



**APLICAÇÃO PARA GERENCIAMENTO AGRÍCOLA**  
**APPLICATION FOR AGRICULTURAL MANAGEMENT**  
**APLICACIÓN PARA LA GESTIÓN AGRÍCOLA**

Gabriel Aparecido Cespedes Estevo<sup>1</sup>, André Luiz da Silva<sup>2</sup>, Fabiana Florian<sup>3</sup>

<https://doi.org/10.47820/recima21.v6i1.7027>

PUBLICADO: 11/2025

**RESUMO**

Este estudo promove uma abordagem prática baseada no desenvolvimento de um sistema de gerenciamento agrícola, auxiliando na gestão de informações para tomadas de decisão do produtor. Foi desenvolvido um *software* utilizando ferramentas gratuitas para demonstrar a possibilidade de pequenos produtores aderirem à agricultura de precisão. O sistema possui tabelas para separação e formulários para inserção de dados, além de se comunicar com a inteligência artificial do Gemini para gerar novas ideias a partir dos dados cadastrados, e com o serviço de previsão do tempo da OpenWeather. O programa demonstrou eficácia no auxílio às tomadas de decisões para o manejo de culturas e manutenções preventivas dos equipamentos. Portanto, o uso de tecnologias sem custo, seguidas de boas práticas, são capazes de oferecer uma evolução e inovação tecnológica para pequenas propriedades rurais.

**PALAVRAS-CHAVE:** Agricultura de precisão. Tomada de decisão. Gerenciamento Agrícola.

**ABSTRACT**

*This study promotes a practical approach based on the development of an agricultural management system, assisting in information management for producer decision-making. Software was developed using free tools to demonstrate the possibility of small producers adopting precision agriculture. The system has tables for data separation and forms for data entry, and it communicates with Gemini's artificial intelligence to generate new ideas from the registered data, and with OpenWeather's weather forecasting service. The program demonstrated effectiveness in assisting decision-making for crop management and preventive maintenance of equipment. Therefore, the use of cost-free technologies, followed by good practices, can offer technological evolution and innovation for small rural properties.*

**KEYWORDS:** Precision agriculture. Decision-making. Agricultural management.

**RESUMEN**

*Este estudio presenta un enfoque práctico basado en el desarrollo de un sistema de gestión agrícola, destinado a apoyar la administración de información para la toma de decisiones del productor. Se desarrolló un software utilizando herramientas gratuitas para demostrar la posibilidad de que pequeños productores adopten la agricultura de precisión. El sistema cuenta con tablas para organización y formularios para el ingreso de datos, además de comunicarse con la inteligencia artificial de Gemini para generar nuevas ideas a partir de los datos registrados y con el servicio de previsión meteorológica de OpenWeather. El programa demostró eficacia en el apoyo a la toma de decisiones para el manejo de cultivos y el mantenimiento preventivo de los equipos. Por lo tanto, el*

---

<sup>1</sup>Graduando do Curso de Sistemas de Informação Gabriel Aparecido Cespedes Estevo da Universidade de Araraquara - UNIARA. Araraquara-SP.

<sup>2</sup>Orientador. Docente do Curso de Sistemas de Informação André Luiz da Silva da Universidade de Araraquara-UNIARA. Araraquara-SP.

<sup>3</sup>Coorientador. Docente do Curso de Sistemas de Informação Fabiana Florian da Universidade de Araraquara-UNIARA. Araraquara-SP.

*uso de tecnologías sin costo, acompañado de buenas prácticas, es capaz de ofrecer evolución e innovación tecnológica para pequeñas propiedades rurales.*

**PALABRAS CLAVE:** Agricultura de precisión. Toma de decisiones. Gestión agrícola.

## 1. INTRODUÇÃO

A ideia de tratar culturas agrícolas considerando fatores como fertilidade do solo, localização e clima antecede a Revolução Industrial. No entanto, os fundamentos da agricultura de precisão surgiram no início do século XX. Foi somente na década de 1980, com o avanço dos microcomputadores, sensores e *softwares* nos Estados Unidos e na Europa, que essa técnica se tornou viável para os produtores rurais (Lamparelli, 2010).

Segundo a *International Society of Precision Agriculture* (ISPA), “A Agricultura de Precisão é uma estratégia de gestão que leva em consideração a variabilidade temporal e espacial para melhorar a sustentabilidade da produção agrícola” (ISPA, 2024, p. 1, tradução nossa).

Portanto, a gestão das atividades realizadas em áreas de cultivo é importante para os produtores, pois fornece informações valiosas que ajudam na tomada de decisões mais acertadas, evitando desperdícios e promovendo a sustentabilidade no campo.

De acordo com Braga e Cunha (2022), a agricultura de precisão busca otimizar as funções produtivas levando em consideração fatores como localidade, tempo e quantidade ideais para as atividades agrícolas, de acordo com os objetivos da produção e as restrições econômicas do negócio. Quando bem aplicada, essa abordagem resulta em maior rentabilidade, menor desperdício de insumos e práticas agrícolas favoráveis ao meio ambiente.

Este trabalho tem o objetivo de desenvolver uma interface interativa, gerenciando e organizando dados agrícolas, visualização da previsão do tempo, criação de um banco de dados e consulta de *insights* estratégicos. O sistema se comunica com um serviço de previsão do tempo que retorna dados climáticos em tempo real e uma inteligência artificial que criará *insights* estratégicos sobre os dados cadastrados no sistema. Toda informação será apresentada de forma analítica ao usuário, complementando de forma estratégica o gerenciamento das atividades agrícolas. Desta maneira, implementar o conceito de agricultura de precisão à propriedade.

A agricultura se encontra pressionada pelo crescimento rápido e contínuo na demanda por alimentos, pois com o crescimento populacional cresce também o consumo *per capita* e a necessidade da população por saúde e qualidade de vida (Silva; Cavichioli, 2020). Neste contexto, a incorporação de tecnologias da informação permite otimizar o uso de recursos e reduzir custos operacionais e desgastes naturais.

Além disso, as alterações climáticas afetam diretamente o desenvolvimento e a produtividade das lavouras. Há fatores meteorológicos que podem tanto beneficiar quanto prejudicar as plantações (Braga; Pinto, 2022). Assim, o acesso a informações climáticas atuais e futuras é fundamental para a tomada de decisões estratégicas no campo. Um sistema que forneça essas informações, aliado às informações contidas no sistema agrícola, representa um avanço em inovação, segurança e confiabilidade.

Apesar das vantagens, a adoção da tecnologia ainda enfrenta barreiras, segundo Ferraz e Pinto (2017), atividades como coleta, cadastro e manutenção dos dados muitas vezes são escritas em folhas de papel e não são tratadas de maneira indevida. Em propriedades familiares este desafio se torna mais presente, pois normalmente a pessoa que gerencia as atividades nessas situações tem mais idade, e não possui aptidão nem prática, conseqüentemente evita o uso de ferramentas tecnológicas (FERRAZ; PINTO, 2017).

Em vista disso, a hipótese desta pesquisa é dar oportunidades aos usuários de utilizarem uma tecnologia instintiva e acessível, enfrentando problemas como o alto custo e complexidade que prejudicam a adoção da agricultura de precisão. Logo, o programa criado para esse estudo, que utiliza sistemas gratuitos e de fácil compreensão, é capaz de oferecer uma oportunidade para os clientes que buscam tecnologias baratas e claras para suas fazendas.

O estudo promove uma abordagem aplicada que envolve o desenvolvimento de uma área de interação que administra e organiza informações sobre os veículos, seções, mapas, atividades, histórico de atividades e insumos em conjunto com dados climáticos enviados por sites de previsão do tempo e *insights* gerados pela inteligência artificial do Gemini. As tecnologias incluídas serão: Spring Boot, responsável pela comunicação entre a interface e o banco de dados, manipulando os dados conforme as interações do usuário; React, utilizado para a construção da interface interativa que apresentará as informações de forma clara e intuitiva ao cliente e permitirá o cadastro de novos dados; e MongoDB, empregado para armazenar os dados estruturados do sistema.

Para o armazenamento dos arquivos de mapas agrícolas, optou-se pelo uso do Firebase Storage, um serviço de armazenamento em nuvem que permite o envio e acesso remoto dos arquivos de forma segura. Em vez de salvar o caminho do arquivo local, será armazenado o link gerado pelo serviço na nuvem, o que garante escalabilidade, confiabilidade e proteção contra perdas locais de dados.

A usabilidade, confiabilidade e eficiência do sistema serão testadas com informações agrícolas reais, e a segurança será assegurada por meio da autenticação de usuários com a tecnologia *JSON Web Token* (JWT), garantindo o acesso ao sistema apenas para usuários autenticados.

A metodologia do estudo conta com etapas de modelagem de dados, desenvolvimento da interface, lógica de aplicação, comunicação entre os componentes, validações, persistência dos dados e mecanismos de segurança. Além disso, serão utilizadas referências e revisões bibliográficas acadêmicas para embasar as decisões técnicas e promover maior qualidade ao estudo.

## **2. REVISÃO BIBLIOGRÁFICA**

Serão expostos nesta seção conceitos sobre agricultura de precisão e das tecnologias utilizadas no desenvolvimento.

## 2.1. Agricultura de precisão

A agricultura de precisão é uma técnica de manejo que analisa os espaços das diferentes áreas rurais calculando aplicações modificadas de local para local para recursos como: fertilizantes, sementes, água, pesticidas, entre outros. Há também a consideração de variações temporais que auxiliam na racionalização dos insumos, no momento, local e dose corretos, trazendo vantagens econômicas e ambientais (Basso *et al.*, 2019, *apud* Ezenne *et al.*, 2019).

A ideia de um ciclo na agricultura de precisão funciona com as seguintes fases: coleta de dados, ajustamento e interpretação dos dados, entrega de recomendações, aplicação no campo e análise dos resultados (Basso *et al.*, 2019, *apud* Geebers; Adamchuck, 2010).

Dessa forma, a agricultura de precisão ajuda os proprietários rurais na tomadas de decisões gerenciais embasadas em dados reais, considerando a variabilidade temporal e espacial da área, contribuindo com a sustentação do meio ambiente e retorno econômico (Basso *et al.*, 2019, *apud* Inamasu *et al.*, 2011). Dito isso, é uma sequência de conhecimentos gerados por máquinas, *softwares* e dispositivos, que devem se dispor de um tratamento para apoio à gestão (Basso *et al.*, 2019, *apud* Inamasu; Bernardi, 2014).

Os desafios da agricultura de precisão se associam ao entender as causas das variações das características dos solos e das plantas, carência de recomendações agrônomas específicas para aplicação à taxa variável, elevado custo para mapeamento do solo em uma escala eficiente, necessidade de melhoras no âmbito técnico de sensoriamento remoto para aplicações à taxa variável, carência de mão de obra especializada em várias modalidades, (equipes técnicas, consultores e pesquisa) e falta de funções padronizadas (Basso *et al.*, 2019).

## 2.2. Arquitetura REST

O estilo arquitetônico *Representational State Transfer* (REST) ou Transferência de Estado Representacional, apresentado por Roy Fielding, nos anos 2000, durante sua tese de doutorado na universidade da Califórnia, é um estilo híbrido derivado de restrições de outros modelos arquitetônicos baseados em redes. Esta arquitetura é comumente usada para a criação de sistemas de hipermídia distribuídos (Fielding, 2000).

A derivação deste modelo baseia-se em algumas restrições que explicarei adiante, seguida dos conceitos de recursos, identificador de recursos e os elementos de dados dessa arquitetura, definidos por Fielding.

### 2.2.1. Restrição Client-Server

A restrição de cliente-servidor definida por Fielding (2000) segregava a interface que o cliente usa do armazenamento de dados, podendo expandir a interface para múltiplas plataformas, deixar o sistema mais escalável e possibilitar a evolução independente dos componentes, atendendo os requisitos de escala da Web.

### 2.2.2. Restrição Uniform Interface

De acordo com Fielding (2000), essa restrição difere a arquitetura REST das demais que são baseadas na rede, pois utiliza uma interface padronizada entre as comunicações dos clientes com o servidor, ou seja, todos os recursos transmitidos pela rede são requisitados e acessados da mesma maneira. Esta interface é eficiente em transferência de dados de hipermídia, agregando vantagens em serviços Web, mas pode não ser eficiente em casos mais específicos.

### 2.2.3. Restrição Stateless

Relacionado ao modelo de cliente-servidor, esta restrição exige que a comunicação seja feita uma vez contendo todas as informações necessárias, pois o servidor não guarda os estados das requisições como em modelos Stateful (Fielding, 2000). As requisições HTTP que seguem esta restrição são autossuficientes, sendo compreendidas perfeitamente pelo servidor.

Essa restrição também é responsável por aprimorar a escalabilidade do sistema, pois não precisa armazenar os estados entre as solicitações dos clientes, porém, pode diminuir seu desempenho pela necessidade de enviar dados repetidos, já que não há um contexto para salvá-los (Fielding, 2000).

### 2.2.4. Restrição Layered System

O Sistema em camadas é essencial nesta arquitetura, promovendo uma cadeia hierárquica bem definida que limita a visão dos componentes para apenas a camada que ele está interagindo, quebrando a complexidade do sistema e encapsulando serviços. Contudo, o sistema em camadas pode sofrer sobrecarga e aumentar a latência, reduzindo o desempenho (Fielding, 2000).

### 2.2.5. Elementos de Dados

Os elementos REST possuem uma interface de comunicação padrão e focam em entender os tipos de dados compartilhados com metadados. O cliente consegue escolher dinamicamente o tipo dos dados, mas a interface oculta a fonte dos dados. Além disso, transfere uma representação dos dados com instruções para uma interface padronizada que consegue interpretar esses comandos (Fielding, 2000).

**Quadro 1.** Elementos de Dados REST

<b>Elementos de Dados</b>	<b>Exemplos Modernos da WEB</b>
Recurso	O conceito alvo de uma referência de hipertexto
Identificador de Recurso	URL, URN
Representação	Documento HTML, imagem JPEG, JSON
Representação dos Metadados	Tipo de Mídia, Data da Última Modificação
Metadados do Recurso	Link de Origem, Alternativas, Variações

Dados de Controle	Se-Modificado-Desde, Controle de Cache
-------------------	----------------------------------------

Fonte: Fielding, 2000

### 2.2.6. Conceito de Recursos

É a forma de representar uma informação, podem ter valores iguais, porém o que realmente importa é o seu significado. Portanto, é possível que um recurso não contenha elementos dentro dele, podendo ser referenciado mesmo sem nenhum registro. A semântica do mapeamento do recurso deve ser estática, pois isso é o que diferencia um recurso de outro (Fielding, 2000).

Um recurso é algo genérico que pode significar qualquer coisa e ter vários tipos e sua representação é definida no momento da solicitação. Se o identificador do recurso estiver implementado corretamente, não há necessidade de alterá-lo de acordo com as mudanças das representações de seus dados (Fielding, 2000).

## 2.3. Spring Boot

*Spring Boot* é um *framework* de código aberto Java e gratuito, com a intenção de simplificar o desenvolvimento de aplicações reduzindo a perda de tempo dos desenvolvedores nas definições de configurações iniciais (Calça, 2022 *apud* Rosali, 2021).

Existe um conceito chamado “convenção sobre configuração”, consiste no próprio *framework* criar as configurações que são escritas na maioria das vezes pelos programadores no início do projeto. A palavra “sobre” significa uma sugestão de configuração, mas se o projeto possuir outro foco o programador precisa personalizá-las (Weissmann, 2015).

Segundo Weissmann(2015), este *framework* possui quatro princípios:

- Iniciar um projeto de forma direta e rápida.
- Sugestão sobre configurações, mas mantendo o conforto para modificação delas de acordo com o foco do projeto.
- Provê requisitos não funcionais pré-definidos, por exemplo, métricas, acesso a base de dados, segurança, servidor de aplicação embarcado etc.
- Reduzir a necessidade de arquivos XML e a geração de código.

## 2.4. React

ReactJs é uma biblioteca de código aberto JavaScript para construção de interfaces de usuário. Lida com a camada de visualização em aplicativos móveis e de página única. Mantida pelo Instagram, Facebook e pela comunidade de empresas e desenvolvedores, tem o objetivo de agregar escalabilidade, simplicidade e rapidez. Seus recursos principais são os componentes com estado, JSX e Modelo de Objeto de Documento Virtual (DOM) (Khuat, 2018).

Segundo Aggarwal (2018) a performance do React se deve ao DOM virtual dentro da memória do computador, pois sempre que uma mudança reflita na página web naquele instante as primeiras alterações são feitas no DOM virtual, depois é processado um algoritmo diff, comparando a DOM virtual da DOM de navegação e apenas os nós mais relevantes do navegador são atualizados, resultando em grande eficiência.

Esta biblioteca oferece nativamente o controle de estados por meio de classes e funções, possibilitando a alteração do estado interno de um componente utilizando o método `setState`, além do uso de *hooks* como `useState` ou `useReducer` (Azzolini, 2021).

## 2.5. MongoDB

MongoDB é um banco de dados NoSQL, não utilizando linhas e colunas para o armazenamento dos dados. Por este motivo ele é orientado a documentos e se baseia no conceito de dados, documentos autocontidos e auto definidos, ou seja, ele deve expressar seu significado de acordo com os dados armazenados em sua estrutura (Souza; De Oliveira, 2019).

Como o MongoDB é orientado a documentos, ele tem como característica apresentar todas as informações importantes em um único documento, conter um identificador único universal (UUID), não precisar de esquemas, proporcionar consultas de documentos através de métodos avançados de agrupamento e filtragem (MapReduce), além de possibilitar redundância e inconsistência (Higor, 2014).

Este banco de dados persiste em seus documentos no formato Binary JSON (BSON), sendo objetos de JSON binários. Este formato tem suporte para *arrays* e *embedded objects* também como o JSON. Os usuários têm a possibilidade de realizar alterações em apenas alguns atributos de um documento sem precisar interagir com o restante da estrutura (Politowski; Maran, 2014).

De acordo com o site oficial do Mongodbcinc (2019), site oficial do MongoDB, afirma que a essência dele é um banco de dados distribuído, tendo integração com alta disponibilidade, escalabilidade horizontal e distribuição geográfica.

Segundo Higor (2014), a vantagem de se utilizar o MongoDB está na performance entregue através dos conceitos dos documentos, aonde as consultas retornam todas as informações necessárias sobre eles. Além disso, ele afirma que é mais simples fazer consultas, visto que não existem *joins* e transações, sendo mais fácil de ajustá-la e escrevê-las.

## 2.6. JSON Web Token (JWT)

JSON Web Token (JWT) tem a finalidade de transmitir informações de forma segura entre o cliente e o servidor usando um objeto JSON (Montanheiro *et al.*, 2017). É um padrão aberto definido pelo documento técnico (RFC 7519), afirmando que o JWT é compacto e representa declarações que servem para ambientes com restrições de espaço, como cabeçalhos de autorizações HTTP e parâmetros de consulta URI (Jones; Bradley; Sakimura, 2015).

As declarações são codificadas e transmitidas como um objeto JSON, usado como carga útil de uma estrutura JSON Web Signature (JWS) ou como um texto de estrutura JSON Web Encryption (JWE), assim as informações são assinadas digitalmente ou são protegidas pelo Código de Autenticação de Mensagens (MAC) e criptografadas (Jones; Bradley; Sakimura, 2015).

Este objeto JSON contém zero ou mais pares de nome/valor, onde nomes são *strings* e valores são valores JSON arbitrários, e esses pares são consideradas as declarações. Este objeto pode possuir espaços em branco ou quebra de linhas antes ou depois dos valores JSON ou caracteres estruturais, além disso, um JWT é uma representação em sequência de partes seguras

para URL separadas por um ponto final, cada parte tem seu valor codificado em base64url e o número de partes do JWT depende da representação do JWT resultante da serialização compacta do JWS ou da serialização compacta do JWE (Jones; Bradley; Sakimura, 2015).

Um dos benefícios por utilizar esta autenticação é por ele ser menos verboso que o XML, simplificando o tamanho do código deixando-o mais compacto e efetivo em ambientes que usam transmissões HTTP. O JWT é amplamente utilizado na internet pois os *tokens* são facilmente processados pela plataforma utilizada pelo cliente.

## 2.7. Firebase

O Firebase é um BaaS (*Backend as a Service*) da Google disponibilizado na web para facilitar o desenvolvimento de aplicações web e mobile, oferecendo uma infraestrutura e o *backend* de forma direta e rápida. Este serviço dispõe de funcionalidades como, armazenamento de arquivos, escalabilidade, serviço de notificações, hospedagem, autenticação de usuários, entre outros serviços (Andrade, 2020).

Um de seus serviços é o Firebase Storage, construído com base na infraestrutura do Google Cloud, podendo armazenar arquivos como PDFs, imagens, vídeos, entre outros tipos de conteúdos. Posteriormente disponibilizados para o cliente. Os SDKs Firebase para Cloud Storage entregam segurança e confiabilidade do Google no download e upload desses arquivos independente da qualidade de rede (GOOGLE, 2025).

## 2.8. Gemini

Lançado inicialmente com o nome de Bard pela Google, é uma *Large Language Model* (LLM) multimodal, ou seja, consegue entender imagens, áudios, textos e outros tipos de arquivos. O Gemini surgiu de uma pesquisa aprofundada sobre LLMs que se iniciou com o artigo Word2Vec em 2013, propondo novos modelos arquitetônicos de mapeamento de palavras como conceitos matemáticos, aliado a um modelo de conversação neural de 2015. Essa base provou como este modelo era capaz de prever frases com base em frases anteriores, trazendo uma experiência de conversação natural (GOOGLE, 2025).

Desde o lançamento do Gemini, usuários o utilizam para escrever e-mails mais envolventes, criar ideias, ajuda para aprender conceitos difíceis e para resolver problemas complexos (GOOGLE, 2025).

Atualmente, esta IA é versátil e útil pois pode ajudar em diversas situações, tanto para fins de produção, curiosidade ou criatividade (GOOGLE, 2025). Porém, existem alguns desafios que ainda estão presentes nesta ferramenta, por exemplo, a precisão da LLM pode ser ruim em alguns casos, especialmente em tópicos factuais ou complexos, ou até mesmo vieses nos dados de treinamento (GOOGLE, 2025).

## 2.9. Open Weather API

É um serviço *online* que fornece dados meteorológicos, como o clima atual e previsões, para desenvolvedores e estudantes implementarem em sistemas web ou mobile (Musah, 2022).



Sua fonte tem como base estações de radar, aeroportos e satélites meteorológicos, além de empresas renomadas como NOAA, Met Office, ECMWF, e Environment Canada. (Openweather, 2025).

Disponibilizam APIs gratuitas e ilimitadas para requisições de previsões climáticas, retornando dados no formato JSON, podendo trazer previsões de três em três horas até 5 dias (Musah, 2022).

Utilizam algoritmos de redes neurais convolucionais e tecnologias de aprendizado de máquina, que são responsáveis por calcular modelos de previsão sofisticados, como Previsão Numérica do Tempo (NWP). Além disso, o uso dessas tecnologias para processamento dos dados, aumenta a rapidez e precisão na montagem de modelos de previsão global, economizando em entregas de previsões para milhares de usuários (Openweather, 2025).

Os dados meteorológicos e históricos precisos ajudam empresas a tomarem decisões mais assertivas, por exemplo, o planejamento de rotas seguras por empresas de transporte e irrigação oportuna por parte de agricultores. Portanto, as aplicações de previsões estão se tornando rotina nas análises diárias das empresas (Openweather, 2025).

### **3. DESENVOLVIMENTO**

Neste capítulo, serão descritas as etapas de desenvolvimento do sistema de gerenciamento agrícola, como a modelagem dos dados, arquitetura e comunicação do sistema, implementação das funcionalidades principais, segurança e integração com as APIs externas.

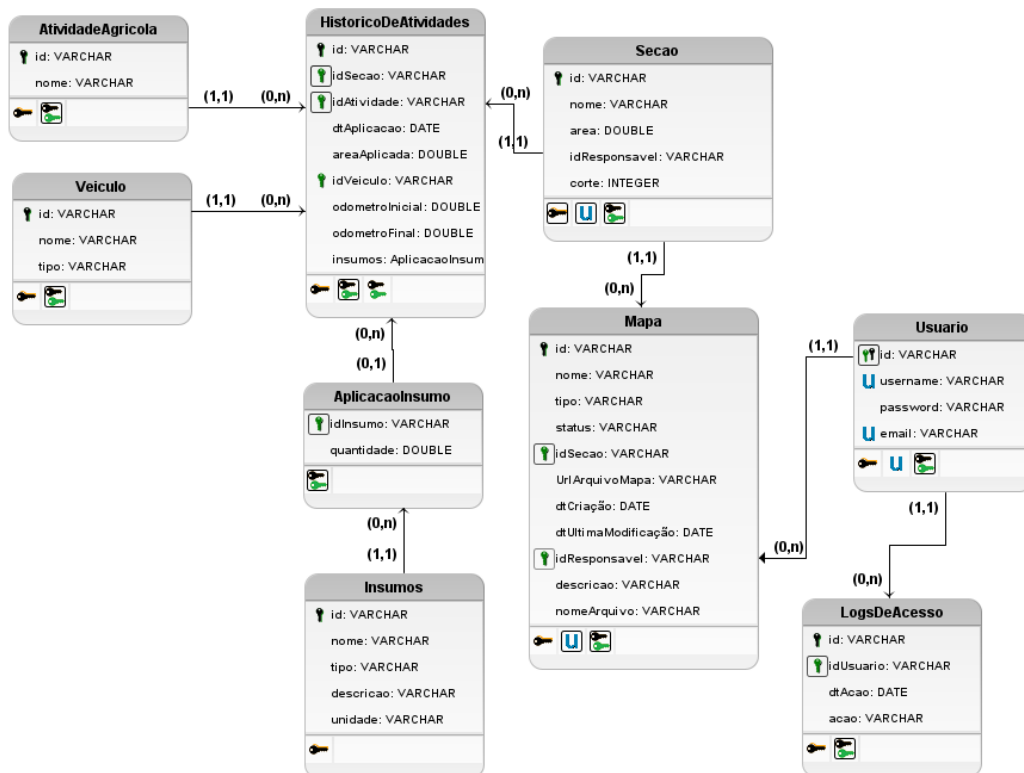
#### **3.1. Modelagem dos Dados**

O primeiro passo foi a criação do modelo de dados, desenvolvida visando atender aos requisitos da aplicação de gerenciamento agrícola, fundamental para a implementação da lógica do sistema garantindo os relacionamentos entre as entidades e suas cardinalidades.

##### **3.1.1. Diagrama de Entidade e Relacionamento (DER)**

A figura 1 representa o modelo lógico das entidades envolvidas no estudo.

Figura 1. Diagrama de Entidade e Relacionamento



Fonte: Autoria Própria

O diagrama apresenta as entidades do sistema (Seção, HistoricoDeAtividades, AtividadeAgricola, Veículo, Mapa, Insumos, Usuário e LogsDeAcesso) e suas cardinalidades. Por exemplo, um usuário pode estar associado à criação de várias seções, e uma seção pode estar associada a vários mapas. A entidade de LogsDeAcesso rastreia as requisições realizadas pelo usuário durante o uso do sistema para fins de auditoria.

A entidade de Role não foi implementada no sistema atualmente, mas será feita futuramente para separar funções e autorizações entre os usuários.

Através deste diagrama, foi possível desenvolver o *back-end*, criando as classes das entidades e suas relações.

### 3.2. Arquitetura e Comunicação do Sistema

Esta seção apresentará como a aplicação foi construída seguindo o modelo arquitetônico REST, explicando como as restrições conceituadas por essa arquitetura foram usadas no sistema.

#### 3.2.1. Client-Server

A separação entre o cliente e servidor são entendidas da seguinte forma:

- **Servidor (*back-end*):** Responsável por realizar as operações de CRUD no banco de dados, implementação da lógica de negócios dos dados e conexão com as APIs externas.

- Cliente (*front-end*): Exibe e manipula os dados através de uma interface usada pelo cliente.

Deste modo, ambas as partes podem evoluir independentemente.

### 3.2.2. Uniform Interface

Todos os recursos disponibilizados pela aplicação serão requisitados através do protocolo HTTP. Para acessar e interagir com os recursos é necessário um *endpoint*, ou Identificador Uniforme de Recurso (URI), que simboliza o endereço do recurso na rede. Portanto, qualquer recurso consegue ser manipulado através dessas URIs em conjunto com a padronização do tipo de comunicação.

Exemplificando, caso o usuário requirite ao meu sistema uma listagem de insumos que estão cadastrados, esta requisição usará o verbo HTTP GET em conjunto o *endpoint* /insumos.

Com todas as rotas dos meus serviços definidas, o quadro 2 apresenta suas devidas funções:

**Quadro 2.** Funções das Rotas de Serviços

Rota(s)	Função
/api-gerenciamen- to-agricola	É a URL base do sistema, onde será complementada pelas outras rotas.
/mapas, /insumos, /secoes, /veiculos, /historico e /atividades	Essas rotas possuem as funções de criar, buscar, atualizar e deletar(CRUD) seus respectivos objetos.
/insights	Possui comunicação com a API externa do Gemini, onde envia os dados de toda a aplicação para a IA com um prompt pré-definido, retornando <i>insights</i> estratégicos.
/temp	Requisita os dados climáticos da API do OpenWeather, a fim de receber informações sobre o tempo naquele momento e previsões futuras.
/autenticacao	Cadastrar ou autenticar um usuário através do seu <i>login</i> e senha.
/logs	É um monitoramento das ações dos usuários, contendo os logs dos endpoints acessados e usados por eles.

Fonte: Autoria Própria

### 3.2.3. Restrição Stateless

A estrutura da requisição é direta e contém todas as informações necessárias para que o servidor entenda o que é desejado e não precise guardar nenhum registro do estado anterior das requisições.

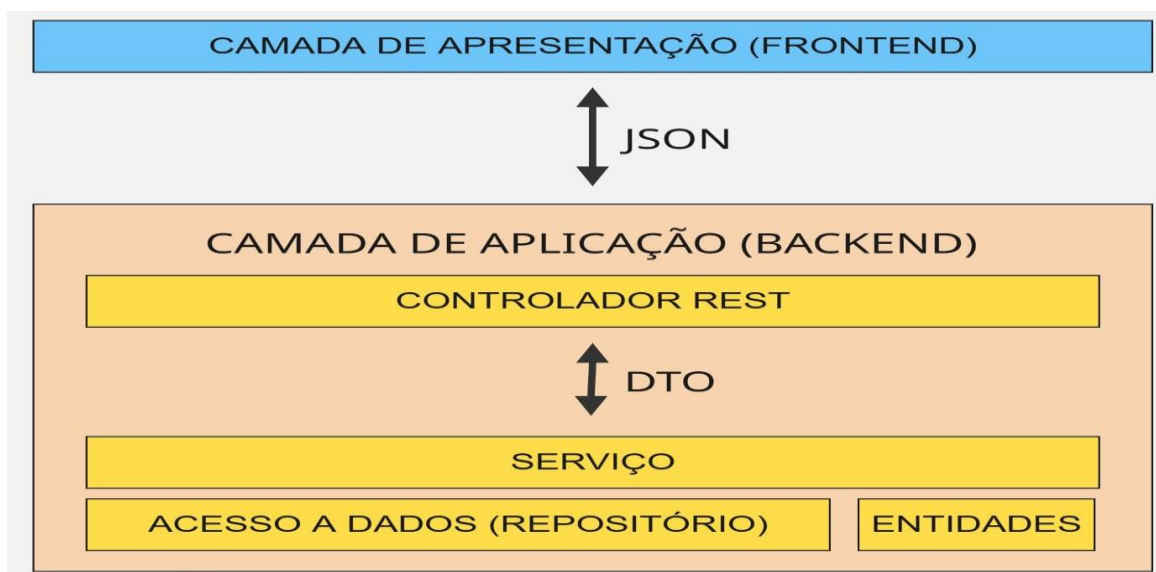
**Tabela 1.** Estrutura das Requisições

Elemento da Requisição	Função
Verbo HTTP	A ação a ser executada
Endpoint	Identificador do recurso a ser acessado
Corpo	Os dados enviados para serem criados ou atualizados
Token JWT	Cabeçalho da requisição onde prova a autenticidade do cliente

Fonte: A autoria Própria

### 3.2.4 Restrição Layered System

A estrutura das camadas lógicas do programa de gerenciamento agrícola é apresentada na figura 2:

**Figura 2.** Camadas Lógicas do Sistema de Gerenciamento Agrícola

Fonte: Adaptado de DevSuperior, 2021

Camada de Apresentação (Front-end), foi desenvolvida em React, responsável pelo ambiente de interação do usuário, onde se comunicará exclusivamente com o *back-end*, sem ter ciência dos serviços externos como do Gemini ou do OpenWeather.

Já a camada de Aplicação (*Back-end*), foi construída em Spring Boot. Esta camada possui outras camadas internas, cada uma com suas funções, são elas:

- Controlador REST: Por meio de métodos HTTP, se comunica e transporta dados do tipo JSON entre o *backend* e o *frontend*. Recebe as requisições do usuário e retorna uma resposta.
- Serviço: Lida com a lógica e regras de negócio do sistema, como validações, cálculos, conversões, entre outros.
- Repositório: Utiliza uma interface chamada *MongoRepository* que abstrai a comunicação direta com o MongoDB, facilitando o desenvolvimento do código.
- Entidades: São as classes que são mapeadas pelo repositório e implementadas no banco de dados.
- *Data Transfer Object (DTO)*: Não é uma camada, são classes usadas para trafegar apenas os dados necessários para a função designada, não possui lógica de negócios ou complexidade.

Este modelo deixa a camada de apresentação desacoplada dos serviços externos, ou seja, caso haja mudanças nos sistemas do OpenWeather ou do Gemini, as alterações serão feitas na camada de aplicação, não sendo necessário fazer manutenção no *front-end*. Além disso, o *back-end* pode ser reutilizado para integração de outras interfaces como um aplicativo de celular.

### 3.3. Principais Funcionalidades

Neste tópico, serão mostradas as principais funcionalidades implementadas, respeitando a arquitetura em camadas para a construção de um sistema robusto e funcional.

#### 3.3.1. CRUD das entidades

As primeiras funções implementadas foram de criar, atualizar, buscar e deletar (CRUD), das entidades mostradas no Diagrama de Entidade e Relacionamento, exceto Usuário e LogsDeAcesso.

No controlador REST, os *endpoints* dos objetos são mapeados pela anotação “@RequestMapping” e os verbos associados às funções pelas anotações “@PostMapping”, “@PutMapping”, “@DeleteMapping” e “@GetMapping”. Além disso, cada método usa a classe “ResponseEntity”, que oferece respostas HTTP completas, enviando ao usuário uma resposta precisa e clara.

Quando o método do controlador recebe uma requisição, é chamado então a classe de serviço, que irá aplicar as regras de negócios e lógica. No caso do CRUD, existem cinco funções básicas, são elas:













- Função de Salvar: Recebe o verbo POST e um objeto JSON para criar um novo documento.
- Função de Listar: Recebe o verbo GET e retorna uma lista de objetos.
- Função de ListarPorId: Recebe o verbo GET mais o ID do documento a ser listado.
- Função Atualizar: Recebe o verbo PUT, o ID do documento e um corpo JSON para atualizar um documento em específico.
- Função Deletar: Recebe o verbo DELETE mais o ID do documento a ser apagado.

Tanto na função de Atualizar como de ListarPorId existem verificações iniciais para ver se aquele documento em específico existe ou não, podendo retornar dois tipos de respostas, 200 para confirmação (“ok”) ou 404, significando que não foi encontrado (“not found”).

Em seguida, foram criadas as páginas de cada entidade no *frontend*, todas contendo o mesmo componente de tabela, para visualização das informações, filtradas ou não, e ações possíveis, como deletar, editar ou visualizar os dados restantes. O formulário é outro componente padrão nessas páginas, mudando apenas os *inputs* de cadastro dependendo de qual entidade será cadastrada. Cada página manipula os dados através dos *hooks* do *React*, como o *useState*, *useEffect* e *useMemo*.

As duas próximas figuras mostram a estrutura dos componentes de tabela e formulário.

**Figura 3.** Componente de Tabela exemplificado na Página de Mapas Agrícolas

Mapas Agrícolas						Total: 6
Nome	Seção	Última Modificação	Tipo	Status	Ações	
Todos	Todos	Todos	Todos	Todos		
T. Engenho	Engenho	Ainda não foi modificado	Mapa de Colheita	Ativo		
T. Girassol	Girassol	Ainda não foi modificado	Mapa de Colheita	Ativo		
T. Valencia Superior	Valência P/ Cima	Ainda não foi modificado	Mapa de Colheita	Ativo		
T. Casa 15	Casa do 15	Ainda não foi modificado	Mapa de Colheita	Ativo		
Sede Velha	Sede Velha	28/10/2025	Mapa de Colheita	Ativo		
T. Sler	Sler	Ainda não foi modificado	Mapa de Colheita	Ativo		

Ativar o Windows  
 Linhas por página: 10 1-6 de 6

Fonte: Autoria Própria

**Figura 4.** Componente de Formulário da Página de Mapas Agrícolas

The image shows a modal form titled "Cadastrar Insumo" (Register Input) with a close button (X) in the top right corner. The form contains the following fields:

- Nome:** A text input field.
- Tipo:** A dropdown menu.
- Unidade de Medida:** A dropdown menu.
- Descrição:** A text input field.

At the bottom right of the form, there are two buttons: "Cancelar" (Cancel) in a green box and "Salvar" (Save) in a blue box. The background of the page shows a search bar with the text "Pesquisar p" and a magnifying glass icon, and a table with a column labeled "Medida".

Fonte: Autoria Própria

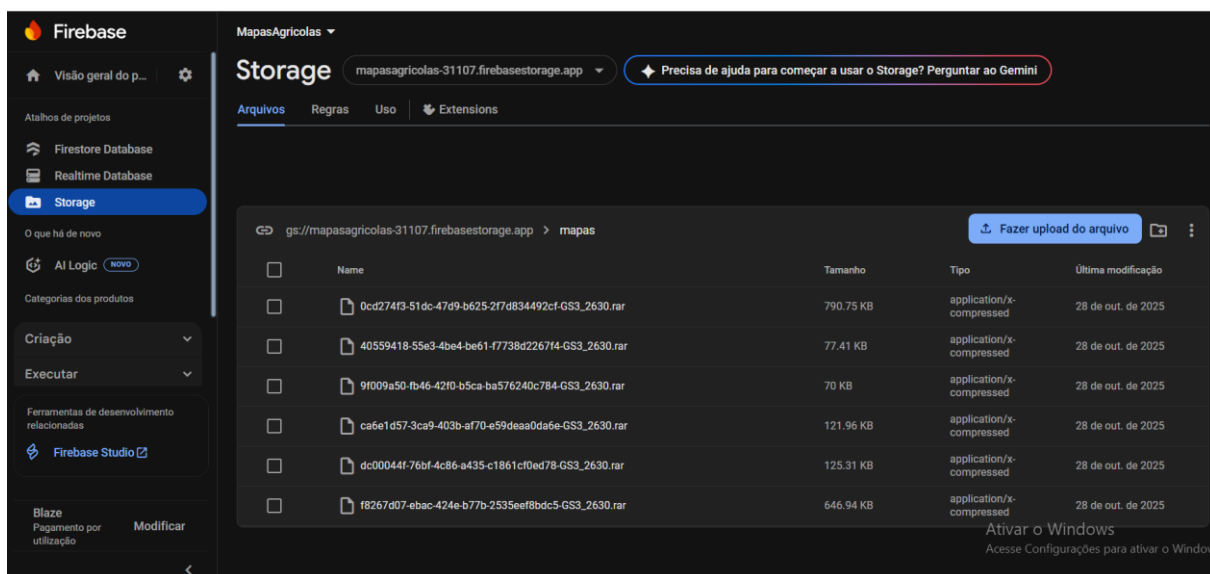
As funções utilizadas no *frontend* são definidas em uma classe de serviço, onde são feitas todas as conexões com as rotas do *backend* utilizando a biblioteca *Axios* para fazer requisições HTTP, além de uma função para adicionar o *token* JWT no cabeçalho da requisição.

### 3.3.2. *Download* e *Upload* de Arquivos no *Firebase Storage*

Esta funcionalidade é responsável pelo armazenamento dos arquivos dos mapas agrícolas em nuvem.

Inicialmente foi criado o que é chamado de *bucket*, que serve para armazenar os arquivos, e o *download* da chave da conta de serviço, que contém os dados sensíveis do meu projeto. O nome desse *bucket* e a chave são criadas no arquivo de configuração do projeto, para garantir a segurança dos dados.

**Figura 5.** Bucket do Firebase Responsável por Armazenar os Arquivos



Fonte: Autoria Própria

Posteriormente, uma classe de configuração do firebase foi criada para inicializar o ambiente de conexão com o bucket do firebase. A implementação da lógica está em uma classe de serviço, contendo as seguintes funções:

- `uploadFile`: Recebe um arquivo e o salva em uma subpasta (`/mapas`) dentro do bucket do firebase storage. O nome deste arquivo passa por um método que gera uma string aleatória para evitar conflitos de nomes.
- `generateDownloadUrl`: Recebe o nome do arquivo e busca ele dentro do bucket, depois gera um link de download deste arquivo que é válido por 10 minutos.
- `deleteFile`: Recebe o nome do arquivo e o deleta do bucket do firebase.

Algumas imagens representando o funcionamento do upload e download de arquivos.

A seguir serão apresentadas algumas figuras que representam o funcionamento desta funcionalidade.

A primeira delas representa o campo onde será selecionado o arquivo `.rar` ou `.zip`.



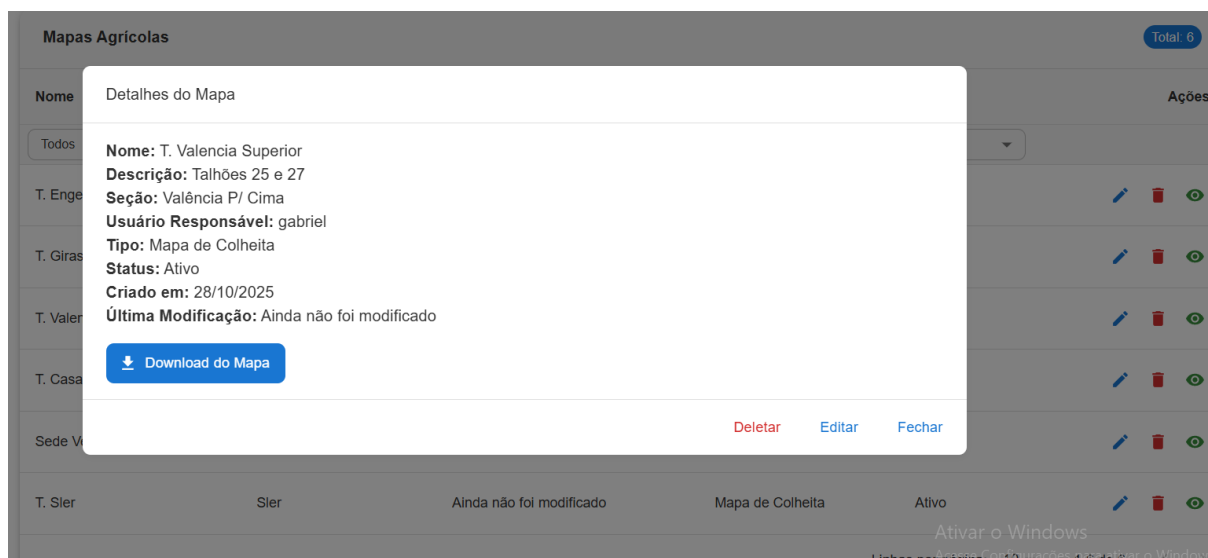
Figura 6. *Upload* do Arquivo feito através do Formulário de Cadastro

The image shows a modal window for uploading a map file. The modal has a title bar and a close button. Inside, there is a dropdown menu for 'Mapa de Colheita' with a downward arrow. Below it is a 'Status' dropdown menu with 'Ativo' selected. Underneath is a section titled 'Arquivo do Mapa' with a file input field containing the text 'Escolher arquivo' and the filename 'GS3\_2630.rar'. At the bottom of the modal are two buttons: 'Cancelar' (green) and 'Salvar' (blue). The background shows a table with columns for 'Nome', 'Status', 'Tipo', and 'Ações', with rows of map data.

Fonte: Autoria Própria

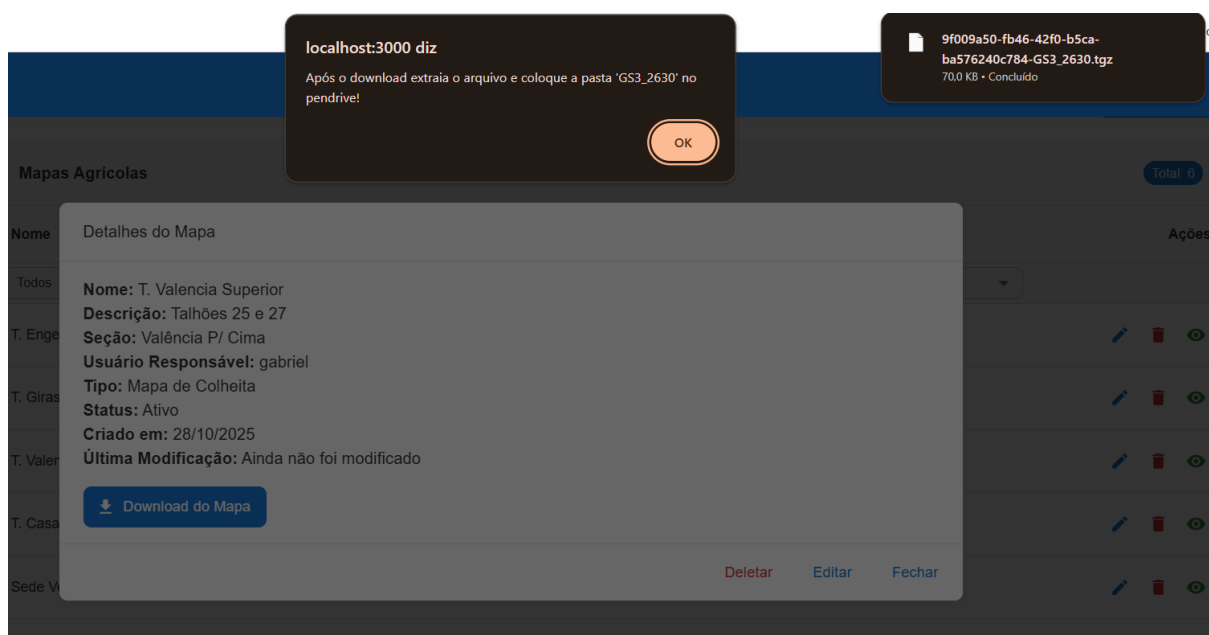
A segunda figura mostra onde está localizado o botão para *download* do arquivo, dentro dos detalhes dos mapas possui o botão para baixar seu respectivo arquivo para serem usados nas máquinas.

Figura 7. Botão de Download do Arquivo



Fonte: Autoria Própria

A última figura retrata a mensagem gerada após a finalização do *download*. O arquivo baixado é automaticamente extraído.

Figura 8. Mensagem de Sucesso e Instruções após o *Download* do Arquivo

Fonte: Autoria Própria

### 3.3.3. Previsão Climática com API do OpenWeather

A conexão com a API da OpenWeather necessitou da criação de uma conta de estudante no site deles para ter acesso a uma chave de permissão para sua API gratuita. Criando as variáveis de URL e chave privada no arquivo de configuração, foi possível estabelecer a comunicação com a API externa.

Esta funcionalidade possui apenas uma função chamada `getTemp`, que realiza a requisição para a API gratuita do OpenWeather passando o nome da cidade como parâmetro. É retornado então um JSON contendo todas as informações da previsão daquele local, com intervalo de 3 em 3 horas em um total de até 5 dias.

No *frontend*, não há a possibilidade de alterar o nome da cidade que será requisitada, pois a área rural onde esse sistema foi implementado está apenas na cidade de Nova Europa e para evitar erros de digitação no nome da cidade, foi decidido fazer a requisição já estruturada pelo sistema. Futuramente, com a expansão do sistema, será possível a escolha de outras cidades por meio de uma lista de seleções, visando também a minimização dos erros de digitação.

A exposição da previsão do tempo é acessada na aba de "Previsão do tempo" na lateral esquerda da interface. Clicando nela, a requisição acontecerá automaticamente e assim que carregada será exibida para o produtor.

**Figura 9.** Consulta da Previsão Climática



Fonte: Autoria Própria

### 3.3.4. Geração de *Insights* com API do Gemini

A geração de *insights* é feita pela API do Gemini, e para funcionar é preciso criar as variáveis da URL da API e uma chave de acesso para autenticação, ambos no arquivo de configuração.

Só há uma função na classe de serviço, chamada *getInsights*, que não recebe nenhum parâmetro. Ela transforma todos os dados do sistema, como insumos, veículos, seções etc., em um arquivo JSON que contém um prompt para dizer o que a IA deve fazer com esses dados. Portanto, é especificado neste prompt que a IA interprete os dados como um profissional agrícola e gere *insights* e estatísticas importantes para o produtor.

Na tela inicial do *frontend* está localizado o botão "Gerar *Insights*", que enviará a requisição. Uma mensagem de espera pela resposta será apresentada, pois a resposta pode demorar cerca de 20 segundos para aparecer na tela do usuário. O Gemini irá retornar a mensagem em JSON que será passada ao *frontend*, apresentada e salva em contexto para que não seja necessário realizar uma nova requisição toda vez que navegar para fora da tela inicial.

Figura 10. Insights Gerados pela Inteligência Artificial do Gemini



Fonte: Aatoria Própria

### 3.3.5. Implementação dos Logs

Os logs foram criados para fazer a auditoria do sistema. Uma classe de componente intercepta todas as requisições do sistema e coleta dados como nome do usuário, URL requisitada e horário e data da requisição. Esses dados coletados geram um novo log e são salvos no banco de dados.

Por enquanto todos podem ver os logs acessando a aba “Logs” na lateral de navegação da interface, porém futuramente apenas o usuário que possuir a permissão de administrador poderá vê-los.

Está figura apresenta a tabela dos logs do sistema.

Figura 11. Tabela dos Logs do Sistema

Logs do Sistema Pesquisar...  Total: 14

Registro de Ações <span style="float: right;">Total: 14</span>		
Usuário	Ação	Data e Hora
Todos	Todos	Todos
gabriel	GET /api-gerenciaimento-agricola/logs	02/11/2025, 12:57:53
gabriel	GET /api-gerenciaimento-agricola/logs	02/11/2025, 12:57:53
gabriel	GET /api-gerenciaimento-agricola/Insumos	02/11/2025, 12:51:07
gabriel	GET /api-gerenciaimento-agricola/Insumos	02/11/2025, 12:51:07
anonymousUser	GET /api-gerenciaimento-agricola/mapas/download/690108e79e9e593fa3454816	02/11/2025, 12:26:06
gabriel	GET /api-gerenciaimento-agricola/mapas	02/11/2025, 12:13:31
gabriel	GET /api-gerenciaimento-agricola/secoes	02/11/2025, 12:13:31

Fonte: Aatoria Própria

### 3.4. Autenticação e Segurança

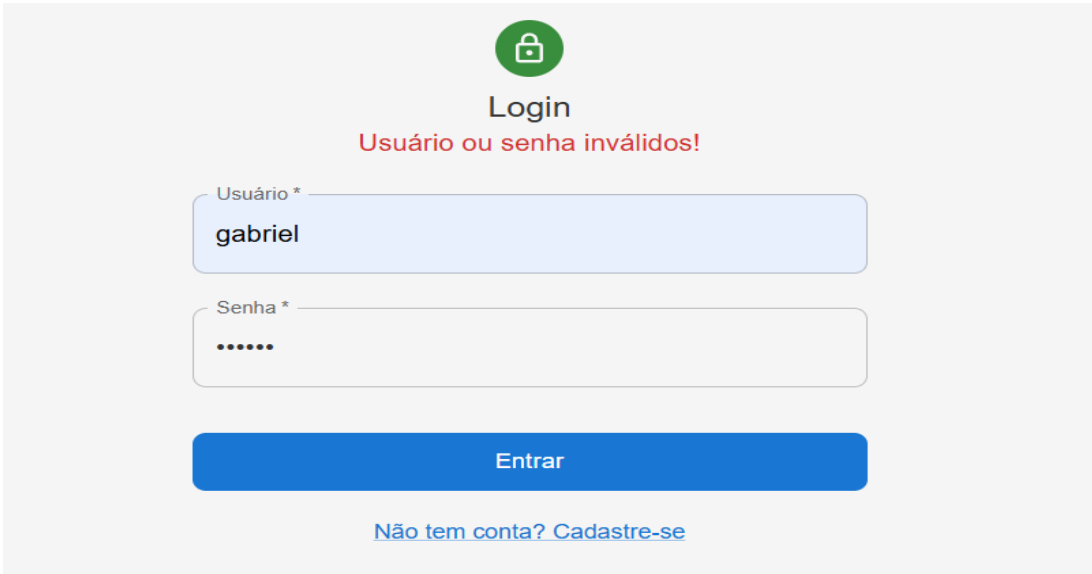
Para assegurar a segurança e autenticidade do sistema, foi utilizado o padrão aberto JWT e validações.

O fluxo começa na tela de cadastro, onde possuem validações de criação de conta como, tamanho mínimo de senha (8 dígitos), confirmação de senha, e-mail já cadastrado ou digitado incorretamente e campos nulos. Posteriormente, o usuário é redirecionado à página de *login*, onde ele acessa o sistema inserindo seu nome de usuário cadastrado e senha cadastrada. Feito o *login*, seu usuário receberá um *token* JWT que será armazenado em *localStorage*.

Todas as ações realizadas por quem está usufruindo do sistema, terão que enviar o *token* JWT em toda requisição, e chegando no servidor ela passa por um filtro de verificação do *token*, caso ele esteja correto o fluxo da requisição continua normalmente, porém, caso esteja incorreto, adulterado ou nulo, o servidor retornará uma resposta com código 401, que significa “*unauthorized*”, ou não autorizado.

A seguir está uma imagem para ilustrar a página de *login* do sistema, aproveitando para mostrar a mensagem de login incorreta.

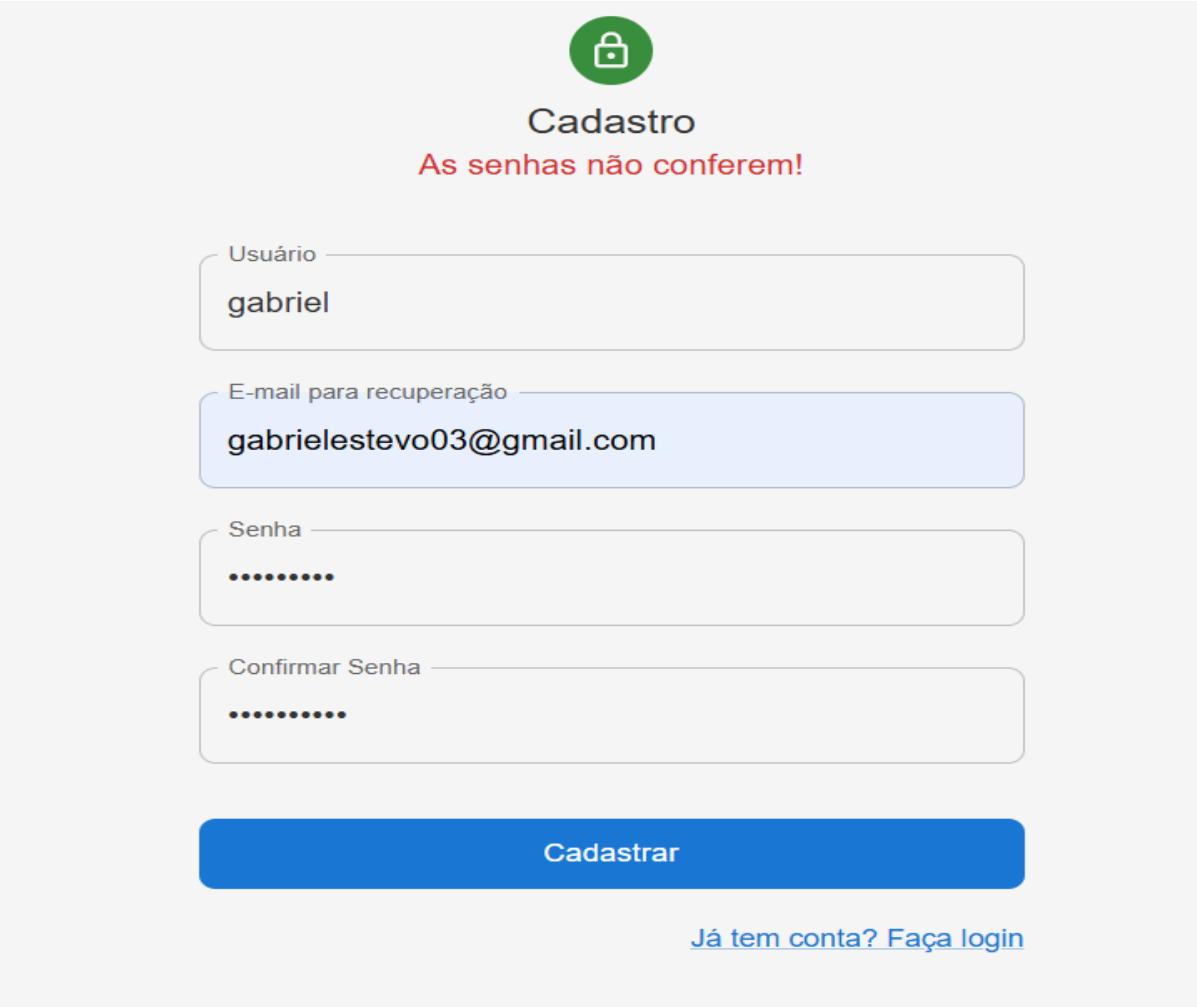
**Figura 12.** Tela de *Login* com Mensagem de Validação



A imagem mostra a interface de login de um sistema web. No topo, há um ícone de cadeado verde dentro de um círculo verde. Abaixo dele, o título "Login" é exibido em uma fonte preta. Logo abaixo, uma mensagem de erro "Usuário ou senha inválidos!" aparece em uma cor vermelha. O formulário de login contém dois campos de entrada: "Usuário \*" com o texto "gabriel" e "Senha \*" com pontos cinza para ocultar o conteúdo. Abaixo dos campos, há um botão azul com o texto "Entrar". Na base do formulário, há um link azul que diz "Não tem conta? Cadastre-se".

Fonte: Autoria Própria

A próxima imagem ilustra a página de cadastro com uma alerta de validação sobre as senhas diferentes.

**Figura 13.** Tela de Cadastro com Mensagem de Validação sobre Senhas

A imagem mostra uma interface de usuário para o processo de cadastro. No topo, há um ícone de cadeado verde dentro de um círculo verde. Abaixo dele, o título "Cadastro" é exibido em uma fonte preta, seguido por uma mensagem de erro em vermelho: "As senhas não conferem!".

Abaixo da mensagem, há quatro campos de entrada de texto:

- O primeiro campo, rotulado "Usuário", contém o texto "gabriel".
- O segundo campo, rotulado "E-mail para recuperação", contém o texto "gabrielestevo03@gmail.com".
- O terceiro campo, rotulado "Senha", contém sete pontos pretos para ocultar o texto.
- O quarto campo, rotulado "Confirmar Senha", também contém sete pontos pretos.

Na base da tela, há um botão azul arredondado com o texto "Cadastrar" em branco. Abaixo do botão, há um link azul que diz "Já tem conta? Faça login".

Fonte: Autoria Própria

#### 4. CONSIDERAÇÕES

A realização do projeto provou que a implementação da tecnologia na propriedade rural não precisa ser necessariamente onerosa. A adoção de ferramentas sem despesas contribuída de uma arquitetura estruturada e boas práticas consegue inovar o campo, tornando os serviços de agricultura mais sustentáveis e precisos.

A integração com sistemas externos de inteligência artificial e previsão do tempo se mostraram de extrema importância na hora de decidir quais ações devem ser realizadas e sugerir ideias que complementem a visão do produtor rural.

Existem alterações futuras significativas para que este sistema possa evoluir ainda mais, como a definição de autorizações de usuários e a elaboração de rastreadores para mapeamento exato dos rastros das máquinas, possibilitando o monitoramento das operações e auditoria dos processos realizados.

Portanto, com os avanços tecnológicos atuais é relevante que os produtores rurais menores consigam acompanhar essa evolução, e com o uso de *software* que utiliza tecnologias gratuitas, é possível que ele não fique para trás e consiga seguir práticas mais sustentáveis e inovadoras.

## REFERÊNCIAS

AGGARWAL, S. *et al.* Modern web-development using reactjs. **International Journal of Recent Research Aspects**, v. 5, n. 1, p. 133-137, 2018. Disponível em: <https://ijrra.net/Vol5issue1/IJRR-05-01-27.pdf>. Acesso em: 28 set. 2025.

ANDRADE, A. P. O **que é Firebase?** [S. l.: s. n.], 2020. Disponível em: <https://www.treinaweb.com.br/blog/o-que-e-firebase>. Acesso em: 10 out. de 2025.

AZZOLINI, M. G. **Derivação de um estilo arquitetural para o front-end de sistemas baseados em react. js com base em projetos open source.** [S. l.: s. n.], 2021. Disponível em: <https://lume.ufrgs.br/bitstream/handle/10183/231869/001133554.pdf?sequence=1>. Acesso em: 26 set. 2025.

BASSOI, L. H. *et al.* Agricultura de precisão e agricultura digital. **TECCOGS: Revista Digital de Tecnologias Cognitivas**, n. 20, 2019. Disponível em: <https://revistas.pucsp.br/teccogs/article/view/48542>. Acesso em: 30 mar. 2025

BRAGA, R.; PINTO, P. A. Alterações climáticas e agricultura. **Inov. Tecnol. Form. Agríc**, v. 12, p. 55, 2009. Disponível em: [https://www.researchgate.net/profile/Pedro-Pinto-42/publication/255703923\\_Alteracoes\\_climaticas\\_e\\_Agricultura/links/00463520420b946bb600000/Alteracoes-climaticas-e-Agricultura.pdf](https://www.researchgate.net/profile/Pedro-Pinto-42/publication/255703923_Alteracoes_climaticas_e_Agricultura/links/00463520420b946bb600000/Alteracoes-climaticas-e-Agricultura.pdf). Acesso em: 30 mar. 2025

CALÇA, J. V. J. **Análise comparativa entre os frameworks Django e Spring Boot.** [S. l.: s. n.], 2022. Disponível em: [http://riccps.eastus2.cloudapp.azure.com/bitstream/123456789/24770/1/informaticanegocios\\_2022\\_2\\_joavictorjustocalca\\_analisecomparativaentreosframeworksdjanquesprin.pdf](http://riccps.eastus2.cloudapp.azure.com/bitstream/123456789/24770/1/informaticanegocios_2022_2_joavictorjustocalca_analisecomparativaentreosframeworksdjanquesprin.pdf). Acesso em: 20 out. 2025.

CUNHA, M.; BRAGA, R. Agricultura de precisão e sustentabilidade. **INESC TEC Science & Society**, v. 1, n. 4, 2022. Disponível em: <https://sciencesociety.inesctec.pt/pt/index.php/inesctecesciedade/article/view/90/185>. Acesso em: 30 mar. 2025.

FERRAZ, C. O.; PINTO, W. F. Tecnologia da Informação para a agropecuária: utilização de ferramentas da tecnologia da informação no apoio a tomada de decisões em pequenas propriedades. **Revista Eletrônica Competências Digitais para Agricultura Familiar**, v. 3, n. 1, p. 38-49, 2017. Disponível em: <https://owl.tupa.unesp.br/recodaf/index.php/recodaf/article/view/48>. Acesso em: 5 nov. 2025.

FIELDING, R. T. **Architectural styles and the design of network-based software architectures.** Irvine: University of California, 2000. Disponível em: [https://www.epai-ict.ch/nexus/repository/course-material/m133/fielding\\_dissertation.pdf](https://www.epai-ict.ch/nexus/repository/course-material/m133/fielding_dissertation.pdf). Acesso em: 22 set. 2025

GOOGLE. **Cloud Storage for Firebase.** [S. l.]: Google, s. d. Disponível em: <https://firebase.google.com/docs/storage?hl=pt-br>. Acesso em: 10 out. 2025.

GOOGLE. **Firebase Storage.** Firebase, [S. l.]: Google, 2023. Disponível em: <https://firebase.google.com/products/storage>. Acesso em: 10 jun. 2025.

GOOGLE. **What is Gemini and how it works.** [S. l.]: Google, 2025. Disponível em: <https://gemini.google/overview/?hl=pt-BR>. Acesso em: 12 out. 2025.

INTERNATIONAL SOCIETY OF PRECISION AGRICULTURE. **What is Precision Agriculture?** [S. l. s. n.], 2023. Disponível em: <https://www.ispag.org/resources/definition>. Acesso em: 10 jun. 2025.

JONES, M.; BRADLEY, J.; SAKIMURA, N. JSON Web Token (JWT). RFC 7519. **Internet Engineering Task Force (IETF)**, May 2015. Disponível em: <https://datatracker.ietf.org/doc/html/rfc7519>. Acesso em: 7 out. 2025.

JWT. **JSON Web Token Introduction**. [S. l.]: JWT, 2017. Disponível em: <https://www.jwt.io/introduction>. Acesso em: 7 out. 2025.

KHUAT, T. **Developing a frontend application using ReactJS and Redux**. Dissertação (Degree Programme in Business Information Technology Bachelor's) — Laurea University of Applied Sciences, Leppävaara, 2018. Disponível em: [https://www.theseus.fi/bitstream/handle/10024/150837/Tung\\_Khuat\\_1301747\\_Thesis.pdf?sequence=1](https://www.theseus.fi/bitstream/handle/10024/150837/Tung_Khuat_1301747_Thesis.pdf?sequence=1). Acesso em: 26 set. 2025.

LAMPARELLI, R. A. C. **Agricultura de Precisão**. [S. l.: s. n.], 2010. Disponível em: <https://www.embrapa.br/agencia-de-informacao-tecnologica/cultivos/cana/producao/avanco-tecnologico/agricultura-de-precisao>. Acesso em: 30 mar. 2025.

MEDEIROS, H. M. Introdução ao MongoDB. **DevMedia**, 2014, Disponível em: <https://www.devmedia.com.br/introducao-ao-mongodb/30792>. Acesso em: 03 out. 2025.

MONGODB INC. **What Is MongoDB**. [S. l.: s. n.], 2019. Disponível em: <https://www.mongodb.com/what-is-mongodb>. Acesso em: 03 out. 2025.

MONTANHEIRO, L. S.; CARVALHO, A. M. M.; RODRIGUES, J. A. Utilização de json web token na autenticação de usuários em apis rest. In: **XIII Encontro Anual de Computação. Anais do XIII Encontro Anual de Computação EnAComp**, p. 186-193, 2017. Disponível em: [https://www.enacomp.com.br/2017/docs/json\\_web\\_token\\_api\\_rest.pdf](https://www.enacomp.com.br/2017/docs/json_web_token_api_rest.pdf). Acesso em: 7 out. 2025.

MUSAH, A. *et al.* An evaluation of the OpenWeatherMap API versus INMET using weather data from two Brazilian cities: Recife and Campina Grande. **Data**, v. 7, n. 8, p. 106, 2022. Disponível em: <https://www.mdpi.com/2306-5729/7/8/106#>. Acesso em: 21 out. 2025.

OPENWEATHER. **Technology**. [S. l.: s. n.], 2025. Disponível em: <https://openweathermap.org/technology>. Acesso em: 21 out. 2025.

PADRÃO em camadas: como usar no Spring REST. Gravado por Nélio Alves. [s. l.: s. n.], 30 mar. 2021. 1 vídeo (83 min). Publicado pelo canal DevSuperior. Disponível em: [https://www.youtube.com/live/ZaNVBhZUFIq?si=oe\\_npYeuzRfSkUkk](https://www.youtube.com/live/ZaNVBhZUFIq?si=oe_npYeuzRfSkUkk). Acesso em: 20 set. 2025.

POLITOWSKI, C.; MARAN, V. Comparação de performance entre postgresql e mongodb. **Escola Regional de Banco de Dados**, Sao Francisco do Sul, 2014. Disponível em: [https://turing.pro.br/anais/ERBD-2014/artigos\\_aceitos/trilha\\_pesquisa/124500\\_1.pdf](https://turing.pro.br/anais/ERBD-2014/artigos_aceitos/trilha_pesquisa/124500_1.pdf). Acesso em: 03 out. 2025.

SILVA, J. M. P.; CAVICHIOLI, F. A. O uso da agricultura 4.0 como perspectiva do aumento da produtividade no campo. **Revista Interface Tecnológica**, v. 17, n. 2, p. 617-618, 2020. Disponível em: [https://revista.fatectq.edu.br/interfacetecnologica/pt\\_BR/article/view/1068](https://revista.fatectq.edu.br/interfacetecnologica/pt_BR/article/view/1068). Acesso em: 20 set. 2025.

SOUZA, E. C.; DE OLIVEIRA, M. R. Comparativo entre os bancos de dados MySQL e MongoDB: quando o MongoDB é indicado para o desenvolvimento de uma aplicação. **Revista Interface Tecnológica**, v. 16, n. 2, p. 8-9, 2019. Disponível em: [https://revista.fatectq.edu.br/interfacetecnologica/pt\\_BR/article/view/664](https://revista.fatectq.edu.br/interfacetecnologica/pt_BR/article/view/664). Acesso em: 19 jun. 2025.



WEISSMANN, H. L. **Spring Boot: Simplificando Spring**. [S. l.: s. n.], 2015. Disponível em: <https://www.devmedia.com.br/spring-boot-simplificando-spring-revista-java-magazine-135/31979>. Acesso em: 26 set. 2025.