Desenvolvimento de um Chatbot para Consulta de Registros de Produção Leiteira *

Matheus Hermes Neu[1], **Roger Luis Hoff Lavarda** [1]

[1]Ciência da Computação – Instituto Federal do Rio Grande do Sul (IFRS)
Campus Ibirubá

Abstract. *Dairy production plays a fundamental role in the Brazilian economy, contributing significantly to the income of thousands of Brazilians. One of the challenges faced by milk producers lies in the accurate collection and efficient management of daily production data. As such, there is much discussion about the application of technology in the agricultural sector to improve dairy production management. In this context, this article aims to develop a chatbot integrated with Telegram to monitor milk production using data from weighing systems. The development demonstrated the chatbot's ability to understand requests in natural language and provide clear and accurate responses. The solution significantly contributes to digital inclusion in rural areas, optimizing production monitoring and supporting producers in decision-making.*

Resumo. *A produção leiteira desempenha um papel fundamental na economia brasileira, contribuindo significativamente para a renda de milhares de brasileiros. Uma das dificuldades encontradas pelos produtores de leite reside na coleta precisa e no gerenciamento eficiente dos dados diários de produção. Assim sendo, muito se discute sobre a aplicação da tecnologia ao campo, a fim de aperfeiçoar a gestão da produção de leite. Neste contexto, este artigo tem por objetivo a criação de um chatbot aplicado ao Telegram para o controle da produção leiteira com dados oriundos de sistemas de pesagem. O desenvolvimento demonstrou a capacidade do chatbot em compreender solicitações em linguagem natural e fornecer respostas claras e precisas. A solução contribui significativamente para a inclusão digital no campo, otimizando o monitoramento produtivo e a tomada de decisões dos produtores.*

1. Introdução

Atualmente, a produção leiteira é de notável importância para a economia brasileira. Segundo dados da USP – Universidade de São Paulo (2022), contribui para o Produto Interno Bruto (PIB) nacional, para a geração de empregos e para ampliação da renda a milhares de brasileiros – da agricultura familiar aos grandes laticínios. No ano de 2020, com o total de 35,44 bilhões de litros de leite, a produção brasileira registrou aumento de 1,72% – comparado ao ano de 2019 – com destaque para os estados de Minas Gerais, Paraná, Rio Grande do Sul, Goiás e Santa Catarina (EMBRAPA – Empresa Brasileira de Pesquisa Agropecuária, 2022). Além disso, em 2021, conforme publicado pelo Ministério da Agricultura, Pecuária e Abastecimento (MAPA), o Brasil ocupou o 4º lugar como maior

*Trabalho de Conclusão de Curso (TCC) do curso Bacharelado em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul – Campus Ibirubá, 2025.

produtor de leite no mundo. Nesse cenário, produziu cerca de 34 bilhões de litros de leite no ano, movimentou mais de R$ 100 bilhões de reais neste mesmo período e gerou, aproximadamente, 4 milhões de empregos no campo (Portal E-Food, 2021).

Em face ao avanço do mercado leiteiro, os produtores de leite buscam o aperfeiçoamento das tecnologias aplicadas na produção da sua matéria-prima. Hoje, conforme publicado na revista MIT Technology Review, em 2021, o Brasil e o mundo se beneficiam com diversas tecnologias de automação: robôs para ordenha; pesagem do animal em cocho; dispositivos para detecção de cio; pesagem de produção; dentre outras. Segundo Lemos et al. (2003), o aumento de produtividade e gestão na produção são os objetivos principais do aporte tecnológico aplicado ao campo. Somado a isso, o estudo cita, ainda, que a internet e a modernização das ferramentas de conectividade no meio rural têm importância crucial para a comunicação do produtor, interação dentro e fora do meio rural, automatização de processos e etapas simples, além de permitir o controle e otimização da produção.

Com o avanço da tecnologia e o desenvolvimento da inteligência artificial, surge a ideia de desenvolver sistemas com habilidade de interagir com o usuário. Nesse contexto, chatbots são sistemas computacionais que objetivam manter uma conversação com o usuário como se fossem seres humanos. Em todo o mundo, tal tecnologia vem despertando interesse de diversas áreas para melhorar a relação ser humano-computador. Atualmente, possuem diversas aplicações, seja no meio acadêmico, social, comercial, entre outros. Neles, o sistema simula uma conversa com o usuário, podendo ela ser em linguagem técnica ou coloquial.

Mais recentemente, os avanços no campo do Processamento de Linguagem Natural (PLN) e o desenvolvimento de Modelos de Linguagem de Grande Escala (LLMs, na sigla em inglês), permitiram um salto qualitativo na capacidade dos chatbots de compreenderem e gerarem linguagem natural com alto grau de coerência e contexto (BROWN et al., 2020). Essas tecnologias ampliaram significativamente o potencial de aplicação dos assistentes virtuais, tornando-os mais eficientes na interpretação de comandos, fornecimento de informações específicas e interação personalizada com os usuários, inclusive em contextos rurais e produtivos.

Diante disso, este trabalho tem por objetivo o desenvolvimento de um chatbot para um aplicativo de mensagens instantâneas, a fim de auxiliar gestores de propriedades leiteiras. Esse assistente virtual pode contribuir de forma significativa para a rotina produtiva, ao permitir a consulta rápida de registros diários de produção, média por animal e por período, entre outros dados. Dessa forma, facilita a tomada de decisão com base em informações atualizadas e organizadas, reduz o tempo gasto com tarefas administrativas e promove maior eficiência na gestão da propriedade rural.

O presente artigo segue dividido em seções. Na Seção 2, é apresentada a Fundamentação Teórica e Tecnológica. Já na Seção 3, são abordados os Trabalhos Correlatos. Para detalhar a sistemática deste estudo, a Seção 4 descreve a Metodologia e Desenvolvimento do Sistema, seguida pela Seção 5, que apresenta os Testes Realizados. Para finalizar, a Seção 6 contém as Considerações Finais.

2. Fundamentação Teórica e Tecnológica

Nesta seção, serão apresentados conceitos e tecnologias utilizados para o desenvolvimento do trabalho proposto.

2.1. Produção leiteira

Hoje, o leite ocupa lugar de destaque na economia brasileira. Seu elevado valor de produção e sua alta rentabilidade trazem consigo geração de empregos e renda para milhares de trabalhadores — dentro e fora do campo (EMBRAPA – Empresa Brasileira de Pesquisa Agropecuária, 2002). Segundo a EMBRAPA – Empresa Brasileira de Pesquisa Agropecuária (2022), em 2020, os cinco maiores estados em produção de leite concentraram mais de 70% do total nacional, sendo eles: Minas Gerais, Paraná, Rio Grande do Sul, Goiás e Santa Catarina.

O panorama da produção leiteira no Brasil revela um setor com grande representatividade no agronegócio, mas que ainda enfrenta desafios relacionados à gestão eficiente, principalmente entre pequenos e médios produtores (IBGE – Instituto Brasileiro de Geografia e Estatística, 2021). Nesse cenário, o uso de tecnologias no agronegócio tem se mostrado essencial para melhorar a produtividade e a sustentabilidade. A digitalização rural e a automação de processos são tendências cada vez mais presentes nas propriedades, promovendo maior precisão no controle da produção, do manejo e da sanidade animal (SILVA; OLIVEIRA; SANTOS, 2022).

Para Brandão (2001), a modernização é singular para a continuidade da atividade leiteira. Com o aumento da demanda de produtos lácteos no mercado nacional, há a necessidade de melhorias contínuas na produção e qualidade do produto. Segundo Okamura (2021), o emprego da tecnologia na produção leiteira está associado ao aumento da produtividade, redução de custos, controle de qualidade da produção e melhoria do bem-estar animal. Dentre essas tecnologias podemos citar: aplicativos de gestão, robôs de ordenha, sensores de monitoramento da saúde do animal, vagões de trato com pesagem e automação de ambientes para controle térmico.

Contudo, um dos principais desafios ainda enfrentados nas propriedades é a gestão eficaz de dados no campo, que exige ferramentas acessíveis, integradas e de fácil uso. Nesse contexto, soluções inovadoras, como o uso de chatbots e sistemas integrados de informação, vêm sendo desenvolvidas para melhorar a tomada de decisão e otimizar o tempo de gestão. Essas soluções impulsionam o mercado nacional e promovem maior sustentabilidade ao setor (SOUZA; PEREIRA, 2023).

2.2. Chatbot

De acordo com Atwell e Abushawar (2015), um chatbot é um agente de software conversacional projetado para interagir com usuários por meio de linguagem natural. O primeiro chatbot reconhecido foi o ELIZA, desenvolvido por Joseph Weizenbaum em 1966. Esse sistema identificava palavras-chave nas entradas dos usuários e as comparava com um conjunto de regras pré-programadas para gerar respostas apropriadas (WEIZENBAUM, 1966). Embora sua capacidade de diálogo fosse limitada, o ELIZA influenciou significativamente o desenvolvimento de tecnologias posteriores nessa área (KLOPFENSTEIN; JOHNSON; RZEPKA, 2017).

Em essência, chatbots têm como objetivo possibilitar a comunicação entre humanos e sistemas computacionais de maneira intuitiva e acessível, por meio de interfaces baseadas em linguagem natural. A estrutura funcional de um chatbot pode ser representada por meio do diagrama ilustrado na Figura 1:

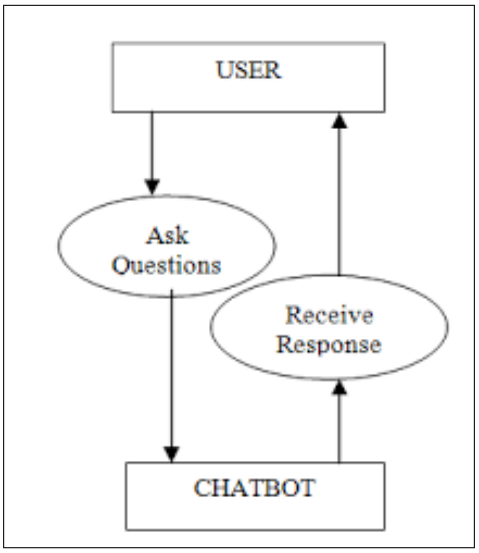

Figura 1. Diagrama de um chatbot (DAHIYA, 2017)

Os chatbots podem ser classificados em duas categorias principais: baseados em regras e baseados em inteligência artificial (IA). Os chatbots baseados em regras operam por meio de comandos específicos e seguem fluxos de conversação predefinidos. Para garantir sua eficácia, exigem que o usuário compreenda esses comandos, sendo necessário guiar a interação de forma estruturada (RAMPINELLI, 2017).

Por outro lado, os chatbots baseados em IA fazem uso de técnicas avançadas, como o Processamento de Linguagem Natural (PLN), o que lhes permite compreender e responder de forma mais flexível às mensagens dos usuários. Além do reconhecimento de padrões, esses sistemas podem aprender com interações anteriores e manter o contexto da conversa, tornando-se mais eficientes ao longo do tempo (KAR; HALDAR, 2016).

2.3. Processamento de Linguagem Natural (PLN)

O Processamento de Linguagem Natural (PLN) é um ramo da inteligência artificial e da linguística computacional que visa permitir que computadores compreendam e gerem linguagem humana. Inicialmente baseado em regras linguísticas e gramaticais simples, o campo evoluiu para métodos estatísticos e hoje depende fortemente de aprendizado de máquina (ML) e redes neurais profundas. Segundo Jurafsky e Martin (2021), a linguagem é a melhor ferramenta natural para comunicação entre humanos, e o PLN torna essa linguagem compreensível para as máquinas. Essa definição ressalta que o PLN atua como ponte entre a língua humana e os sistemas computacionais, sendo aplicado em tarefas como tradução automática, análise de sentimento, resumo de textos e assistentes virtuais.

Nas últimas décadas, o Processamento de Linguagem Natural (PLN) tem passado por avanços significativos, impulsionados principalmente por inovações em arquiteturas de redes neurais. Um marco fundamental foi a introdução da arquitetura Transformer,

proposta por Vaswani et al. (2017). Diferentemente de modelos anteriores baseados em redes recorrentes, os Transformers utilizam o mecanismo chamado self-attention, que permite ao modelo avaliar simultaneamente todas as posições de uma sequência textual, capturando relações contextuais entre palavras distantes e eliminando a necessidade de processar os dados em ordem sequencial. Isso resultou em modelos mais profundos, eficientes e paralelizáveis, capazes de lidar com dependências de longo alcance em textos complexos.

Com essa base, surgiram os chamados modelos pré-treinados em larga escala, que são treinados inicialmente em grandes volumes de texto para aprender representações gerais da linguagem, podendo ser adaptados para tarefas específicas com menor quantidade de dados rotulados. Um exemplo emblemático é o BERT (DEVLIN et al., 2018), que utiliza uma abordagem bidirecional para entender o contexto tanto à esquerda quanto à direita de cada palavra, destacando-se em tarefas como a compreensão de perguntas (question answering) e inferência textual. Outro marco é o GPT-3 (BROWN et al., 2020), um modelo autorregressivo com 175 bilhões de parâmetros, capaz de realizar diversas tarefas linguísticas apenas a partir de exemplos fornecidos no momento da consulta — uma técnica conhecida como few-shot learning. Essa capacidade permite que o GPT-3 gere textos coerentes, traduções e respostas a perguntas com qualidade próxima à humana, mesmo sem treinamento específico para essas tarefas.

Apesar dessas conquistas, o PLN ainda enfrenta desafios importantes. A ambiguidade semântica da linguagem natural, as variações culturais e linguísticas e a escassez de dados anotados para idiomas menos representados continuam dificultando o desenvolvimento de sistemas universais. Além disso, o crescente tamanho e complexidade dos modelos levantam preocupações éticas e ambientais. Conforme destacado por Bender et al. (2021), o custo financeiro e energético para treinar grandes modelos de linguagem (LMs) tem aumentado exponencialmente, contribuindo para impactos ambientais significativos. Também, Weidinger et al. (2022) propuseram uma taxonomia dos riscos sociais associados a esses modelos, incluindo a disseminação de desinformação por meio de textos sintéticos difíceis de detectar. Em suma, apesar do PLN transformar diversas aplicações — de tradutores automáticos a assistentes virtuais — ainda há desafios como o viés dos dados, o consumo elevado de recursos e a garantia de confiabilidade, que exigem pesquisas contínuas e responsáveis.

2.4. Modelos de Linguagem de Grande Escala (LLMs)

Os Modelos de Linguagem de Grande Escala (LLMs) são uma categoria específica dentro do PLN caracterizada por redes neurais profundas, geralmente baseadas em Transformers, treinadas em conjuntos de dados textuais massivos. Esses modelos possuem dezenas a centenas de bilhões de parâmetros; por exemplo, Brown et al. (2020) descrevem o GPT-3 com 175 bilhões de parâmetros – cerca de 10 vezes mais que qualquer modelo anterior não-esparso. Essa escala massiva permite que os LLMs realizem aprendizado in-context: eles podem executar tarefas novas apenas a partir de instruções textuais, sem ajuste de parâmetros, demonstrando desempenho competitivo em tradução, respostas a perguntas e outros problemas de NLP. Em aplicações práticas, LLMs vêm alimentando chatbots conversacionais avançados e assistentes virtuais capazes de gerar textos fluentemente. Essa capacidade de geração e entendimento contextual os torna poderosos em ferramentas de atendimento automático, composição de e-mails, auxílio à escrita criativa e outras

interações baseadas em linguagem.

Para adaptar LLMs às intenções humanas específicas, pesquisadores desenvolveram técnicas de refinamento pós-treinamento. Ouyang et al. (2022) aplicaram aprendizado por reforço com feedback humano (RLHF) para ajustar modelos pré-treinados, originando o InstructGPT. Em testes humanos, mesmo um InstructGPT com 1,3 bilhões de parâmetros foi preferido ao GPT-3 de 175 bilhões: suas saídas apresentaram maior veracidade e menos conteúdo tóxico. Isso confirma que o simples aumento do tamanho não garante adequação às demandas do usuário – o processo de alinhamento via RLHF é crucial para reduzir respostas indesejáveis.

Outra abordagem recente é a Geração Aumentada por Recuperação (RAG). Segundo Lewis et al. (2020), um sistema RAG combina o componente paramétrico de um LLM com uma base de conhecimento não-paramétrica (por exemplo, um índice de textos) que é acessada via mecanismos de recuperação. Esse esquema permite consultar informações externas durante a geração de texto. Nos experimentos originais, os modelos RAG obtiveram resultados de estado da arte em tarefas de perguntas e respostas, produzindo respostas mais específicas e factuais do que modelos somente paramétricos.

Apesar de sua versatilidade, os LLMs apresentam riscos significativos. Ouyang et al. (2022) enfatizam que "tornar os modelos maiores não os torna inerentemente melhores em seguir a intenção do usuário" – eles podem gerar respostas inverídicas, enviesadas ou tóxicas se não forem cuidadosamente controlados. Weidinger et al. (2022) destacam riscos sociais como a potencial utilização de LLMs para gerar desinformação em larga escala: esses modelos podem produzir conteúdo sintético difícil de distinguir de texto humano, facilitando campanhas maliciosas. Há também preocupações com vieses herdados dos dados de treino, perda de privacidade (já que treinar em grandes dados pode memorizar informações sensíveis) e uso indevido em atividades criminosas. Além disso, o elevado consumo computacional dos LLMs envolve custos ambientais e de acesso desiguais. Em suma, embora os LLMs ampliem muito as capacidades do PLN, é fundamental mitigar seus perigos éticos e sociais com práticas responsáveis de desenvolvimento e uso.

2.5. API Rest

Uma interface de programação de aplicação (API) define as regras que você precisa seguir para se comunicar com outros sistemas de software (AWS, 2023). Antes de 2000, não havia um padrão sobre como projetar ou usar uma API. Sua integração exigia o uso de protocolos de mensagens, como SOAP (Simple Object Access Protocol), que eram notoriamente complexos de construir, manipular e difíceis de depurar (SCHULTHESS, 2017).

Isso mudou com o estilo de arquitetura REST (Representational State Transfer), proposto por Roy Fielding no início dos anos 2000, que consistia em criar padrões que permitissem dois servidores se comunicarem e trocarem dados em qualquer lugar do mundo utilizando geralmente o protocolo HTTP (Hypertext Transfer Protocol) (FIELDING, 2000). Sendo REST baseado em recursos, Fielding se refere a estes utilizando os métodos (GET, POST, PUT, DELETE) como ações (Recuperar, Criar, Atualizar, Excluir) que qualquer sistema orientado a recursos deve ser capaz de fornecer.

De forma geral, os recursos em REST são definidos de forma única pelas URLs, e separados por sua representação no código. Representação, em sua essência, é um con-

junto de bytes e metadados que descrevem dados, como por exemplo o JSON (AWS, 2023). Sendo assim, o acesso a um recurso que forneça a lista de todos usuários de um sistema, por exemplo, seria uma requisição do tipo GET para a URL 'minha-Api.com/usuarios' e seu retorno uma representação JSON contendo uma lista de usuários.

2.6. Telegram Bot API

O Telegram é um aplicativo de mensagens gratuito para celulares e computadores baseado em nuvem. Tem como foco a praticidade, segurança nas informações compartilhadas e velocidade de performance. Criado em 2013, o Telegram é utilizado no mundo inteiro como aplicativo de conversação por usuários comuns e também desenvolvedores, por se tratar de uma ferramenta open source, com a capacidade de oferecer diversos serviços customizáveis (Telegram, 2021). Dentre eles, o principal, é a criação de bots, que são utilizados para diversas funcionalidades.

Bots são pequenos aplicativos executados inteiramente no aplicativo Telegram. Os usuários interagem com bots por meio de interfaces flexíveis que podem suportar qualquer tipo de tarefa ou serviço (Telegram, 2021). Por exemplo: o Rastreio Bot, que ajuda os usuários a rastrear suas encomendas pelos correios.

A criação de bots é simplificada com o auxílio da Telegram Bot API, uma interface baseada em HTTP criada para o desenvolvimento de bots para o Telegram (Telegram, 2021). Ela fornece as ferramentas necessárias para registro de bots escritos em várias linguagens de programação, tais como Python e Javascript. O desenvolvimento nessas linguagens permite aos bots executarem tarefas ou fornecer algum tipo de serviços aos seus usuários.

2.7. Gemini

O Gemini é uma família de modelos de linguagem desenvolvida pelo Google DeepMind, projetada para competir com outras inteligências artificiais generativas de ponta, como o GPT da OpenAI. Os modelos Gemini são treinados com grandes quantidades de dados e têm a capacidade de compreender e gerar linguagem natural, código e até conteúdos multimodais (Google DeepMind, 2023). A principal característica do Gemini está em sua integração com os serviços do Google, o que permite aplicações mais contextualizadas e relevantes em sistemas baseados em inteligência artificial.

A utilização do Gemini em sistemas computacionais pode oferecer melhorias em tarefas como análise de dados, processamento de linguagem natural, geração de código e suporte a sistemas de recomendação. Devido à sua arquitetura fundamentada em mecanismos avançados de redes neurais profundas, o Gemini atinge elevados níveis de precisão em tarefas complexas, consolidando-se como uma ferramenta importante para o desenvolvimento de soluções inteligentes (SILVA, 2024). No contexto de aplicações web, pode ser empregado para automatizar interações, gerar respostas em assistentes virtuais ou otimizar processos de desenvolvimento.

2.8. Node.js e NestJS

O Node.js é um ambiente de execução de código JavaScript no lado do servidor. Lançado em 2009, ele permite que desenvolvedores utilizem JavaScript para construir aplicações de backend de forma eficiente e escalável, especialmente em sistemas que exigem alta

concorrência e operações assíncronas (Node.js Foundation, 2025). Sua principal vantagem reside no modelo de I/O não bloqueante e orientado a eventos, tornando-o ideal para aplicações em tempo real, como chats, sistemas de notificação e APIs RESTful (TILKOV; VELTEN, 2010). Além disso, seu ecossistema conta com uma vasta biblioteca de pacotes disponíveis via NPM (Node Package Manager), o que facilita a integração de funcionalidades e acelera o desenvolvimento de software (CANTELLI, 2023).

Nesse contexto, o NestJS surge como um framework progressivo para construção de aplicações backend escaláveis com Node.js, utilizando TypeScript como linguagem principal. Inspirado em arquiteturas como a do Angular, o NestJS adota um modelo baseado em módulos, injeção de dependência e programação orientada a objetos, promovendo a organização do código e facilitando a manutenção de sistemas complexos (KOSSMANN, 2023). O framework integra-se eficientemente com bibliotecas modernas como TypeORM, Prisma e Mongoose, permitindo o desenvolvimento de APIs robustas com suporte a bancos de dados relacionais e não relacionais.

O NestJS tem ganhado destaque no desenvolvimento de aplicações empresariais e APIs modernas devido à sua ênfase em boas práticas de engenharia de software, como a separação de responsabilidades, o uso de decorators e testes automatizados (NestJS, 2025). Sua compatibilidade com o ecossistema do Node.js, aliada à segurança, escalabilidade e clareza arquitetural, torna-o uma escolha apropriada para projetos que exigem desempenho e organização, especialmente em ambientes de produção (FERNANDES, 2022).

3. Trabalhos Correlatos

Nesta seção, serão abordados três trabalhos que têm como objetivo utilizar chatbots para soluções de problemas propostos.

O primeiro trabalho, de García, Bernal e Muñoz (2017), objetivou desenvolver um jogo e um bot no sistema multiplataforma de mensagens Telegram. Trata-se de uma aplicação online que aproveita a plataforma para lançar e prover informações dos usuários no jogo. Através de um web service (Node.js) para comunicação em tempo real entre cliente e servidor (Node.js), os usuários registrados no bot o utilizam para se comunicar com o jogo através de comandos. Estes comandos, enviados via Telegram, chegam à aplicação e esta responde com uma URL (endereço eletrônico) para lançamento do jogo no próprio aplicativo - ou em um navegador do dispositivo. Os comandos podem ser disparados de uma conversa particular (single ou multiplayer) ou de grupos (group). Em sua conclusão, os autores relatam problemas no desenvolvimento do projeto, tais como: insuficiência de documentação para bot de jogos no Telegram, GET de informação pela URL, uso de sockets, cache de navegador, dentre outros. Contudo, eles também descrevem que os estudos trouxeram conhecimentos à equipe em programação e na utilização de frameworks.

O segundo trabalho, de Setiaji e Paputung (2018), traz o relato de estudantes que desenvolveram uma aplicação para compartilhamento de informações de interesse acadêmico. Através de um bot do Telegram, os usuários recebem mensagens com informações referentes a eventos, horários das aulas e mentorias. O bot funciona de duas maneiras: como mídia de informação e compartilhando informações de geolocalização. Com um bot administrador, usuários adicionam informações de interesse em um servidor (Python) com um banco de dados (MySQL). O servidor consulta a base de dados e, através

de um webhook[1], se comunica com o bot que distribui as informações para os usuários. Os autores concluem de forma satisfatória a utilização de webhooks na comunicação entre servidor e bot, com redução de tempo de resposta e disponibilidade diária.

O terceiro trabalho, de Mostaço et al. (2018), teve como objetivo implementar um chatbot para interação remota do utilizador com redes de sensores sem fio (WSN) em vinhedos. A fim de obter uma melhor experiência do usuário, foram utilizados o Telegram Bot API em conjunto com a plataforma IBM Watson Conversation para desenvolvimento do diálogo do chatbot. Os sensores dispostos nos vinhedos se comunicam com uma API REST que registra os dados em um banco de dados. Através da interface do Telegram, o usuário interage com o bot por mensagens ou opções disponíveis em menus, que são interpretados pelo assistente IBM Watson, que por sua vez envia a solicitação a API de dados dos sensores. Com base na pesquisa aqui descrita, foi possível fornecer um chatbot baseado em IA, auxiliando o especialista na aquisição de dados de clima e solo por meio de uma rede de sensores sem fio.

Tabela 1. Comparativo entre trabalhos com chatbots

Artigo	Aplicação	Tecnologias	Tipo de Chatbot
(GARCíA; BERNAL; MUñOZ, 2017)	Chatbot para lançamento de partidas de um jogo	Telegram, Node.js, Socket	Baseado em Regras
(SETIAJI; PAPUTUNG, 2018)	Chatbot para compartilhamento de informações acadêmicas	Telegram, Python, MySQL, Webhook	Baseado em Regras
(MOSTAçO et al., 2018)	Chatbot para consumo de dados de sensores em vinhedo	Telegram, IBM Watson, Node.js, WSN	Baseado em IA

A Tabela 1 apresenta uma comparação entre os trabalhos relacionados. Os dois primeiros trabalhos utilizam do mesmo tipo de chatbot proposto neste trabalho. Porém, o último estudo aborda um tema mais semelhante, pois retrata a realidade de produtores do agronegócio. Todos utilizam o Telegram como ferramenta de interface de chatbot, além das tecnologias e arquiteturas semelhantes ao trabalho proposto, mas com aplicações em áreas diferentes.

4. Metodologia e Desenvolvimento do Sistema

Esta seção descreve as principais etapas do desenvolvimento da solução, incluindo aspectos relacionados à arquitetura, tecnologias adotadas, modelagem de dados e funcionalidades implementadas.

4.1. Arquitetura geral e tecnologias utilizadas

O desenvolvimento deste trabalho foi organizado em etapas metodológicas sequenciais, com o objetivo de construir uma solução robusta e funcional para a consulta de dados de

[1]callbacks HTTP definidos pelo usuário que permitem a notificação automática entre sistemas quando eventos específicos ocorrem (COMARTIN, 2022)

produção leiteira por meio de um chatbot integrado ao Telegram. A Figura 2 ilustra a arquitetura geral do sistema, evidenciando a interligação entre os principais componentes tecnológicos envolvidos.

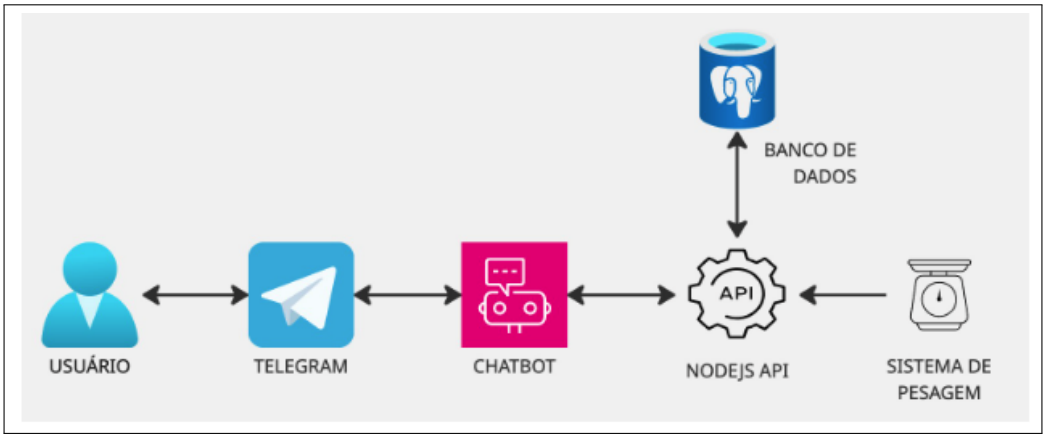

Figura 2. Arquitetura geral

A arquitetura proposta conecta um sistema de pesagem de leite a uma API desenvolvida em Node.js, responsável por receber e armazenar os dados em um banco de dados PostgreSQL. Um chatbot, integrado a essa API, possibilita a consulta automatizada das informações, utilizando o Telegram como interface de comunicação com o usuário. Com isso, produtores e técnicos podem acessar de forma prática e eficiente os registros das pesagens leiteiras diretamente pelo aplicativo de mensagens.

O fluxo inicia-se com o envio de uma mensagem pelo usuário no Telegram, contendo uma solicitação em linguagem natural, como, por exemplo, a produção total de determinado período. Essa mensagem é interceptada pelo chatbot, que interpreta o conteúdo com base em regras pré-definidas e, quando necessário, realiza requisições à API. A API processa a solicitação, consulta o banco de dados e retorna as informações solicitadas ao chatbot, que, por sua vez, as formata e envia a resposta de volta ao usuário por mensagem do Telegram.

Na sequência, são descritas as fases do processo de desenvolvimento tecnológico e suas contribuições para a composição do sistema final.

4.2. Modelagem e armazenamento de dados

Inicialmente, procedeu-se com a modelagem de dados, etapa fundamental para garantir a correta organização, integridade e armazenamento das informações de produção. Optou-se pelo sistema de gerenciamento de banco de dados PostgreSQL, reconhecido por sua robustez, confiabilidade e suporte a tipos de dados complexos. Nesta fase, foram definidas as estruturas de dados e os relacionamentos necessários para representar de forma fidedigna o contexto de uma propriedade leiteira, culminando na criação do esquema lógico e físico do banco de dados. O modelo de dados consiste basicamente em entidades como: usuário, propriedade, preço (por litro) e produção, representado na Figura 3.

A tabela **User** gerencia os dados de acesso e identificação dos usuários, como e-mail, telefone e o chatId do Telegram. Já a tabela **Property** armazena informações sobre as propriedades rurais, como nome e localização. A relação entre usuários e propriedades

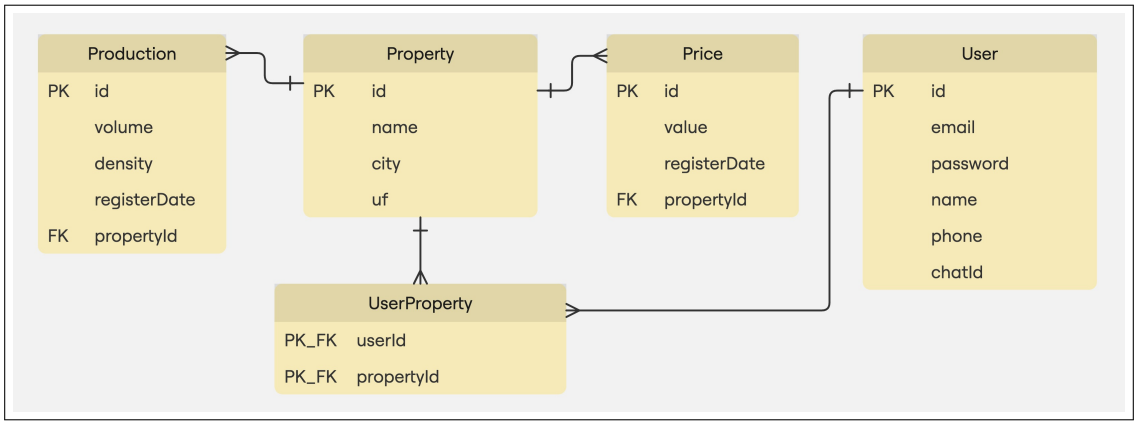

Figura 3. Diagrama de dados

é estabelecida pela tabela **UserProperty**, uma tabela de associação entre propriedade e seus usuários. As tabelas restantes são responsáveis pelos dados de produção. A tabela **Production** registra o volume e a densidade do leite produzido diariamente em cada propriedade. Por fim, a tabela **Price** armazena o valor do litro de leite para cada propriedade em uma data específica.

4.3. Implementação da API REST

Com a estrutura de dados estabelecida, a etapa subsequente consistiu no desenvolvimento de uma API RESTful utilizando Node.js com o framework NestJS. Esta API atua como uma camada de serviço intermediária (middleware), responsável por abstrair e gerenciar as operações de manipulação dos dados no banco de dados PostgreSQL. Foram implementados endpoints específicos para receber novas informações de produção, bem como para realizar buscas parametrizadas e agregar dados conforme as requisições. A tabela a seguir detalha os endpoints que compõem a API.

Tabela 2. Endpoints da API

Endpoint	Método	Descrição
/property	POST	Registrar propriedade
/property	GET	Buscar propriedades
/production	POST	Registrar produções de leite
/production	GET	Buscar produções por propriedade e período
/price	POST	Registrar preços de leite
/price	GET	Buscar preços por propriedade e período
/user	POST	Criar usuário
/user	PATCH	Atualizar usuário
/user	GET	Buscar usuários
/user-property	POST	Relacionar usuário a uma propriedade
/auth/signup	POST	Registrar conta de usuário
/auth/signin	POST	Fazer login
/auth/refresh	POST	Atualizar token de autenticação

Dentre os endpoints (funções) do sistema citados acima, os primordiais são aqueles diretamente relacionados aos dados de produção, como os de registro e consulta, apre-

sentados nas subseções 4.3.1 e 4.3.2, respectivamente. O sistema utiliza autenticação baseada em JWT (JSON Web Token) para proteger os endpoints e garantir que apenas usuários autorizados tenham acesso às funcionalidades da API. Esse mecanismo é amplamente adotado para autenticação e autorização em aplicações web e APIs, sendo fundamental para assegurar a integridade e a confidencialidade das informações (JONES, 2015).

4.3.1. Registro de dados de produção

Este endpoint é destinado ao registro de novos dados de produção no sistema. Ele permite o cadastro em lote, otimizando a comunicação ao aceitar múltiplos registros em uma única requisição. Utiliza o método POST (padrão REST) e está disponível no endpoint production. A requisição deve conter um array de objetos, em que cada objeto representa uma produção específica, com informações como o identificador da propriedade, o volume coletado e a data do registro. O corpo da requisição deve seguir o formato JSON, como apresentado na Figura 4.

Figura 4. JSON para registro de produção

4.3.2. Consulta à dados de produção

Com o objetivo de possibilitar a análise e visualização dos dados registrados, este endpoint permite a consulta das produções com base em filtros específicos. Utiliza o método GET (padrão REST), disponível no endpoint production. A busca é realizada com base no identificador da propriedade — obtido por meio da associação entre **chatId** e **UserProperty** — e em um intervalo de datas (inicial e final). Os parâmetros da requisição são:

- **propertyId**: Identificador único da propriedade.
- **startDate**: Data de início do período da consulta (formato AAAA-MM-DD).
- **finalDate**: Data de término do período da consulta (formato AAAA-MM-DD).

A resposta é composta por um array de objetos JSON, onde cada objeto representa um registro diário de produção que atende aos critérios da busca. A estrutura da resposta inclui, ainda, o preço do leite no dia, calculado dinamicamente pelo sistema no momento da consulta (caso já esteja cadastrado para o período).

```
[
  {
    "id": 1,
    "volume": 1000,
    "density": 1.032,
    "price": 2.15,
    "registerDate": "2025-05-01"
  }
]
```


Figura 5. JSON de resposta com dados de produção

4.4. Desenvolvimento do chatbot

A terceira etapa concentrou-se no desenvolvimento do chatbot propriamente dito, utilizando a biblioteca node-telegram-bot-api para a interação com a plataforma de mensagens Telegram. O diferencial desta solução reside na integração com a inteligência artificial generativa Gemini, do Google. Esta IA foi empregada para interpretar as mensagens enviadas pelos usuários em linguagem natural, realizar o reconhecimento de intenções e extração de entidades para consultas específicas, e formatar as respostas de maneira clara, concisa e humanizada.

Um aspecto crucial desta fase foi a elaboração de prompts detalhados para o Gemini. Estes prompts foram escritos para contextualizar a IA sobre o domínio específico do sistema, instruindo-a sobre como processar as diversas variações de perguntas dos usuários e como estruturar os dados retornados pela API para construir respostas úteis, precisas e contextualmente relevantes. Na Figura 6, pode-se observar o fluxo da interação entre a mensagem do usuário e a resposta do chatbot.

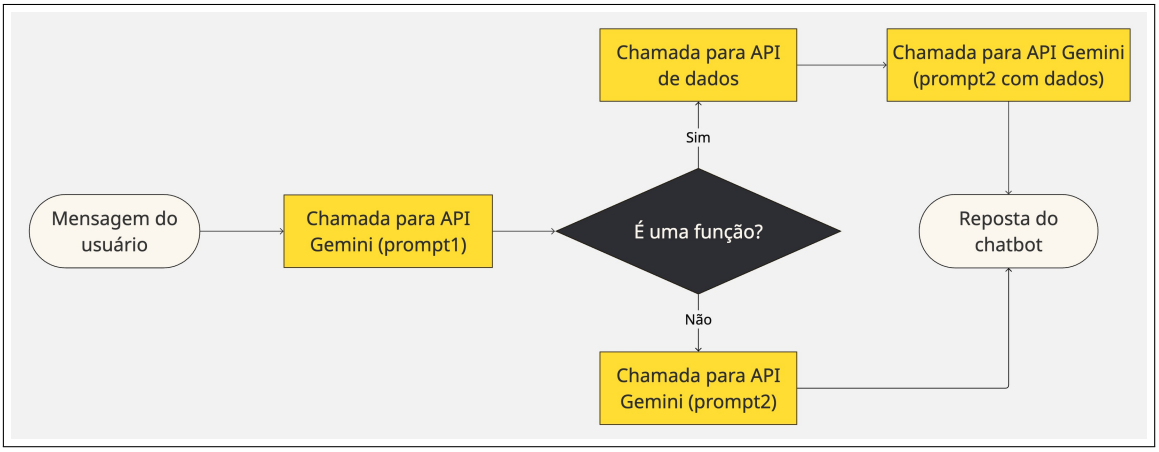

Figura 6. Fluxograma do chatbot desenvolvido

Este chatbot opera através de um fluxo de duas etapas principais, orquestradas por dois prompts (instruções) distintos para a IA Gemini, desenvolvidos pelo autor com base nas práticas de engenharia de prompts para LLMs, com foco na integração com funções

de backend. O primeiro, prompt 1, atua como um intérprete e roteador: ele analisa a mensagem do usuário para determinar sua intenção. Com base nessa análise, ele decide se pode gerar uma resposta conversacional direta ou se precisa acionar uma ”função”para buscar ou atualizar dados específicos, como registros de produção de leite ou preços. Sua saída é um JSON que ou contém a resposta final ou as instruções exatas para chamar a API de dados necessária. Um trecho da especificação do prompt inclui:

”Você é o Chatbot Assistente da MilkBot, especializado em gerenciamento de fazendas de leite e seu objetivo é: 1) Interpretar a solicitação do usuário e decidir como responder. 2) Se precisar buscar ou atualizar dados no banco, você chama uma das funções: - find_milk_registers (start_date, end_date) Descrição: Busca os registros de produção de leite em determinado intervalo de datas. [...]”

Caso uma função seja acionada e os dados sejam obtidos, o processo avança para a segunda etapa com o prompt 2. Este prompt atua como um sintetizador: ele recebe os dados brutos retornados pela API, juntamente com o contexto da pergunta original do usuário. Sua tarefa é traduzir essas informações técnicas em uma resposta final coesa, em linguagem natural e fácil de entender. Essa arquitetura de dois prompts separa a lógica de decisão da formatação da resposta, garantindo que o chatbot possa interagir de forma inteligente tanto em conversas gerais quanto ao fornecer informações precisas baseadas em dados.

Por fim, para que o chatbot se tornasse acessível aos usuários finais na plataforma Telegram, foi realizado o cadastro e configuração do bot utilizando o BotFather. Este é um bot oficial do Telegram que gerencia todos os bots na plataforma, auxiliando no processo de criação, personalização e fornecendo o token de acesso, essencial para a comunicação entre o código do chatbot e os servidores do Telegram.

5. Testes realizados

Finalmente, para validar a funcionalidade, a usabilidade e a eficácia do sistema integrado, foi conduzida uma breve etapa de testes utilizando um conjunto de dados reais de uma propriedade leiteira da cidade de Quinze de Novembro - RS, como detalhado na Figura 7.

Os dados foram adquiridos por meio de um produtor que anotava, em um calendário, a produção diária, conforme o recolhimento realizado pela empresa de laticínios. O eixo Y do gráfico representa o volume diário, e o eixo X, a data de recolhimento.

Esses dados foram inseridos através dos endpoints da API utilizando o Postman[2] e, posteriormente, consultados via interações com o chatbot, simulando diversos cenários de uso e tipos de perguntas. A Figura 8 mostra a primeira interação para consulta de dados do teste de validação.

A interação é uma pergunta enviada ao chatbot sobre a produção leiteira do mês de maio. Essa interação foi processada pelo modelo Gemini com o uso do primeiro prompt, responsável por interpretar e organizar a solicitação do usuário. A análise resultou num JSON que contém informações da função da API que deverá ser requisitada, como demonstrado na Figura 9.

O log mostra que a função **find_milk_registers**, descrita no prompt 1, foi identifi-

[2]software usado para envio de requisições em endpoints de API

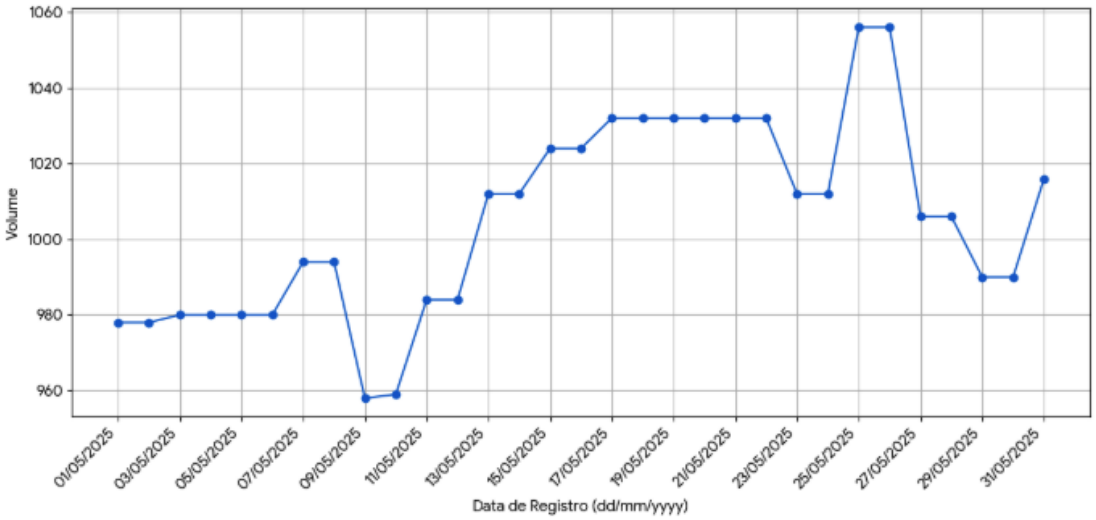

Figura 7. Gráfico de produção

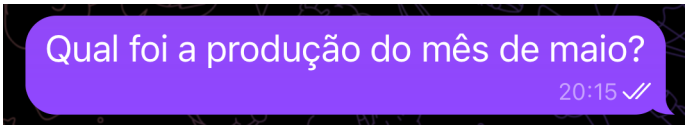

Figura 8. Primeira interação do teste

```
DEBUG: Resposta do Prompt1 limpa:
 {
  analysis: 'O usuário deseja saber a produção de leite no mês de maio do ano atual (2025). Por
tanto, precisamos buscar os registros de produção de leite do dia 01/05/2025 até 31/05/2025.',
  mensagem: null,
  funcao: {
    name: 'find_milk_registers',
    arguments: { start_date: '2025-05-01', end_date: '2025-05-31' }
  }
}
```

Figura 9. Log de resultado do gemini na primeira interação

cada. Isso indica que o sistema reconheceu a necessidade de realizar uma consulta à API de dados. Além disso, os argumentos referentes ao período (start_date e end_date) foram extraídos com sucesso, resultando na busca por registros de produção de leite dentro do intervalo especificado. Com os dados obtidos, o prompt 2 é preenchido, incorporando as informações necessárias para uma nova chamada ao Gemini. O resultado desse preenchimento é apresentado no na Figura 10.

Com o prompt 2 devidamente preenchido com os dados retornados pela API, o Gemini é novamente acionado. O processamento desta etapa resultou na mensagem descrita na Figura 11.

```
System (prompt):
Você é o Chatbot Assistente da MilkBot.
O usuário solicitou: Qual foi a produção do mês de maio?

Você chamou a função "find_milk_registers" com os argumentos:
{"start_date":"2025-05-01","end_date":"2025-05-31"}

O retorno da função foi:
"Producao total do periodo: 31157\nReceita: 101260.25\nInformacao de cada dia:\n2025-05-01: vol
ume 978, preco R$3.25;\n2025-05-02: volume 978, preco R$3.25;\n2025-05-03: volume 980, preco R$
3.25;\n2025-05-04: volume 980, preco R$3.25;\n2025-05-05: volume 980, preco R$3.25;\n2025-05-06
: volume 980, preco R$3.25;\n2025-05-07: volume 994, preco R$3.25;\n2025-05-08: volume 994, pre
co R$3.25;\n2025-05-09: volume 958, preco R$3.25;\n2025-05-10: volume 959, preco R$3.25;\n2025-
05-11: volume 984, preco R$3.25;\n2025-05-12: volume 984, preco R$3.25;\n2025-05-13: volume 101
2, preco R$3.25;\n2025-05-14: volume 1012, preco R$3.25;\n2025-05-15: volume 1024, preco R$3.25
;\n2025-05-16: volume 1024, preco R$3.25;\n2025-05-17: volume 1032, preco R$3.25;\n2025-05-18:
volume 1032, preco R$3.25;\n2025-05-19: volume 1032, preco R$3.25;\n2025-05-20: volume 1032, pr
eco R$3.25;\n2025-05-21: volume 1032, preco R$3.25;\n2025-05-22: volume 1032, preco R$3.25;\n20
25-05-23: volume 1012, preco R$3.25;\n2025-05-24: volume 1012, preco R$3.25;\n2025-05-25: volum
e 1056, preco R$3.25;\n2025-05-26: volume 1056, preco R$3.25;\n2025-05-27: volume 1006, preco R
$3.25;\n2025-05-28: volume 1006, preco R$3.25;\n2025-05-29: volume 990, preco R$3.25;\n2025-05-
30: volume 990, preco R$3.25;\n2025-05-31: volume 1016, preco R$3.25"
```

Figura 10. Prompt 2 preenchido com dados de produção

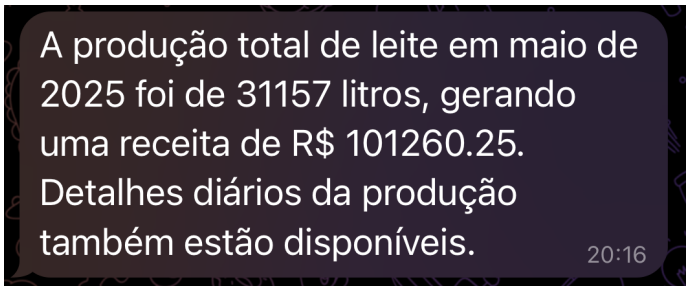

Figura 11. Resposta final do chatbot ao usuário

O segundo prompt foi utilizado para gerar uma resposta mais clara e bem estruturada ao usuário. Isso demonstra que o chatbot consegue transformar os dados brutos em respostas compreensíveis, o que melhora a experiência de uso. Uma segunda interação foi realizada, muito parecida com a primeira, porém com a adição de um novo parâmetro: o número de vacas da propriedade (35). Nessa nova solicitação, o usuário questiona qual foi a produção média diária por animal no mês de maio, buscando uma informação mais detalhada e personalizada.

Conforme ilustrado na Figura 12, o chatbot compreende a pergunta e responde prontamente com o cálculo aproximado da média diária por vaca, informando que, no mês de maio, a produção média foi de aproximadamente 28,7 litros por animal. Essa interação demonstra a capacidade do sistema de interpretar variáveis fornecidas pelo usuário e adaptar suas respostas de forma contextualizada.

Os testes foram desenhados para verificar a capacidade de interpretação semântica do Gemini frente a diferentes formulações de perguntas, a precisão dos dados retornados pela API, a integridade da comunicação entre os componentes e a clareza e utilidade das respostas geradas, permitindo identificar e corrigir eventuais falhas, além de otimizar a experiência geral do usuário.

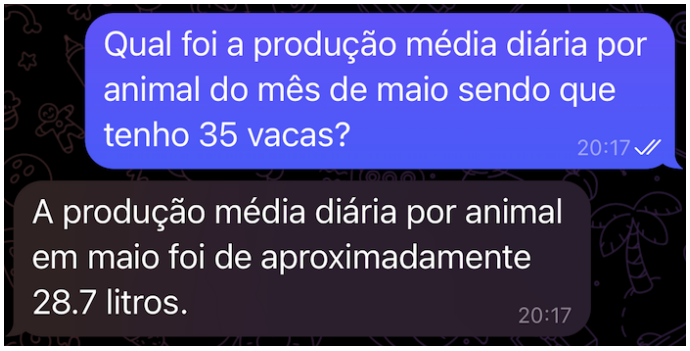

Figura 12. Segunda interação do teste

6. Considerações finais

O presente trabalho apresentou o desenvolvimento de uma solução tecnológica baseada em chatbot, com foco em melhorar o acesso de produtores rurais aos dados de produção leiteira. A aplicação foi implementada na plataforma de mensagens instantâneas Telegram, proporcionando uma interface acessível, leve e de fácil utilização para consulta de informações relevantes à gestão da propriedade.

Para fins de demonstração, testes pontuais foram realizados para evidenciar a capacidade do chatbot em compreender solicitações formuladas em linguagem natural, interpretar variáveis contextuais (como período e número de vacas) e fornecer respostas de forma clara, precisa e contextualizada. A solução demonstrou potencial para contribuir para a inclusão digital no campo, oferecendo praticidade a produtores que, muitas vezes, não têm familiaridade com sistemas informatizados complexos.

A ferramenta desenvolvida pode ser uma aliada importante na modernização da gestão da produção leiteira, especialmente em pequenas e médias propriedades. Seu uso promove maior controle produtivo, facilita a tomada de decisões baseada em dados atualizados e reduz a incidência de erros associados a registros manuais. Ao incorporar a tecnologia de forma acessível à rotina rural, o sistema contribui diretamente para uma agricultura mais conectada, eficiente e orientada por dados.

Como perspectivas futuras, a expansão e validação do sistema em um ambiente real de produção são cruciais. Sugere-se a realização de testes com uma amostra maior de dados e de usuários, para avaliar a robustez e a usabilidade da ferramenta em diferentes contextos. Além disso, propõe-se a evolução do projeto na integração com sistemas embarcados que automatizam a coleta de dados diretamente no ambiente produtivo. Destaca-se também o potencial de expansão da ferramenta para outras culturas agrícolas ou atividades pecuárias, bem como a implementação de um dashboard web para visualização gráfica e detalhada das informações. Recomenda-se ainda a integração com outras plataformas de mensagens, como o WhatsApp, e o aperfeiçoamento dos prompts de interação, com a utilização de modelos de linguagem mais avançados, visando respostas ainda mais naturais e eficientes.

Referências

ATWELL, E.; ABUSHAWAR, B. Machine Learning for Arabic Natural Language

Processing: N-grams and Beyond. *International Journal of Corpus Linguistics*, v. 20, n. 2, p. 216–240, 2015.

AWS. *O que é uma API?* 2023. Disponível em: ⟨https://aws.amazon.com/pt/what-is/restful-api/⟩.

BENDER, E. M. et al. On the dangers of stochastic parrots: Can language models be too big? In: *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (FAccT)*. [S.l.: s.n.], 2021.

BRANDãO, C. A. A modernização da agropecuária e suas implicações. *Revista Economia e Sociedade*, v. 10, n. 16, p. 75–94, jul/dez 2001.

BROWN, T. B. et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, v. 33, 2020.

CANTELLI, R. Desenvolvimento web com Node.js: explorando o potencial do JavaScript no backend. *Revista Brasileira de Computação Aplicada*, v. 12, n. 1, p. 55–64, 2023.

COMARTIN, D. Building a webhooks system with event driven architecture. *Tech Blog*, mar 2022. Publicado em 23 mar. 2022; acesso em 4 ago. 2025. Disponível em: ⟨https://codeopinion.com/building-a-webhooks-system-with-event-driven-architecture/⟩.

DAHIYA, M. A tool of conversation: chatbot. *International Journal of Computer Sciences and Engineering*, v. 5, n. 5, p. 37–43, 2017.

DEVLIN, J. et al. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint*, 2018.

EMBRAPA – Empresa Brasileira de Pesquisa Agropecuária. *Produção de leite no Sudeste do Brasil*. Brasília: [s.n.], 2002. Sistemas de Produção EMBRAPA. Disponível em: ⟨https://sistemasdeproducao.cnptia.embrapa.br/⟩.

EMBRAPA – Empresa Brasileira de Pesquisa Agropecuária. *Anuário do leite 2022*. Brasília: EMBRAPA, 2022. Disponível em: ⟨https://www.embrapa.br/busca-de-publicacoes/-/publicacao/1144110⟩.

FERNANDES, C. Arquiteturas modernas de APIs REST com NestJS: uma abordagem orientada a serviços. In: *Anais da Conferência Nacional de Engenharia de Software*. [S.l.: s.n.], 2022. v. 6, n. 1, p. 203–212.

FIELDING, R. T. *Architectural styles and the design of network-based software architectures*. Tese (Doutorado) — University of California, Irvine, 2000. Disponível em: ⟨https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm⟩.

GARCíA, J. d. O.; BERNAL, H. G.; MUñOZ, E. C. Telegram bot: track and field. *Universidad Complutense de Madrid*, 2017. Disponível em: ⟨https://eprints.ucm.es/id/eprint/49965/⟩.

Google DeepMind. *Gemini 1: Our largest and most capable AI model*. 2023. DeepMind. Disponível em: ⟨https://deepmind.google/technologies/gemini/⟩.

IBGE – Instituto Brasileiro de Geografia e Estatística. *Produção da Pecuária Municipal 2021*. Rio de Janeiro: [s.n.], 2021. Disponível em: ⟨https://www.ibge.gov.br/⟩.

JONES, M. B. *JSON Web Token (JWT)*. [S.l.], 2015. Disponível em: ⟨https://tools.ietf.org/html/rfc7519⟩.

JURAFSKY, D.; MARTIN, J. H. *Speech and Language Processing*. 3. ed. [S.l.]: Stanford University, 2021. Draft.

KAR, S.; HALDAR, S. Intelligent Chatbot for Guided Conversational Search. In: *IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*. New York: IEEE, 2016.

KLOPFENSTEIN, L. C.; JOHNSON, E.; RZEPKA, B. The Rise of Bots: A Survey of Conversational Interfaces, Patterns, and Key Design Issues. In: *Proceedings of the 2017 Conference on Designing Interactive Systems*. Edinburgh: ACM, 2017.

KOSSMANN, R. *NestJS: arquitetura modular e desenvolvimento de aplicações com TypeScript*. São Paulo: Novatec, 2023.

LEMOS, M. B. et al. Tecnologia, especialização regional e produtividade: um estudo da pecuária leiteira em Minas Gerais. *Revista de Economia e Sociologia Rural*, v. 41, n. 4, p. 767–790, 2003.

LEWIS, P. et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 2020.

MOSTAçO, G. M. et al. AgronomoBot: a smart answering chatbot applied to agricultural sensor networks. *IEEE Latin America Transactions*, v. 16, n. 12, 2018. Disponível em: ⟨https://www.researchgate.net/publication/327212062⟩.

NestJS. *Overview – NestJS Documentation*. 2025. Disponível em: ⟨https://docs.nestjs.com/⟩.

Node.js Foundation. *About Node.js*. 2025. Disponível em: ⟨https://nodejs.org/en/about⟩.

OKAMURA, R. *Tecnologia digital na produção de leite*. 2021. Venturus. Disponível em: ⟨https://www.venturus.org.br/tecnologia-digital-na-producao-de-leite/⟩.

OUYANG, L. et al. Training language models to follow instructions with human feedback. *arXiv preprint*, 2022.

Portal E-Food. *Brasil produz mais de 34 bilhões de litros de leite por ano*. 2021. Disponível em: ⟨https://portalefood.com.br⟩.

RAMPINELLI, F. *Chatbots no atendimento a clientes: tudo que você precisa saber*. 2017. DDS Blog. Disponível em: ⟨https://www.dds.com.br/blog/index.php/chatbots-atendimento-tudo-que-voce-precisa-saber⟩.

SCHULTHESS, C. *History of REST APIs*. 2017. Mobapi Blog. Disponível em: ⟨https://www.mobapi.com/history-of-rest-apis/⟩.

SETIAJI, H.; PAPUTUNG, I. V. Design of Telegram bots for campus information sharing. In: *IOP Conf. Ser.: Mater. Sci. Eng.* [S.l.: s.n.], 2018. v. 325, n. 1, p. 012005.

SILVA, J. L. d.; OLIVEIRA, M. G. d.; SANTOS, R. T. d. Transformação digital no campo: desafios e oportunidades no agronegócio brasileiro. *Revista Tecnologia e Sociedade*, v. 18, n. 49, p. 112–129, 2022.

SILVA, J. P. d. Modelos de linguagem multimodal: uma revisão sobre as principais abordagens contemporâneas. *Revista de Tecnologias Inteligentes*, v. 10, n. 2, p. 77–89, 2024.

SOUZA, F. M.; PEREIRA, L. C. Soluções digitais para a gestão da produção leiteira: uma revisão sistemática. *Revista Brasileira de Zootecnia*, v. 52, 2023. Disponível em: ⟨https://www.rbz.org.br/article/10.37496/rbz5220230123⟩.

Telegram. *A evolução do Telegram*. 2021. Disponível em: ⟨https://telegram.org/evolution⟩.

TILKOV, S.; VELTEN, S. Node.js: Using JavaScript to Build High-Performance Network Programs. *IEEE Internet Computing*, v. 14, n. 6, p. 80–83, 2010.

USP – Universidade de São Paulo. *PIB do Agro cresce 8,36% em 2021*. Piracicaba: [s.n.], 2022. CEPEA. Disponível em: ⟨https://www.cepea.esalq.usp.br/⟩.

VASWANI, A. et al. Attention is all you need. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2017. p. 5998–6008.

WEIDINGER, L. et al. A taxonomy of risks of large language models. *arXiv preprint arXiv:2202.00686*, 2022.

WEIZENBAUM, J. ELIZA – A Computer Program For the Study of Natural Language Communication Between Man and Machine. *Communications of the ACM*, v. 9, n. 1, p. 36–45, 1966.