA

No cotidiano da empresa, clientes entram em contato constantemente com os colaboradores do departamento de logística da WEG em busca de informações sobre seus pedidos. Nesse sentido, a problemática abordada neste trabalho refere-se à constante necessidade dos clientes de obter informações atualizadas sobre seus pedidos e embarques. Este cenário cria uma situação em que os trabalhadores do departamento de logística são frequentemente interrompidos em suas atividades regulares para atender às demandas dos clientes. Essa atividade pode tomar uma parte considerável do dia do colaborador para conseguir responder todos os e-mails de clientes com as informações requisitadas.

Outra questão levantada nessa problemática é a disponibilidade do coordenador para responder às requisições. O tempo médio de espera até a requisição ser atendida por um colaborador é de, atualmente, 6h[1], podendo chegar a até dois dias[1] após a chegada do e-mail. Os diferentes colaboradores que recebem essas mensagens possuem diferentes demandas, entretanto o tempo de busca e resposta da informação pode variar entre 5min a 20min[1], dependendo do quão complexa é a informação desejada. Ademais, o tempo médio de busca é por volta de 10min[1] para cada e-mail. Individualmente, não é um valor muito elevado, mas como os e-mails são respondidos de forma sequencial, esse número escala consideravelmente com a quantidade. Cerca de 15 e-mails[1] chegam por dia para os coordenadores, assim, ao final de uma semana de trabalho, mais um dia produtivo é alocado a apenas responder e-mails sobre informações de pedidos e/ou embarques. Esta é uma atividade repetitiva e padronizada, onde o trabalho mecânico torna essa tarefa um estorvo, resultando em uma diminuição da produtividade e eficiência operacional.

A fim de resolver essa questão, foi proposto um projeto que visa a automatização do processo de coleta e fornecimento de informações sobre pedidos e embarques. A ideia central do projeto é desenvolver uma solução tecnológica capaz de lidar com essas consultas de forma rápida e eficiente, eliminando a necessidade de intervenção humana constante. Com isso, os clientes poderão obter as informações de que precisam de maneira autônoma, reduzindo a carga de trabalho sobre os colaboradores do departamento de logística.

Especificamente, o projeto propõe a criação de um chatbot, uma ferramenta tecnológica que permite interações em tempo real com os clientes. Este chatbot apresenta capacidade de operar 24 horas por dia, 7 dias por semana, oferecendo uma maior flexibilidade de horários e aumentando a eficiência do atendimento. O objetivo

[1] Valores analisados e informados por colaboradores do departamento de logística.

do chatbot é compreender as solicitações dos usuários e fornecer respostas precisas sobre a informação desejada referente aos seus pedidos e embarques.

Além de melhorar a eficiência operacional, este projeto tem o potencial de elevar significativamente a satisfação dos clientes, proporcionando um serviço de atendimento constante e acessível. Os clientes poderão obter as informações que desejam a qualquer momento, sem precisar esperar pelo horário de atendimento ou pela disponibilidade dos colaboradores. Isso não só agiliza o processo de atendimento, como também melhora a experiência do cliente com a empresa, reforçando a imagem da WEG como uma organização inovadora e centrada nas necessidades de seus clientes.

1.2 OBJETIVO GERAL

Este projeto tem como objetivo principal desenvolver um sistema robusto e eficiente que permita consultas rápidas e precisas, de forma intuitiva e facilitada, às informações de pedidos e/ou embarques. A premissa central é criar uma ferramenta que otimize o fluxo de trabalho e melhore a experiência dos usuários, sejam eles membros das equipes internas ou clientes externos.

Para atingir esse objetivo, é proposto, em parceria com a empresa, o desenvolvimento e a implementação de um chatbot. Este chatbot será responsável por realizar a busca e apresentação de informações de pedidos e embarques, automatizando os processos de busca das equipes de vendas e logística. Ao integrar esta solução tecnológica, espera-se reduzir significativamente o tempo gasto pelos colaboradores em tarefas repetitivas e aumentar a velocidade na disseminação de informações.

1.3 OBJETIVOS ESPECÍFICOS

Os objetivos específicos que esperam-se alcançar com o desenvolvimento do projeto são:

- Estudar e analisar chatbots e sua aplicação em um ambiente industrial com escopo corporativo, analisando impactos de sua incorporação em um ambiente real de trabalho;
- Estudar programas, softwares e métodos de consulta a base de dados e demais ferramentas e frameworks de desenvolvimento de chatbots necessários e existentes;
- Entender o processo fabril de um material, passando por todos seus processos dentro da fábrica, e como é extraída a informação sobre ele;
- Desenvolver um sistema de chatbot capaz de entender linguagem natural;

- Implementar a integração entre o chatbot e os meios de consulta para retornar ao usuário informações sobre pedidos ou embarques específicos através de solicitações do usuário por meio de linguagem natural, fazendo busca de informações nas bases de dados da empresa;
- Teste e análise de resultados.

ESTRUTURA DO DOCUMENTO

Além da Introdução apresentada, há mais 7 capítulos que retratam o conceitos do desenvolvimento do projeto do *chatbot*. No capítulo 2 é feita uma descrição da empresa e local de trabalho do autor, com seus processos e produtos. No capítulo 3 é apresentada uma fundamentação teórica, além de exemplos de *chabots* e assistentes virtuais em utilização no mercado. No capítulo 4 são expostos os requisitos gerais, funcionais e não-funcionais, além dos fluxos de informação e materiais, diagrama do processo. No capítulo 5 são descritas as ferramentas utilizadas para o projeto feito. No capítulo 6 é feita uma análise da implementação do projeto na Empresa. No capítulo 7 são analisados os testes e feita a avaliação dos resultados do projeto. No último capítulo é feita a conclusão e perspectivas futuras para o *chatbot*.

2 DESCRIÇÃO DO LOCAL

Quando se pensa em motores, normalmente o primeiro pensamento que vem à mente das pessoas são os carros ou veículos a combustão. Alguns podem associá-los aos velozes carros da Formula 1, ou até mesmo às motos de corrida, e até mesmo aos enormes “monster trucks”. No setor automobilístico, diversas empresas estão presentes no mercado e trabalham com motores, incluindo a BMW, Volkswagen, Ford e Mitsubishi. Por outro lado, ao mencionar motores elétricos, algumas empresas específicas surgem à mente, como Tesla, General Electric, Siemens e, notadamente no Brasil, a WEG. Com destaque para a WEG, ela possui uma história rica desde sua origem até os dias atuais, desenvolvendo vários projetos e ampliando cada vez mais seus horizontes. O projeto proposto foi desenvolvido dentro de um dos setores da WEG.

2.1 DESCRIÇÃO DA EMPRESA

A WEG é uma empresa fundada em Jaraguá do Sul, Santa Catarina, no dia 16 de setembro de 1961, fruto da união de forças de um eletricista, de um administrador e de um mecânico. Originalmente, a empresa focava na fabricação e venda de motores elétricos. No entanto, a partir da década de 1980, a WEG começou a expandir suas atividades, diversificando seu portfólio para incluir a produção de componentes eletroeletrônicos, produtos para automação industrial, transformadores de força e distribuição, tintas líquidas e em pó, e vernizes eletroisolantes (WEG, 2024).

Figura 1 – Logo WEG.

Fonte: Arquivo da empresa (2024).

Com o passar dos anos, a empresa se consolidou não apenas como fabricante de motores, mas também como fornecedora de sistemas elétricos industriais completos. Atualmente, a WEG é reconhecida como a quinta maior empresa do Brasil e uma multinacional com presença global. A empresa possui filiais em diversos países, incluindo os Estados Unidos, Alemanha, Colômbia, China, entre outros, o que demonstra sua capacidade de competir e operar em mercados internacionais.

Hoje, a WEG conta com uma ampla gama de setores, abrangendo áreas como energia, com destaque para painéis solares e geradores eólicos, transformadores, motores industriais e comerciais, automação industrial, tintas e vernizes, e soluções digitais. A empresa tem se destacado no mercado de energia renovável, com investimentos significativos em tecnologias sustentáveis, como energia solar e eólica, reafirmando seu compromisso com a inovação e a sustentabilidade.

A diversificação e expansão contínua do portfólio de produtos e serviços da WEG refletem a visão estratégica da empresa de crescer e se adaptar às demandas do mercado global. A capacidade de inovar e a busca constante por excelência tecnológica têm sido pilares fundamentais do sucesso da WEG, permitindo à empresa não apenas acompanhar, mas liderar tendências em vários segmentos da indústria.

Como resultado dessa trajetória de crescimento e inovação, a WEG se tornou um exemplo de sucesso empresarial no Brasil e no mundo, reconhecida por sua qualidade, eficiência e compromisso com o desenvolvimento sustentável. A empresa continua a investir em pesquisa e desenvolvimento, garantindo que esteja na vanguarda das inovações tecnológicas e preparada para enfrentar os desafios e oportunidades do futuro.

2.2 DESCRIÇÃO DO LOCAL DE TRABALHO

O local onde o projeto descrito no presente relatório foi desenvolvido é o Departamento de Logística da WEG. É por esse departamento que toda informação sobre pedidos, nacionais ou internacionais, passa antes de sair da empresa. O local de trabalho do autor é na seção de Gestão de Estoques do departamento de logística. O time de gestão de estoques é responsável por análise de indicadores operacionais ligados a estoque, e análise e suporte a filiais internacionais.

O projeto surgiu dado um acontecimento constante durante o cotidiano dos trabalhadores do departamento. Clientes ou filiais internacionais, desejando saber informações sobre seus pedidos, entram em contato com a seção de exportação do departamento diariamente. Esse acontecimento acaba por tomar tempo dos colaboradores que precisam parar seus afazeres e tarefas para dar atenção e responder tais perguntas. Esse fato gerou a necessidade de um sistema capaz de automatizar esse processo. Assim, a ideia do chatbot surgiu como uma possível solução ao problema.

2.3 PROCESSOS

Uma parte importante para o entendimento completo do problema é a contextualização e explicação dos processos envolvendo o ambiente em que o problema se encontra. Dentro desse contexto, um pedido na empresa passa por diversos estágios até chegar a porta do cliente, e cada um deles será explicado a seguir.

2.3.1 Processo de pedido

Dado seu exponencial crescimento, uma enorme gama de produtos são produzidos pela WEG todos os dias, sendo eles tanto para o mercado interno, quando para o externo. Cada produto tem sua peculiaridade de produção, entretanto, em situações normais, sempre que um cliente deseja realizar um pedido para a empresa, segue-se o processo a seguir:

Inicialmente, o cliente entra em contato com o time de vendas da empresa, especificando qual tipo de material deseja e as quantidades necessárias. Esse contato pode ser feito por diversos meios, como telefone, e-mail, ou através do site da empresa. Durante essa fase, os representantes de vendas fornecem ao cliente todas as informações necessárias, incluindo detalhes sobre os produtos, prazos de entrega, e condições de pagamento. Após a confirmação de todos os detalhes, o pedido é formalmente colocado.

O pedido, então, é passado ao time de Planejamento e Controle da Produção (PCP) da fábrica. Com o PCP, o pedido é adicionado ao cronograma de produção. Nesta etapa, o time de PCP avalia a capacidade de produção disponível, a disponibilidade de matérias-primas e os prazos de entrega solicitados pelo cliente. Com base nessas informações, o pedido é agendado para produção em uma data específica. Se houver necessidade de ajustar o cronograma para acomodar o pedido, o time de PCP fará os ajustes necessários para otimizar a produção.

Com o pedido agendado, a produção é iniciada conforme planejado. O processo de produção envolve várias etapas, desde a preparação das matérias-primas até a fabricação, montagem (se aplicável), e inspeção de qualidade final.

Após a conclusão da produção, é feita a etapa de pagamento. Essa etapa é automatizada para a maioria dos pedidos, entretanto há aqueles que necessitam da atenção do time de vendas. Assim, para esses casos, o time de vendas é notificado de que o pedido está pronto. Neste ponto, o time de vendas entra em contato com o cliente para providenciar a cobrança do pagamento. A confirmação do pagamento é essencial para que o pedido seja liberado para entrega.

Uma vez confirmado o pagamento, o pedido é liberado para entrega. O processo de entrega varia em tempo e método, dependendo de diversos fatores, como a localização do cliente, o meio de transporte escolhido (terrestre, aéreo, marítimo), e a urgência da entrega. Para pedidos locais, a entrega pode ser feita por transportadoras parceiras ou pela frota própria da empresa. Para pedidos internacionais, o envio pode envolver processos adicionais de documentação e alfândega. A parte do “como” a entrega é realizada será detalhada mais adiante neste relatório, incluindo rastreamento de pedidos e outros tipos de informações.

2.3.2 Armazenamento na exportação

Nesta etapa, após o pedido ser produzido, ele é armazenado no estoque de exportação (WEX). No caso de armazenamento, os pedidos possuem uma data prevista de embarque, quando serão despachados, mas enquanto a data de embarque está distante eles permanecem na WEX. É nesta etapa que embarques são criados. Embarques consistem de um ou mais pedidos de um mesmo cliente que serão despachados em uma determinada data, dependendo da forma que será transportada. Os embarques possuem várias informações diferentes, como o modo de transporte, podendo ser marítimo, aéreo ou viário, diferentes destinos, datas estimadas de saídas ou chegadas, número de tracking, código do container, responsável pelo transporte, entre outros. Várias informações estão disponíveis no sistema da empresa quando se trata dos embarques.

Tendo o embarque de um pedido sido criado e o container ter sido fechado, o container está pronto para ser despachado.

2.3.3 Processo de despacho

No processo de transporte de mercadorias, é fundamental considerar os diferentes tipos de meios disponíveis para assegurar que as cargas cheguem ao destino com eficiência e segurança. Os modos de transporte utilizados pela WEG são o aéreo, o marítimo e o viário, cada um com suas características específicas, vantagens e limitações.

Focando especificamente no transporte marítimo, este é amplamente utilizado para o transporte de grandes volumes de mercadorias, especialmente em longas distâncias, devido à sua capacidade de carga e custo-benefício. Nesse tipo de transporte, os pedidos são colocados em containers, que oferecem proteção e facilitam o manuseio e a movimentação das mercadorias. Esses containers são então transportados para o porto de origem, onde passam por processos de verificação e documentação antes de serem carregados nos navios.

2.3.4 Processo de saída do porto

Uma vez a bordo, os navios de carga partem em direção ao porto de destino, seguindo rotas marítimas específicas. Ademais, desde o momento que as mercadorias chegam no porto, o sistema da WEG não realiza acompanhamento em tempo real, nesse caso um outro sistema realiza esse acompanhamento, o GT Nexus. Esse programa é de um fornecedor externo à empresa. É nesse programa que os armadores depositam informações sobre os containers que estão carregando.

As informações nesse sistema são baseados em eventos. A cada ponto chave no trajeto do navio um novo evento é disparado, alguns exemplos de eventos são,

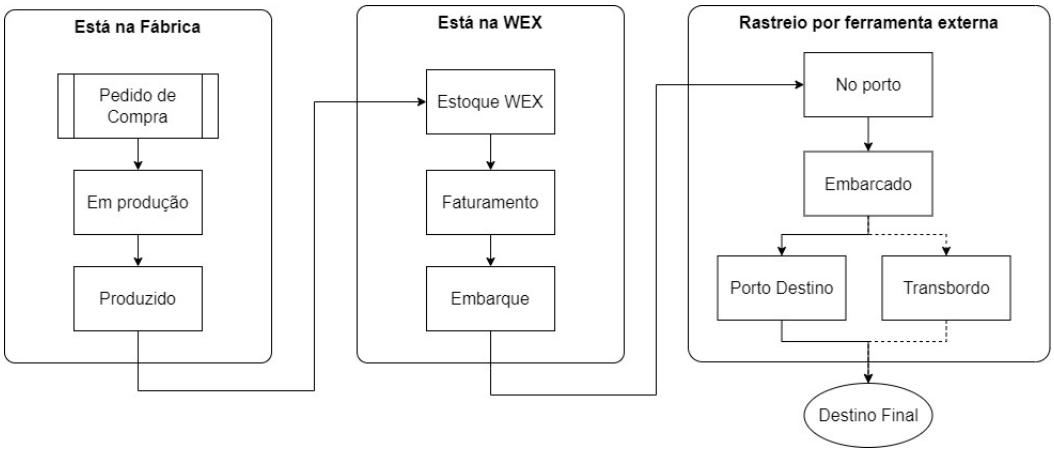
chegada ao porto de carregamento, saída do porto de carregamento, chegada ao porto de destino, entre outros. Cada evento é acompanhado por uma data do acontecimento disparo do evento, assim é possível ter uma noção de onde está o navio.

Uma outra forma de saber onde está o navio é através do site do armador responsável pelo transporte. Utilizando um código associado ao container, pode-se verificar em tempo real onde está o navio. Esse acompanhamento é feito até chegar ao porto de destino, onde é descarregado do navio.

2.3.5 Caminho do pedido

Para uma melhor visualização dos processos mencionados anteriormente, e como é a sequência das atividades, foi construído o diagrama apresentado na figura 2.

Figura 2 – Caminho do pedido.



Fonte: Arquivo do autor (2024).

3 METODOLOGIA

Para desenvolver o projeto mencionado, foi utilizada a metodologias Design Thinking para o planejamento e desenvolvimento das partes. Dessa forma, o desenvolvimento do projeto se deu de forma gradual e iterativa, permitindo a adaptação constante às necessidades e feedbacks dos usuários, garantindo que o produto final fosse altamente funcional e eficiente.

3.1 DESIGN THINKING

Design Thinking é uma abordagem centrada no ser humano para a inovação, que utiliza um conjunto de metodologias e processos para resolver problemas complexos de maneira criativa e eficaz. Originada no campo do design, essa metodologia tem sido amplamente adotada em diversas áreas, como negócios, engenharia, educação e serviços públicos, devido à sua capacidade de gerar soluções inovadoras e centradas nas necessidades reais dos usuários.

A metodologia Design Thinking pode ser dividida em quatro etapas principais: Imersão, Definição, Ideação e Prototipação.

3.1.1 Imersão

Nesta fase a equipe inicia a compreensão do desafio que terá pela frente, explorando-o sob as perspectivas tanto da empresa quanto da pessoa usuária ou cliente final. A etapa imersiva pode ser dividida em dois momentos distintos. O primeiro é focado em conhecer o problema e dar visibilidade para a equipe do projeto, fazendo com que se familiarize com o contexto por meio de pesquisa e conhecimentos prévios. Já no segundo momento, há um aprofundamento no problema e, por isso, é comum o envolvimento de usuários ou outras pessoas-chave dentro do contexto de projeto (ALURA, 2024b).

3.1.2 Definição

Na segunda etapa do processo de design thinking, começa-se a convergir. Ou seja, refinar os dados e informações da etapa anterior e definir o escopo do projeto. Para facilitar esse processo, os insights gerados são sintetizados e organizados, sendo possível identificar padrões e ter uma maior compreensão do problema a ser resolvido. Nesta fase, a equipe também consegue esclarecer os limites do projeto e também os recursos necessários para a próxima etapa, de criação (ALURA, 2024b).

3.1.3 Ideação

A ideação é a etapa que usa todo o conhecimento sobre o problema para gerar ideias e filtrar quais delas serão levadas adiante para teste. Debates feitos em cima de diferentes pontos de vista são comuns aqui. Desta forma, é altamente recomendável contar com uma diversidade de perfis entre as pessoas envolvidas, incluindo também aquelas que se beneficiarão pelas soluções. Esta etapa é mais um momento de divergir ideias, explorar possibilidades criativas e propor soluções para o problema em debate. O *brainstorming* é a ferramenta mais comum, pois facilita a proposição de ideias e estimula a participação de todos os membros da equipe na geração de soluções (ALURA, 2024b). No contexto desse projeto, essa etapa foi realizada com a equipe da sessão onde o autor trabalha.

3.1.4 Prototipação

A etapa final do processo de design thinking é crucial para validar as ideias geradas, sendo o momento ideal para colocar a mão na massa e ajustar detalhes importantes do projeto. O objetivo é reunir as ideias mais promissoras geradas no processo de ideação, priorizando aquelas com maior potencial e probabilidade de sucesso.

Com o protótipo em mãos, é possível testar o produto junto aos usuários, refinando e aprimorando até que se torne uma solução alinhada às necessidades, capaz de gerar resultados positivos. Um ponto importante é que, embora comumente apresentada como a fase final, a prototipação pode ocorrer em paralelo às outras etapas (ALURA, 2024b).

É importante frisar que o processo de Design Thinking não é linear. Ou seja, é possível sobrepor e revisitar as etapas conforme novos insights são descobertos, priorizando a flexibilidade e a adaptação.

3.2 IMPLEMENTAÇÃO DA METODOLOGIA

Seguindo a metodologia Design Thinking, começa-se com a etapa de imersão. Conversando com o supervisor e as pessoas dentro da área onde o problema está ocorrendo foi construída uma ideia inicial do problema a ser atacado. Assim, foram estudadas e analisadas as bibliografias, ferramentas e frameworks já existentes sobre os vários assuntos abarcados, tais como inteligência artificial, chatbots e bancos de dados. Esta fase inclui, também, o estudo detalhado dos processos da empresa, visando entender como as informações sobre pedidos e embarques são gerenciadas atualmente. Além disso, foi realizada uma pesquisa minuciosa dos dados existentes na empresa que poderão ser utilizados pelo chatbot, garantindo que todas as informações relevantes estejam disponíveis para serem integradas na nova solução.

A etapa de definição aconteceu logo em sequência, e foram levantados onde cada uma das informações desejadas levantadas na etapa de imersão poderia ser encontrada e sua devida importância. Filtrando aquelas que não são requisitadas. Foi nessa etapa que foi definido o leque de quais informações e delimitar as áreas de interesse. Dessa forma, foi definido que inicialmente não seriam analisados pedidos em fábrica para essa versão do chatbot.

Agora, na etapa de ideação, os diálogos a serem trocados entre usuários e o chatbot foram cuidadosamente concebidos. Para isso, foram mapeados os possíveis cenários de interação, identificando as perguntas mais frequentes e as informações mais solicitadas pelos usuários. Com base nessa análise, foram desenvolvidos os scripts de conversação do chatbot, utilizando técnicas de processamento de linguagem natural para garantir que as respostas sejam precisas e contextualizadas.

Agora, na etapa de prototipação, foram criados inicialmente os métodos e funções utilizando a linguagem Python para coletar os dados necessários. Estes métodos foram projetados para acessar, processar e formatar os dados de forma eficiente. Os métodos foram agregados em uma biblioteca autoral, desenvolvida especificamente para este projeto, facilitando sua utilização e manutenção futura. Esta biblioteca servirá como uma base sólida sobre a qual o chatbot será construído, garantindo a consistência e a confiabilidade das informações fornecidas.

Um piloto do chatbot foi disponibilizado para um grupo de usuários, dentro da empresa, permitindo a realização de testes iniciais em um ambiente controlado. Durante essa fase piloto, foram coletados feedbacks dos usuários sobre a funcionalidade e a usabilidade do chatbot. As melhorias e adaptações necessárias são feitas com base nesses feedbacks, permitindo refinar o sistema antes de sua implementação em larga escala.

Após a fase piloto, o chatbot seria gradualmente implementado no setor de logística da empresa. Infelizmente, esta etapa, não pôde ser realizada em decorrência de alguns problemas enfrentados dentro da empresa. Entretanto, a implementação, quando ocorrer, será acompanhada de treinamentos para os colaboradores, assegurando que todos estejam familiarizados com a nova ferramenta e saibam como utilizá-la efetivamente.

Em resumo, a metodologia adotada para o desenvolvimento deste projeto é projetada para ser flexível e adaptativa, garantindo que o chatbot atenda plenamente às necessidades da empresa e de seus clientes. Através de um processo iterativo de desenvolvimento, testes e refinamento, o objetivo é criar uma ferramenta que não apenas melhore a eficiência operacional, mas também eleve a qualidade do atendimento ao cliente, reforçando o compromisso da WEG com a inovação e a excelência.

4 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo será tratada e apresentada a fundamentação teórica correspondente ao desenvolvimento do projeto, abrangendo conceitos tratados neste documento a partir da pesquisa realizada.

4.1 *CHATBOTS* E ASSISTENTES VIRTUAIS

Chatbots e assistentes virtuais são ferramentas muito úteis no dia a dia para a realização de tarefas, suas capacidades auxiliar na realização de tarefas repetitivas. Começando a fundamentação do que são *chatbots*. Um programa pode ser considerado um *chatbot* quando ele suporta uma conversação em (na maioria dos casos) linguagem natural. Já um assistente virtual é definido se suas ações dedicam-se a lidar com necessidades particulares de um ou alguns usuários com objetivos semelhantes dentro de algumas áreas de domínio (RABELO; ZAMBIASI; ROMERO, 2023).

4.1.1 ChatGPT

O ChatGPT (OPENAI, 2022) é um modelo de linguagem avançado desenvolvido pela OpenAI, baseado na arquitetura GPT (Generative Pre-trained Transformer). Ele utiliza técnicas de *Deep Learning* para gerar texto de forma coerente e contextualizada, a partir de entradas fornecidas pelos usuários.

Figura 3 – OpenAI Logo.

Fonte: (OPENAI, 2024)

O ChatGPT é parte da série de modelos GPT da OpenAI, que começou com o lançamento do GPT-1. O GPT-1 foi um avanço significativo, utilizando transformadores, uma arquitetura introduzida em um artigo de pesquisa (VASWANI *et al.*, 2017) pela equipe do Google em 2017 (MEDIUM, 2022). A OpenAI seguiu com o GPT-2, um modelo muito maior e mais poderoso, capaz de gerar texto de alta qualidade, o que chamou a atenção do público e da mídia. Em seguida, veio o GPT-3, ainda mais avançado, com 175 bilhões de parâmetros, tornando-se um dos maiores modelos de linguagem já criados.

Como apontado anteriormente, o ChatGPT é baseado na arquitetura Transformer, que utiliza mecanismos de atenção para processar e gerar texto. Esses me-

canismos permitem que o modelo dê peso diferente a palavras ou partes do texto, dependendo de sua relevância para a tarefa em questão. O modelo é pré-treinado em grandes corpora de texto da internet, o que lhe dá uma base ampla de conhecimento e habilidade para gerar texto em diversos contextos.

4.1.2 Siri

Siri (APPLE, 2024a) é o assistente virtual da Apple, sendo exclusiva da Apple, integrada em seus aparelhos, lançado pela primeira vez em 2011, junto com o iPhone 4S.

Figura 4 – Siri no Apple Watch.

Fonte: (APPLE, 2024b).

Algumas das funções da Siri são:

- **Comandos de voz:** Siri permite que os usuários realizem diversas tarefas por meio de comandos de voz, como enviar mensagens, fazer chamadas, definir alarmes, configurar lembretes, procurar informações na web, e controlar configurações do dispositivo.
- **Integração com o Ecossistema Apple:** Siri está profundamente integrada com o ecossistema da Apple, incluindo iPhones, iPads, Apple Watches, Macs, Apple TVs e HomePods. Isso permite uma experiência contínua e integrada entre diferentes dispositivos
- **Ações Personalizadas:** Siri pode realizar ações personalizadas com base em aplicativos de terceiros que suportam a funcionalidade SiriKit. Isso permite que os desenvolvedores integrem seus aplicativos com Siri para oferecer comandos de voz específicos.

- **Siri Shortcuts:** Introduzido no iOS 12, o Siri Shortcuts permite que os usuários criem atalhos personalizados para ações frequentes ou complexas. Esses atalhos podem ser ativados por comandos de voz personalizados.

4.1.3 Alexa

A Amazon Alexa é a assistente virtual desenvolvida pela Amazon, lançada em 2014. Foi primeiramente utilizada como sistema embarcado nos alto-falantes Amazon Echo. Alexa foi projetada para interagir com o usuário por meio de comandos de voz e podendo realizar várias tarefas, como:

Figura 5 – Alexa Logo.

Fonte: (AMAZON, 2024)

- **Controle de Casa Inteligente:** É possível integrar com a Alexa vários produtos smart, como lâmpadas, ar-condicionado, aquecedores, tomadas, televisões, entre outros, para serem controlados por comando de voz.
- **Reprodução de Músicas e Podcasts:** Alexa pode reproduzir músicas e podcasts por meio de sua integração com aplicativos de áudio, como Spotify, Youtube Music, Deezer, Amazon Music, etc.
- **Responder Perguntas e Buscar Informações:** Alexa consegue responder a uma vasta gama de perguntas e buscar diferentes tipos de informações na internet, como previsão do clima, resolver contas, ler notícias, informar sobre o trânsito, dar a resposta para a vida, o universo e tudo mais, entre outros.
- **Listas e Alarmes:** Com Alexa é possível definir alarmes, fazer lista de compras, organizar agenda, entre outros.
- **Comandos Personalizados:** Utilizando o aplicativo da Alexa, é possível definir rotinas personalizadas ativadas por comando de voz. Essas rotinas podem ser usadas para controlar dispositivos smart, fazer a Alexa falar um texto predefinido, etc.

4.1.4 Cortana (Descontinuado)

A Cortana é a assistente pessoal criada pela Microsoft para ajudar pessoas a terem mais produtividade e ganhar tempo em tarefas corriqueiras no computador (CANALTECH, 2022). A assistente virtual estava incorporada no sistema operacional Windows 10, lançada em 2014. O nome desse assistente surgiu como uma referência a AI assistente presente nos jogos da saga Halo. Entretanto, a assistente de voz da Cortana no Windows foi descontinuada na primavera de 2023, e em outono de 2023 nos aplicativos como Microsoft Teams, Outlook e o Microsoft 365. As funções da Cortana eram:

- **Gerenciar Calendários e Reuniões**
- **Criar e Gerenciar Listas**
- **Configurar Alarmes e Lembretes**
- **Localizar Dados e Abrir Aplicativos**
- **Comandos de Voz e Palavra de Ativação**

4.1.5 Microsoft Copilot

O Microsoft Copilot (MICROSOFT, 2024b) é um assistente inteligente desenvolvido pela Microsoft, projetado para integrar-se aos produtos do Microsoft 365 (anteriormente conhecido como Office 365), como Word, Excel, PowerPoint, Outlook e Teams. Ele utiliza tecnologias de IA para ajudar os usuários a serem mais produtivos e eficientes em suas tarefas diárias. Possui também integração com o DALL-E 3 da OpenAI, possibilitando, assim, a geração de imagens com base em scripts fornecidos pelo usuário.

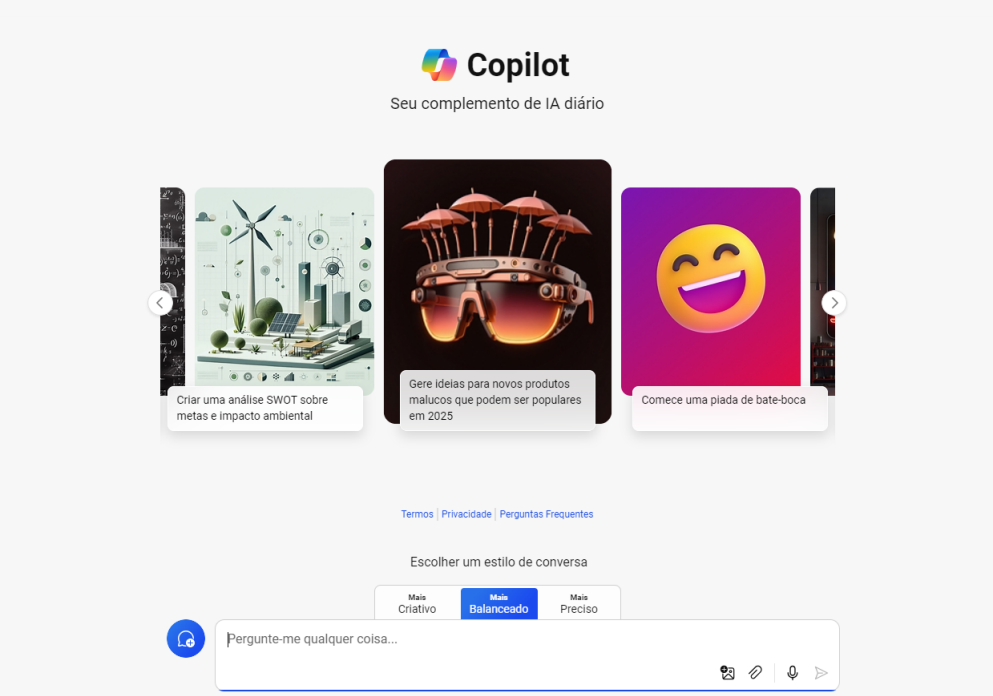
O Microsoft Copilot foi introduzido em 2023 como parte dos esforços da Microsoft para incorporar IA generativa em seus produtos e serviços. A ideia por trás do Copilot é oferecer uma ajuda contínua e contextualizada, baseada nas necessidades específicas dos usuários, para aprimorar a produtividade e facilitar a execução de tarefas complexas.

O Microsoft Copilot oferece uma ampla gama de funcionalidades, ajustadas para diferentes aplicativos do Microsoft 365, alguns exemplos são nos aplicativos:

4.1.5.1 Word:

- *Geração de Texto:* Assistência na escrita de documentos, sugerindo frases, parágrafos ou mesmo seções inteiras dependendo do contexto fornecido.
- *Revisão e Edição:* Sugestões para melhorar a gramática, estilo e clareza do texto.

Figura 6 – Interface no Site do Copilot.



Fonte: (MICROSOFT, 2024c)

- *Resumos:* Criação de resumos automáticos de documentos longos.

4.1.5.2 Excel:

- *Análise de Dados:* Auxilia na análise de grandes volumes de dados, identificando tendências, padrões e anomalias.
- *Geração de Fórmulas:* Sugestões para fórmulas complexas e automação de cálculos.
- *Visualização de Dados:* Criação de gráficos e tabelas baseados nos dados fornecidos.

4.1.5.3 PowerPoint:

- *Criação de Apresentações:* Geração automática de slides com base no conteúdo textual fornecido.
- *Design e Layout:* Sugestões de design para melhorar a estética e a clareza das apresentações.
- *Edição de Conteúdo:* Ajuste e refinamento do conteúdo dos slides.

4.1.6 Bixby

Bixby é um assistente virtual desenvolvido pela Samsung Electronics e lançada em 2017. Bixby é o assistente virtual da Samsung, lançado pela primeira vez em 2017. Ele foi desenvolvido para oferecer uma experiência integrada e intuitiva em dispositivos Samsung, incluindo smartphones, tablets, smart TVs e eletrodomésticos inteligentes. Algumas das funcionalidades desse assistente são:

- **Comandos de voz:** Bixby permite que os usuários executem várias tarefas por meio de comandos de voz. Isso inclui enviar mensagens, fazer chamadas, definir alarmes, abrir aplicativos e controlar configurações do dispositivo.
- **Bixby Vision:** Uma funcionalidade que utiliza a câmera do dispositivo para reconhecer objetos, traduzir texto, escanear códigos QR e fornecer informações sobre locais e produtos.
- **Bixby Routines:** Permite automatizar tarefas com base no comportamento do usuário. Por exemplo, ativar o modo silencioso à noite ou ligar o Wi-Fi ao chegar em casa.
- **Integração com o Ecossistema Samsung:** Bixby está integrado a uma ampla gama de dispositivos Samsung, permitindo controle de smart TVs, geladeiras inteligentes, lavadoras, entre outros.

4.1.7 Google Assistant

O Google Assistant é um assistente virtual desenvolvido pelo Google, lançado pela primeira vez em maio de 2016. Ele está integrado em diversos dispositivos, incluindo smartphones, alto-falantes inteligentes, relógios, carros e outros dispositivos habilitados para a Internet das Coisas (IoT). O Google Assistant é projetado para ajudar os usuários a realizar tarefas diárias, responder perguntas, controlar dispositivos inteligentes e muito mais, utilizando comandos de voz e interações em linguagem natural.

O Google Assistant utiliza uma combinação de processamento de linguagem natural (NLP), machine learning e integração com os serviços do Google para funcionar eficientemente.

4.2 SOFTWARES PARA CHATBOTS

No mercado atual, após a explosão da AI mostrando seus benefícios, existem várias opções de soluções para aqueles que desejam desenvolver seu próprio chatbot. Existem tanto aqueles que é preciso definir todos os parâmetros e construir seu

próprio modelo de LLM, quanto aqueles que fornecem uma estrutura pronta para o funcionamento.

4.2.1 Microsoft Bot Framework

Microsoft Bot Framework (MICROSOFT, 2024d) é um serviço disponibilizado pela Microsoft que permite de forma simples e prática a criação e publicação de bots de AI conversacional de nível empresarial. Junto do framework um outro software auxilia na criação do bot, o Bot Framework Composer. Esse programa é uma tela de criação visual de código aberto para desenvolvedores e equipes multidisciplinares projetarem e construírem experiências conversacionais com Linguagem natural e uma composição sofisticada de respostas de bot.

4.2.2 Wit.ai

O Wit.ai (WIT.AI, 2024) é uma plataforma de desenvolvimento de linguagem natural (NLP) que foi adquirida pelo Facebook em 2015. Ele permite que desenvolvedores integrem facilmente recursos de processamento de linguagem natural em seus aplicativos e dispositivos.

O principal diferencial do Wit.ai é sua simplicidade e facilidade de uso. Ele oferece uma API simples e intuitiva que permite aos desenvolvedores criar e treinar modelos de linguagem natural para entender comandos de voz e texto de maneira eficaz. A plataforma suporta várias linguagens e oferece ferramentas para personalizar e aprimorar o entendimento do contexto das interações.

4.2.3 IBM Watsonx Assistant

O IBM Watsonx Assistant (IBM, 2024b) é uma plataforma avançada de inteligência artificial desenvolvida pela IBM. Ele é projetado para permitir que empresas criem assistentes virtuais altamente sofisticados e personalizados, capazes de interagir de maneira natural com os usuários

Figura 7 – IBM Logo.

Fonte: (IBM, 2024a)

A principal força do IBM Watsonx Assistant reside em sua capacidade de processamento de linguagem natural (NLP), que permite compreender e responder às

perguntas dos usuários de maneira contextual e precisa. Ele utiliza técnicas avançadas de aprendizado de máquina para melhorar continuamente suas respostas e entender melhor o contexto das interações.

4.2.4 Bibliotecas e Frameworks Python

Atualmente, há uma vasta gama de bibliotecas Python especializadas em inteligência artificial, cada uma com suas funcionalidades distintas. Estas bibliotecas são ferramentas essenciais para desenvolvedores e pesquisadores que buscam criar, treinar e implementar modelos de IA em diversas aplicações, desde tarefas simples até projetos complexos.

Algumas dessas bibliotecas são de baixo nível, exigindo que o usuário treine a IA desde o início. Isso significa que o desenvolvedor precisa inserir manualmente os dados de treinamento, ajustar os hiperparâmetros do modelo, avaliar a precisão e interpretar as respostas do modelo gerado. Esse processo pode ser mais trabalhoso e demandar um conhecimento profundo dos fundamentos de machine learning e do funcionamento interno das redes neurais, mas também oferece maior flexibilidade e controle sobre cada etapa do desenvolvimento do modelo.

Por outro lado, existem bibliotecas de alto nível que simplificam significativamente o processo de desenvolvimento de IA. Estas bibliotecas permitem o uso de modelos de LLM (Large Language Models) pré-treinados, que são capazes de gerar respostas com base em grandes conjuntos de dados em que foram previamente treinados. Utilizar esses modelos pré-treinados pode economizar tempo e recursos, além de tornar a IA acessível a um público mais amplo, que pode não ter a mesma profundidade de conhecimento técnico.

Entre as bibliotecas que possibilitam a construção e treinamento de redes neurais e/ou modelos de IA personalizados para atuar como um LLM, destacam-se TensorFlow, PyTorch e Ludwig. TensorFlow e PyTorch são particularmente renomadas no ramo de desenvolvimento de machine learning e inteligência artificial. Elas são amplamente utilizadas tanto na academia quanto na indústria, devido à sua robustez, flexibilidade e grande comunidade de suporte. TensorFlow, desenvolvido pelo Google, é conhecido por sua capacidade de escalar facilmente de projetos de pesquisa para aplicações de produção. PyTorch, desenvolvido pelo Facebook, é apreciado pela sua facilidade de uso e abordagem intuitiva, sendo preferido por muitos pesquisadores devido à sua natureza dinâmica.

Ludwig, por sua vez, é uma biblioteca de mais alto nível que se destaca pela simplicidade de uso, permitindo a criação de modelos de machine learning sem a necessidade de escrever código. Isso a torna uma excelente opção para iniciantes ou para projetos onde a rapidez de desenvolvimento é essencial.

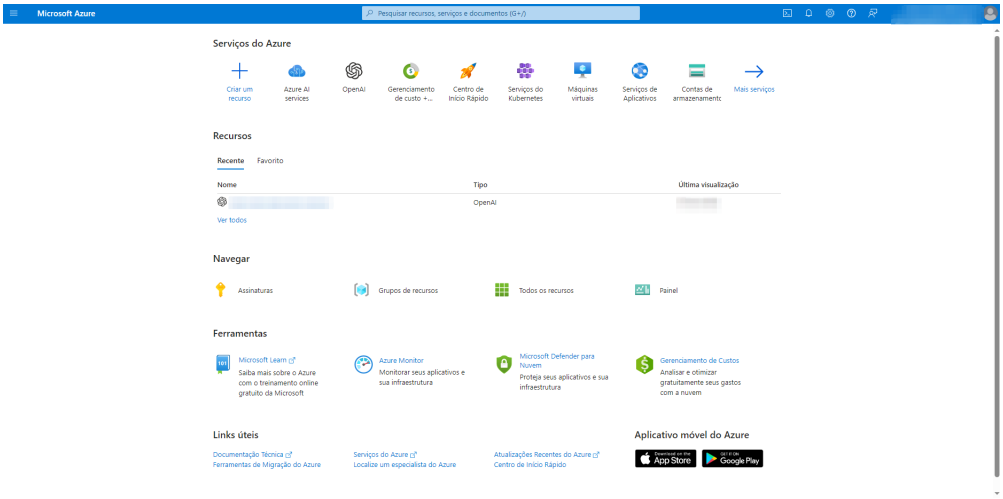
Além das bibliotecas mencionadas, existem frameworks de mercado que se des-

tacam na integração e aplicação de IA em soluções comerciais. Entre eles, LangChain, LlamaIndex e AutoGen são algumas das opções disponíveis.

4.2.5 Azure AI Services

Azure AI Services (MICROSOFT, 2024a) são serviços disponibilizados pela Microsoft por meio da plataforma da Azure que permitem a construção de apps utilizando APIs e modelos de AI customizáveis. Para utilizar os serviços da Azure, é necessário, primeiramente, criar uma conta Microsoft e vinculá-la a Azure. Depois disso, dentro do site (figura 8) é possível acessar os serviços de AI da Azure.

Figura 8 – Site Microsoft Azure.



Fonte: Arquivo do autor (2024).

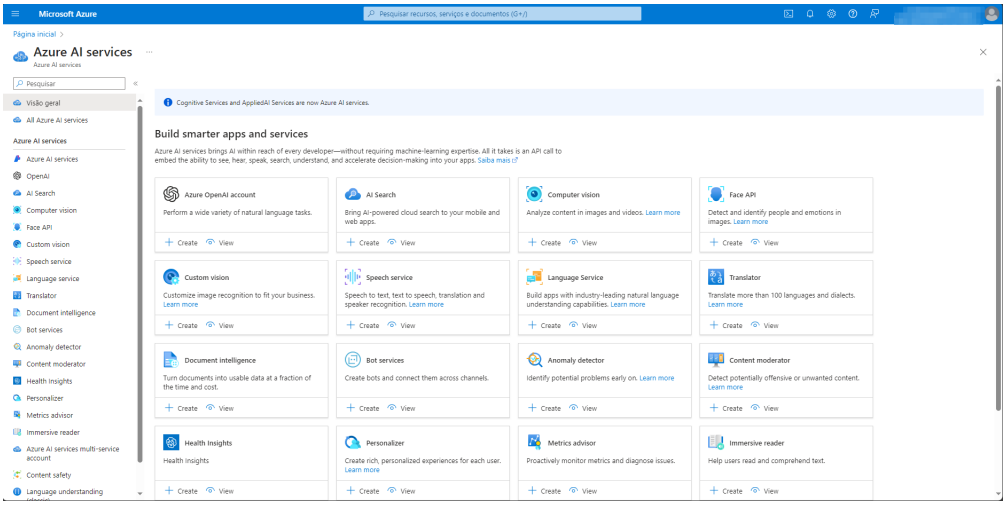
A Azure disponibiliza uma gama de opções de serviços (MICROSOFT, 2024f) que podem ser escolhidos e utilizados. Desde compreensão de texto em linguagem natural até reconhecimento facial, como pode ser visto na figura 9. Cada um dos serviços tem suas peculiaridades e aplicações, podendo geral um grande valor operacional, auxiliando no processo de automatização.

Para ter acesso aos serviços de AI é necessário criar um recurso dentro da página. Nesse estágio, é definido todos os detalhes de custos, e versões da aplicação que deseja utilizar.

4.3 BANCO DE DADOS

Um banco de dados (DB) é um sistema de armazenamento de informações que permite a coleta, o armazenamento, a recuperação e a manipulação de dados de maneira estruturada e eficiente.

Figura 9 – Site Azure AI Services.



Fonte: Arquivo do autor (2024).

Trata-se de uma rede integrada de dados que serve para gerenciar e acessar informações de forma confiável.

O que é essencial para muitas aplicações, desde empresas que gerenciam informações de clientes até aplicações científicas e governamentais que lidam com grandes volumes de dados.

Nos dias atuais, existem vários tipos de banco de dados, incluindo relacionais, NoSQL e outros, cada um com suas próprias características e usos específicos.

Independentemente do tipo, os bancos de dados desempenham um papel fundamental na tomada de decisões e no suporte a operações críticas em organizações e sistemas de informação.

Existem vários tipos de bancos de dados, cada um projetado para atender a diferentes necessidades e cenários de aplicação. Os principais tipos de bancos de dados incluem (ALURA, 2024a):

- **Banco de Dados Relacional (RBDMS)**: Este é o tipo mais comum de banco de dados. Os dados são organizados em tabelas com linhas e colunas, e a relação entre os dados é estabelecida por meio de chaves primárias e estrangeiras. Exemplos populares incluem MySQL, PostgreSQL, Oracle e Microsoft SQL Server.

- **Banco de Dados NoSQL**: Bancos de dados NoSQL são projetados para lidar com dados não estruturados, semiestruturados ou altamente variáveis. Eles podem ser baseados em documentos (como o MongoDB), em colunas (como o Apache Cassandra), em gráficos (como o Neo4j) ou em chave-valor (como o Redis).

- **Banco de Dados em Memória:** Esses bancos de dados armazenam dados na memória principal do sistema para acesso ultra-rápido. Exemplos incluem Redis, Memcached e algumas opções em bancos de dados relacionais que suportam armazenamento em memória.
- **Banco de Dados de Gráficos:** Projetados para armazenar e consultar dados relacionais complexos, esses bancos de dados são eficazes para análises de rede e relações entre entidades. O Neo4j é um exemplo popular de banco de dados de gráficos.
- **Banco de Dados em Nuvem:** Bancos de dados hospedados na nuvem, que podem ser gerenciados por provedores de serviços em nuvem, como Amazon Web Services (AWS), Microsoft Azure ou Google Cloud. Exemplos incluem Amazon RDS, Azure SQL Database e Google Cloud SQL.
- **Banco de Dados de Coluna:** Esses bancos de dados armazenam dados em formato de coluna em vez de linhas, o que os torna eficientes para consultas analíticas. O Apache Cassandra é um exemplo de banco de dados de coluna.
- **Banco de Dados de Tempo Real:** Projetados para processamento em tempo real e análise de fluxos de dados em alta velocidade. Exemplos incluem Apache Kafka e Apache Flink.
- **Banco de Dados de Séries Temporais:** Especializados em armazenar e consultar dados de séries temporais, comumente usados em IoT e monitoramento. O InfluxDB é um exemplo popular.
- **Banco de Dados Distribuído:** Projetados para lidar com grandes volumes de dados distribuídos em vários servidores ou nós. O Hadoop HDFS e o HBase são exemplos de bancos de dados distribuídos.
- **Banco de Dados de Texto Completo:** Projetados para pesquisa e consulta de texto completo em grandes volumes de documentos. O Elasticsearch e o Solr são exemplos populares.
- **Banco de Dados Blockchain:** Usados para armazenar registros imutáveis e descentralizados. O Bitcoin e o Ethereum são exemplos de blockchains com recursos de banco de dados.

A escolha do tipo de banco de dados depende das necessidades específicas de um projeto, incluindo a estrutura dos dados, os requisitos de escalabilidade, o desempenho, a consistência e outros fatores.

Cada tipo de banco de dados tem suas vantagens e limitações, e a seleção adequada é crucial para o sucesso de uma aplicação ou sistema.

4.3.1 MySQL

MySQL (MYSQL, 2024) é um sistema de gerenciamento de banco de dados relacional (RDBMS) de código aberto que goza de grande popularidade. Criado inicialmente pela MySQL AB, o projeto passou para as mãos da Sun Microsystems em 2008, antes de ser adquirido pela Oracle Corporation em 2010. Reconhecido por sua simplicidade operacional, desempenho estável e adoção massiva na comunidade de desenvolvimento, o MySQL continua a ser uma escolha preferencial para a gestão de dados.

4.3.2 Oracle Database

O Oracle Database (ORACLE, 2024) é um RDBMS desenvolvido pela Oracle Corporation. É um dos sistemas de banco de dados mais populares e amplamente utilizados em todo o mundo, especialmente em ambientes empresariais devido à sua confiabilidade, escalabilidade e desempenho.

Figura 10 – Logo Oracle.

Fonte: Oracle (2024).

4.3.3 SQL Server

O SQL Server (MICROSOFT, 2024e) é também um RDBMS, desenvolvido pela Microsoft. Ele é uma das principais opções para armazenamento e recuperação de dados em aplicações empresariais e é amplamente utilizado em todo o mundo.

4.3.4 MongoDB

MongoDB (MONGODB, 2024) é um banco de dados NoSQL, ou seja, não utiliza o modelo relacional tradicional para armazenar dados. Em vez disso, ele usa um modelo de documento flexível, semelhante a JSON, o que o torna ideal para lidar com dados não estruturados ou semi-estruturados.

4.4 ETAPAS PARA DESENVOLVIMENTO DO CHATBOT

No desenvolvimento desse projeto, alguns passos foram tomados. Primeiramente, foi feito o estudo do que seria o ideal do projeto, quais necessidades precisam

ser atingidas, como é feita a resolução do problema. A segunda etapa foi a busca de tecnologias já existentes e conceitos para resolver a questão e discernir qual é a melhor solução. A próxima etapa foi a de implementar a solução na empresa, desenvolvendo o sistema do *chatbot*. Por seguinte, tendo a ferramenta implementada, fazer a validação dos resultados. E, por fim, a avaliação do projeto.

4.4.1 Investigação do problema

A primeira etapa é a mais fundamental para o desenvolvimento correto e eficiente do projeto. A investigação e entendimento do problema é essencial para que haja o direcionamento e desenvolvimento de uma metodologia para o projeto.

Na WEG, o processo de localização de pedidos é feita manualmente. Ao receber um e-mail de algum cliente solicitando a posição atual do pedido ou alguma informação relacionada, é preciso acessar o sistema de gestão empresarial para encontrá-lo. Para tal, verificou-se a necessidade de um programa capaz de acessar o DB da empresa e retornar automaticamente a informação dos pedidos.

Para a interação intuitiva com o usuário, decidiu-se a utilização de um *chabot* como interface. Utilizando de linguagem natural, pode ser criado um sistema de chat capaz de responder toda pergunta envolvendo os pedidos.

4.4.2 Análise soluções

Após a etapa de entendimento do problema, verificou-se as ferramentas disponíveis no mercado para serem utilizadas no projeto. Dentre elas estavam bibliotecas de Python, frameworks para o desenvolvimento de chatbots e programas prontos de AI.

A crescente popularidade dos chatbots tem levado ao surgimento de muitas ferramentas para a construção deles. Essas opções variam desde serviços de processamento de linguagem natural (NLP) em nível básico, que auxiliam na codificação de intenções e frases de treinamento, até plataformas abrangentes de desenvolvimento de baixo código, que cobrem a maioria das etapas no processo de criação de chatbots. Escolher a melhor ferramenta de desenvolvimento de chatbot para uma necessidade específica é difícil. Tomar uma decisão incorreta pode resultar em não conformidade com os requisitos técnicos do chatbot ou com as políticas da empresa de desenvolvimento de software. Alguns sites e blogs informais comparam algumas opções disponíveis para construir chatbots, e pesquisadores identificaram aspectos a serem considerados no design de chatbots, como funcionalidade, integração, análise e garantia de qualidade (PÉREZ-SOLER *et al.*, 2021).

Dentre as opções avaliadas todas as citadas na subseção 4.2, há aquelas que se destacaram, sendo elas o Microsoft Bot Framework, Wit.ai e as bibliotecas de Python LangChain, LlamaIndex e AutoGen. Cada uma delas, apesar de serem semelhantes,

apresentava diferentes ferramentas para o desenvolvimento do Chatbot. Por questões de familiaridade da empresa, a escolha ficou limitada entre as três bibliotecas de python.

Ao final, a opção escolhida para o desenvolvimento foi a utilização da biblioteca LangChain, de Python, que usa a API Azure OpenAI para o desenvolvimento do chatbot e a biblioteca SQL Alchemy para a conexão com a DB.

4.4.3 Implementação da solução proposta

A implementação da solução não foi finalizada devido a restrições de segurança no sistema da empresa, pois, para uma aplicação nesse contexto, é necessária homologação da solução pelo departamento de segurança da informação. Assim, o serviço do bot não pôde ser colocado em um servidor para ser aberto para testes em ambiente real. Dessa forma, o desenvolvimento e testes ficaram limitados a um servidor local, com o código funcionando no computador da empresa.

4.4.4 Validação da solução

Seguindo a linha do tópico anterior, não foi possível realizar testes em um ambiente real de trabalho, limitando-se a testes locais utilizando conexão via LAN para acessar a interface do chatbot e fazer requisições via API.

4.4.5 Avaliação do projeto

A avaliação do projeto se deu pelo atendimento dos requisitos de projeto. Dessa forma, esse trabalho terá a visão de ser uma prova de conceito para a aplicação futura no ambiente de trabalho da WEG.

5 REQUISITOS DO PROJETO

Os requisitos de projeto são descrições detalhadas e específicas das funcionalidades, características, restrições e qualidades que um produto, sistema ou serviço deve possuir para atender às necessidades e expectativas do cliente ou usuário final. Eles servem como a base para o desenvolvimento, implementação e avaliação do projeto, garantindo que todos os envolvidos tenham um entendimento claro do que será entregue e como será alcançado.

Neste capítulo serão apresentados os requisitos funcionais e não funcionais do projeto, além da modelagem do sistema em UML.

5.1 SISTEMA DA EMPRESA

Na WEG, diversos clientes podem pedir a mais variada combinação de produtos dentro do catálogo. E cabe a empresa ser capaz de completar tais pedidos. Toda informação relacionada a pedidos é tratada dentro da ferramenta de ERP da WEG. Assim, é necessário acessar essas informações para o projeto ser capaz de responder perguntas.

5.1.1 Programa para coleta de dados

Dado o escopo inicial do projeto ser tratar do problema na parte da logística da WEG, foi limitado o leque de busca de informações, pegando informações apenas de pedidos que já estão no estoque de exportação (WEX) ou exportados, estando já em návio, caminhão ou avião, indo para o destino final.

5.2 *CHATBOT*

Dentro dos diferentes tipos de bots, aqueles que suportam conversação (também chamado de “assistente habilitado para voz”) e processamento de linguagem (principalmente natural) relacionado a um determinado assunto são considerados chatbots (RABELO; ZAMBIASI; ROMERO, 2023).

O chatbot desenvolvido nesse projeto é uma tecnologia que visa melhorar a experiência do usuário ao permitir interações mais eficientes e intuitivas com sistemas complexos. Em sua essência, ele atua como um intermediário inteligente entre o usuário e as informações que ele busca, transformando processos tradicionalmente manuais em interações automatizadas e instantâneas.

No âmbito deste projeto específico, o objetivo central é proporcionar aos usuários uma maneira rápida e conveniente de acessar dados sobre pedidos ou embarques. Isso envolve não apenas responder a perguntas diretas, como o status de um pedido

ou informações de rastreamento, mas também entender e processar consultas mais complexas que possam surgir.

5.3 ESTRUTURAÇÃO DO *CHATBOT*

5.3.1 Criação do Bot

A etapa de criação e configuração do bot é de suma importância, pois é aqui que uma Large Language Model (LLM) "crua" recebe um "propósito" e tem seu comportamento definido para responder de maneira específica. Neste projeto, o propósito atribuído à LLM é interpretar a mensagem do usuário, identificar quais informações estão sendo buscadas e determinar a qual pedido ou embarque essas informações se referem.

Durante essa fase, a LLM é configurada para entender e responder adequadamente às interações dos usuários. Essa configuração envolve diversos passos cruciais:

- **Configuração:** A configuração de uma instância do bot é de extrema importancia. Nesse projeto é configurada uma instância da API da Azure OpenAI, trazendo um modelo de LLM do ChatGPT 3.5.
- **Definição do Propósito:** É essencial definir claramente o propósito da LLM. Para este projeto, o objetivo é que a LLM atue como um assistente inteligente, capaz de interpretar as perguntas dos usuários sobre pedidos ou embarques, identificar os dados relevantes (como números de pedidos) e fornecer respostas precisas e contextualizadas.
- **Treinamento e Customização:** A LLM é treinada e personalizada com dados específicos do domínio em questão. Isso inclui o uso de exemplos ("Few-shot examples") que demonstram como a LLM deve responder a diferentes tipos de perguntas. Esses exemplos ajudam a modelar o comportamento da LLM, ensinando-a a reconhecer padrões nas solicitações dos usuários e a gerar respostas apropriadas.

Através desse processo de criação e configuração, a LLM "crua" se transforma em uma ferramenta poderosa e orientada a propósito, capaz de interpretar as mensagens dos usuários e fornecer respostas informativas e contextualmente relevantes.

5.3.1.1 Configuração

Na parte de configuração, primeiramente, é preciso importar as bibliotecas, funções e classes necessárias do LangChain. Após isso, faz-se a configuração de uma instância da LLM que será utilizada no projeto por meio da API do Azure OpenAI, armazenando-a em uma variável, como mostrado na figura 11.

Figura 11 – Configuração do modelo da Azure OpenAI.

Fonte: Arquivo do autor (2024).

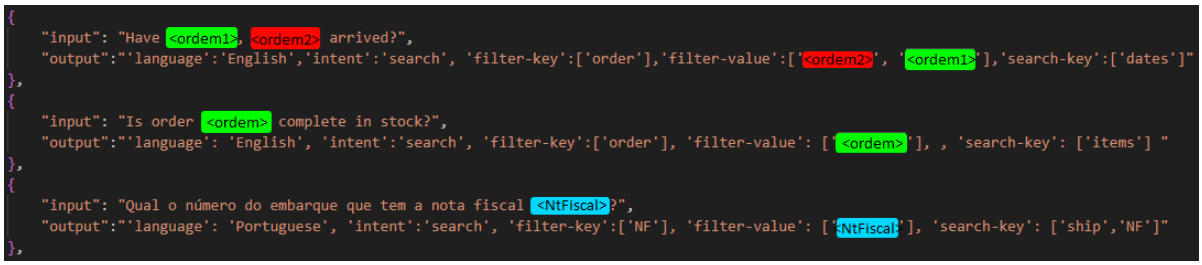
Dada a configuração do LLM, pode-se seguir com a configuração do bot propriamente dito. Para isso, faz-se necessário a definição de um template para o bot. No template, são passados para o bot instruções em texto de como ele deve se comportar, tipos de informação que deve ou não deve informar. Outra coisa que pode ser passada para o bot por meio do prompt são exemplos.

5.3.1.2 Exemplos

Uma das estratégias possíveis para criar um prompt usando LangChain é utilizando “Few-shot examples” (LANGCHAIN, 2024a). Essa estratégia consiste em fornecer exemplos dentro do prompt para melhorar a performance do modelo. Esses exemplos servem como base para o bot gerar suas próprias respostas.

Os exemplos devem incluir uma possível mensagem do usuário e a resposta que o bot deveria dar para essa mensagem. No contexto deste trabalho, os exemplos para o primeiro bot devem apresentar uma mensagem potencial do usuário e uma seleção destacando o número do pedido ou embarque, a informação que o usuário deseja, e a língua em que a pergunta está sendo feita.

Figura 12 – Exemplos de treino do modelo.



Fonte: Arquivo do autor (2024).

A implementação dessa estratégia começa com a construção de um conjunto

de exemplos representativos. Cada exemplo deve ilustrar claramente a estrutura da interação esperada entre o usuário e o bot. Por exemplo, se um usuário enviar uma mensagem solicitando o status de um pedido, o exemplo no prompt deve mostrar como identificar o número do pedido e a informação específica solicitada. Isso pode ser feito da maneira apresentada na figura 12. Por volta de 100 exemplos foram passados para os dois bots desenvolvidos nesse projeto. Esse número de exemplos foi sufuciente para que ambos os bots tivessem um comportamento satisfatório, mas planeja-se aumentar ainda mais o número de exemplos a fim de tornar o chatbot ainda mais robusto. Na imagem da figura 12, os termos “<ordem>” e “<NtFiscal>”, representam um número de ordem e nota fiscal, respectivamente.

5.3.1.3 Respostas

Da mesma forma que o primeiro bot, o segundo bot, responsável por interpretar os resultados da *query*, utiliza a técnica de “Few-shot examples” para gerar respostas precisas e contextualmente apropriadas para o usuário. Este segundo bot recebe a pergunta do usuário, o resultado da *query* na forma de um CSV, e o idioma em que a pergunta foi feita, e com base nessas informações, formula uma resposta adequada.

A utilização de “Few-shot examples” envolve fornecer ao bot exemplos de interações entre usuário e bot, que incluem tanto a pergunta do usuário quanto a resposta correta que o bot deve fornecer. Esses exemplos ajudam o bot a entender como transformar os dados brutos obtidos na *query* em respostas compreensíveis e úteis para o usuário.

A inclusão de tais exemplos no prompt permite que o segundo bot aprenda a mapear o resultado da *query* diretamente para uma resposta formulada de maneira clara e direta, levando em consideração o contexto da pergunta original do usuário. A técnica de “Few-shot examples” pode ajudar o bot a lidar com possíveis variações nas perguntas dos usuários. Diferentes usuários podem formular a mesma pergunta de várias maneiras, e os exemplos ajudam o bot a reconhecer essas variações e fornecer respostas precisas independentemente da forma como a pergunta foi feita.

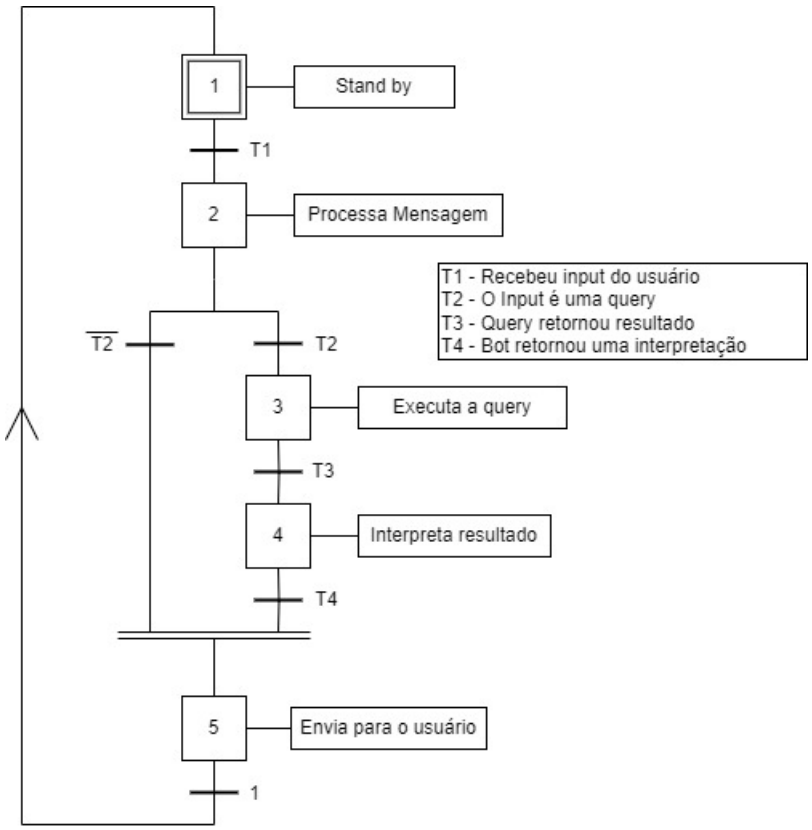
Utilizando dessa técnica, é aprimorada a capacidade do bot de interpretar os resultados da *query* e gerar respostas que não só respondem às perguntas dos usuários, mas também o fazem de maneira que seja fácil de entender e relevante para o contexto específico de cada interação.

5.3.2 Comportamento Esperado

O comportamento esperado do chatbot pode ser representado utilizando o Grafcet demonstrado na figura 13. O Grafcet foi escolhido para representar o sistema por ser possível, por meio dele, representar um sistema complexo de forma simplificada e visual por meio de estados. Dessa forma, é possível visualizar a forma como o com-

portamento do sistema ocorre de forma sequencial, podendo haver diferentes ações dependendo do eventos ocorridos.

Figura 13 – Grafcet do comportamento esperado do Chatbot.



Fonte: Arquivo do autor (2024).

O estado 1 é o estado inicial do sistema. Nesse estado o sistema está em *stand by* esperando pelo input de uma pergunta de um usuário. Quando há o input, o sistema passa para o estado 2, nesse estado, o bot utilizando da API da Azure OpenAI interpreta a entrada do usuário e decide se a resposta dele será uma *query* para buscar informações no DB ou será uma mensagem normal para responder ao input. Desse útimo estado, o sistema se divide, se o que o bot retornar não for uma *query* o sistema pula direto para o estado 5. Se for uma *query*, ele irá para o estado 3, nesse ponto o sistema executa uma *query* no banco de dados buscando pela informação solicitada pelo usuário. Ao final dessa busca, ao retornar um dataframe com as informações, o sistema passa para o estado 4, onde é feita a interpretação aos valores na tabela gerada, retornando, então, uma mensagem sumarizando os dados alcançados. Desse ponto, terminada a interpretação, passa-se para o estado 5, nesse estado é retornado para o usuário a mensagem gerada pelo bot, tanto a vinda do estado 2, quanto a do 4. Depois dessa etapa, o sistema volta para o estado inicial, esperando um novo input do usuário.

5.4 REQUISITOS

Os requisitos de projeto são estabelecidos após uma análise detalhada da proposta inicial. Este processo é crucial para garantir que todas as expectativas e objetivos sejam claramente definidos e compreendidos antes do início efetivo do desenvolvimento.

5.4.1 Requisitos funcionais

Os requisitos funcionais descrevem os requisitos mínimos do projeto, aqueles que devem ser atendidos para o funcionamento do projeto de forma efetiva. Os requisitos levantados, por meio das reuniões iniciais do projeto junto do supervisor e colegas de trabalho do autor, foram:

1. O *chatbot* deve ter um comportamento responsivo, retornando a partir de inputs do usuário.
2. O *chatbot* deve ser capaz de interagir com o banco de dados da empresa.
3. O *chatbot* deve retornar informações precisas do banco de dados.
4. O *chatbot* deve responder o usuário de forma concisa e clara.
5. O *chatbot* deve interagir por meio de linguagem natural.
6. O *chatbot* deve ser robusto às perguntas de usuários.
7. O *chatbot* não deve retornar informações confidenciais da empresa.
8. O *chatbot* deve ter uma interface de chat.
9. O *chatbot* deve ter memória do histórico de mensagens.
10. Transformar o código do *chatbot* em uma API para ser incorporado em um sistema da empresa.
11. Utilizar requisições REST para comunicação com a API do Chatbot.

5.4.2 Requisitos não funcionais

Os requisitos não funcionais do sistema são:

- Gerar arquivos em formato PDF e retornar para o usuário com os dados solicitados, se houver um número elevado de pedidos.
- É esperado que o *chatbot* funcione 24h por dia, 7 dias por semana.

- É esperado que o *chatbot* acelere o processo de atendimento a clientes.
- É esperado que o *chatbot* acelere o processo de busca de informações a pedido e/ou embarques.
- É esperado que o *chatbot* seja intuitivo e fácil de usar.
- É esperado que o *chatbot* melhore a experiência do usuário.
- É esperado que o *chatbot* automatize o processo de busca de informações de pedidos e/ou embarques.
- É esperado que o *chatbot* reduzir a carga de trabalho de colaboradores em relação a busca de informações de pedidos e/ou embarques.
- É esperado que o *chatbot* ajude a otimizar o fluxo de trabalho.
- É esperado que o *chatbot* acelerar processo de disseminação de informações.

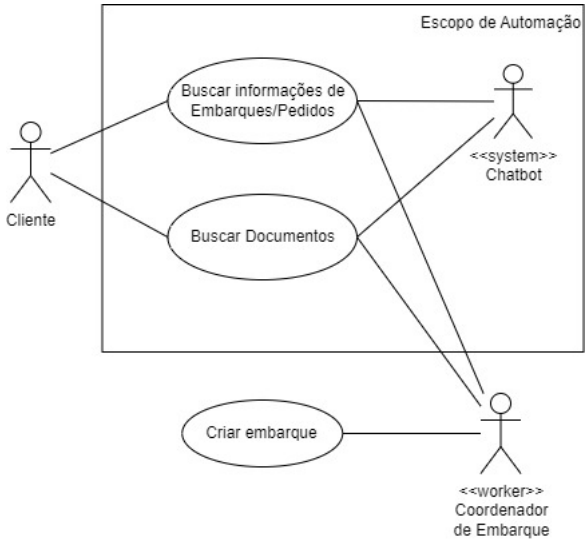
5.5 MODELAGEM EM UML

Uma etapa crucial em qualquer projeto de desenvolvimento de software é a modelagem do sistema. Este processo permite que as ideias e os requisitos sejam transformados em uma estrutura compreensível e organizada, servindo como base para todo o ciclo de desenvolvimento. Entre as diversas estratégias disponíveis para a modelagem de sistemas, a utilização da UML (Unified Modeling Language) destaca-se por sua eficácia e abrangência. A UML é uma linguagem de modelagem padronizada que permite a criação de uma variedade de diagramas, cada um focado em aspectos específicos do sistema. Esses diagramas incluem, mas não se limitam a, diagramas de casos de uso, diagramas de classes, diagramas de sequência, diagramas de atividades e diagramas de estados. Cada um desses elementos contribui para uma visão detalhada e integrada do sistema, desde a arquitetura de alto nível até as interações mais específicas entre os componentes. Além disso, a UML facilita a documentação do sistema de maneira estruturada e compreensível. Isso não só ajuda durante o desenvolvimento inicial, mas também é valioso para a manutenção e evolução do software ao longo do tempo.

5.5.1 Diagrama de Casos de Uso

O diagrama de casos de uso é um diagrama comportamental da UML que captura os requisitos funcionais do sistema. Ele descreve como os usuários (atores) interagem com o sistema para realizar um conjunto de atividades ou objetivos específicos (casos de uso). Para o projeto do Chatbot, os casos de uso e atores foram levantados como apresentado na figura 14

Figura 14 – Diagrama de Casos de Uso.



Fonte: Arquivo do autor (2024).

Três agentes foram nomeados, sendo eles: o Cliente, que faz a requisição de informações, o coordenador de embarque, que possui a gunção de criar embarques, buscar informações de pedidos ou embarques e buscar informações de documentação, e por último o Chatbot, que possui a função de buscar informações de pedidos ou embarques e de documentação. No caso de uso de “Buscar informações de pedidos ou embarques” está incorporado nele a busca por pedidos na WEX, por busca de pedidos exportados e busca de pedidos na fábrica.

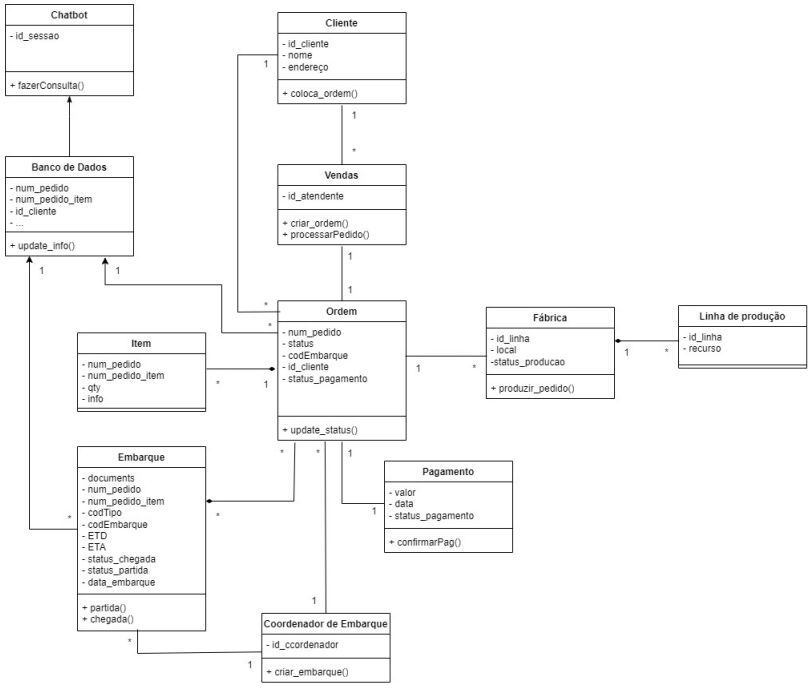
5.5.2 Diagrama de Classe

O diagrama de classes é um dos diagramas estruturais da UML que descreve a estrutura de um sistema mostrando suas classes, atributos, métodos e os relacionamentos entre os objetos. Este diagrama é fundamental para o design orientado a objetos, pois permite definir a arquitetura estática do sistema. Nesse projeto o Diagrama de Classe ficou definido como mostrado na figura 15.

5.5.3 Diagrama de Deployment

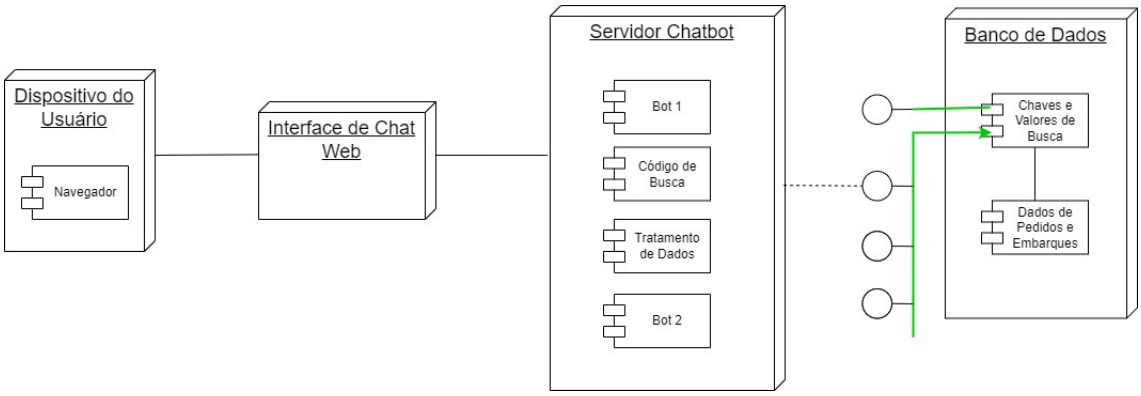
O diagrama de deployment (ou diagrama de implantação) é um diagrama estrutural da UML que mostra a disposição física dos artefatos de software em nós de hardware. Ele descreve a topologia do sistema em termos de hardware e software. Nesse projeto, a estrutura no software esperada durante a etapa inicial de planejamento é como apresentado na figura 16.

Figura 15 – Diagrama de Classe.



Fonte: Arquivo do autor (2024).

Figura 16 – Diagrama de Deployment.



Fonte: Arquivo do autor (2024).

5.6 CASOS DE USO

Todo o projeto se comportará de forma semelhante, como o banco de dados é apenas um, os diferentes casos de uso são relacionados a diferentes estados do pedido dentro no caminho do pedido (Fig. 2) até o cliente.

5.6.1 Solicitar informações de pedidos ou embarques

O funcionamento básico do bot é fundamentado na interação com o usuário, que fornece as informações necessárias para que o sistema realize as operações desejadas. Esse mecanismo de entrada é crucial, pois determina a resposta e as ações subsequentes do bot, garantindo que as solicitações dos usuários sejam atendidas de maneira eficaz e precisa.

Quando um usuário interage com o bot, ele pode fazer uma série de consultas ou solicitações. No contexto da WEG, essas entradas podem variar amplamente, desde a busca por informações específicas sobre um pedido até o rastreamento de embarques dentro das instalações da empresa ou até mesmo fora delas. Entretanto, a principal mudança são a localização e diferentes status atuais do pedido, podendo estar em produção, no estoque da WEX, ou já exportado. Assim os casos de uso seguem como:

5.6.1.1 Solicitar informações de Pedido já Exportado

Pedidos já exportados não estão diretamente registrados no sistema da empresa, eles estão disponíveis através da plataforma GT Nexus da Infor Nexus, utilizada pelos armadores para fazer o rastreio de seus navios e containers embarcados. As informações de rastreio são baseadas em eventos, isto é, sempre que um navio chega a um ponto de interesse, ele registra como um evento. Um exemplo seria, navio zarpando ou chegando ao porto de destino ou o descarregamento do container no porto. As informações são puxadas desse sistema de tracking e colocados no DB que é utilizado nesse trabalho.

5.6.1.2 Solicitar informações de Pedido no Estoque de Exportação (WEX)

No caso de um pedido no estoque de exportação da empresa, há várias informações que podem ser adquiridas no DB. Entretanto, no DB há informações de vários dias de estoque, então há a necessidade de filtrar os valores para o dia mais recente de registro.

5.6.1.3 Solicitar informações de Pedido em Produção

Na caso de uso de pedidos em produção, o sistema do chatbot não está mapeando esse tipo de pedidos. Nesse caso, o chatbot deve retornar que o usuário deve entrar em contato com o responsável de vendas que fez o pedido para a fábrica para saber informações do pedido.

5.6.1.4 Falha: Solicitar informações de Pedido Inexistente

No caso de haver uma busca por um número de pedido inexistente no banco, o tratamento será igual ao pedido em produção, pedindo para o usuário entrar em

contato com o responsável de vendas ou verificar se o número de pedido está correto.

5.6.1.5 Emissão de PDFs quando houver um elevado número de pedidos

Todo embarque é composto por pedidos, e todo pedido é composto por itens. No caso, algumas buscas podem gerar um número elevado de pedidos, o que pode ser prejudicial para o chatbot, dessa forma, é possível minimizar esse problema colocando as informações em um PDF e retornando para o usuário.

6 IMPLEMENTAÇÃO

Este capítulo trata da unificação dos sistemas abordados anteriormente, seguindo os requisitos funcionais explicitados na subsessão 5.4.1, detalhando o processo de implementação desses sistemas no ambiente corporativo da empresa. Inicialmente, revisaremos as principais funcionalidades e características de cada sistema individualmente, destacando suas contribuições e benefícios específicos. Em seguida, descreveremos o processo de integração, abordando os desafios técnicos e as soluções adotadas para garantir o funcionamento da operação.

6.1 FERRAMENTAS UTILIZADAS

Nesta parte serão apresentadas as ferramentas utilizadas no desenvolvimento do projeto juntamente com como cada elemento foi aplicado para a concepção do projeto.

6.1.1 LangChain

LangChain é um framework que funciona com Python para a criação de chat*bot*s, assistentes e agentes virtuais. LangChain disponibiliza ao usuário um framework para o desenvolvimento de *bot*s utilizando Large Language Models (LLM). Com essa ferramenta, pode-se treinar *bot*s para terem diversas funções, apenas necessitando declarar em prompt como ele deve atuar.

Apesar de ter uma funcionalidade simples e fácil de usar, o framework não possui uma LLM própria, necessitando a utilização de uma externa. Para isso, foi utilizada uma API da Azure com um modelo de LLM da OpenAI.

6.1.1.1 API Azure OpenAI

A API da Azure OpenAI é um serviço disponibilizado pela Microsoft, ligado aos serviços de AI do Azure, para a utilização da LLM do ChatGPT. Esse serviço tem acesso limitado a empresas, sendo necessário fazer a solicitação para a Microsoft para a utilização.

Dentro da configuração da API, pode-se escolher qual modelo de LLM é desejado utilizar e qual versão também. Como cada modelo possui um custo de utilização atrelado a quantidade de tokens utilizados. Os custos são calculados utilizando a quantidade de tokens de entrada na API, limitados a 16 mil tokens, e tokens de saída da resposta da LLM.

Dois exemplos de custos são apresentados abaixo:

6.1.1.1.1 GPT-3.5-Turbo-0125-16K

Esse modelo é a versão mais antiga do ChatGPT disponibilizada gratuitamente no site da OpenAI, quando utilizando o modelo unicamente pelo site oficial do ChatGPT. Quando usada a API pela Azure, o custo em dólares a cada mil tokens de entrada e saída são:

- Entrada: $ 0,0005
- Saída: $ 0,0015

6.1.1.1.2 GPT-4-8K

Modelo de LLM mais atual, até o momento de escrita dessa dissertação, da OpenAI. Possui um limite máximo de 8 mil tokens de entrada. O custo de utilização da API em dólares por cada mil tokens é:

- Entrada: $ 0,03
- Saída: $ 0,06

Como pode ser visto nos exemplos, escolher um modelo na utilização da API é de suma importância. Diferentes modelos possuem características singulares entre si. Ainda tomando o exemplo acima, apesar de ser ideal utilizar a LLM mais atual, devido as suas capacidades, ela se torna inviável financeiramente.

Como o objetivo desse projeto é desenvolver um chat*bot* que seja utilizado um elevado número de vezes, é preciso que o projeto seja financeiramente viável.

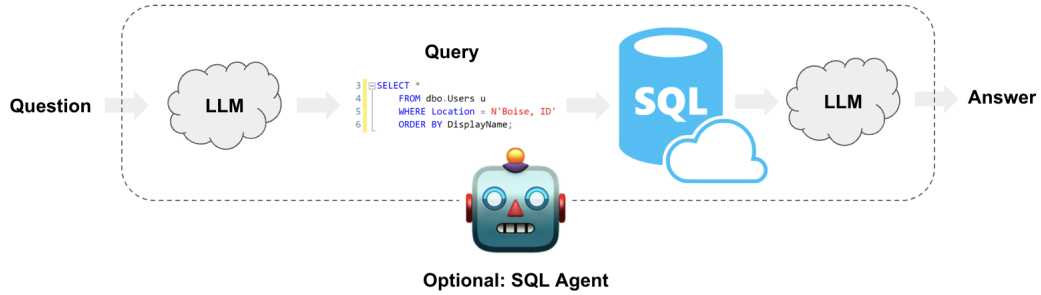
6.1.1.2 Agentes

Por padrão, LLMs não podem realizar ações, apenas retornam texto. Entretanto, utilizando LangChain é possível desenvolver agentes. Agentes são sistemas que usam uma LLM como um mecanismo para determinar quais ações devem ser tomadas e quais devem ser as entradas para essas ações. Os resultados dessas ações podem então ser realimentados no agente e determinar se mais ações são necessárias ou se não há problema em concluir.

No caso desse projeto, um agente poderia ser usado para ser realizada a busca pelas informações no DB da empresa. O processo do agente é dado como segue:

1. Há o input do usuário;
2. A LLM interpreta a pergunta;
3. O agente constrói uma *query* com base na interpretação da LLM;

Figura 17 – Demonstração visual do agente.



Fonte: (LANGCHAIN, 2024b).

4. O agente executa a *query* e recebe um retorno;
5. A LLM interpreta o resultado da *query* e gera uma resposta para o usuário.

Nessa abordagem, o trabalho de treinar o *bot* e de fazer as buscas são menores, apenas sendo necessário disponibilizar para o *bot* exemplos de *queries* (Fig. 18), nome de tabela e colunas para que o *bot* seja capaz de gerar corretamente respostas.

Figura 18 – Exemplo genérico para treino do bot.

```
{
    "input":"Qual é a localização do pedido <pedido>?",
    "query":"""SELECT tl.NUM_PEDIDO, tp.LOCAL, tp.DATA FROM TABELA_PEDIDO tp
                LEFT JOIN TABELA_LOCAL tl ON tl.EMBARQUE = tp.EMBARQUE
                WHERE tl.NUM_PEDIDO = '<pedido>'"""
}
```

Fonte: Arquivo do Autor.

6.1.2 SQLAlchemy

SQLAlchemy (SQLALCHEMY, 2024) é uma biblioteca SQL Toolkit e ORM (Object-Relational Mapping) para Python que fornece uma interface flexível e poderosa para interagir com bancos de dados relacionais. Ele facilita a execução de operações de banco de dados, desde consultas SQL simples até transações complexas e mapeamento objeto-relacional.

6.2 FUNCIONALIDADE

Este projeto de desenvolvimento de um chat*bot* visa criar uma ferramenta robusta e eficiente para interação automatizada com usuários, oferecendo respostas

rápidas e precisas a partir de uma base de dados específica. Este chat*bot* pode ser dividido em quatro partes principais, cada uma desempenhando um papel essencial na funcionalidade geral do sistema. As partes são: a interface de chat, o *bot* que recebe a mensagem do usuário, o programa de coleta de dados do banco de dados e um segundo *bot* responsável por interpretar o resultado da *query* para o usuário. As últimas três partes fazem parte da API que compõem o Chat*bot*. A estrutura de funcionamento do Chat*bot* pode ser vista no Grafcet da figura 13 apresentado anteriormente neste documento.

6.2.1 Comportamento

O comportamento do chat*bot*, mapeado pelos requisitos funcionais do projeto, ocorre da seguinte forma. Um usuário, acessa a interface de chat (requisito 8), e pode enviar uma mensagem para o *bot* (requisito 5). A mensagem é assim convertida em uma requisição REST e é enviada para a API do *Bot* armazenada em um servidor (requisitos 10 e 11). O *bot* é responsivo (requisito 1), e sua ativação ocorre quando o usuário envia uma mensagem. A primeira etapa, depois da ativação, consiste em analisar a pergunta do usuário e determinar qual busca deve ser feita (requisito 3). Diferentes mensagens podem gerar diferentes respostas, mas há aquelas que, embora diferentes em texto e função sintática, buscam a mesma informação, um exemplo desse cenário seria com as mensagens “Onde está o embarque?” e “Gostaria de saber a localização do embarque”, nesse caso a robustez do *Bot* deve determinar que as duas frases tem a mesma conotação (requisito 6). Nesse sentido, o chat*bot* examina a pergunta e o contexto, por meio de memória das mensagens anteriores (requisito 9), em que está inserida. Após essa análise inicial, o *bot* identifica um exemplo relacionado à pergunta, levando em consideração o contexto atual e conotação da mensagem do usuário. Os exemplos passados para o *bot* garantem a robustez e qualidade das respostas respectivas.

As perguntas do usuário podem conter termos chave (localização, status, etc.), ou referências a eles, que determinam se um script de busca deve ser executado e, em caso afirmativo, os termos chave também direcionam quais informações o *bot* deve buscar no DB. Se o diálogo não contiver esses termos chave, o chat*bot* simplesmente responde ao usuário e a dinâmica encerra, retornando ao estado inicial. No entanto, se for necessário executar uma busca, o *bot* consultará o banco de dados do Logistics Integration System (LIS) (requisito 2).

Existem cenários onde podem ocorrer falhas durante a busca de informações, tais como: inexistência de informações sobre o pedido/embarque, ou a solicitação de uma busca sem que o número da ordem ou embarque seja fornecido. Grande parte desses erros está sendo mapeada e retornada ao usuário na forma de mensagens no chat, informando-o sobre a falha ocorrida.

Após a busca das informações, os dados são tratados para serem passados ao segundo *bot*, responsável por gerar a resposta para o usuário. Esse segundo *bot* recebe as informações provenientes da busca e as resume, considerando o contexto da pergunta original do usuário, para gerar uma resposta adequada e concisa (requisitos 3 e 4). Finalmente, essa resposta é retornada ao usuário em forma de texto, completando o ciclo de interação.

6.2.2 Desenvolvimento

O desenvolvimento do projeto se deu por etapas, desenvolvidas de forma cíclica. Etapas essas que não necessariamente foram desenvolvidas em sequência, em vários momentos houve a sobreposição de tarefas pois uma nova ferramenta foi disponibilizada ou um processo foi reiterado no fluxo de trabalho. As etapas do projeto foram divididas principalmente pela disponibilidade das ferramentas utilizadas nesse trabalho.

A primeira etapa foi a familiarização com as ferramentas de SQL e Banco de Dados da empresa. Inicialmente, um programa simples de busca utilizando SQLAlchemy, sem a utilização do chat*bot* foi desenvolvido a fim de haver a compreensão de como o programa deveria funcionar em um caso ideal e como deveria ser construída a *query* responsável pela coleta de informações.

A próxima etapa segue com o desenvolvimento de uma primeira versão de chat*bot* para verificar o funcionamento do programa utilizado e como é feita a configuração dos parâmetros. Nessa parte, foram testadas diferentes formas de configurar o *Bot*, tendo assim as versões apresentadas na sequência.

Por fim, entendendo as ferramentas a serem utilizadas. Foi começado o desenvolvimento do chat*bot*.

6.2.2.1 Versões desenvolvidas do chat*bot*

No desenvolvimento do projeto três versões do *bot* foram criadas:

6.2.2.1.1 Utilizando Agente

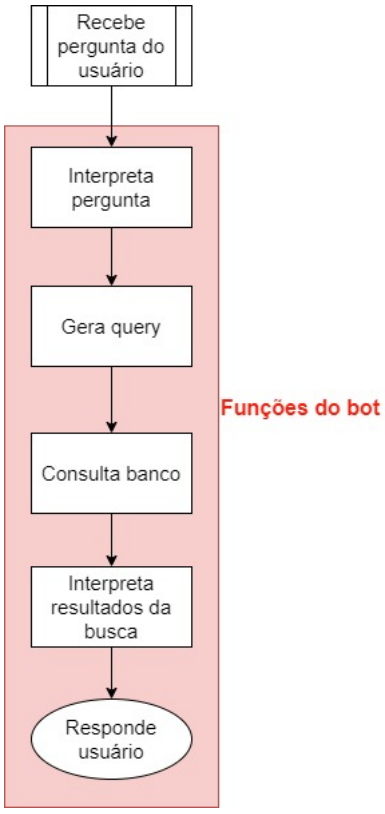
Em um primeiro cenário houve a utilização de agentes do LangChain. Com essa ferramenta do frameworks foi possível construir um *bot* capaz de receber a entrada do usuário, interpretar a mensagem, gerar uma consulta, executar a consulta, interpretar os resultados e retornar a resposta ao usuário. A principal vantagem desse método é que o *bot* tem total controle sobre as execuções das buscas, funcionando eficientemente sem a necessidade de uma grande quantidade de informações prévias.

Contudo, essa abordagem apresenta uma desvantagem significativa. A precificação do uso da API é baseada em tokens, e como o *bot* controla completamente

os processos de busca, o custo pode se tornar elevado. O funcionamento do agente envolve várias etapas: receber a mensagem do usuário (tokens de entrada), gerar uma consulta com uma *query* (tokens de saída), receber o resultado da consulta (tokens de entrada) e interpretar a resposta para gerar uma mensagem ao usuário (tokens de saída), podendo ainda o *bot* executar uma nova *query* para responder completamente a pergunta. Cada uma dessas etapas consome tokens, e o acúmulo desses custos pode aumentar de forma imperceptível.

Além disso, a utilização contínua e intensiva do *bot* pode levar a uma escalada nos custos operacionais, tornando esse sistema economicamente inviável a longo prazo. Portanto, apesar das vantagens em termos de controle e eficiência, essa combinação de fatores tornou essa abordagem menos prática para resolver o problema de forma sustentável. Foi necessário, então, considerar alternativas que equilibrem a eficiência do *bot* com um modelo de custo mais gerenciável, garantindo a viabilidade do sistema sem comprometer seu desempenho.

Figura 19 – Representação das funções do Bot (Versão 1).



Fonte: Arquivo do Autor.

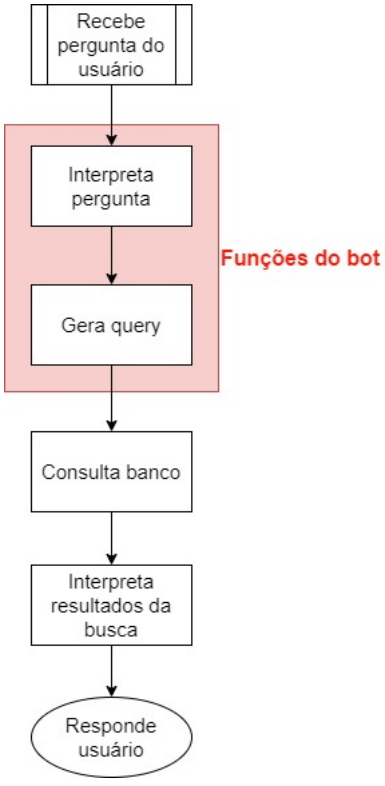
6.2.2.1.2 Bot construindo só Query

A fim de diminuir o custo do *bot*, a segunda versão foi desenvolvida. Nessa versão, o *bot* foi planejado para apenas realizar a contrução da *query*. Diferentemente do agente, essa versão apenas utiliza a LLM para contruir a *query* que fará a busca no banco de dados e a busca em si seria feito por um código a parte. Nesse sistema o programa utilizaria da LLM para interpretar a pergunta e gerar a *query*, sendo assim, cortando os custos de analisar a resposta da *query* e devolvê-la para o usuário.

Entretanto, utilizando esse método, o custo ainda era elevado. Como a *query* desenvolvida busca por informações de diferentes tabela, que necessitam diferentes tipos de tratamentos, seu tamanho acabou sendo considerável. Dessa forma, os exemplos acabam por serem grandes também.

Outra desvantagem dessa abordagem é que o texto de saída para o usuário é fixo, nesse sentido, não somente a formatação do texto, como também a língua acabam por ser parâmetros fixos, o que pode ser um inconveniente para a implantação do projeto.

Figura 20 – Representação das funções do Bot (Versão 2).



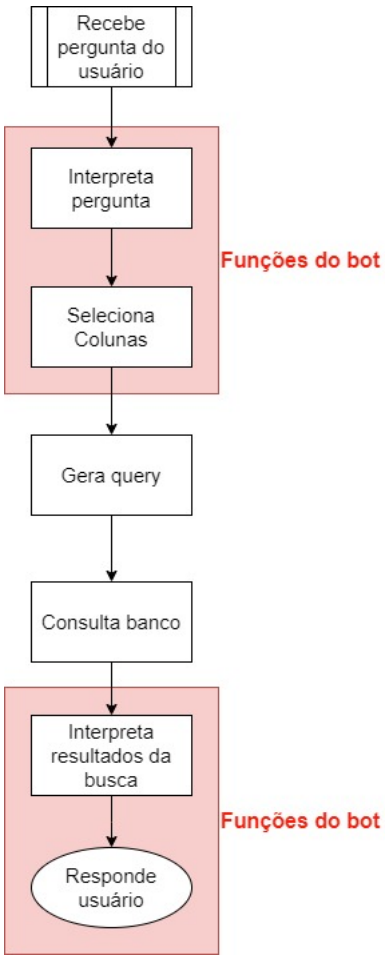
Fonte: Arquivo do Autor.

6.2.2.1.3 Bot só selecionando colunas e interpretando mensagem

Esta versão é a mais recente até o momento de escrita desse trabalho. No desenvolvimento dessa versão notou-se que a *query* criada pelo *Bot* na versão anterior possuía sua estrutura praticamente idêntica para todas as chamadas. Nesse contexto, pensou-se em modificar o *bot* mais uma vez. Dessa forma, foi desenvolvido o *bot* que apenas seleciona as colunas de cada tabela de forma a buscar as informações requiridas pelo usuário. Essa abordagem reduziu em muito os custos de utilização da API para a construção da *query*. Entretanto, como desejáva-se colocar o artifício de o *bot* poder ter conversações em diversas línguas, utilizou-se de uma segunda instância do *bot* para interpretar os resultados da *query* e retornar para o usuário.

Essa abordagem permitiu uma maior possibilidade de difusão e universalidade de utilização dos usuários com o projeto. Assim, apesar de o não haver significativa diferença de custo entre a segunda versão para a terceira, as respostas finais utilizando linguagem natural tiveram uma considerável melhora qualitativa.

Figura 21 – Representação das funções do Bot (Versão 3).



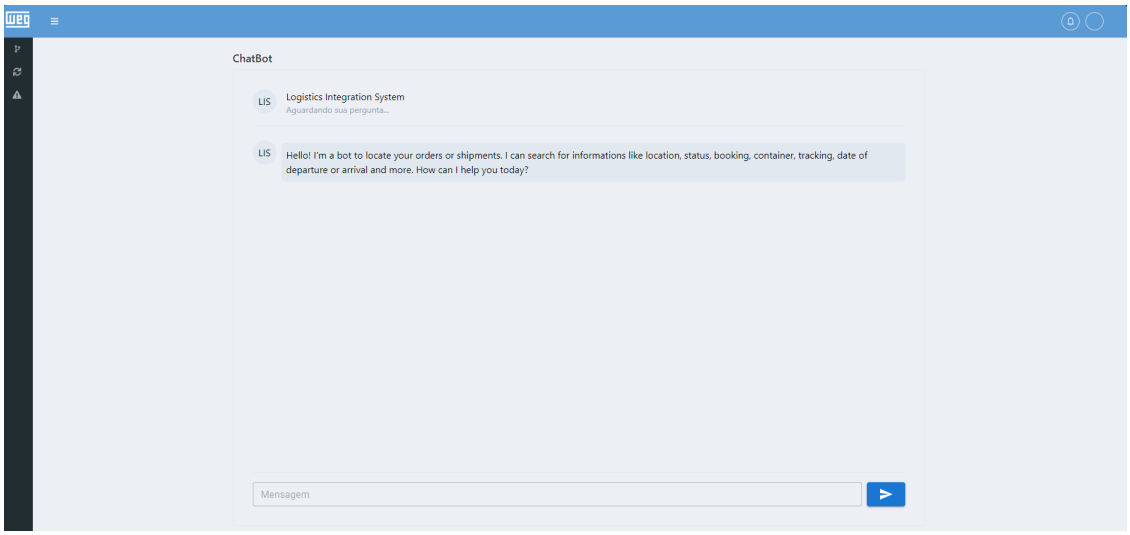
Fonte: Arquivo do Autor.

Vale notar que nessa versão do *bot* a única informação sensível passada para o *bot* são os nomes de colunas passados nos exemplos.

6.2.3 Interface de Chat

A interface de chat é o ponto de interação entre o usuário e o chat*bot*. Trata-se de uma plataforma onde o usuário insere suas perguntas e recebe as respostas geradas pelo *bot*. Nesse projeto foi tentado ser incorporado como uma parte do sistema da LIS. A interface foi projetada para ter uma aparência “limpa”, intuitiva e amigável, proporcionando uma experiência de usuário fluida e sem fricções. Sua função é servir de interface e intermédio para a comunicação entre o usuário e o sistema do chat*bot*. Uma ilustração do Chat pode ser visto na figura 22.

Figura 22 – Interface de Chat do projeto.



Fonte: Arquivo do Autor.

O design da interface foi proposto pelo autor do projeto, que delineou os aspectos visuais e funcionais que a interface deveria ter para proporcionar uma experiência de usuário eficiente e intuitiva. No entanto, o desenvolvimento técnico da interface foi realizado por um colega do mesmo setor. Essa decisão foi tomada porque o programa seria implementado no sistema da LIS, e esse colega era o responsável pela manutenção e desenvolvimento do sistema. Portanto, ficou sob a responsabilidade dele a tarefa de desenvolver a interface de chat. O frontend da interface foi desenvolvido utilizando ReactJS, já no backend foi utilizado Node.JS. ReactJS é uma biblioteca JavaScript desenvolvida pelo Facebook para a construção de interfaces de usuário. É especialmente útil para criar aplicações web dinâmicas. Node.js é um ambiente de execução JavaScript que permite aos desenvolvedores executar código JavaScript no lado do servidor.

As mensagens de usuário são enviadas para o código que executa o Chat*bot* por meio de requisições REST. Dessa forma, as mensagens são passadas no formato *json*, recebendo da API uma resposta no mesmo formato.

6.2.4 *Bot* que recebe mensagem do usuário

O primeiro *bot*, é onde há a interpretação da mensagem do usuário. O *bot* é desenvolvido para receber a mensagem do usuário e interpretá-la, diferenciando a mensagem em sua intenção. Caso a intenção do usuário seja a busca de uma informação de um pedido ou embarque que necessita de uma busca no banco de dados, o *bot* gerará uma resposta selecionando quais tipos de informações o usuário deseja, seguindo os exemplos mostrados na subsessão 5.3.1.2. O *bot* receberá uma pergunta em linguagem natural do usuário, mas retornará uma resposta no formato de um dicionário para o sistema. Essa resposta conterá o numero de embarque ou pedido e os conteúdos desejados pelo usuário, assim, o código é capaz de fazer a busca correta. Em caso de a mensagem do usuário não requisitar uma busca, a parte da resposta do *bot* ficará vazia, que é representada por “order”, tendo apenas a parte de “content”, que estará preenchida com a mensagem do *bot*.

6.2.5 Programa de Coleta de Dados do DB

A WEG possui um software de planejamento de recursos empresariais (ERP), onde ali é feito e registrado tudo relacionado a pedidos. Entretanto o acesso a esse sistema é restrito, para isso o departamento de logística possui uma “cópia” do sistema em um banco de dados. A LIS, sistema onde o projeto seria incorporado, possui um banco de dados próprio. Nele são puxadas tabelas do ERP para serem utilizadas por todos os serviços disponibilizados na plataforma. O projeto utiliza dessas tabelas para fazer as consultas de pedidos e embarques.

Para fazer a consulta a essas tabelas será utilizado da linguagem Python com as bibliotecas SQLAlchemy e Pandas, a primeria para fzer a consulta no DB e a segunda para armazenar e tratar os dados. Python é amplamente reconhecido por sua simplicidade e versatilidade, tornando-se uma escolha popular para o desenvolvimento de programas de coleta e análise de dados. Uma das bibliotecas mais poderosas para trabalhar com bancos de dados em Python é a SQLAlchemy, que oferece uma interface completa para a criação e manipulação de bancos de dados relacionais. Neste texto, será apresentado como foi utilizado Python e SQLAlchemy para desenvolver o programa de coleta de dados.

O primeiro passo para interagir com um banco de dados usando SQLAlchemy é criar uma engine de conexão. A engine é o ponto central de comunicação entre o programa e o banco de dados. A engine criada pode ser vista na figura 23.

Figura 23 – Engine para conexão com o DB.

```
import sqlalchemy as sa
from sqlalchemy import create_engine, URL

# CRIA ENGINE PARA CONECTAR NO DB DA LIS
engine = create_engine(URL.create(
    host=os.getenv("DB_HOST"),
    username=os.getenv("DB_USERNAME"),
    password=os.getenv("DB_PASSWORD"),
    database=os.getenv("DB_DATABASE"),
)
)
```

Fonte: Arquivo do Autor.

Uma vez estabelecida a engine, ela serve como um ponto de entrada para realizar operações de consulta ao banco de dados. Com essa engine em mãos, a biblioteca pandas pode ser utilizada para executar consultas SQL diretamente no banco de dados. A pandas é uma biblioteca essencial para análise de dados em Python, oferecendo ferramentas robustas para manipulação e análise de dados. O método “read_sql_query” de pandas permite executar uma consulta SQL e armazenar os resultados em um DataFrame, como mostrado na figura 24.

Figura 24 – Método de Pandas para executar SQL.

```
# Faz a execução da query
result = pd.read_sql(base_query,engine)
```

Fonte: Arquivo do Autor.

Ao executar a *query* usando pandas, os resultados são armazenados em um DataFrame. O DataFrame é uma estrutura de dados bidimensional, similar a uma tabela, que facilita o trabalho com dados tabulares em Python. Com o DataFrame, é possível realizar as operações de manipulação de dados necessárias para a geração da resposta ao usuário. Um manipulação utilizada para tratar o banco de dados e retornar dados mais “limpos” é a de apagar duplicatas, que pode ser visto na figura 25.

Tendo removido as duplicatas, o próximo passo seria definir se as informações requisitadas pelo usuário serão retornadas como um arquivo PDF ou como uma mensagem gerada pelo segundo *bot*.

Figura 25 – Método de Pandas para remover duplicatas.

```
# Remove duplicatas do DataFrame
result = result.drop_duplicates()
```

Fonte: Arquivo do Autor.

6.2.6 *Bot* para interpretar resultados

O segundo *bot* utilizado neste projeto é encarregado de interpretar os resultados obtidos e retornar uma mensagem ao usuário. Para operar, ele recebe a pergunta do usuário, a língua em que foi feita e os resultados da busca no banco de dados no formato CSV. A interpretação ocorre pela associação dos dados adquiridos com a pergunta formulada. Desta maneira, o *bot* pode resumir as informações para devolvê-las de forma compreensível. Por exemplo, um embarque contém 100 itens diferentes, mas todos possuindo a mesma data de chegada. Se perguntado por essa data, ele pode sumarizar essa informação permitindo assim uma resposta sucinta.

Da mesma forma que o primeiro *Bot*, esse também recebe exemplos, sendo eles responsaveis por delimitar como devem ser feitas as respostas.

7 ANÁLISE DE RESULTADOS

Conforme apresentado no subsessão 4.4.5, o projeto não pôde ser implementado no sistema da empresa, sendo dessa forma, o projeto se avaliará pelo cumprimento dos requisitos levantados na subsessão 5.4. A avaliação será feita ponto a ponto a partir dos requisitos.

7.1 CHATBOT

Para verificar a veracidade, praticidade e sustentabilidade da aplicação, o projeto foi submetido a testes com colaboradores do setor de logística, especificamente com coordenadores de embarque. Estes profissionais são os principais responsáveis por atender às perguntas dos clientes relacionadas a pedidos e embarques. Dessa maneira, o chatbot pôde ser avaliado em um ambiente real, utilizando perguntas genuínas dos clientes, o que também possibilitou a obtenção de feedback valioso sobre melhorias potenciais e sobre diferentes tipos de informações que poderiam ser úteis de buscar, bem como diferentes métodos de pesquisa.

7.1.1 Análise de funcionamento

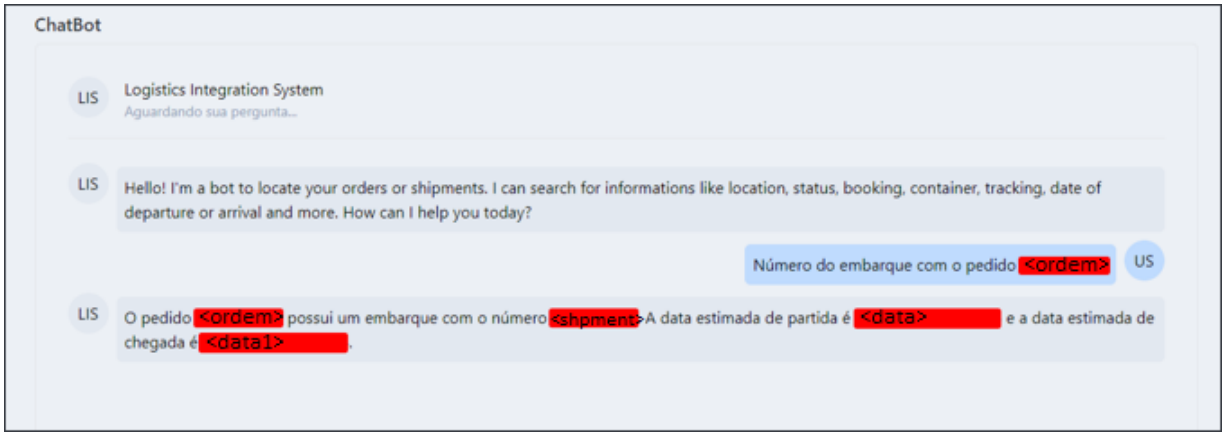
Ao analisar o comportamento do chatbot desenvolvido, constata-se que ele atende à maioria dos requisitos funcionais estabelecidos. Ao pontuar cada um deles, pode-se afirmar que o bot apresenta comportamento responsivo, respondendo prontamente às mensagens do usuário (Fig. 26, 27, 28 e 29) (requisito 1). Ele é capaz de buscar informações de forma eficiente no banco de dados da empresa (requisito 2), localizando corretamente as informações desejadas (requisito 3).

O bot compreende e responde em linguagem natural, graças ao modelo de linguagem da API da Azure OpenAI (requisito 5), sendo capaz de interpretar mensagens por meio do contexto, utilizando a memória das interações anteriores (requisito 9). Dessa forma, ele gera respostas claras e concisas (Fig. 26, 27, 28 e 29) (requisito 4).

Conforme ilustrado na Figura 26, foi desenvolvida uma interface de chat para a aplicação, que se comunica por meio de requisições REST com a API do chatbot (requisitos 8, 10 e 11). Em relação ao requisito 7, que estipula a não divulgação de informações sensíveis da empresa, uma das principais preocupações durante a evolução das versões do bot foi a segurança da informação. Para garantir essa segurança, a cada iteração foram limitados ainda mais os tipos de dados transmitidos para a IA. Na versão mais recente do bot, são passados apenas os nomes das colunas, assegurando que nenhuma informação sensível seja divulgada. Adicionalmente, há um script que detecta a presença dos nomes das colunas e elimina a mensagem potencialmente

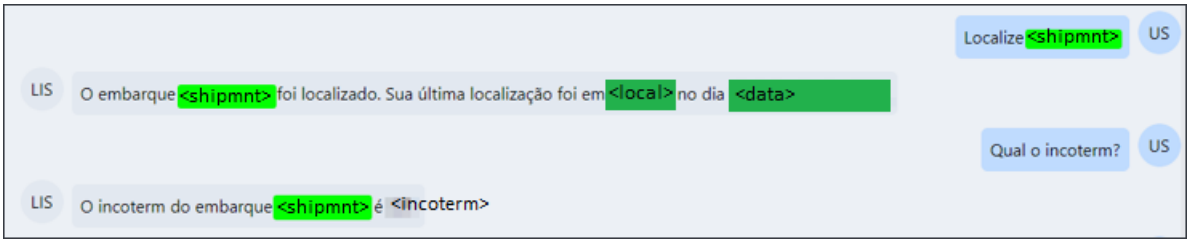
sensível, caso necessário.

Figura 26 – Exemplo de pergunta e resposta (1/4).



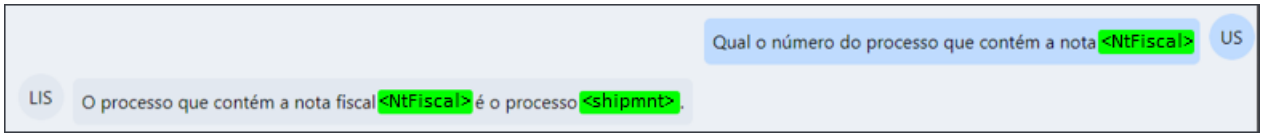
Fonte: Arquivo do autor (2024).

Figura 27 – Exemplo de pergunta e resposta (2/4).



Fonte: Arquivo do autor (2024).

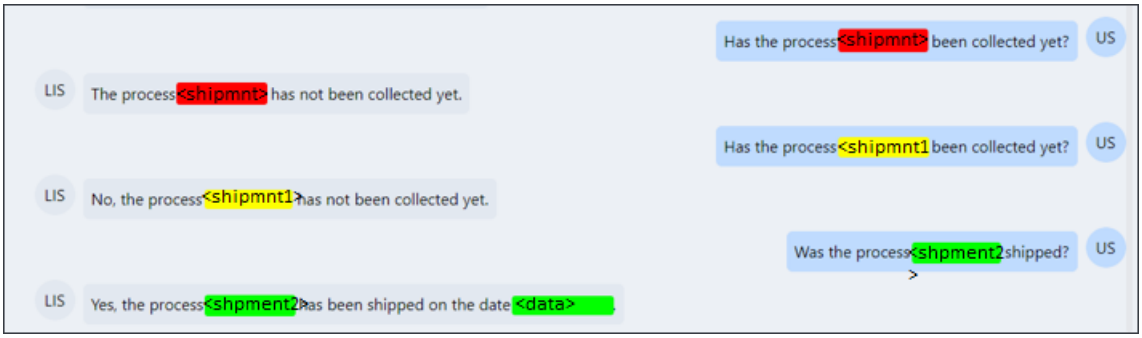
Figura 28 – Exemplo de pergunta e resposta (3/4).



Fonte: Arquivo do autor (2024).

Agora, outra análise relevante envolve os custos operacionais. Quando um cliente ou filial faz uma solicitação por e-mail, o analista precisa interromper suas atividades para responder à pergunta. Entretanto, há o tempo de espera apontado na introdução desse relatório, aproximadamente 6h. Além do mais, o tempo médio de busca de informação pelo analista é de aproximadamente 10 minutos por e-mail, variando conforme a complexidade e quantidade de dados requisitados. Em contrapartida,

Figura 29 – Exemplo de pergunta e resposta (4/4).



Fonte: Arquivo do autor (2024).

o chatbot é capaz de realizar buscas no banco de dados em um tempo médio de 1 minuto, independentemente da complexidade da pergunta, por requisição. Isso gera uma economia de aproximadamente **22,5 horas-homem**[1] por dia.

Além de sua eficiência em termos de tempo, o chatbot tem o potencial de ser acessado a qualquer hora do dia, sendo capaz de **reduzir o tempo de espera para zero**, proporcionando um método direto e contínuo de acesso à informação, uma vez que os colaboradores humanos não estão disponíveis em tempo integral para responder às perguntas. Essa disponibilidade contínua do chatbot resulta em uma melhora qualitativa do atendimento e quantitativa na velocidade de busca. O aumento de velocidade de busca está estimado em aproximadamente **10 vezes** mais rápida que a média humana. Agora, em termos de custo, a utilização do chatbot, para uma consulta, representa uma redução substancial de **99,19%** ao analisar o custo médio de uma consulta[2] do bot com o custo de uma consulta de um analista, levando em consideração o custo da hora[3] no setor de logística .

Portanto, a implementação do chatbot não só atende aos requisitos funcionais, como também oferece uma solução extremamente eficiente e econômica para a WEG, melhorando significativamente o atendimento e a agilidade na obtenção de informações.

Assim, colocando em destaque os objetivos alcançados, pontua-se que o chatbot desenvolvido tem a capacidade de:

- Redução de 22,5 horas-homem para consultas de dados.
- Redução de 100% no tempo de espera.
- Redução de 99,19% no custo operacional para a tarefa de busca de informações.

[1] Calculado com os valores médios e número de colaboradores responsáveis por responder clientes
[2] Valor não divulgado para não comprometer o valor sigiloso
[3] Valor sigiloso, não pôde ser divulgado

- Acelerar o processo de busca de informações em até 10 vezes.
- Automatização do processo de busca e atendimento ao cliente.
- Melhorar o fluxo de trabalho, reduzindo carga horária do colaborador e acelerando o processo de busca de informações.

Entretanto, apesar de possuir um potencial de ser uma ferramenta que pode ajudar em muito o setor que será aplicada, essa ferramenta possui algumas limitações dependente do banco de dados e limite de tokens que utiliza. Como todo projeto, o chatbot possui algumas limitações. No setor de logística, algumas das tabelas do banco de dados utilizadas no projeto são atualizadas apenas uma vez por dia. Essa limitação implica que o sistema pode conter informações potencialmente desatualizadas em comparação com o ERP da empresa. Isso pode levar a respostas que não refletem a situação mais recente, especialmente em consultas sensíveis ao tempo.

Outra limitação significativa está relacionada aos exemplos fornecidos ao bot e à memória de mensagens. Devido às restrições de quantidade de tokens que o bot pode processar e ao custo associado na API da Azure, há um limite na quantidade de exemplos que podem ser utilizados e a quantidade de mensagens que o bot consegue lembrar, pois ambos são passados no prompt. A falta de exemplos suficientes pode resultar em respostas ocasionalmente inconsistentes ou incoerentes, pois o bot pode não ter sido exposto a uma variedade suficiente de casos para aprender de maneira abrangente.

Ambas as limitações são críticas para o sucesso e precisão do chatbot no ambiente operacional da empresa. A gestão adequada desses desafios será essencial para maximizar a utilidade e eficácia do chatbot, garantindo que ele possa fornecer respostas precisas e relevantes aos usuários dentro das restrições operacionais e financeiras estabelecidas.

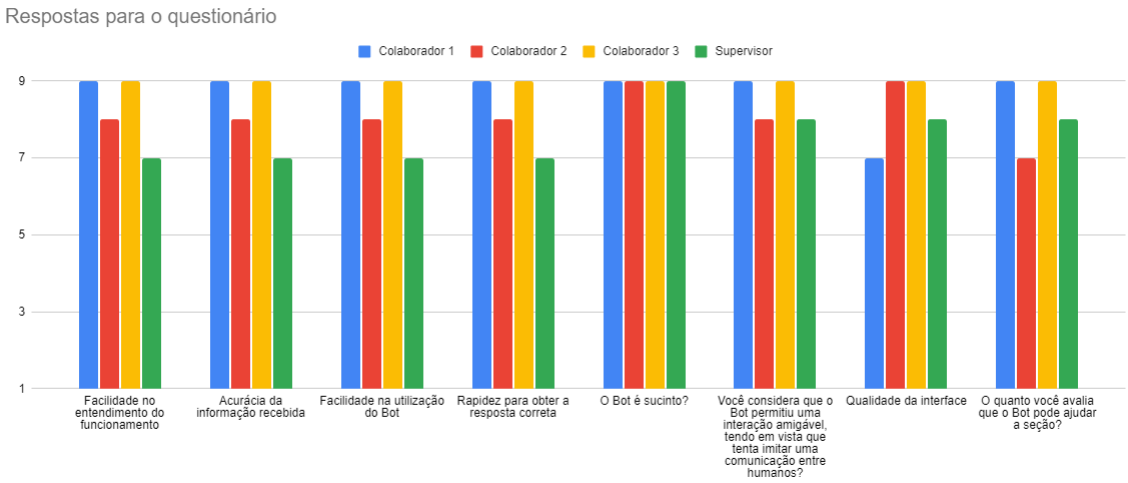
Para o caso de treinamento do modelo da Azure, é possível realizar o fine-tuning do modelo. Fine-tuning de um modelo de LLM refere-se ao processo de ajuste ou refinamento de um modelo pré-treinado em uma tarefa específica ou domínio de interesse. Modelos de LLM, como os baseados em arquiteturas Transformer, são geralmente treinados em grandes quantidades de texto não rotulado para aprender representações linguísticas gerais. No entanto, para aplicar esses modelos a tarefas específicas, é necessário ajustá-los aos dados e ao contexto específico dessa tarefa. O fine-tuning permite que o modelo pré-treinado adapte-se aos padrões e características dos dados específicos da nova tarefa, melhorando sua capacidade de realizar essa tarefa específica de forma precisa. Dessa forma, pode-se treinar o modelo tornando desnecessário a passagem de exemplos no prompt, tornando o sistema mais barato e confiável.

7.1.2 Avaliação dos colaboradores e supervisor

O projeto foi apresentado aos colaboradores, explicando sua finalidade, a metodologia das buscas e seu funcionamento detalhado. Após essa apresentação, foram conduzidos testes práticos, nos quais os colaboradores interagiram diretamente com o chatbot, fornecendo feedbacks que permitiram o sistema evoluir ainda mais. Em seguida, foi aplicado um questionário para avaliar a eficácia e a utilidade do projeto com base na experiência prática dos usuários. Como a visão dos colaboradores representa a ponta final do sistema, onde há a pergunta e recebe uma resposta, para eles, o sistema do chabot funciona como uma caixa preta. Assim, os pontos avaliados são em relação a requisitos não funcionais do sistema, como intuitividade e facilidade de utilização do chat, acelerar processo de busca e melhora da experiência do usuário, e alguns dos requsitos funcionais que tangenciam aos usuário e não somente ao sistema do chatbot. Sendo assim, os requisitos funcionais avaliados são os requisitos 1, 2, 3, 4, 5, 6 e 8. O requisito 1 está implicito na pergunta 3, o requisito 2, 3 e 6 estão na pergunta 2, o requisito 4 é a questão 5, o requisito 8 está na pergunta 7.

O questionário, cuja estrutura está ilustrada na Figura 30, permitia que os colaboradores dessem notas de 1 a 9 pontos, sendo 1 a menor nota e 9 a maior. Os resultados foram consolidados em um gráfico, que demonstrou uma avaliação positiva do projeto tanto por parte dos colaboradores quanto do supervisor da empresa.

Figura 30 – Gráfico de respostas.



Fonte: Arquivo do autor (2024).

Além das notas, o questionário incluía campos para comentários, permitindo que os colaboradores expressassem suas opiniões detalhadas sobre cada aspecto avaliado no questionário. Alguns dos comentários destacavam a praticidade do chatbot

e a clareza das informações fornecidas, assim como ideias para expandir as funcionalidades da ferramenta. Alguns dos comentários foram:

- “Atende a necessidade de responder questionamentos de clientes - especialmente filiais, sem exigir uma interação humana”.
- “Por que será bem útil para perguntas rápidos a respeito de datas e informações repetitivas que diariamente temos que responder as filiais no exterior”.
- “Automatiza o processo respondendo perguntas que viriam a ser um e-mail”.

O supervisor, por sua vez, foi mais incisivo em suas observações, destacando a eficiência e apontando situações específicas de funcionamento e possíveis melhorias que deveriam ser feitas. Ademais comentou dos benefícios que o chatbot pode trazer para o setor. Seus comentários incluíram:

- “Pode automatizar as respostas de diversas perguntas que hoje são feitas por email”.
- “Hoje nas atividades mapeadas na seção, o principal ladrão de tempo são os emails com consultas”.

Este feedback foi crucial para identificar áreas de melhoria e validar a aplicabilidade do chatbot no ambiente de trabalho real. A avaliação positiva dos coordenadores de embarque e do supervisor reforça a relevância do projeto, indicando que a integração do chatbot pode trazer benefícios significativos, como maior eficiência na resposta às perguntas dos clientes e redução do tempo gasto pelos colaboradores em tarefas repetitivas. Com base nos resultados obtidos, podemos concluir que o projeto não só é viável e sustentável, mas também possui grande potencial de impacto positivo nas operações do setor de logística.

8 CONCLUSÃO

Analisando todo o desenvolvimento do projeto, apesar de não ter sido possível incorporá-lo no sistema da LIS, é possível dizer que o projeto, como prova de conceito, se deu como um sucesso, tendo em vista os resultados alcançados com os testes locais. Esses testes serviram como uma prova de conceito sólida. O projeto de chatbot permite não apenas a automatização de processos, mas também a otimização de funções que demandam atividades repetitivas.

Durante os testes com os coordenadores de embarque, ficou claro que o chatbot atendeu às expectativas ao responder eficientemente às perguntas de clientes sobre pedidos e embarques. Sua capacidade de compreender linguagem natural, manter o contexto das conversas anteriores e interagir de forma precisa com o banco de dados da empresa foi um ponto chave para dizer que esse projeto foi um sucesso. Além disso, a interface intuitiva facilitou a utilização da tecnologia por parte dos colaboradores durante os testes, contribuindo para um melhor aproveitamento da ferramenta.

Embora ainda não tenha sido implementado, o projeto mostrou uma possível economia significativa de tempo e recursos. A possibilidade de redução no tempo de resposta das consultas, devido à velocidade e precisão do chatbot, representa uma melhoria substancial na eficiência operacional. Os comentários positivos dos colaboradores, que reconheceram o potencial do chatbot em simplificar suas rotinas e liberá-los para tarefas mais estratégicas, corroboram com o sucesso do projeto. Portanto, conclui-se que o projeto do chatbot não apenas alcançou seus objetivos iniciais, como também demonstrou ser uma solução viável e promissora para ser implementada na empresa.

Diante dessas considerações, algumas atividades futuras são sugeridas para o aprimoramento do projeto:

- Implantar o projeto na LIS.
- Realizar o Fine-tuning do modelo com uma quantidade maior de exemplos para garantir maior robustez às perguntas de usuário.
- Realizar um estudo da possibilidade de aumentar a frequência de atualização das tabelas do DB.
- Adicionar mais funções, permitindo uma maior diversidade de funcionalidades além de conseguir informações de pedidos e embarques. Podendo incorporar mais atividades além da busca de informações.

REFERÊNCIAS

ALURA. **Banco de Dados: o que é, principais tipos e um guia para iniciar**. [*S.l.*: *s.n.*], 2024a. https://www.alura.com.br/artigos/banco-de-dados#quais-sao-os-principais-tipos-de-bancos-de-dados?/. Acessado em: 14/05/2024.

ALURA. **Design Thinking: o que é, quais as principais etapas e como aplicar**. [*S.l.*: *s.n.*], 2024b. https://www.alura.com.br/artigos/design-thinking. Acessado em: 20/04/2024.

AMAZON. **Alexa Logo Guidelines**. [*S.l.*: *s.n.*], 2024. https://developer.amazon.com/en-US/alexa/branding/alexa-guidelines/brand-guidelines/the-alexa-logo. Acessado em: 09/06/2024.

APPLE. **Siri**. [*S.l.*: *s.n.*], 2024a. https://www.apple.com/br/siri/. Acessado em: 09/06/2024.

APPLE. **Usar a Siri em todos os dispositivos Apple**. [*S.l.*: *s.n.*], 2024b. https://support.apple.com/pt-br/105020. Acessado em: 01/06/2024.

CANALTECH. **O que é Cortana no Windows**. [*S.l.*: *s.n.*], 2022. https://canaltech.com.br/windows/o-que-e-cortana-no-windows-dicas-para-usar/. Acessado em: 09/06/2024.

FORBES. **How To Use Intelligent Automation For Revenue Generation**. [*S.l.*: *s.n.*], 2023. https://www.forbes.com/sites/serenitygibbons/2023/10/05/how-to-use-intelligent-automation-for-revenue-generation/. Acessado em: 14/05/2024.

FORBES. **The Transformative Power Of AI In Logistics**. [*S.l.*: *s.n.*], 2024. https://www.forbes.com/sites/forbestechcouncil/2024/06/21/the-transformative-power-of-ai-in-logistics/. Acessado em: 22/06/2024.

IBM. **IBM Logo Guidelines**. [*S.l.*: *s.n.*], 2024a. https://www.ibm.com/brand/experience-guides/developer/brand/logo#ibm-8-bar-logo. Acessado em: 09/06/2024.

IBM. **IBM Watsonx Assistant**. [*S.l.*: *s.n.*], 2024b. https://www.ibm.com/br-pt/products/watsonx-assistant. Acessado em: 21/04/2024.

LANGCHAIN. **Prompting strategies**. [*S.l.*: *s.n.*], 2024a. https://python.langchain.com/v0.1/docs/use_cases/sql/prompting/. Acessado em: 20/04/2024.

LANGCHAIN. **Quickstart**. [*S.l.*: *s.n.*], 2024b. https://python.langchain.com/v0.1/docs/use_cases/sql/quickstart/. Acessado em: 20/04/2024.

MEDIUM. **Large Language Models, GPT-1 — Generative Pre-Trained Transformer**. [*S.l.*: *s.n.*], 2022. https://towardsdatascience.com/large-language-models-gpt-1-generative-pre-trained-transformer-7b895f296d3b. Acessado em: 30/05/2024.

MICROSOFT. **Azure AI Services**. [*S.l.*: *s.n.*], 2024a. https://azure.microsoft.com/en-us/products/ai-services/. Acessado em: 24/04/2024.

MICROSOFT. **Copilot**. [*S.l.*: *s.n.*], 2024b. https://www.microsoft.com/pt-br/microsoft-copilot. Acessado em: 09/06/2024.

MICROSOFT. **Interface Copilot**. [*S.l.*: *s.n.*], 2024c. https://copilot.microsoft.com/. Acessado em: 09/06/2024.

MICROSOFT. **Microsoft Bot Framework**. [*S.l.*: *s.n.*], 2024d. https://dev.botframework.com/. Acessado em: 20/04/2024.

MICROSOFT. **Sql Server**. [*S.l.*: *s.n.*], 2024e. https://www.microsoft.com/pt-br/sql-server/sql-server-2019-features. Acessado em: 30/05/2024.

MICROSOFT. **What are Azure AI Services?** [*S.l.*: *s.n.*], 2024f. https://learn.microsoft.com/en-us/azure/ai-services/what-are-ai-services. Acessado em: 09/06/2024.

MOHAMAD SUHAILI, Sinarwati; SALIM, Naomie; JAMBLI, Mohamad Nazim. Service chatbots: A systematic review. **Expert Systems with Applications**, v. 184, p. 115461, 2021. Acessado em: 26/04/2024. ISSN 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2021.115461. Disponível em: https://www.sciencedirect.com/science/article/pii/S0957417421008745.

MONGODB. **MongoDB**. [*S.l.*: *s.n.*], 2024. https://www.mongodb.com/pt-br. Acessado em: 30/05/2024.

MYSQL. **MySQL**. [*S.l.*: *s.n.*], 2024. https://www.mysql.com/. Acessado em: 30/05/2024.

OPENAI. **ChatGPT**. [*S.l.*: *s.n.*], 2022. https://openai.com/chatgpt/. Acessado em: 20/05/2024.

OPENAI. **OpenAI Brand Guidelines**. [*S.l.*: *s.n.*], 2024. https://openai.com/brand/. Acessado em: 19/06/2024.

ORACLE. **Oracle Database**. [*S.l.*: *s.n.*], 2024. https://www.oracle.com/. Acessado em: 30/05/2024.

PÉREZ-SOLER, Sara; JUÁREZ-PUERTA, Sandra; GUERRA, Esther; LARA, Juan de. Choosing a Chatbot Development Tool. **IEEE Software**, v. 38, n. 4, p. 94–103, 2021. DOI: 10.1109/MS.2020.3030198.

RABELO, Ricardo J; ZAMBIASI, Saulo P; ROMERO, David. Softbots 4.0: supporting cyber-physical social systems in smart production management. **International Journal of Industrial Engineering and Management**, v. 14, n. 1, p. 63–93, 2023.

SQLALCHEMY. **SQLAlchemy**. [*S.l.*: *s.n.*], 2024. https://www.sqlalchemy.org/. Acessado em: 02/04/2024.

VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N; KAISER, Łukasz; POLOSUKHIN, Illia. Attention is All you Need. *In*: ADVANCES in Neural Information Processing Systems. [*S.l.*]: Curran Associates, Inc., 2017. v. 30. Disponível em: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

WEG. **História WEG**. [*S.l.*: *s.n.*], 2024. https://www.weg.net/institutional/BR/pt/history. Acessado em: 20/05/2024.

WIT.AI. **Wit.ai**. [*S.l.*: *s.n.*], 2024. https://wit.ai/. Acessado em: 20/04/2024.