UNIVERSIDADE FEDERAL DE UBERLÂNDIA
INSTITUTO DE GEOGRAFIA - IG
CURSO DE ENGENHARIA DE AGRIMENSURA E CARTOGRÁFICA
CAMPUS MONTE CARMELO

MATHEUS ALBUQUERQUE DE MELO

DESENVOLVIMENTO DE UM PLUGIN NO QGIS PARA TRATAMENTO DE DADOS DISPONIBILIZADOS PELO INMET

MONTE CARMELO
2024

MATHEUS ALBUQUERQUE DE MELO

DESENVOLVIMENTO DE UM PLUGIN NO QGIS PARA TRATAMENTO DE DADOS DISPONIBILIZADOS PELO INMET

Trabalho de Conclusão de Curso apresentado ao Instituto de Geografia da Universidade Federal de Uberlândia como requisito para obtenção do título de bacharel em Engenharia de Agrimensura e Cartográfica.

Orientadora: Profª Dra. Ana Cláudia Martinez

MONTE CARMELO

2024

MATHEUS ALBUQUERQUE DE MELO

DESENVOLVIMENTO DE UM PLUGIN NO QGIS PARA TRATAMENTO DE DADOS DISPONIBILIZADOS PELO INMET

Trabalho de Conclusão de Curso apresentado ao Instituto de Geografia da Universidade Federal de Uberlândia como requisito para obtenção do título de bacharel em Engenharia de Agrimensura e Cartográfica.

Aprovado em: _____/_____/_____

Banca Examinadora:

_____

Profª. Dra. Ana Cláudia Martinez (Orientadora)
Universidade Federal de Uberlândia (UFU)

_____

Profª. Dra. Luziane Ribeiro Indjai (Coorientadora)
Universidade Federal de Uberlândia (IG/UFU)

_____

Engª. Andréia Ferreira Soares
Universidade Federal de Uberlândia (UFU)

RESUMO

Os bancos de dados armazenam informações sobre diversas variáveis climáticas, como radiação, temperatura, umidade, pressão atmosférica e precipitação. A coleta, processamento e compartilhamento dessas informações são realizadas por meio de software específicos, tais dados são obtidos por estações convencionais ou automáticas, além disso, é possível obter dados por meio dos sensores orbitais ou por sistema de radar. Com uma vasta gama de dados torna-se possível a elaboração de isolinhas, gráficos e mapas. Esses sistemas dão às pessoas capacidade de rastrear a localização e o movimento de objetos no espaço em volta da terra. Esses dados podem ser processados para criar superfícies contínuas que exibem variáveis como temperatura, pressão ou altitude. O banco de dados do Instituto Nacional de Meteorologia (INMET) que armazena informações meteorológicas, é atualizado a cada hora. No entanto, para análise de dados com datas anterior aos últimos 6 meses ou para análise temporal, o excesso de dados torna-se exorbitante para cada estação. Mediante isso, a elaboração de um complemento no software QGIS tem como objetivo trabalhar com as estações meteorológicas obtidas no banco de dados do INMET, com o intuito de otimizar tempo, retornando os dados de acordo com o período e gerando um arquivo shapefile georreferenciado para cada estação.

Palavras-chave: Plugin, INMET, QGIS

ABSTRACT

Databases store information about various climatic variables, such as radiation, temperature, humidity, atmospheric pressure, and precipitation. The collection, processing, and sharing of this information are carried out through specific software, with data obtained from conventional or automatic stations, as well as through orbital sensors or radar systems. With a vast array of data, it becomes possible to create isolines, graphs, and maps. These systems provide people with the ability to track the location and movement of objects in the space around the Earth. These data can be processed to create continuous surfaces displaying variables such as temperature, pressure, or altitude. The database of the National Institute of Meteorology (INMET), which stores meteorological information, is updated every hour. However, for data analysis dating back more than the last 6 months or for temporal analysis, the excess of data becomes overwhelming for each station. Therefore, the development of an add-on in QGIS aims to work with meteorological stations obtained from the INMET database, with the aim of optimizing time, retrieving data according to the period, and generating a georeferenced shapefile for each station.

Keywords: Plugin, INMET, QGIS

LISTA DE FIGURAS

LISTA DE MAPAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

A compreensão dos dados meteorológicos desempenha um papel fundamental na agricultura, pois as condições climáticas têm um impacto direto no desenvolvimento das culturas e na produtividade agrícola (Oliveira; Assunção, 2009). Dessa forma, a utilização de um calendário agrícola adequado, que leva em consideração os parâmetros climáticos da zona de produção, torna-se vital para o planejamento das atividades agrícolas, resultando em uma produtividade mais eficiente. Um planejamento eficaz melhora imediatamente a produção agrícola, proporcionando aos agricultores condições mais favoráveis.

A agricultura é altamente dependente das condições climáticas em comparação com outros setores econômicos (Hamada; Ghini; Gonçalves, 2006). Devido ao seu alto risco decorrente dessa dependência, é fundamental identificar e mapear as áreas ideais para o cultivo, levando em consideração fatores como clima e precipitação, para reduzir esses riscos e evitar a perda de safras. Dessa forma, os agricultores podem então decidir o que plantar e usar programas apropriados para a área, aumentando assim a produtividade agrícola e auxiliando os agricultores a enfrentar de forma mais eficaz os desafios climáticos. Além disso, torna-se essencial investir em sistemas de monitoramento climático e previsão do tempo (Maciel; Azevedo; Andrade Júnior, 2009).

Dentre os fatores climáticos que influenciam diretamente a agricultura, podemos citar a temperatura, umidade relativa do ar, velocidade e direção dos ventos, radiação solar e até nebulosidade (Oliveira; Assunção, 2009). Para isso, O Instituto Nacional de Meteorologia (INMET) criou o Sistema de Suporte à Decisão na Agropecuária (SISDAGRO), com o intuito de auxiliar usuários no planejamento e gestão agrícola para tomadas de decisão mais assertivas.

Embora os fatores climáticos contribuam significativamente na área agrícola, o seu uso não se limita apenas a isso, já que grande parte do estudo e análise de risco ambiental necessitam de dados meteorológicos para o mapeamento de risco de incêndio florestal, bem como auxiliar na prevenção de novos riscos. Esses índices são adaptados de acordo com a área de interesse, sendo geralmente utilizado por equipes de combate a incêndios, principalmente para as empresas florestais (Vettorazzi; Ferraz, 1998).

Para um planejamento eficaz relacionado aos dados climáticos, é essencial utilizar um banco de dados que abranja diversas variáveis em um espaço temporal predefinido. A partir desses dados, torna-se possível realizar análises e obter sustentação para a elaboração de mapas climáticos e avaliação de riscos em áreas de estudo (Oliveira; Assunção, 2009). Melhorando assim os métodos de detecção e monitoramento de incêndios florestais, e contribuindo para um

melhor planejamento do controle, bem como para o avaliar os efeitos produzidos pelo fogo sobre o ambiente (Batista, 2004).

Sendo assim, softwares de Sistema de Informação Geográfica (SIG) são amplamente utilizados para elaboração de mapas (Vettorazzi; Ferraz, 1998). O ambiente computacional permite analisar dados de forma integrada, como intuito de soluções rápidas e precisas, por abranger diversas ferramentas e aplicações distintas de acordo com o usuário.

Entre os softwares relacionados, o *Quantum Geographic Information System* (QGIS) é um software SIG que oferece todas as condições necessárias para análise de fenômenos espaciais. Além disso, graças ao seu design intuitivo e com diversas funcionalidades, o software vem conquistando a popularidade entre os usuários (Pejovic et al, 2014). Entre suas vantagens, o seu maior benefício está no seu acesso gratuito, já que o software é mantido atualmente por grupos de voluntários, que frequentemente atribuem atualizações e correções de bugs.

De forma similar a outros softwares SIG, o QGIS permite aos usuários elaborem mapas com uso de diversas camadas, bem como a utilização de múltiplas projeções cartográficas (Pejovic et al, 2014). O mapa pode ser usado para diversas finalidades e salvo em diferentes formatos, seja no formato raster ou vetor. Além disso, os dados vetoriais podem ser representados de diversas formas, entre elas de um ponto, polilinha ou polígono, enquanto os dados de raster precisam ser georreferenciados primeiro e salvos mais tarde em diferentes formatos (Trevisan; Molin, 2014).

Outros aspectos positivos no software é a questão do código aberto, além disso, o QGIS permite a integração de inúmeros plugins, uma vez que ele foi projetado com esta finalidade, facilitando assim a implementação de novas funções ou ferramentas (Pejovic et al, 2014). Vale ressaltar que inúmeros plugins já foram incrementados no software, além disso, possui vários arquivos online em constante expansão, possibilitando assim novos plugins inseridos por programadores profissionais (Pejovic et al, 2014).

Dessa forma, este trabalho tem como objetivo a elaboração de um plugin no software QGIS para o tratamento dos dados meteorológicos disponibilizados pelo INMET. Considerando que a análise de cada estação meteorológica gera uma quantidade significativa de valores para cada hora do período considerado, faz-se necessário otimizar o serviço e garantir maior acurácia na verificação dos dados presentes em cada estação, levando em conta o intervalo de tempo desejado. Além disso, busca-se eliminar dados ausentes ou estações que não apresentem informações para o período pré-estabelecido, garantindo a qualidade e completude dos dados utilizados na análise climática.

1.1 OBJETIVO GERAL

O objetivo é otimizar a análise temporal e espacial dos dados meteorológicos fornecidos pelo Instituto Nacional de Meteorologia (INMET) por meio do desenvolvimento de um plugin para o software QGIS.

1.2 OBJETIVOS ESPECÍFICOS

Para alcançar o objetivo geral, serão estabelecidos os seguintes objetivos específicos:

- Estudar e compreender o funcionamento do software QGIS e a sua estrutura de plugins.
- Analisar o formato dos dados disponibilizados pelo INMET.
- Desenvolver um plugin que permita a seleção dos dados meteorológicos de interesse e a geração de arquivos shapefile georreferenciados para cada estação.
- Testar e validar o plugin desenvolvido em diferentes cenários.

1.3 JUSTIFICATIVA

A agrimensura é uma área do conhecimento que utiliza dados espaciais para o planejamento, monitoramento e execução de projetos relacionados à gestão territorial, ambiental e agrícola. Nesse contexto, a utilização de dados meteorológicos é essencial para a tomada de decisões mais precisas e assertivas. O INMET é um dos principais fornecedores de dados meteorológicos no Brasil, disponibilizando informações coletadas por estações convencionais e automáticas em todo o território nacional. No entanto, a análise temporal e espacial desses dados pode ser um desafio, principalmente para pesquisadores e profissionais que trabalham com grandes volumes de informações. Nesse sentido, o desenvolvimento de um plugin para o software QGIS contribui para uma melhor tomada de decisões e planejamento de projetos, permitindo uma análise mais eficiente e confiável dos dados coletados pelo INMET.

2 REFERENCIAL TEÓRICO

2.1 ESTAÇÕES AUTOMÁTICAS E CONVENCIONAIS

Segundo Tanner (1990), embora o uso de estações meteorológicas convencionais seja amplamente utilizado, o avanço tecnológico tornou possível a criação de estações automáticas que coletam continuamente dados sem intervenção humana. Essas estações automatizadas são consideradas mais precisas e confiáveis porque eliminam o erro humano na coleta de dados e são capazes de detectar mudanças sutis nas condições climáticas.

Tanner (1990) também observou que as estações meteorológicas tradicionais são instrumentos manuais que requerem observadores treinados para fazer leituras regulares a cada uma a três horas. Embora precisos, seu uso pode ser limitado devido ao tempo e esforço necessários para coletar e registrar os dados.

Pereira (2008) por sua vez, acrescenta que as estações convencionais e automáticas adquirem resultados significativos e semelhantes, enfatizando que no presente momento não seria viável a substituição das estações convencionais pelas automatizadas no Brasil, pois o custo de manutenção seria alto e a precisão não mudaria muito.

A capacidade de coletar dados continuamente e em tempo real fornecida por estações automatizadas permite uma análise mais precisa e detalhada das condições meteorológicas, bem como a fácil integração dos dados coletados em sistemas de informações geográficas (GIS) e modelos numéricos de previsão do tempo (PNM), permitindo uma análise mais abrangente e integrada das condições climáticas (Tanner, 1990).

2.2 PLUGIN

O potencial dos plugins é destacado por Dietrich et al. (2007) em seu artigo como uma estratégia promissora para desenvolver sistemas complexos que exigem reconfigurações em tempo de execução. Em sua análise, os plugins são simples de serem implementados e por isso existem competições entre vários plugins que oferecem funcionalidades comparáveis.

Papajorgji (2004), conclui a pesquisa dizendo que o uso de complementos é fundamental para otimização de serviço, ajudando economicamente e atribuindo esforços a outros fatores de pesquisa. Por fim, justifica dizendo que a praticidade de utilizar um plugin e como ele auxilia outros usuários, o desenvolvimento de tais componentes tende a incentivar futuras criações e melhorias de modelos para melhor contribuir com os resultados desejados.

Para Garrido et al (2019), o uso de um complemento nos softwares amplia determinados recursos, tornando assim o software cada vez mais completo. Ampliando opções de customização na interface do software, além de permitir melhores acessibilidades e outras funcionalidades de acordo com o que o usuário solicitar.

2.3 SISTEMA DE INFORMAÇÃO GEOGRÁFICA

Segundo Rosa (2013), o geoprocessamento é um conjunto de tecnologias e técnicas que permitem coletar, armazenar, processar, analisar e visualizar dados geográficos. Esses dados podem incluir informações sobre a topografia, o clima, a vegetação, o uso do solo, as infraestruturas urbanas, entre outros aspectos relacionados ao espaço geográfico.

O geoprocessamento é composto por diversas ferramentas, como o Sistema de Informação Geográfica (SIG), a cartografia digital, o sensoriamento remoto, o sistema de posicionamento global e a topografia, que possibilitam a análise de informações com referências geográficas. Rosa (2005) enfatiza que o geoprocessamento é uma ferramenta eficaz para a tomada de decisões em diversas áreas. Além de servir de base para o planejamento urbano (para cadastramento imobiliário e posicionamento espacial), o geoprocessamento também é fundamental na gestão ambiental, contribuindo significativamente para a conservação de áreas e de recursos naturais. Na agricultura, além da sua aplicação na gestão de bacias hidrográficas, é também utilizado para um melhor planejamento no uso do solo, resultando numa abordagem mais integrada e sustentável à gestão dos recursos hídricos.

Para facilitar a atualização e o uso de dados georreferenciados de múltiplas fontes para o planejamento e a otimização de tarefas específicas, o SIG é construído com bancos de dados digitais gerenciados. O SIG é utilizado para combinar e analisar dados provenientes de diversas fontes, incluindo imagens digitais de satélite, mapas de tipos de solo, vegetação, flora e fauna, topografia, hidrologia e cartas climáticas, entre outros. Em um sistema computacional, sua principal função é armazenar, recuperar, analisar e gerar mapas (Liu, 2007).

Existem vários softwares disponíveis para a utilização de informações geográficas, sendo os mais conhecidos o ArcGIS e o QGIS, sendo este último gratuito para a comunidade. É difícil afirmar qual software é melhor, pois existem várias variáveis a serem consideradas na avaliação. O quadro 1 apresenta algumas comparações entre os softwares ArcGIS, QGIS, MapInfo e GRASS GISS, levando em conta a implementação de plugins, métodos de download, requisito mínimo e tipos de licença.

Quadro 1 – Comparação entre softwares

COMPARAÇÃO ENTRE SOFTWARES				
Software	**ArcGIS**	**QGIS**	**MapInfo**	**GRASS GIS**
Licença	Pago	Gratuito	Pago	Gratuito
Download Plugin	Site Oficial	Próprio Software	Terceiros	Site Oficial
Desen. Plugin	ESRI ou Terceirizado	Comunidade	Terceiros	Comunidade

Requisito Mínimo	Memória: 8GB Espaço: 4GB Processador: I3 ou superior	Memória: 2GB Espaço: 2,5GB Processador i3 ou superior	Memória: 4GB - Espaço: 1,5GB Processador I3 ou superior	Memória: 4GB Espaço: 1GB Processador i3 ou superior

Fonte: O autor (2024)

No entanto, além dessas variáveis mencionadas, outras considerações importantes podem incluir a interface do usuário, recursos específicos, capacidade de processamento, suporte da comunidade, integração com outras ferramentas e a experiência do usuário (Gisgeography, 2023). É recomendado que os usuários avaliem suas necessidades específicas e experimentem todos os softwares para determinar qual deles atende melhor às suas demandas (Trindade, 2021).

2.4 MAPEAMENTO DE RISCO DE INCÊNDIO

A ocorrência de incêndios está diretamente relacionada com a umidade do ar e o vento (Freire et al., 2002). A estimativa de risco de incêndio a curto prazo é fundamental para tomadas de decisão atualizadas em relação às atividades de pré-supressão e supressão em um plano de mitigação de incêndios. Essa estimativa pode ser usada para diversos fins, especialmente na prevenção e combate a incêndios, quando apresentada em forma de mapas de risco especializados. Esses mapas, que variam de cobertura local a global, são produzidos com diferentes resoluções espaciais e podem abranger períodos curtos ou longos (Freire et al., 2002).

De acordo com Deppe et al. (2004), um dos aspectos mais importantes a serem analisados nos índices de risco de incêndio é a disponibilidade de dados meteorológicos. O autor ressalta que a precisão desses índices é diretamente influenciada pela densidade de estações meteorológicas disponíveis. Quanto maior a quantidade dessas estações, melhores serão as estimativas obtidas. Além disso, o autor destaca que essas informações contribuem significativamente para a prevenção e combate de incêndios, assim como para o planejamento de ações de manutenção e monitoramento de incêndios.

2.5 LINGUAGEM DE PROGRAMAÇÃO PYTHON

Desenvolvido por Guido Van Rossum na década de 1980, Python é uma linguagem de alto nível orientada a objetos com semântica dinâmica. Sua simplicidade torna a programação mais acessível, enquanto seu suporte a módulos e pacotes promove a reutilização de código e a

programação modular. A compatibilidade do Python com vários sistemas operacionais e sua capacidade de suportar múltiplas linguagens contribuíram para o seu rápido crescimento em popularidade. Atualmente é a linguagem líder para análise de dados e ganhou adoção significativa na comunidade científica (Caelum, 2020).

Para aumentar a eficiência e produtividade no desenvolvimento em Python, muitos profissionais optam por utilizar um ambiente de desenvolvimento integrado (IDE). Entre as opções disponíveis, destaca-se o PyCharm, um software desenvolvido pela JetBrains. O PyCharm oferece uma ampla gama de recursos avançados, como edição de código inteligente, depuração integrada, gerenciamento de projetos e suporte a controle de versão, tornando o processo de desenvolvimento ainda mais facilitado (Saabith; Vinothraj; Fareez, 2021).

Conforme apresentado por Urano (2023), um dos recursos de destaque do PyCharm é "Code Intelligence", que abrange a notável capacidade do IDE de compreender, avaliar e interagir com o código-fonte do seu projeto. Este recurso permite diversas funcionalidades, como:

- **Compreensão de linguagem:** PyCharm possui uma compreensão profunda da linguagem Python, permitindo ao IDE compreender seus meandros, abrangendo elementos como recuo, identificadores, importações, classes e funções.
- **Sugestão de código:** PyCharm oferece sugestões de código que podem economizar tempo e minimizar erros, sugerindo automaticamente conclusões de código quando você começa a digitar um identificador ou função.
- **Detecção instantânea de erros:** À medida que você escreve seu código, o IDE o examina continuamente e identifica prontamente quaisquer erros de sintaxe ou lógica. Ao detectar esses problemas imediatamente, você pode evitar que surjam problemas ao executar o código.
- **Refatoração de código:** PyCharm fornece um kit de ferramentas robusto para refatoração de código, permitindo reorganizar, renomear e otimizar seu código com segurança. Isto garante atualizações precisas e automáticas de todas as referências do código refatorado, garantindo precisão em todo o processo.

O Code Intelligence do PyCharm é um recurso inestimável para desenvolvedores Python, pois aumenta muito a eficiência, promove a limpeza do código e mitiga a ocorrência de erros frequentes.

2.6 BANCO DE DADOS

Um banco de dados, conforme definidos por Date (2004), são sistemas informatizados que abrigam arquivos e informações organizados e estruturados para fins de manutenção de registros. O conceito de bancos de dados foi introduzido pela primeira vez no início da década de 1960 por Edgar Frank, pesquisador da IBM, e desde então passou por avanços significativos. Esses avanços permitiram que os usuários armazenassem e recuperassem com eficiência grandes quantidades de dados.

Além disso, o sistema de gerenciamento de banco de dados (SGBD) funciona como um mediador entre o usuário e o banco de dados, gerenciando o acesso aos dados e ocultando o intrincado funcionamento interno do banco de dados. De acordo com Rob e Coronel (2011), a inclusão do SGBD entre as aplicações do usuário e o banco de dados apresenta benefícios notáveis, incluindo a capacidade de compartilhar dados entre várias aplicações e usuários, melhorar a segurança dos dados, facilitar a integração mais eficiente dos dados e oferecer uma perspectiva coesa de as operações da organização.

MATERIAL E MÉTODOS

3.1 MATERIAL

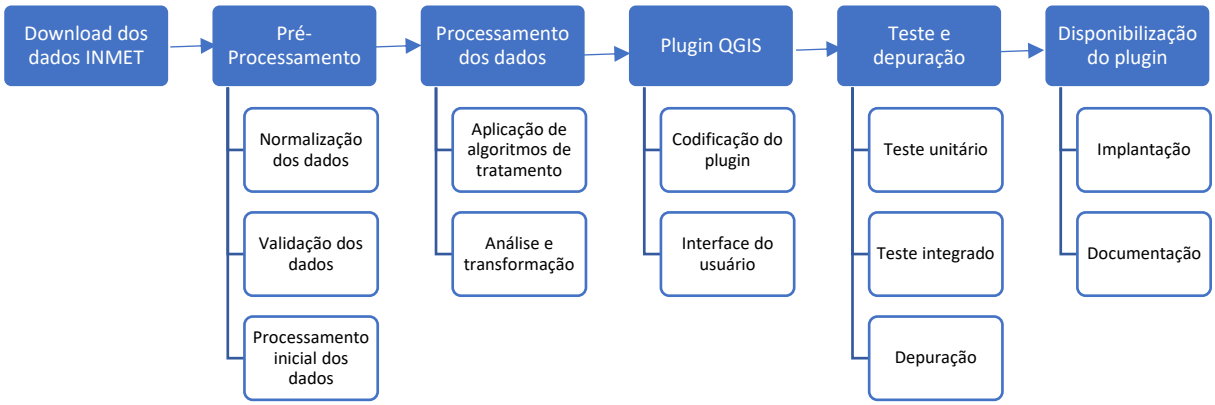
O material utilizado para o desenvolvimento do plugin inclui:

- Software QGIS na versão 3.22.12 (sendo requisito ter a versão a partir da 3.0).
- Ambiente PyCharm 2023.2 para elaboração de scripts e manipulação de dados INMET.
- QtDesing 3.22.12 para criação da interface gráfica, inclusas no QGIS para manipulação e geoprocessamento dos dados.
- Notepad++ versão 8.5.7.

3.2 MÉTODOS

Compreender a metodologia utilizada é uma etapa fundamental para entender o projeto proposto. Nesse sentido, foi elaborado um fluxograma metodológico, demonstrado na Figura 1, que busca explicar de forma clara os processos envolvidos e fornece uma representação objetiva do desenvolvimento do plugin.

Figura 1 - Fluxograma do trabalho



Fonte: O autor (2024)

A metodologia deste trabalho foi dividida em duas etapas principais. A primeira parte, o pré-processamento, concentrou-se na leitura e tratamento dos arquivos de dados. Isso inclui a verificação de possíveis métodos a serem implementados, além da detecção e correção de erros para garantir o funcionamento adequado do plugin. Na segunda parte, o front-end, foi dedicada à criação da interface do plugin para permitir que os usuários interagissem com as informações meteorológicas de maneira intuitiva e eficaz.

Essa divisão foi necessária para garantir que os dados brutos fossem adequadamente tratados, o que envolveu a verificação de inconsistências e a padronização das informações para análise posterior. Por fim, a união de ambas as partes para assim obter um plugin completamento funcional.

3.2.1 PRÉ-PROCESSAMENTO

Nesse processo, foi realizado o download de todas as planilhas disponíveis, abrangendo o período de 7 de maio de 2000, quando foram disponibilizadas as primeiras informações pelo INMET, até dezembro de 2023. A partir desses dados foi utilizado o software PyCharm (2023.2) para desenvolver o script em Python.

Para realizar a leitura dos arquivos, foi utilizado a biblioteca Pandas para o tratamento de diversos tipos de dados tabulares. Nesse contexto, foi utilizado o comando "pd.read_csv", uma vez que todos os arquivos disponibilizados pelo INMET possuíam a extensão “.cvs” (valores separados por vírgula). Além disso, outra vantagem dessa biblioteca é a capacidade de escolher em qual linha a tabela começa, através do comando "header" que, conforme ilustrado na Figura 2, a tabela tem início na linha 8.

Figura 2 - Código de leitura do arquivo

```
import pandas as pd

caminho = "INMET_CO_DF_A001_BRASILIA_01-01-2009_A_31-12-2009.CSV"
arquivo = pd.read_csv(caminho, sep=';', header=8, encoding='ISO-8859-1', decimal=',')
print(arquivo)
```

Fonte: O autor (2024)

A partir dessas informações, foi elaborada uma função para alterar os nomes das colunas de data, hora e informações de radiação global, a fim de adequá-las à nova formatação da tabela ajustada pelo INMET. A Figura 3 demonstra os comandos utilizados para realizar essas alterações.

Figura 3 - Código implementado para padronização das colunas

```
import pandas as pd

diretorio = r"..\2019\INMET_CO_GO_A014_GOIAS_01-01-2019_A_31-12-2019.CSV"
arquivo = pd.read_csv(diretorio, sep=';', header=8, encoding='ISO-8859-1', decimal=',')

colunas_mapeadas = {'Data': 'DATA (YYYY-MM-DD)',
                    'Hora UTC': 'HORA (UTC)',
                    'RADIACAO GLOBAL (Kj/m²)': 'RADIACAO GLOBAL (KJ/m²)'}
arquivo.rename(columns=colunas_mapeadas, inplace=True)
```

Fonte: O autor (2024)

Após ajustar a leitura dos arquivos padrões, a próxima etapa foi a obtenção de todas as informações do cabeçalho, incluindo as coordenadas tridimensionais, o nome e o código da estação, a região e o estado onde estão localizadas. Para isso, foi implementada uma função para extrair a tabela do cabeçalho presente em cada arquivo, cujo início se dá na linha um e termina na linha sete. Além disso, foram feitos alguns ajustes, visto que as duas tabelas estão presentes na mesma planilha. Como a tabela principal possuía um número maior de colunas, houve um acréscimo de colunas na tabela do cabeçalho. Dessa forma, foi necessário remover essas implementações indesejadas. A Figura 4 mostra o código implementado para obtenção e tratamento das informações no cabeçalho.

Figura 4 - Obtenção das informações no cabeçalho

```
def cabecalho(diretorio):
    col = [...]
    header_info = pd.read_csv(diretorio, sep=';', nrows=7, encoding='ISO-8859-1', decimal=',', header=None,
                              names=col)
    deletar = ['2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15', '16', '17', '18']
    header_info = header_info.drop(columns=deletar)

    if header_info.shape[0] != 7:
        return False

    header_info = header_info.T
    header_info.columns = ["REGIAO", "UF", "ESTACAO", "CODIGO", "LATITUDE", "LONGITUDE", "ALTITUDE"]
    header_info = header_info[1:]

    colunas_numericas = ["LATITUDE", "LONGITUDE", "ALTITUDE"]
    header_info[colunas_numericas] = header_info[colunas_numericas].apply(lambda x: x.str.replace(',', '.'))
    return header_info
```

Fonte: O autor (2024)

Após implementar essas funções, a próxima aplicação do código foi a normalização das informações presentes na tabela. Essa necessidade se deve ao fato da alteração do formato na tabela, pois nos primeiros modelos os valores ausentes na tabela eram representados como “-9999”, o que posteriormente foi alterado para células vazias. Outro ajuste que também foi necessário fazer foi no formato das datas, sendo que no primeiro modelo a separação era feita por hifens e na nova tabela foi alterada por barra. O mesmo foi feito na coluna “horas”, onde no primeiro formato eram representadas com a simbologia padrão de “00:00”, e na nova tabela foi alterado para “00:00 UTC”, simbolizando o Tempo Universal Coordenado (UTC). A Figura 5 mostra a função elaborada para normalização dessas informações.

Figura 5 - Formatação das colunas datas e horas

```
        def padronizar_data(data):
            formatos = ['%d/%m/%Y', '%Y/%m/%d', '%Y-%m-%d', '%d-%m-%Y', '%d-%m-%Y %H:%M:%S']
            for formato in formatos:
                try:
                    return datetime.strptime(data, formato).strftime('%Y-%m-%d')
                except ValueError:
                    pass
            return data

        arquivo['DATA'] = arquivo['DATA'].apply(padronizar_data)
        arquivo['DATA'] = pd.to_datetime(arquivo['DATA'])

        arquivo['HORA'] = [f'{hora[:2]}:00' for hora in arquivo['HORA']]
        arquivo['HORA'] = pd.to_datetime(arquivo['HORA'], format='%H:%M').dt.time

        arquivo['DATA'] = [datetime.combine(data, hora) - timedelta(hours=fuso) for data, hora
                           in zip(arquivo['DATA'], arquivo['HORA'])]
        arquivo.replace(-9999, None, inplace=True)
        return arquivo
    else:
        print('Arquivo Alterado!')
```

Fonte: O autor (2024)

Outra implementação no código envolveu o uso dos comandos “pd.errors.ParserError”, “pd.errors.EmptyDataError” e “Exception”. Cada uma dessas funções é utilizada para lidar com diferentes tipos de erros que podem ocorrer durante o processamento dos dados. o “ParserError” é acionado quando ocorre um erro durante a análise do arquivo CSV, o “EmptyDataError” é utilizado quando o arquivo está vazio, e o “Exception**”** é uma classe base para exceções em Python, sendo útil para capturar erros genéricos e exceções não esperadas durante a execução do código. A Figura 6 ilustra os comandos aplicados.

Figura 6 - Implementação de código para tratamento de erros

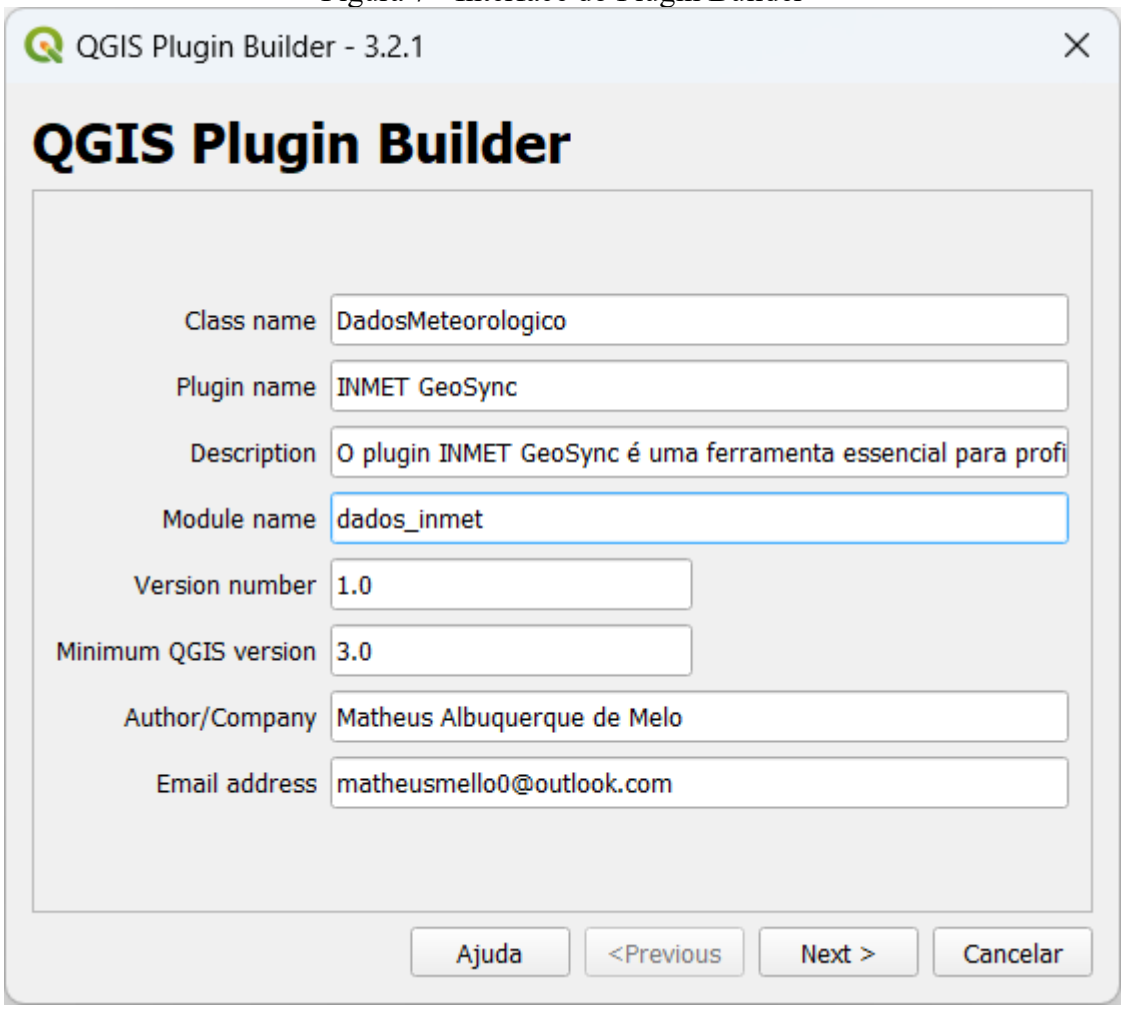
```
except pd.errors.ParserError:
    print(diretorio, '\n')
    print('Erro ao analisar o arquivo CSV')

except pd.errors.EmptyDataError:
    print(diretorio, '\n')
    print('O arquivo está vazio')

except Exception as e:
    print(diretorio, '\n')
    print(f'Erro inesperado: {str(e)}')
return salvar_dados
```

Fonte: O autor (2024)

3.2.2 FRONT-END

Após a conclusão do tratamento de dados e da geração do produto, a próxima etapa envolve a criação da interface do plugin. Para isso, o QGIS oferece diversas ferramentas que permitem aos usuários gerar o escopo do complemento preenchendo apenas as informações necessárias. Uma dessas ferramentas é o Plugin Builder, um complemento que facilita a geração de uma estrutura completa de plugin. A Figura 7 ilustra a interface do Plugin Builder e as informações iniciais necessárias para a elaboração do plugin.

Figura 7 - Interface do Plugin Builder



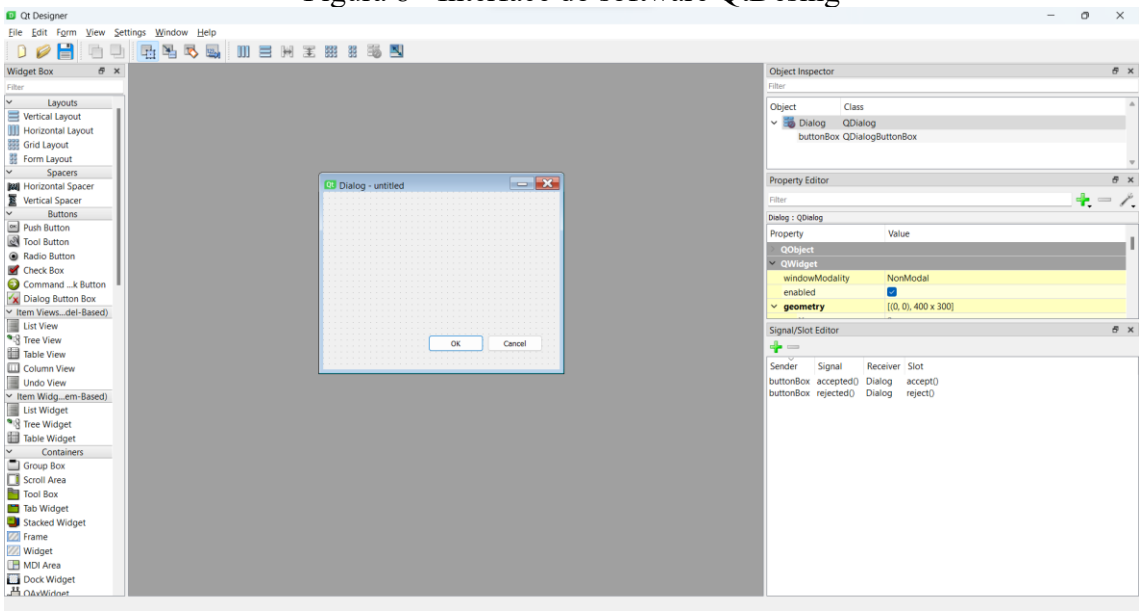
Fonte: O autor (2024)

Além do tipo visual do plugin, existem várias outras características a serem preenchidas ao criar um plugin no Plugin Builder do QGIS. Como por exemplo:

- Class Name: Nome que será atribuída a classe do código contigo no plugin.
- Plugin name: Nome pelo qual o plugin será referenciado
- Description: Descrição detalhada sobre o plugin que estará disponível no repositório oficial do QGIS.
- Module name: Nome que será dado ao módulo principal do plugin
- Version Number: Número da versão que estará disponível do plugin.
- Mínimum Version Versão mínima do QGIS necessária para o plugin funcionar corretamente.
- Author/Company: Nome do autor ou da empresa que elaborou o plugin.
- Email adress: Endereço do e-mail para contato.

Após o preenchimento dos dados, uma pasta foi gerada contendo vários arquivos necessários para o funcionamento adequado do plugin, desde o script base (.py), o arquivo com a interface do plugin (.ui), metadados com as informações do plugin (.metadata), arquivos de documentação sobre o plugin, arquivos de configuração (.ini), entre outros. Entre os arquivos citados, o responsável por armazenar informações sobre a interace gráfica do plugin é a extensão .ui. Para trabalhar com esse arquivo, foi utilizado o software QtDesign, que já está incluso no próprio software QGIS. A Figura 8 mostra a interface do software.

Figura 8 - Interface do software QtDesing



Fonte: O autor (2024)

A interface do plugin foi projetada para oferecer uma experiência de usuário intuitiva e eficaz. Permitindo o usuário escolher o tipo de informação desejada, a definição do intervalo de tempo (hora e data) e a especificação do local para salvar os produtos gerados. Para implementar essa funcionalidade, foi utilizado várias ferramentas e componentes fornecidos pelo Qt Design, incluindo:

1 - Line Edit: Uma caixa de texto na qual os usuários podem inserir um diretório ou onde podem ver o diretório selecionado.

2 - Push Button: Um botão de ação que abre uma caixa de diálogo para que os usuários escolham o diretório onde os arquivos do INMET estão localizados.

3 - QgsCheckableComboBox: Semelhante ao combobox, o QgsCheckableComboBox lista itens dos quais o usuário pode escolher um ou mais itens como resultado.

4 - QgsDateTimeEdit: Uma ferramenta específica disponível apenas no Qt Design em conjunto com o QGIS, que exibe um calendário para que o usuário possa escolher a data de interesse.

5 - QComboBox: Mostra uma lista de itens dos quais o usuário pode escolher um para a seleção dos dados.

6. Spin Box: A ferramenta Spin Box é usada para inserção de valores numéricos de forma intuitiva e controlada em uma interface gráfica.

7. QTimeEdit: Ferramenta usada para inserção de informações de horário.

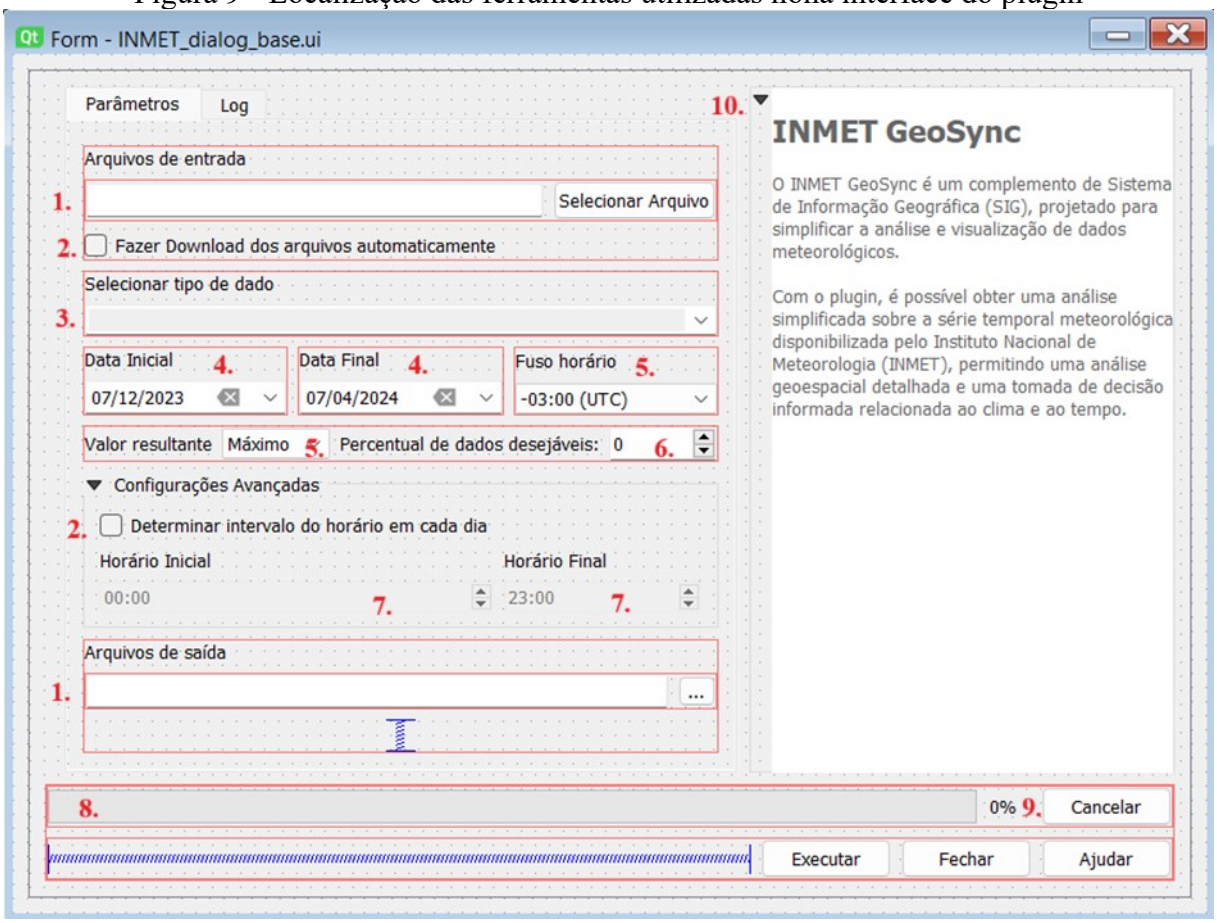
8. Progress Bar: Ferramenta utilizada para visualizar o andamento do processamento de uma tarefa em andamento.

9. Push Button: Um botão de ação cujo a função será executada ao clicar sobre a ferramenta.

10 – CollapsibleGroupBox: Oculta ou exibe ferramenta(s) ou informação(ões) que estejam dentre dessa variável.

A Figura 9 ilustra a localização dessas ferramentas na interface do plugin, destinadas a suas respectivas aplicações específicas.

Figura 9 - Localização das ferramentas utilizadas nona interface do plugin



Fonte: O autor (2024)

Antes de salvar o modelo, foram aplicadas algumas interações desejadas diretamente no QtDesigner para reduzir a necessidade de escrita em Python. Na Figura 10 podemos observar as informações relevantes.

Figura 10 - Comandos atribuídos no QtDesing

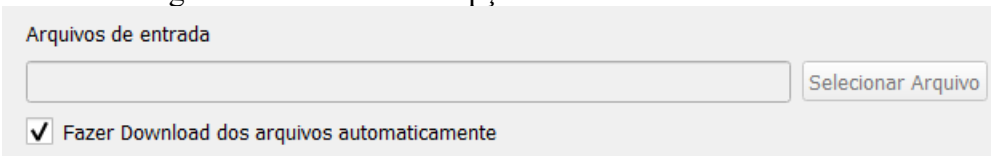
Sender	Signal	Receiver	Slot
hora_fixo	clicked(bool)	hora_inicial	setEnabled(bool)
hora_fixo	clicked(bool)	hora_final	setEnabled(bool)
download	clicked(bool)	select_dir	setDisabled(bool)
download	clicked(bool)	caminho_select	setDisabled(bool)

Fonte: O autor (2024)

O item "hora_fixo" é uma checkbox localizado nas configurações avançadas. Caso o usuário escolha ativá-la, ele poderá especificar o intervalo de horário a ser considerado ao longo do(s) dia(s).

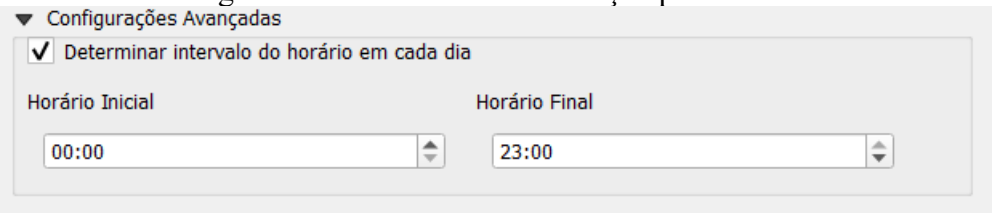
Outra opção é a caixa de seleção “download”, que possibilita o download direto pelo site, eliminando a necessidade do usuário inserir manualmente o caminho da pasta. Quando essa opção está ativada, as escolhas relacionadas à pasta e ao caminho são desabilitadas conforme ilustra a Figura 11. Essas interações tornam o uso do complemento mais fácil e proporcionam maior flexibilidade nas configurações, permitindo que os usuários escolham entre as opções de download e personalizem o intervalo de horário conforme necessário (Figura 12).

Figura 11 - Interface da opção de download automático



Fonte: O autor (2024)

Figura 12 - Interface da delimitação por horário

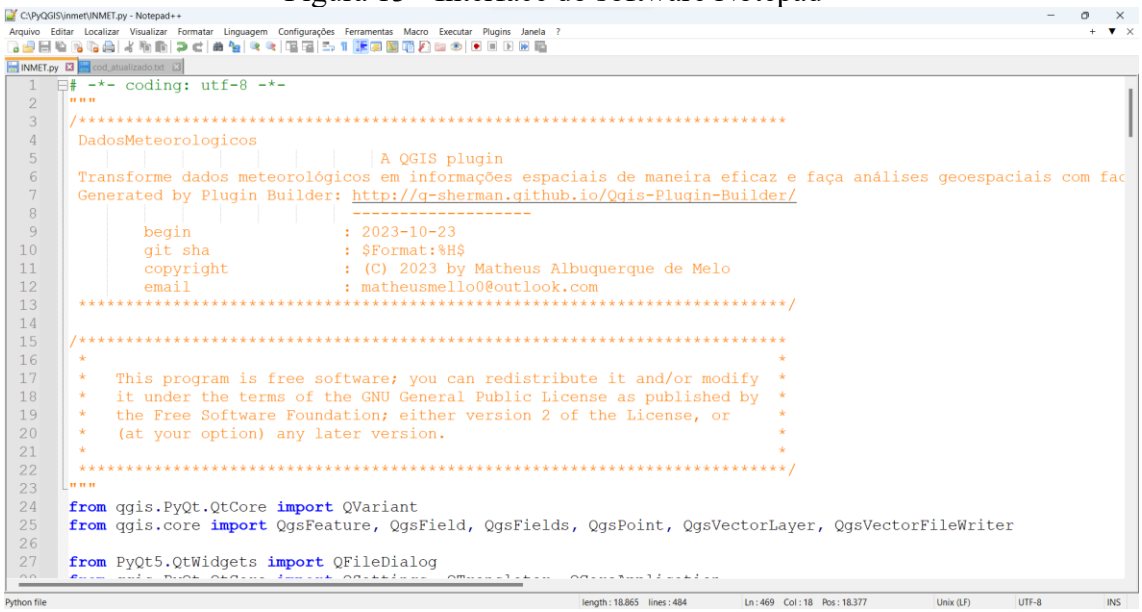


Fonte: O autor (2024)

Após a conclusão da etapa de criação da interface, o próximo passo envolveu a integração das duas partes (back e front-end) para permitir o funcionamento do plugin, uma vez que, até esse momento, ambas estavam operando de forma independente. Para realizar essa integração, foi utilizado o arquivo .py gerado durante a criação do plugin inicial.

Essa etapa de união das funcionalidades da interface com o código do plugin é essencial para que o complemento funcione de forma eficaz e permita aos usuários utilizar as opções de configuração e aquisição de dados de maneira integrada. Para isso, foi utilizado o software Notepad++ uma vez que o editor de texto também é utilizado como código de fonte aberto. Conforme ilustra na Figura 13.

Figura 13 - Interface do software Notepad++



Fonte: O autor (2024)

O primeiro ajuste necessário está relacionado à escolha das datas e horários, garantindo que as informações finais não possam ser anteriores às informações iniciais. Para resolver essa questão, foram criadas duas funções separadas, uma para a data e outra para o horário, conforme ilustrado na Figura 14.

Figura 14 - Código implementado para validação de data e hora

```
def validar_horarios(self):
    hora_inicial = self.dlg.hora_inicial.time()
    hora_final = self.dlg.hora_final.time()

    if hora_inicial > hora_final:
        self.dlg.hora_inicial.setTime(hora_final)


def validar_datas(self):
    data_inicial = self.dlg.dt_inicial.date()
    data_final = self.dlg.dt_final.date()

    if data_inicial > data_final:
        self.dlg.dt_inicial.setDate(data_final)
```

Fonte: O autor (2024)

Essas funções desempenham um papel na validação e garantia de que os intervalos de datas e horários selecionados pelos usuários estejam corretos, evitando escolhas inválidas que poderiam afetar a aquisição de dados do INMET. Esse cuidado na validação dos parâmetros de entrada contribui para a consistência e confiabilidade do plugin.

De maneira semelhante, ambas as funções foram projetadas para garantir que, caso o usuário insira uma informação inicial maior do que a informação final, elas sejam ajustadas automaticamente para serem iguais à informação final. Da mesma forma, se o usuário reduzir a informação final, a informação inicial também será reduzida de forma correspondente, mantendo a consistência dos intervalos de datas e horários.

Quanto à funcionalidade que permite ao usuário escolher o diretório dos arquivos do INMET e o local de saída, foram criadas duas funções específicas para esses propósitos. Ambas as funções permitem que o caminho do diretório seja selecionado pelo usuário e são responsáveis por armazenar o caminho nas variáveis apropriadas, simplificando assim a interação do usuário com o plugin e facilitando a configuração das opções de entrada e saída. Conforme ilustra na Figura 15.

Figura 15 - Código implementado para a definir entrada e saída de arquivos

```
def pasta(self):
    """Selecionar o diretório aonde está a pasta e subpastas das estações a serem tratadas"""
    pasta_selecionada = QFileDialog.getExistingDirectory(caption="Escolha a pasta dos arquivos",
    options=QFileDialog.ShowDirsOnly)
    self.dlg.caminho_select.setText(pasta_selecionada)


def definirsaida(self):
    salvar = str(QFileDialog.getSaveFileName(caption="Escolha a pasta dos arquivos",
    filter="Shapefile (*.shp)")[0])
    self.dlg.dir_saida.setText(salvar)
```

Fonte: O autor (2024)

A abordagem implementada para realizar o download diretamente do site consistiu na reutilização do link https://portal.inmet.gov.br/uploads/dadoshistoricos. Este link automaticamente direciona para o download dos dados mais recentes do ano em curso. A partir disso, os arquivos compactados são extraídos e posteriormente os mesmos são excluídos, deixando apenas os arquivos necessários para o processamento. Esses arquivos são salvos na aba de downloads e podem ser visualizados pelo usuário ou reaproveitados para futuros processamentos. A Figura 16 ilustra o código implementado para esta função.

Figura 16 - Código implementado para download automático de arquivos

```
def download_arquivos_inmet(data_inicial, data_final):
    for ano in range(data_inicial, data_final+1):
        url = f"https://portal.inmet.gov.br/uploads/dadoshistoricos/{ano}.zip"
        save_path = f"{ano}.zip"

        downloads_path = Path.home() / "Downloads"
        extraction_path = downloads_path / "INMET_DOWNLOAD" / f"{ano}"


        if not os.path.exists(extraction_path):
            try:
                download_zip_file(url, save_path)
                extract_zip_file(save_path, extraction_path)
                print("Arquivo ZIP baixado e extraído com sucesso!")
            except requests.exceptions.RequestException as e:
                print(f"Erro ao baixar o arquivo: {e}")
            except zipfile.BadZipFile:
                print("Erro ao extrair o arquivo ZIP.")
            finally:
                if os.path.exists(save_path):
                    os.remove(save_path)
        else:
            print("Os dados já foram baixados e extraídos anteriormente.")

download_arquivos_inmet(data_inicial, data_final)
```

Fonte: O autor (2024)

Em relação ao método utilizado para obter os resultados das variáveis climáticas, foram definidas quatro opções: selecionar o valor máximo, mínimo, médio e a soma, juntamente com uma ou mais informações climáticas. Para aplicar esse método, foi implementado um laço de repetição para verificar qual variável foi selecionada pelo usuário e, em seguida, aplicar o método correspondente. Os resultados foram salvos inicialmente no dataframe. A Figura 17 ilustra o código implementado para o funcionamento desse procedimento.

Figura 17 - implementação da variável selecionada

```
for var, comando in zip(['Máximo', 'Mínimo', 'Média', 'Soma'], ['max', 'min', 'mean', 'sum']):
    if var == variavel:
        salvar_dados = salvar_dados.groupby(['ESTACAO', 'REGIAO', 'UF', 'CODIGO', 'LATITUDE', 'LONGITUDE',
        'ALTITUDE']).agg({coluna: comando for coluna in colunas_dados}).reset_index()
```

Fonte: O autor (2024)

Para lidar com possíveis ausências de dados meteorológicos em determinadas estações, seja por manutenção ou defeitos, foi desenvolvida uma funcionalidade que permite ao usuário selecionar o percentual mínimo de dados desejáveis para considerar uma estação aceitável durante o período de interesse (conforme observado na Figura 18). Ao permitir que os usuários determinem o nível mínimo de dados aceitável, mesmo em condições menos ideais, o sistema busca mitigar os impactos das lacunas nos dados meteorológicos, garantindo uma distribuição mais equitativa das informações disponíveis em conformidade com a região de interesse.

Figura 18 - Código implementado para percentual dos dados disponíveis.

```
for coluna in colunas:

    num_dados_totais = (arquivo['DATA'].count())
    num_vazios = arquivo[coluna].isnull().sum()
    num_dados = num_dados_totais - num_vazios

    arquivo[coluna] = arquivo[coluna].astype(float)
    percentual = self.dlg.percentual.value()

    if (num_dados * 100 / num_dados_totais) >= percentual:
        if variavel == "Máximo":
            info_estacao[coluna] = arquivo[coluna].max()
        elif variavel == "Mínimo":
            info_estacao[coluna] = arquivo[coluna].min()
        elif variavel == "Média":
            info_estacao[coluna] = arquivo[coluna].mean()
        elif variavel == "Soma":
            info_estacao[coluna] = arquivo[coluna].sum()
    else:
        info_estacao[coluna] = None
```

Fonte: O autor (2024)

Além disso, a etapa subsequente teve como objetivo transformar os dados salvos no formato *dataframe* em um arquivo do tipo *shapefile*, com referência às coordenadas de latitude, longitude e altitude, além de todas as informações solicitadas pelos usuários e as informações padrão para cada estação. O código implementado para esse propósito é mostrado na Figura 19, onde o sistema de referência de coordenadas SIRGAS2000 (EPSG:4674) é implementado e um formato de geometria de ponto é fornecido para o arquivo de saída.

Figura 19 - Conversão de tabela para arquivo shapefile

```
salvar_dados['geometry'] = salvar_dados.apply(lambda row: Point(row['LONGITUDE'], row['LATITUDE']), axis=1)

gdf = gpd.GeoDataFrame(salvar_dados, geometry='geometry')
gdf.crs = pyproj.CRS('EPSG:4674')

dir_shapefile = self.dlg.dir_saida.text()
gdf.to_file(dir_shapefile)

nome = dir_shapefile[dir_shapefile.rfind("/")+1:dir_shapefile.rfind(".")]
camada = QgsVectorLayer(dir_shapefile, nome, 'ogr')

QgsProject.instance().addMapLayer(camada)
```

Fonte: O autor (2024)

Além das bibliotecas padrões inseridos ao gerar a estrutura inicial do complemento, também foram implementadas outras bibliotecas para manipular funções específicas, como leitura de arquivos, navegação pelo sistema de arquivos e outras funcionalidades. A Figura 20 demonstra todas as bibliotecas que foram utilizadas no desenvolvimento do plugin.

Figura 20 - Bibliotecas utilizadas para elaboração do plugin

```
from qgis.PyQt.QtCore import QVariant
from qgis.core import QgsFeature, QgsField, QgsFields, QgsPoint, QgsVectorLayer, QgsVectorFileWriter

from PyQt5.QtWidgets import QFileDialog
from qgis.PyQt.QtCore import QSettings, QTranslator, QCoreApplication
from qgis.PyQt.QtGui import QIcon
from qgis.PyQt.QtWidgets import QAction, QFileDialog
from qgis.core import *
from PyQt5.QtCore import QTime, Qt, QDateTime, QDate
from PyQt5.QtWidgets import QMessageBox

# Initialize Qt resources from file resources.py
from .resources import *
# Import the code for the dialog
from .INMET_dialog import DadosMeteorologicosDialog
import os.path
import processing
import sys, os
from osgeo import ogr
from PyQt5.QtWidgets import QDialog, QDateTimeEdit
import pandas as pd
from datetime import datetime, timedelta
import geopandas as gpd
from shapely.geometry import Point
import pyproj
```

Fonte: O autor (2024)

Essas bibliotecas atribuem funcionalidades do plugin e permitir a manipulação eficiente dos dados do INMET, bem como a interação com o sistema operacional. Cada biblioteca foi escolhida com um propósito específico para atender às necessidades do complemento.

Todas as funcionalidades da interface, que anteriormente estavam apenas como elementos visuais, foram totalmente implementadas, foram criadas as conexões entre essas funcionalidades e suas funções correspondentes. Com essas implementações o código é capaz de fornecer informações essenciais para a coleta de dados meteorológicos de maneira eficaz (Figura 21).

Figura 21 - Implementação para iteratividade do plugin com o código fonte

```
513     def run(self):
514         """Run method that performs all the real work"""
515
516         self.dlg = DadosMeteorologicosDialog()
517
518
519         self.dlg.dt_inicial.dateChanged.connect(self.validar_datas)
520         self.dlg.dt_final.dateChanged.connect(self.validar_datas)
521         self.dlg.hora_inicial.timeChanged.connect(self.validar_horarios)
522         self.dlg.hora_final.timeChanged.connect(self.validar_horarios)
523
524         self.dlg.executar.clicked.connect(self.validar_campos)
525
526         self.dlg.dt_inicial.setTime(QTime(0, 0))
527         self.dlg.dt_final.setTime(QTime(23, 0))
528
529         if self.first_start == True:
530             self.first_start = False
531
532         self.dlg.select_dir.clicked.connect(self.pasta)
533         self.dlg.salvar_arquivo.clicked.connect(self.definirsaida)
534
535         self.dlg.show()
536         result = self.dlg.exec_()
```

Fonte: O autor (2024)

4 RESULTADO E DISCUSSÕES

Na análise dos arquivos, foi observado que algumas tabelas apresentavam diferenças nos nomes das colunas, com leves alterações no tipo de informação ao longo do tempo. Por isso, foi necessário padronizar os arquivos disponibilizados pelo INMET. Entre as análises realizadas, constatou-se que ocorreram mudanças na formatação dos arquivos entre os anos de 2018 e 2019, bem como na representação dos dados, conforme ilustrado no Quadro 2.do no quadro 2.

Quadro 2 - Alteração no nome das colunas

Ano	Coluna Data	Coluna Hora	Informação Data	Informação Hora
2018	DATA (YYYY-MM-DD)	HORA (UTC)	2018-01-01	23:00
2019	Data	Hora UTC	2019/01/01	2300 UTC

Fonte: O autor (2024)

Além disso, também foram observadas modificações no cabeçalho dos arquivos. A primeira alteração identificada foi a remoção da formatação da data na linha que descreve a fundação da estação. A segunda alteração envolveu a substituição de caracteres acentuados por símbolos de interrogação, como ilustrado na Figura 22.

Figura 22 - Comparação do cabeçalho entre arquivos de 2018 e 2019

	A	B
1	REGIÃO:	CO
2	UF:	DF
3	ESTAÇÃO:	BRASILIA
4	CODIGO (WMO):	A001
5	LATITUDE:	-15,78944444
6	LONGITUDE:	-47,92583332
7	ALTITUDE:	1159,54
8	DATA DE FUNDAÇÃO (YYYY-MM-DD):	07/05/2000

	A	B
1	REGI?O:	CO
2	UF:	DF
3	ESTAC?O:	BRASILIA
4	CODIGO (WMO):	A001
5	LATITUDE:	-15,789343
6	LONGITUDE:	-47,925756
7	ALTITUDE:	1160,96
8	DATA DE FUNDAC?O:	07/05/2000

Fonte: O autor (2024)

Outra alteração evidenciada foi entre 2019 e 2020 cujo a coluna “RADIACAO GLOBAL (KJ/m²)” teve uma leve alteração na simbologia, deixando assim o “J” minúsculo, conforme ilustrado na Figura 23.

Figura 23 - Análise da alteração do nome da coluna radiação

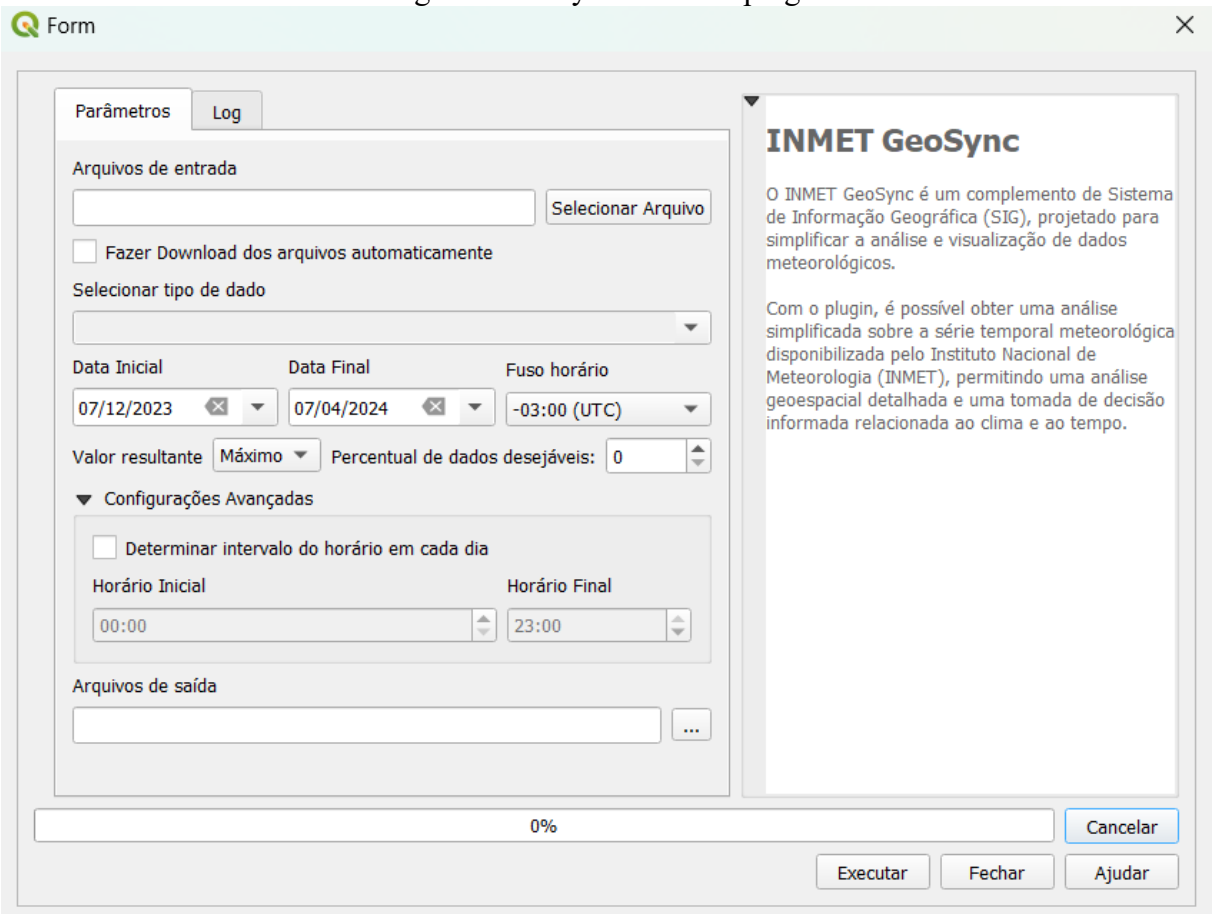
3	ESTAC?O:	BRASILIA	
4	CODIGO (WMO):	A001	
5	LATITUDE:	-15,789343	
6	LONGITUDE:	-47,925756	
7	ALTITUDE:	1160,96	
8	DATA DE FUNDAC?O:	07/05/2000	
9	Data	Hora UTC	RADIACAO GLOBAL (KJ/m²)
10	01/01/2019	0000 UTC	
11	01/01/2019	0100 UTC	

3	ESTACAO:	BRASILIA	
4	CODIGO (WMO):	A001	
5	LATITUDE:	-15,789343	
6	LONGITUDE:	-47,925756	
7	ALTITUDE:	1160,96	
8	DATA DE FUNDACAO:	07/05/2000	
9	Data	Hora UTC	RADIACAO GLOBAL (Kj/m²)
10	01/01/2020	0000 UTC	
11	01/01/2020	0100 UTC	

Fonte: O autor (2024)

Após a padronização dos arquivos ao longo dos anos e a conclusão do plugin, os usuários podem acessar uma ampla gama de informações meteorológicas em diferentes períodos, proporcionando uma visão abrangente e detalhada das condições climáticas. A Figura 24 ilustra a interface final do plugin, mostrando de forma clara e organizada as diversas opções disponíveis. Com esta ferramenta à disposição, os usuários podem gerar uma variedade de resultados potenciais, desde previsões de curto prazo até análises climáticas mais abrangentes, permitindo uma melhor compreensão e tomada de decisões informadas em relação ao clima.

Figura 24 - Layout final do plugin



Fonte: O autor (2024)

Na figura 25, é apresentada a tabela de atributos com diversas informações disponibilizadas após o processamento dos dados meteorológicos. Entre elas, destacam-se as informações padrão, como o nome da estação, região, unidade federativa (UF), código da estação, latitude, longitude e altitude. Além disso, podem ser incluídas informações solicitadas pelo usuário, como temperatura, precipitação, radiação, pressão atmosférica, dados sobre o vento, umidade do ar, entre outros.

Figura 25 - Tabela de atributos

	ESTACAO	REGIAO	UF	CODIGO	LATITUDE	LONGITUDE	ALTITUDE	PRECIPIT	PRESSAO_ES	PRESSAO_MA	PRESSAO_MI	RADIACAO	TEMP_BULBO
1	CAMPO MAIOR	NE	PI	A376	-4.86416666	-42.14555555	125	56,600000...	1002,10000...	1002,3999999...	1001,299999...	8208,89999...	36,700000000...
2	QUIXADA	NE	CE	A369	-4.97888888	-39.05722222	193	49,399999...	994,200000...	994,50000000...	994,0000000...	7186,10000...	37,200000000...
3	BEBEDOURO	SE	SP	A764	-20.94805555	-48.47138888	542	22,600000...	957,299999...	957,29999999...	957,2000000...	6988,10000...	37,399999999...
4	CACHOEIRA PAU...	SE	SP	A769	-22.68888888	-45.00555555	586	56,000000...	954,799999...	954,89999999...	954,6000000...	6817,80000...	35,799999999...
5	CAROLINA	NE	MA	A205	-7.33722221	-47.45972222	182.88	0	993,200000...	992,60000000...	991,7000000...	6275,89999...	34,899999999...
6	IBOTIRAMA	NE	BA	A439	-12.19305555	-43.21333333	425.12	38,000000...	969,700000...	969,79999999...	969,6000000...	5310,00000...	35,700000000...
7	ALVORADA DO G...	NE	PI	A336	-8.37638888	-43.85944444	225	22,600000...	990,799999...	990,79999999...	990,7000000...	5297,30000...	36,600000000...
8	CARATINGA	SE	MG	A554	-19.73583333	-42.13722221	609.25	55,799999...	951,899999...	952,00000000...	951,8999999...	5296,30000...	35,799999999...
9	SAO JOSE DOS A...	S	RS	A829	-28.7486111	-50.05777777	1228.59	4,8000000...	886,500000...	886,60000000...	886,2999999...	5282,69999...	29,399999999...
10	NOVA UBIRATA	CO	MT	A929	-13.41111111	-54.75222222	466.48	20,800000...	963,500000...	963,60000000...	963,3999999...	5136,30000...	32,100000000...
11	VITORIA DA CON...	NE	BA	A414	-14.88638888	-40.80138888	879.38	NULL	922,100000...	922,20000000...	922,0000000...	5076,10000...	32,200000000...
12	MANHUACU	SE	MG	A556	-20.26333333	-42.18277777	819.47	NULL	928,700000...	928,70000000...	928,6000000...	5032,39999...	32,500000000...
13	CIDADE GAUCHA	S	PR	A869	-23.35916666	-52.93194443	365.79	43,600000...	976,600000...	976,60000000...	976,2999999...	4996,00000...	36,799999999...
14	BREJO GRANDE	NE	SE	A421	-10.47388888	-36.48194443	6.11	15,400000...	1016,70000...	1017,2000000...	1016,200000...	4987,39999...	34,700000000...
15	ITAPORANGA	NE	PB	A373	-7.31833332	-38.14083333	292	32,799999...	984,000000...	984,00000000...	983,2000000...	4941,60000...	37,799999999...

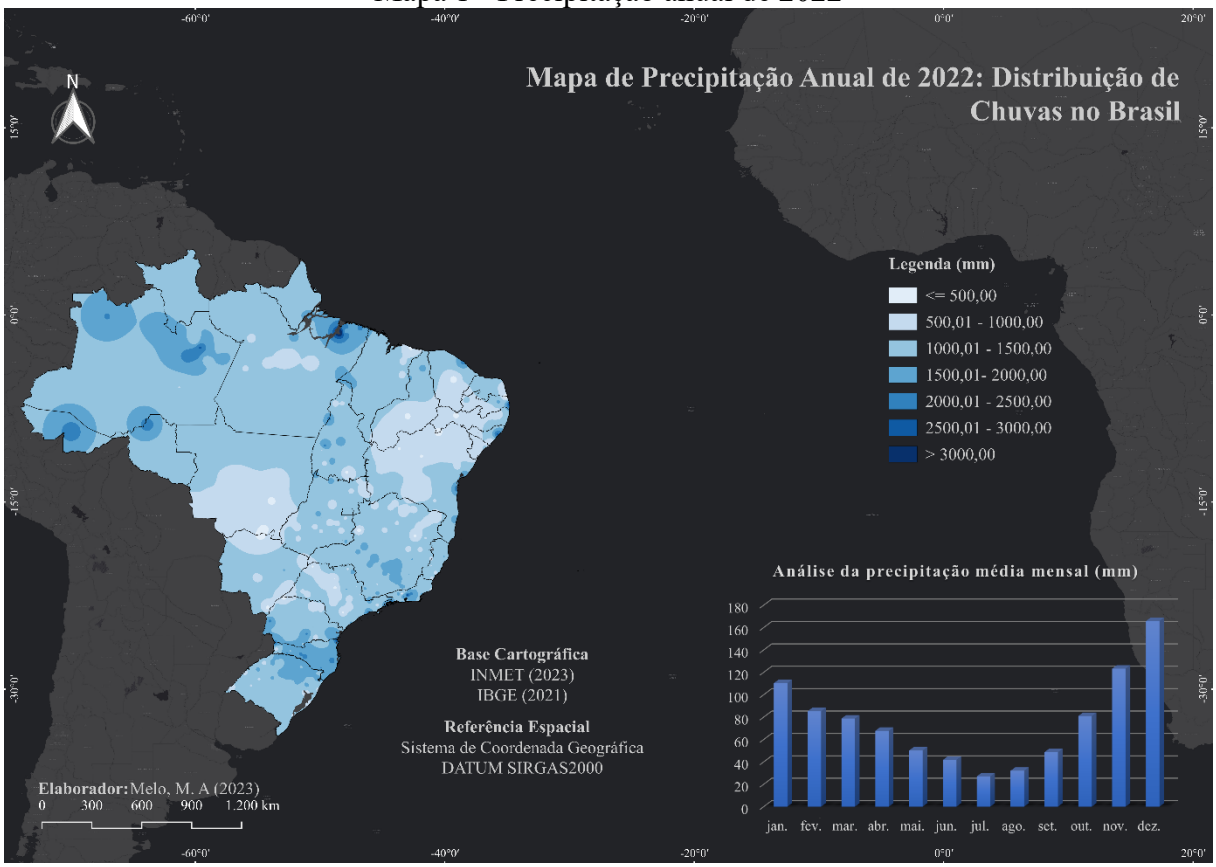
Fonte: O autor (2024)

4.1 MAPA DE PRECIPITAÇÃO

O mapa de precipitação é uma ferramenta fundamental na avaliação das condições meteorológicas em uma determinada região. Utilizando os dados fornecidos pelo INMET, é possível criar mapas de precipitação que oferecem uma representação espacial das áreas afetadas por chuvas durante o período de estudo. Esses mapas fornecem informações valiosas para entender os padrões de chuva em diferentes regiões e épocas do ano, o que é crucial para identificar áreas propensas a eventos de chuva intensa. Além da análise ser essencial para o planejamento de medidas de prevenção de enchentes, irrigação agrícola e uma variedade de

outras aplicações relacionadas ao clima. No Mapa 1 é possível analisar a precipitação em todo o território brasileiro durante o ano de 2022

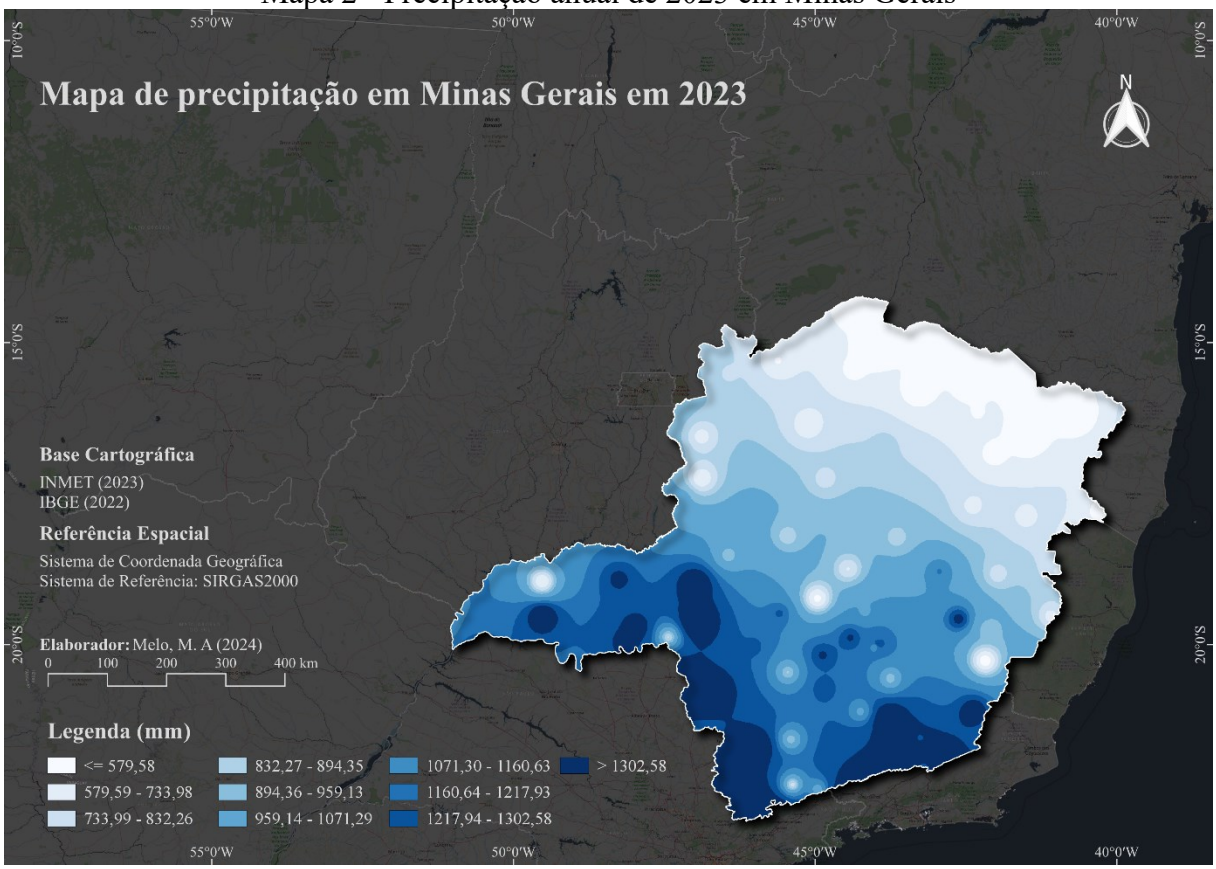
Mapa 1 - Precipitação anual de 2022



Fonte: O autor (2024)

Além disso, foi elaborado um segundo mapa (conforme ilustrado no Mapa 2) de precipitação, com o propósito de avaliar a precipitação total ocorrida no ano de 2023 no estado de Minas Gerais. Destaca-se que tais informações têm aplicabilidade em todo o território brasileiro ou em regiões específicas de interesse.

Mapa 2 - Precipitação anual de 2023 em Minas Gerais



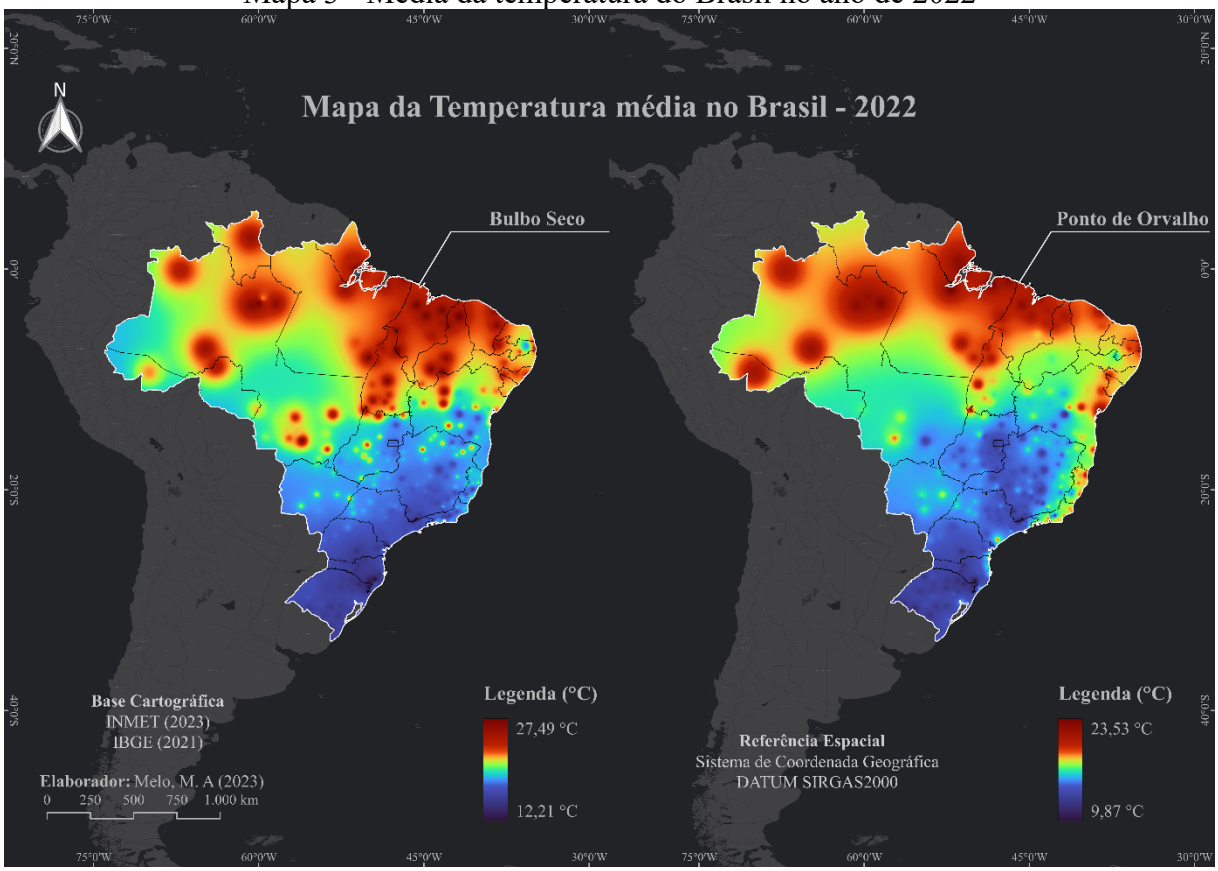
Fonte: O autor (2024)

4.2 MAPA DE TEMPERATURA

Outro resultado que pode ser obtido com os dados do INMET são os mapas de temperatura, que, de forma semelhante aos mapas de precipitação, desempenham um papel essencial em várias análises de estudo meteorológicos. Além disso, o plugin oferece diversos tipos de dados de temperatura. Isso inclui a temperatura de bulbo seco, que representa a temperatura ambiente, e a temperatura de ponto de orvalho, que é a temperatura na qual o vapor de água se condensa em líquido. O mapa 3 ilustra a média de ambas as temperaturas para o ano de 2022.

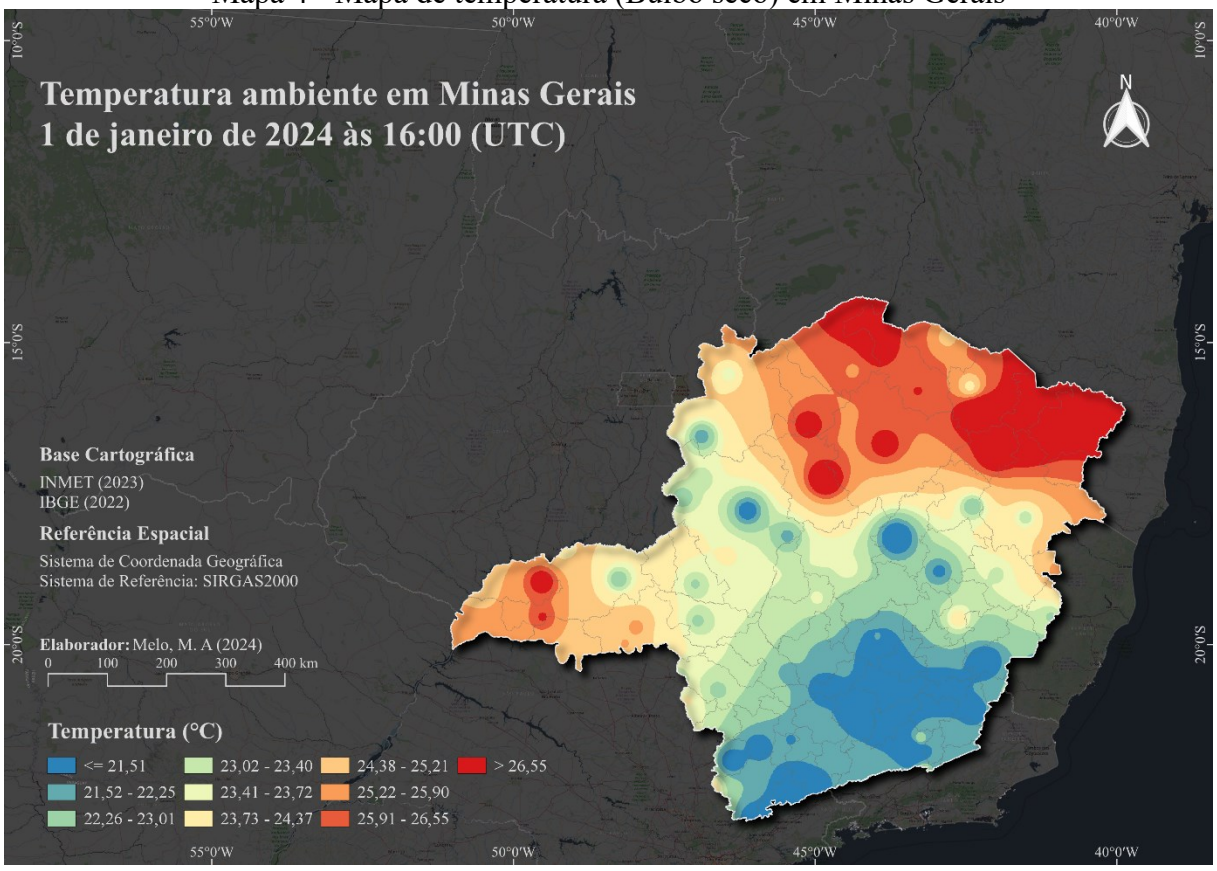
No Mapa 4, é retratada a temperatura máxima em Minas Gerais no dia 1º de janeiro às 16:00h (UTC), correspondendo a 13:00h no horário de Brasília. Neste caso, foram utilizadas as informações de temperatura de bulbo seco. Enquanto isso, o Mapa 5 ilustra a temperatura no ponto de orvalho para a mesma região e período, representando a temperatura na qual o vapor de água se condensa em líquido.

Mapa 3 - Média da temperatura do Brasil no ano de 2022



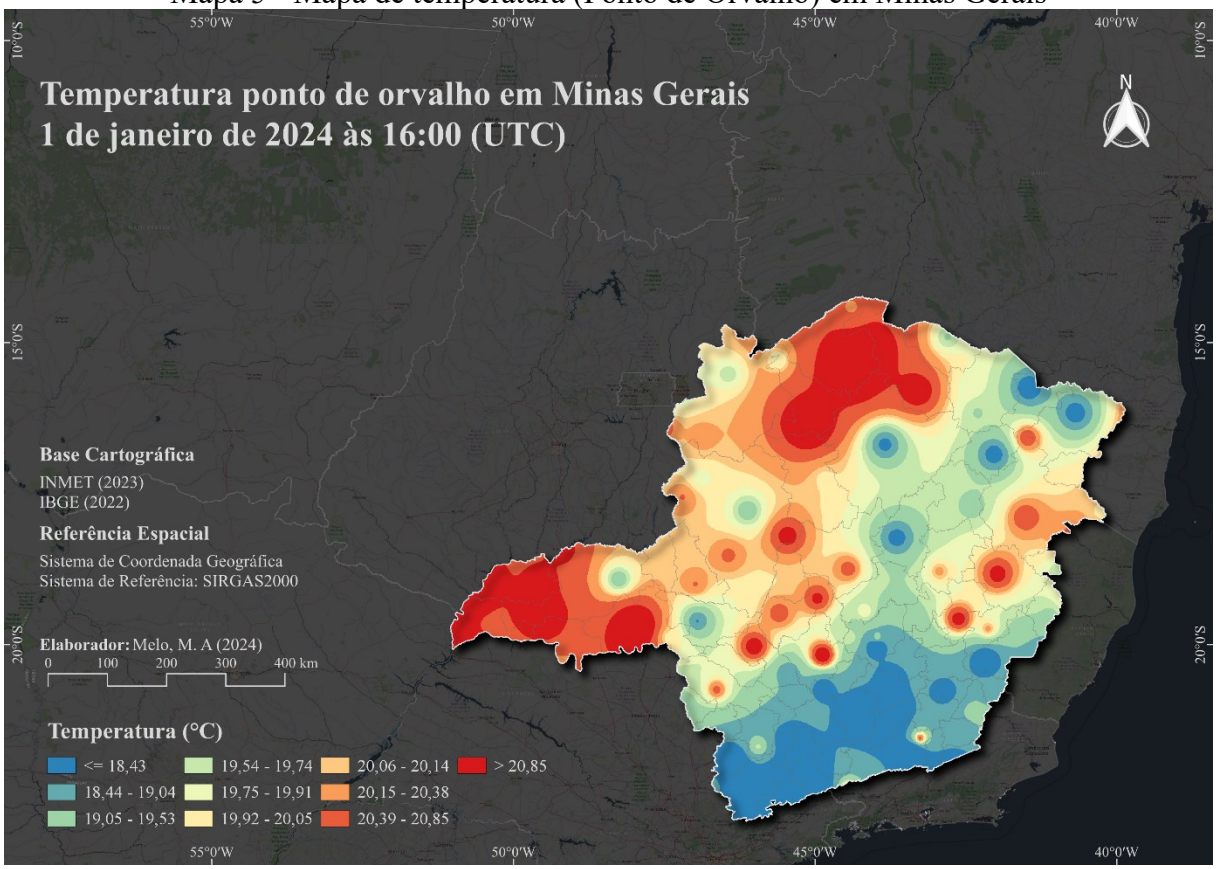
Fonte: O autor (2024)

Mapa 4 - Mapa de temperatura (Bulbo seco) em Minas Gerais



Fonte: O autor (2024)

Mapa 5 - Mapa de temperatura (Ponto de Orvalho) em Minas Gerais

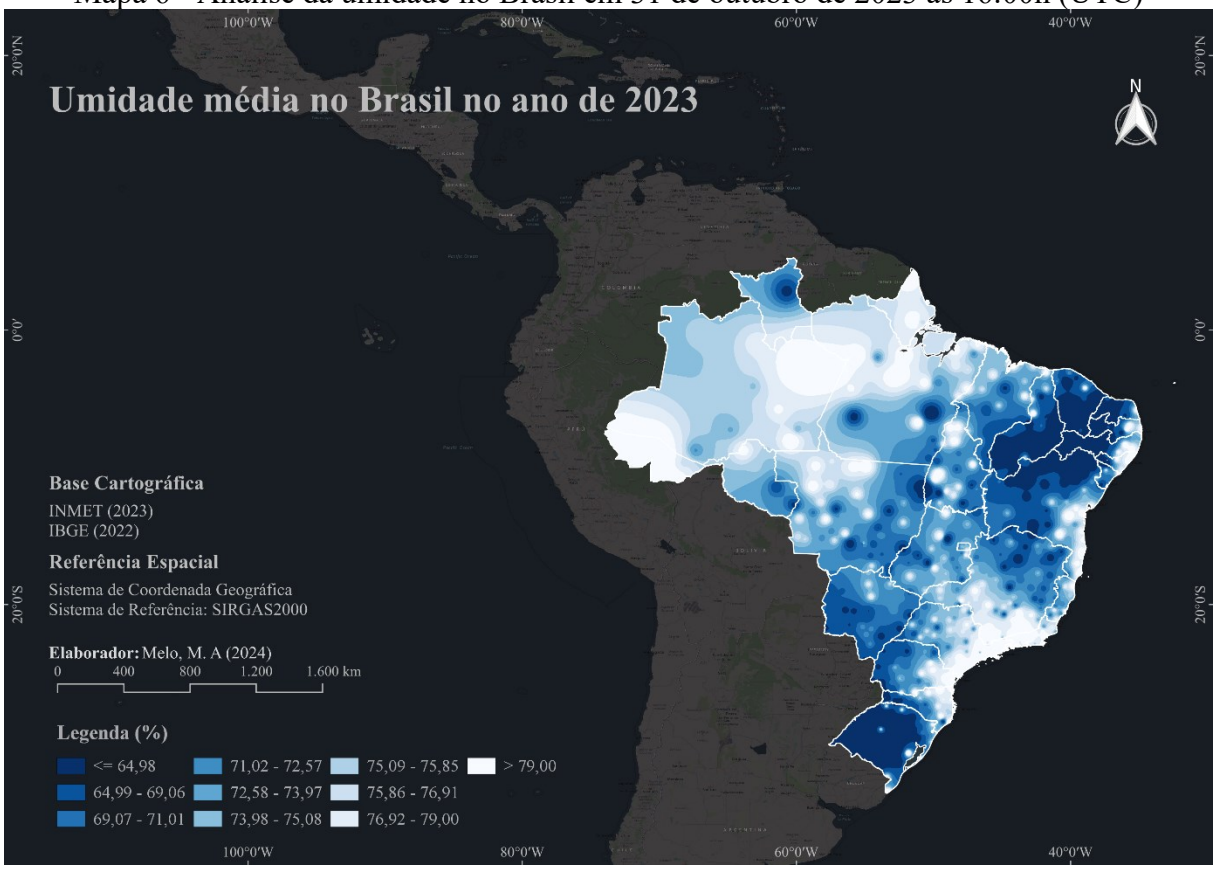


Fonte: O autor (2024)

4.2 UMIDADE RELATIVA DO AR

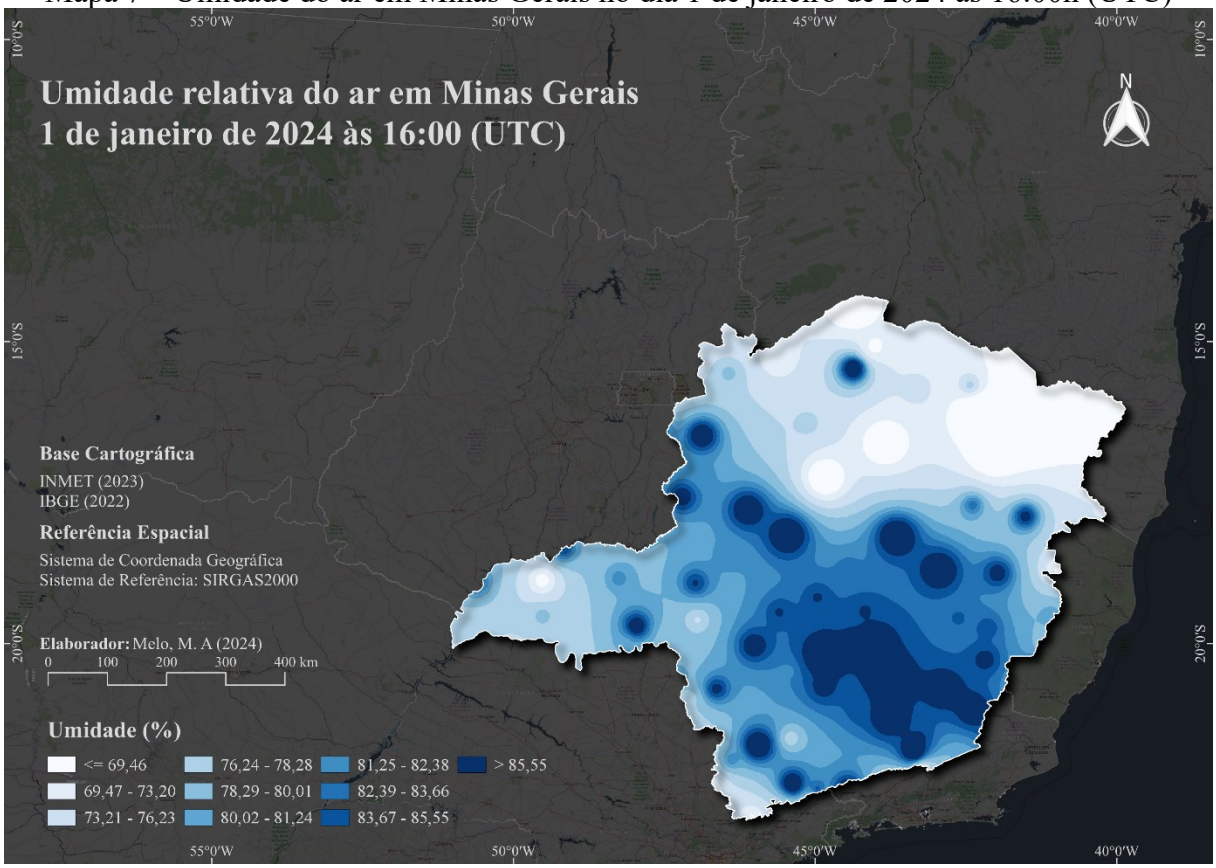
A umidade relativa do ar influencia diretamente a formação de nevoeiro, chuvas e tempestades. Na agricultura, a umidade relativa do ar desempenha um papel crítico no crescimento das plantas, afetando a taxa de evaporação da água do solo e a transpiração das plantas. Níveis ideais de umidade relativa são essenciais para maximizar a produção agrícola e minimizar o estresse hídrico nas culturas. Por outro lado, condições de umidade excessiva ou insuficiente podem levar a problemas como o desenvolvimento de doenças nas plantas, perda de colheitas e redução da qualidade dos produtos agrícolas. Portanto, a monitorização cuidadosa e a compreensão dos dados de umidade relativa do ar são cruciais tanto para os meteorologistas na previsão do tempo quanto para os agricultores na gestão eficaz das suas culturas. O Mapa 6 mostra a média umidade relativa do ar no ano de 2023 em todo território brasileiro, já o Mapa 7 mostra os mesmos dados de umidade relativa do ar para o dia 1 de janeiro de 2024 no estado de Minas Gerias.

Mapa 6 - Análise da umidade no Brasil em 31 de outubro de 2023 às 16:00h (UTC)



Fonte: O autor (2024)

Mapa 7 – Umidade do ar em Minas Gerais no dia 1 de janeiro de 2024 às 16:00h (UTC)

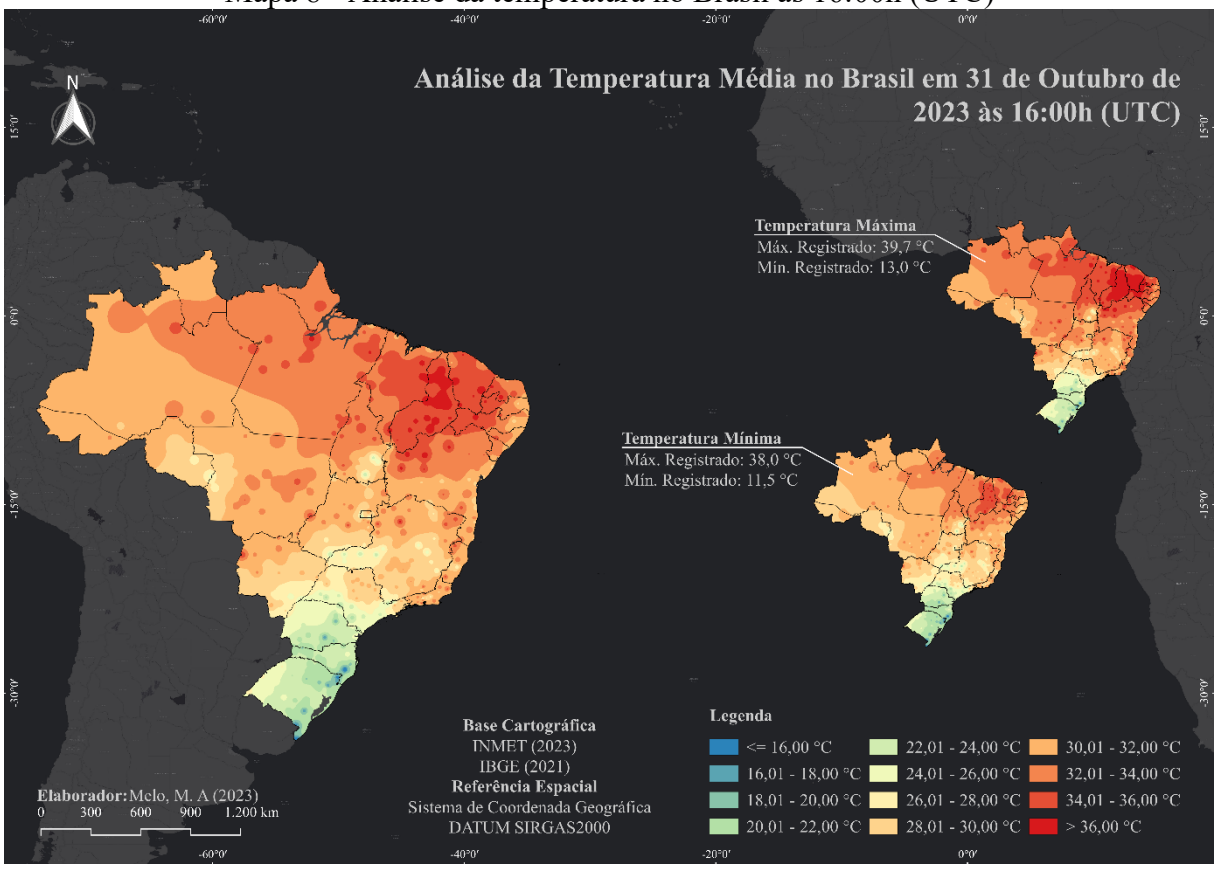


Fonte: O autor (2024)

4.3 VARIÁVEIS DE HORÁRIO ANTERIOR

Embora seja possível atribuir valores médios, máximos e mínimos com base nos dados disponibilizados pelo INMET, em alguns casos existem informações sobre variáveis correspondentes ao horário anterior. Portanto, para aqueles que buscam detalhes mais precisos, a aplicação desse método pode não ser a ideal. Isso evidencia a necessidade de utilizar informações em nível horário e considerar os fusos horários. Para ilustrar essa questão, o mapa 8 demonstra uma análise das temperaturas máximas e mínimas, bem como a média desses valores no dia 31 de outubro de 2023. Nessa análise, foi utilizado o horário de 16:00h UTC, que corresponde a 13:00h no horário local (GMT -3), e a informação retratada refere-se ao horário anterior. Logo a informação final é obtida no horário das 12:00h no fuso horário de Brasília.

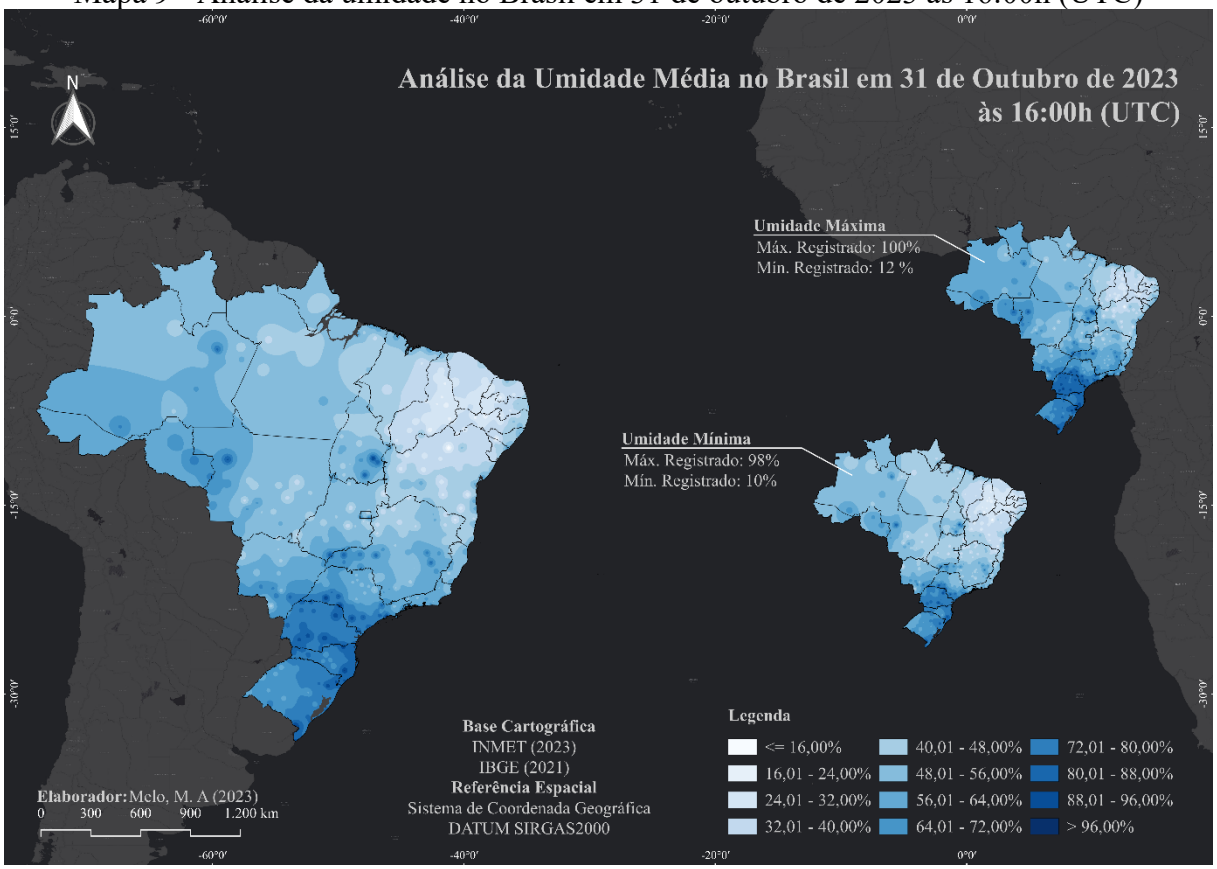
Mapa 8 - Análise da temperatura no Brasil às 16:00h (UTC)



Fonte: O autor (2024)

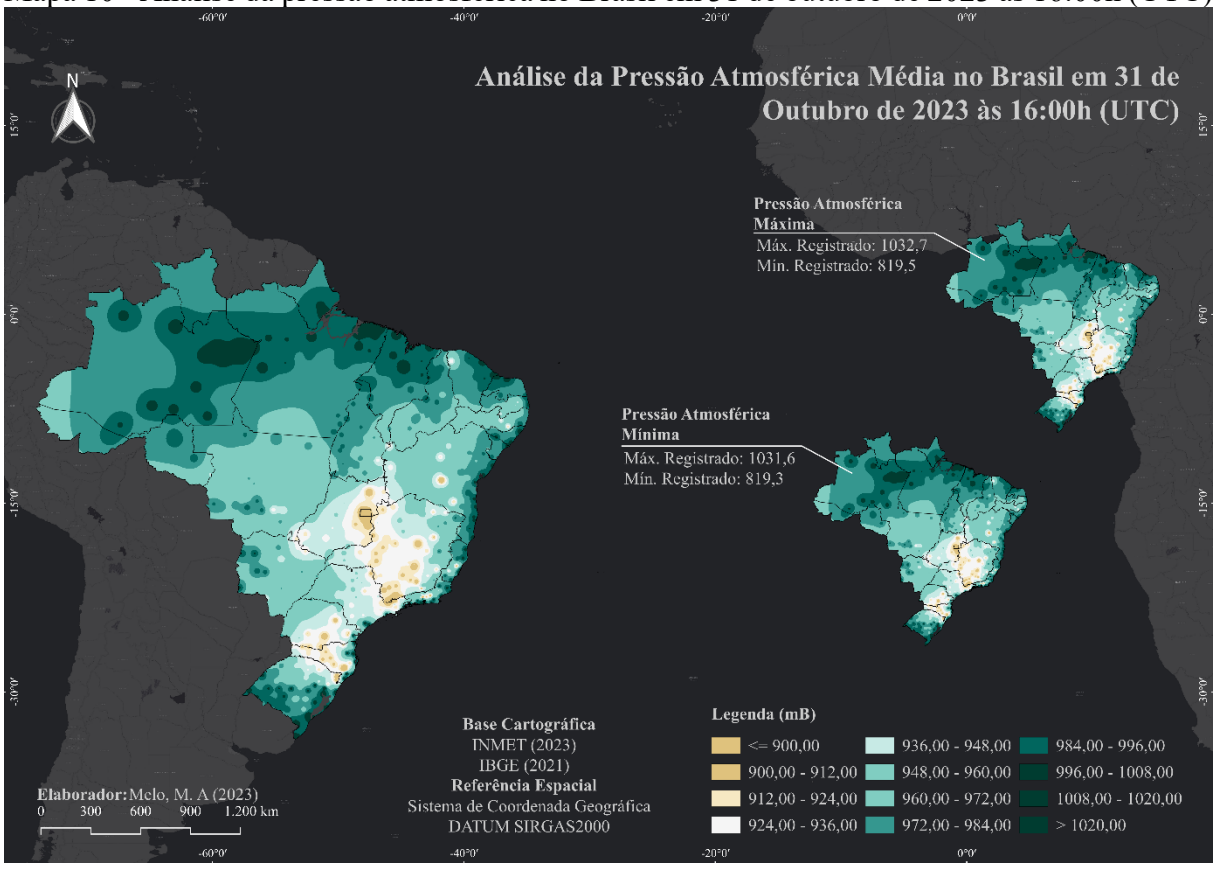
Além de analisar os valores anteriores das temperaturas máximas e mínimas, também temos a capacidade de realizar análises semelhantes para a pressão atmosférica e a umidade relativa do ar. Da mesma forma, foram criados mapas para ambas as variáveis, considerando a mesma data e hora do Mapa de temperatura (Mapa 8). Nos mapas 9 e 10, é possível visualizar as análises da pressão atmosférica e da umidade relativa do ar, respectivamente.

Mapa 9 - Análise da umidade no Brasil em 31 de outubro de 2023 às 16:00h (UTC)



Fonte: O autor (2024)

Mapa 10 - Análise da pressão atmosférica no Brasil em 31 de outubro de 2023 às 16:00h (UTC)

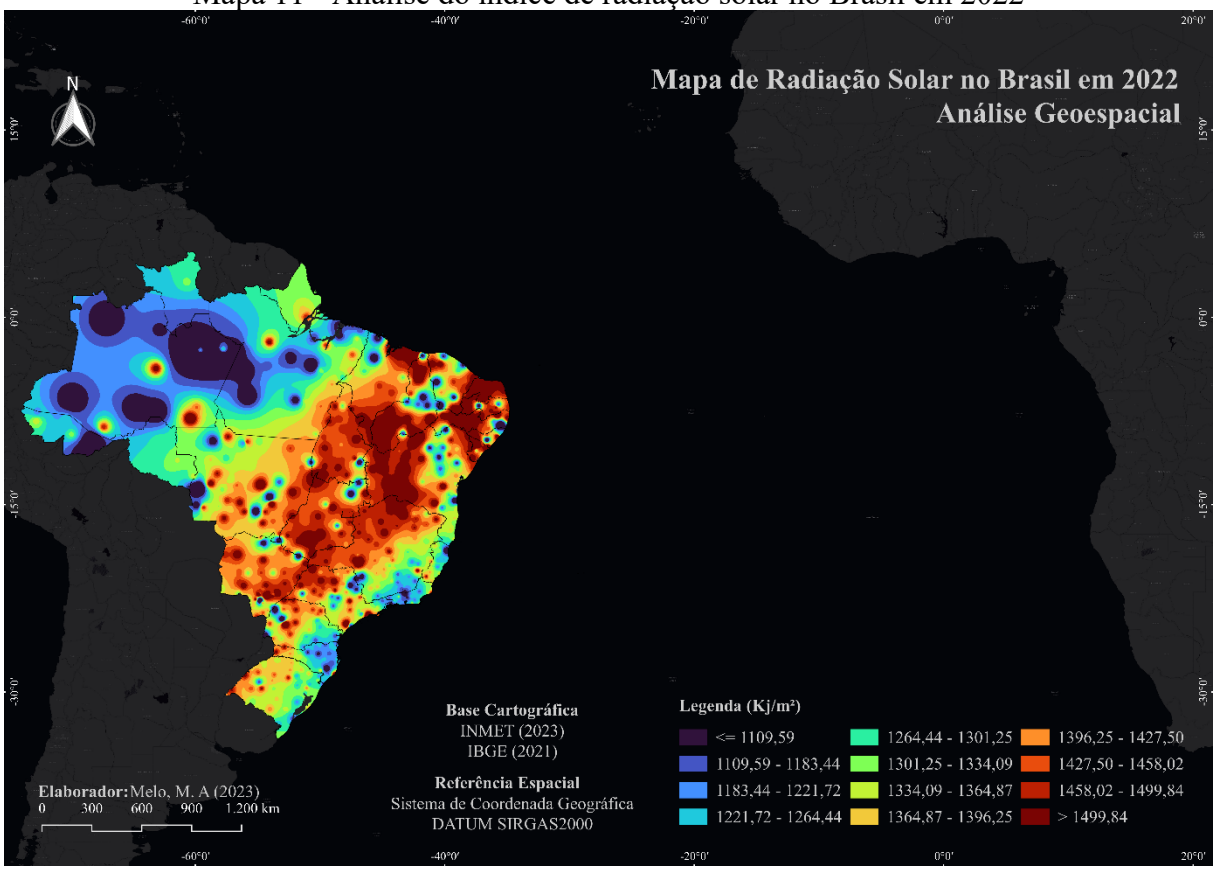


Fonte: O autor (2024)

4.4 RADIAÇÃO SOLAR

Outra informação de grande relevância que pode ser obtida a partir dos dados do INMET refere-se aos índices de radiação solar. Além de representar uma valiosa fonte de energia renovável, a radiação solar é tema de pesquisa em diversos estudos científicos, com destaque para a área agrícola. Com base nessa importância, foi elaborado um mapa que apresenta os dados de radiação solar, ilustrando a média ao longo do ano de 2022 em todo o território brasileiro. Conforme é ilustrado no mapa 11.

Mapa 11 - Análise do índice de radiação solar no Brasil em 2022

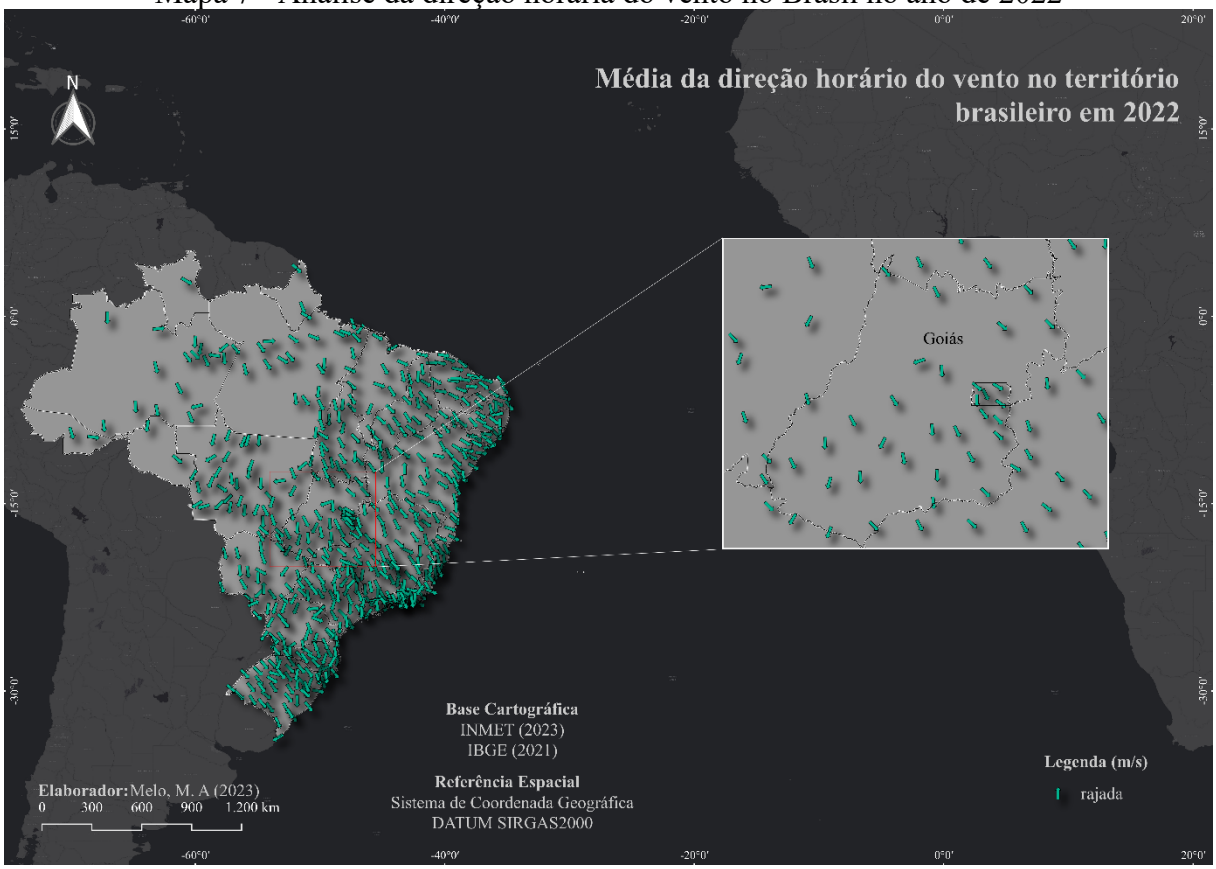


Fonte: O autor (2024)

4.5 ANÁLISE DO VENTO: DIREÇÃO, RAJADAS E VELOCIDADE

O vento desempenha um papel essencial no clima e nas condições meteorológicas de uma região. Para isso, é necessário analisar diversos aspectos, incluindo a direção horária do vento, as rajadas máximas e a velocidade horária. A direção horária do vento, expressada em graus, indica a orientação da qual o vento está soprando em um determinado momento, fornecendo informações sobre os padrões de circulação atmosférica. O Mapa 12 mostra a direção horário do vento no Brasil no ano de 2022.

Mapa 7 - Análise da direção horária do vento no Brasil no ano de 2022



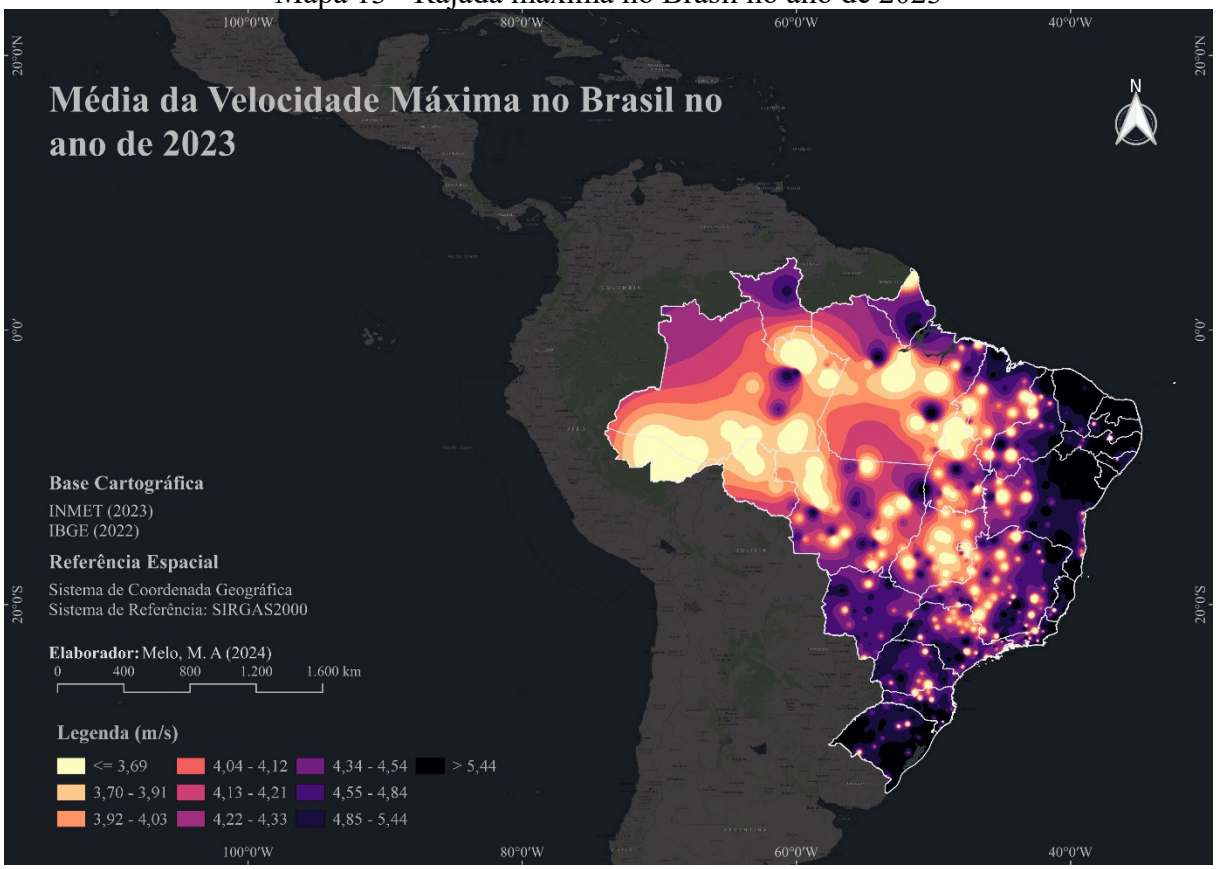
Fonte: O autor (2024)

Por outro lado, as rajadas máximas representam os picos de velocidade do vento em um período específico, destacando momentos de maior intensidade que podem afetar a estabilidade de estruturas agrícolas e ecossistemas sensíveis, além de serem importantes para a segurança das atividades ao ar livre. O Mapa 13 exibe a média das rajadas máximas em todo o Brasil durante o ano de 2023. Enquanto isso, o Mapa 14 oferece uma análise mais detalhada desses mesmos dados, focando especificamente no estado de Minas Gerais, proporcionando insights mais localizados e relevantes para a região.

Além disso, a velocidade horária do vento, medida em metros por segundo, fornece uma indicação clara da força do vento ao longo do tempo, sendo crucial para prever seu impacto em infraestruturas, vegetação e outras áreas sensíveis. O Mapa 15 mostra os dados obtidos sobre a velocidade horário do vento (m/s) no território brasileiro no ano de 2022. Já o Mapa 16 apresenta esses mesmos dados, com foco em Minas Gerais, proporcionando uma análise mais detalhada da área.

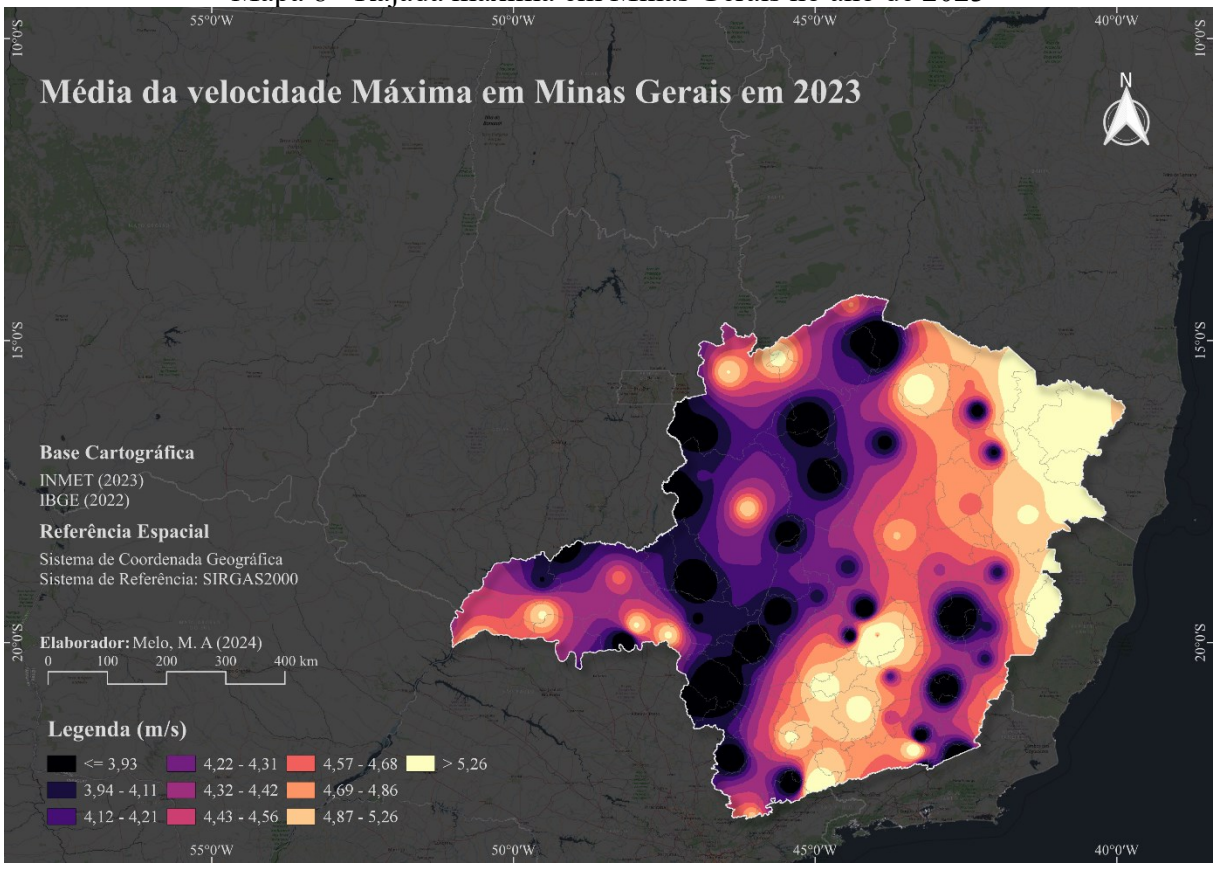
Portanto, a análise integrada desses aspectos do vento é essencial para uma compreensão completa das condições climáticas e para a tomada de decisões informadas em setores como agricultura, conservação ambiental e gestão de recursos naturais.

Mapa 13 - Rajada máxima no Brasil no ano de 2023



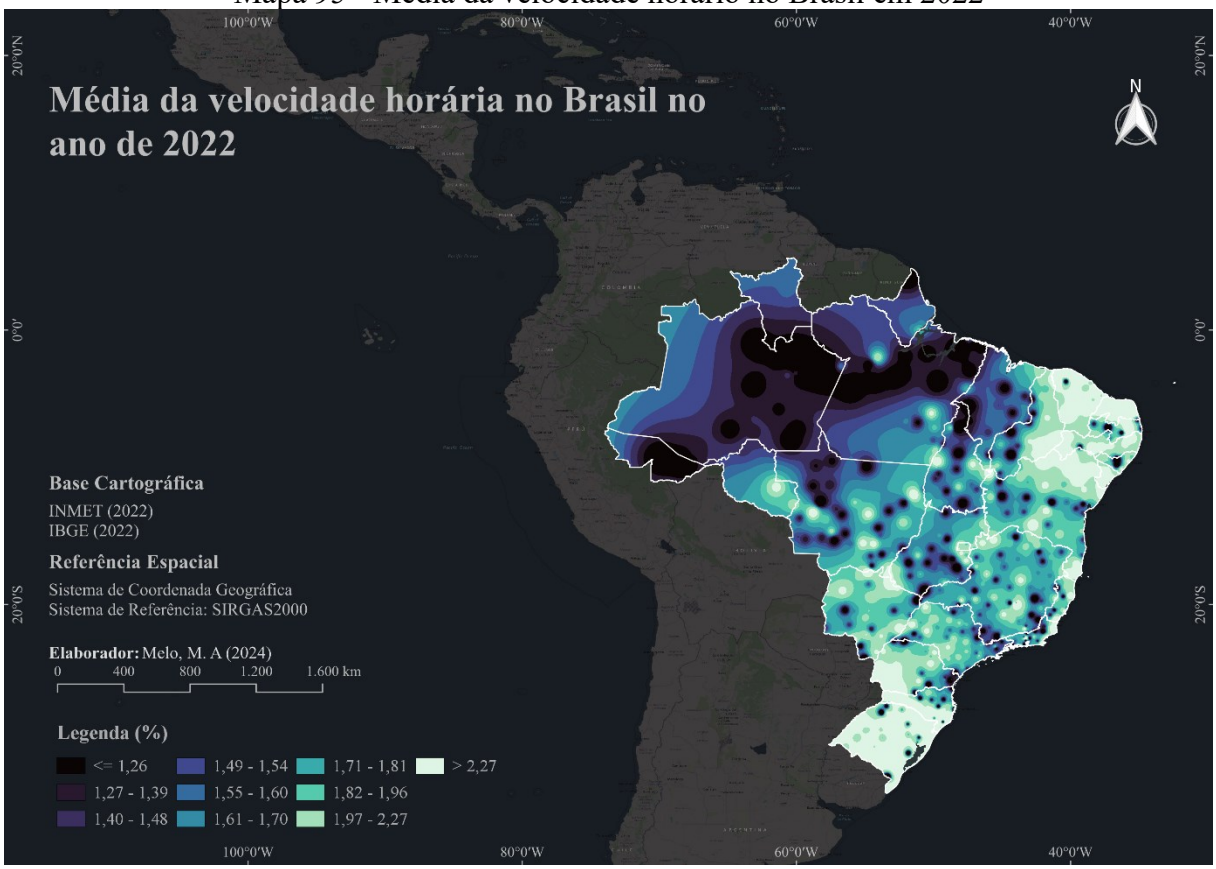
Fonte: O autor (2024)

Mapa 8 - Rajada máxima em Minas Gerais no ano de 2023



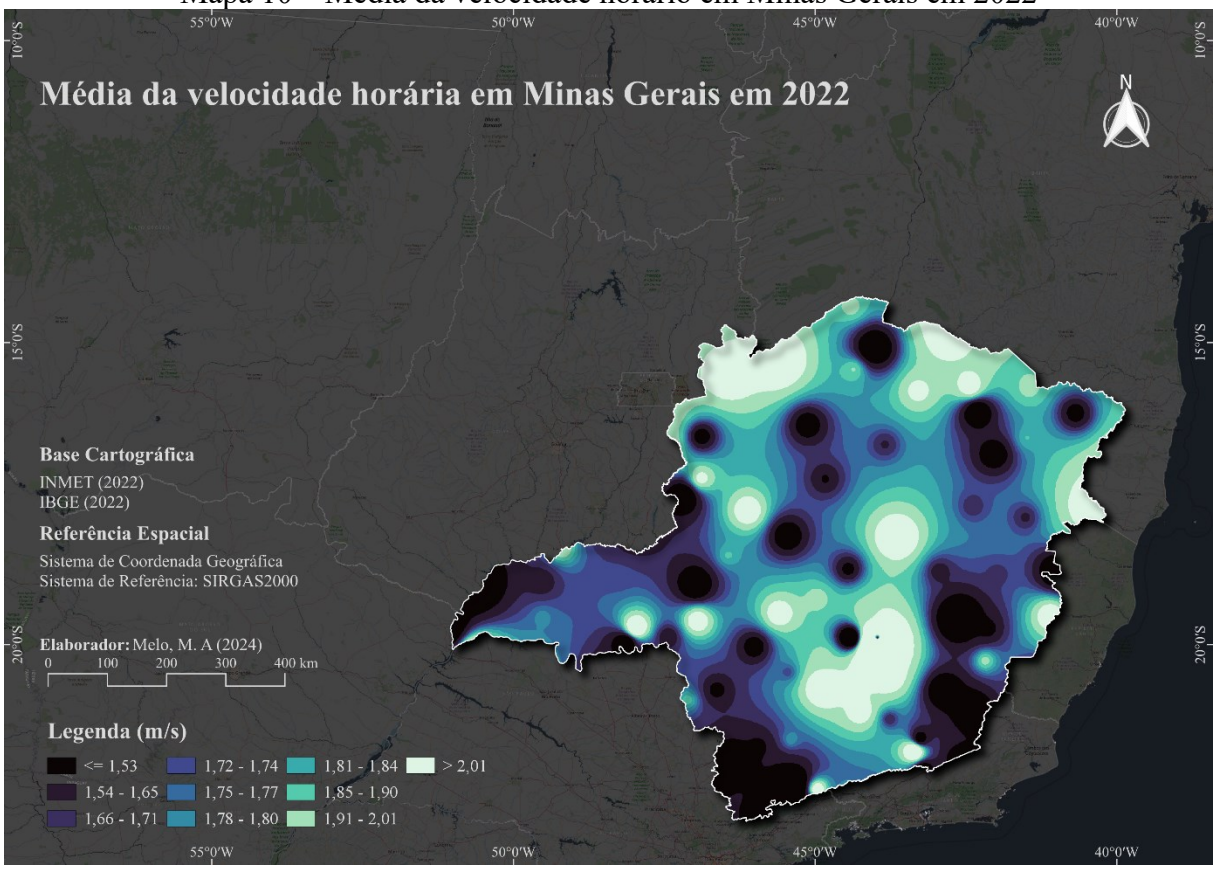
Fonte: O autor (2024)

Mapa 95 - Média da velocidade horário no Brasil em 2022



Fonte: O autor (2024)

Mapa 10 – Média da velocidade horário em Minas Gerais em 2022



Fonte: O autor (2024)

4.6 TESTE DE USUÁRIOS

Durante a fase de teste do plugin, 20 participantes da área de Engenharia de Agrimensura e Cartográfica foram selecionados para avaliar o desempenho e a precisão dos resultados, bem como medir a satisfação dos usuários ao utilizarem essa nova ferramenta para obter informações climáticas. A análise incluiu a identificação de possíveis falhas durante a coleta e processamento de dados, juntamente com o feedback sobre a experiência geral com o plugin. Essas informações foram essenciais para aprimorar o produto, implementando melhorias e sugestão de possíveis trabalhos futuros.

Após a utilização do plugin, foi elaborado 4 perguntas aos usuários, e que após isso foi solicitado o nível de satisfação com o produto atual. Entre essas perguntas foram.

1. **Facilidade de uso**: Em uma escala de 1 a 5, o quão fácil foi utilizar o plugin para acessar e tratar os dados meteorológicos do INMET no QGIS?
2. **Usabilidade da Interface**: Qual é a sua opinião sobre a interface do plugin em termos de design e usabilidade? As opções de navegação e organização dos recursos foram intuitivas e fáceis de entender?
3. **Eficiência e Desempenho**: Você notou alguma lentidão ou problemas de desempenho ao utilizar o plugin para processar grandes conjuntos de dados meteorológicos? Se sim, poderia especificar?
4. **Funcionalidades**: O plugin ofereceu as funcionalidades necessárias para realizar o tratamento de dados meteorológicos conforme suas expectativas? Há alguma funcionalidade adicional que você gostaria de ver implementada?
5. **Feedback adicional**: Por favor, compartilhe quaisquer outros comentários, sugestões ou dificuldades que você enfrentou ao usar o plugin. Se houver algum problema específico ou recurso que você gostaria de destacar, por favor, sinta-se à vontade para fornecer detalhes adicionais.

4.6.1 Facilidade de uso

O teste de usuário realizado no plugin revelou uma satisfação significativa por parte dos usuários em relação à facilidade e praticidade na obtenção de informações dos dados INMET. Os participantes destacaram a conveniência de ter acesso a uma gama mais ampla de variáveis climáticas, as quais anteriormente eram negligenciadas devido ao processo demorado de tratamento de dados em apenas uma estação meteorológica. Esta abordagem mais abrangente permitiu uma compreensão mais completa e detalhada das condições climáticas, atendendo às

necessidades de diferentes usuários e aplicações. O Quadro 3 apresenta o nível de satisfação com a facilidade de uso do plugin

Quadro 3: Nível de satisfação com a facilidade de uso

Avaliação da Facilidade de Uso				
Péssimo	Ruim	Regular	Bom	Ótimo
0	0	0	12	8

Fonte: O Autor (2024).

4.6.2 Usabilidade da interface

No geral, todos os usuários relataram que a interface é bastante intuitiva. Apesar de algumas sugestões terem sido recomendadas, o plugin apresenta uma grande similaridade com outras ferramentas no QGIS. O feedback em relação à interface do plugin pode ser encontrado no Quadro 4, que mostra a avaliação da usabilidade da interface do plugin.

Quadro 4: Avaliação na interface do plugin

Avaliação da usabilidade da interface do plugin				
Muito difícil	Difícil	Normal	Fácil	Muito fácil
0	0	6	9	5

Fonte: O Autor (2024).

4.6.3 Eficiência e Desempenho

Alguns usuários relataram que o processo de obtenção das informações era um tanto demorado. Além disso, houve casos em que o plugin apresentou travamentos, especialmente quando os usuários saíam da interface do aplicativo ou utilizavam outros programas simultaneamente.

A demora e os travamentos observados durante o teste do plugin podem ser atribuídos a vários fatores. Primeiramente, a vasta quantidade de arquivos contendo informações climáticas, com atualizações ocorrendo a cada hora e provenientes de diversas estações meteorológicas ao longo de diferentes anos. Esse volume de informações exige um tempo considerável para ser processado e integrado de forma coerente pelo plugin, resultando em uma aparente demora na obtenção dos resultados desejados pelos usuários.

É importante ressaltar que, apesar dos contratempos, após a conclusão do processamento, o software QGIS geralmente retoma seu funcionamento normal. Isso sugere

que os problemas de demora e travamento estão relacionados principalmente ao processamento dos dados pelo plugin, e não a questões estruturais do ambiente de execução.

4.6.4 Funcionalidades

De modo geral, as expectativas dos usuários em relação ao plugin foram atendidas. Ele fornece informações gerais dos dados retornando valores máximos, mínimos, médios ou somas conforme solicitado pelo usuário. Além disso, a opção de filtrar os dados com base no horário ou no percentual mínimo desejado adicionou um valor significativo para os usuários. Isso garante que apenas as estações que atendam ao critério mínimo estabelecido pelo usuário sejam consideradas para análise durante um determinado período. O recurso de intervalo horário é particularmente útil ao lidar com dados de radiação global, uma vez que as informações podem não estar disponíveis a cada hora devido à ausência de radiação durante certos períodos sem luz solar.

4.6.5 Feedback adicional

Embora o plugin tenha alcançado um nível significativo de satisfação, alguns feedbacks foram sugeridos. Entre eles:

- **Renomeação dos comandos no plugin**: Embora os comandos sejam compreensíveis, foram sugeridas possível para melhorar ainda mais a compreensão e facilidade de uso para os usuários.
- **Otimização do tempo de resposta**: Embora o tempo de processamento possa ser um pouco demorado, especialmente em intervalos de tempo maiores, foram sugeridas revisões no código ou um planejamento mais eficiente para reduzir o tempo de espera até a obtenção do resultado esperado.

5 CONSIDERAÇÕES FINAIS

O presente trabalho introduziu uma ferramenta que tem o potencial de facilitar a criação de produto através de bases de dados confiáveis, elaborando assim um plugin destinado a simplificar a utilização dos dados meteorológicos disponibilizado pelo INMET. Este plugin tem a capacidade de realizar uma variedade de análises meteorológicas, abrangendo todo o território nacional. Além disso, ele é uma ferramenta valiosa para análises temporais desde o início da implantação das estações automáticas no país.

É importante destacar que, embora os dados sejam confiáveis, a falta de informações em determinados intervalos pode representar um fator crítico a ser considerado. A perda de dados de uma estação de interesse pode comprometer a confiabilidade do produto, especialmente quando ocorre manutenção ou desativação dessa estação durante o período de interesse. Isso pode resultar em uma quantidade reduzida de informações disponíveis para análise, levando a estimativas menos precisas na avaliação da área em questão.

A opção de download automático é uma alternativa que visa otimizar tempo e recursos. No entanto, devido à natureza não oficial desse método, sua confiabilidade pode ser prejudicada no futuro, tornando-o inválido em datas subsequentes. Além disso, os downloads realizados dessa forma estarão limitados aos dados do mês anterior.

É importante ressaltar que, como já evidenciados mudanças no formato das tabelas, é possível que ocorram alterações no futuro. Portanto, é essencial realizar verificações periódicas nos arquivos disponibilizados para garantir a integridade e atualização das informações.

Prevê-se que as ferramentas desenvolvidas neste trabalho desempenharão um papel significativo na criação de novos mapas meteorológicos, com o objetivo de avaliar as condições climáticas em determinadas regiões e auxiliar no mapeamento de riscos de incêndio. O plugin INMET GeoSync estará disponível para download por qualquer usuário através do link fornecido no GitHub M-Melo5/plugin-inmet (github.com). Adicionalmente, há a expectativa de que este plugin seja em breve incluído no repositório de complementos do QGIS, compatível com a versão 3.0 ou superior.

6 TRABALHOS FUTUROS

Com base nas avaliações dos usuários e nas oportunidades de aprimoramento identificadas durante os testes, algumas melhorias podem ser implementadas no plugin como trabalhos futuros:

6.1 OTIMIZAÇÃO DO CÓDIGO

Conforme citado, o tempo de processamento é um fator crucial para o usuário, principalmente se o processo for repetitivo. Nesse sentido, otimizar o código existente ou implementar métodos para melhorar o tempo de resposta pode ser fundamental para eliminar possíveis travamentos e reduzir o tempo de espera. Uma abordagem eficaz para isso pode ser permitir que o usuário selecione apenas as estações de interesse dentro da região específica em que está trabalhando, eliminando assim o processamento desnecessário de dados não relevantes.

6.2 IMPLEMENTAÇÃO DAS ESTAÇÕES CONVENCIONAIS

O INMET possui dois tipos de estações, as convencionais e automáticas. No entanto, no momento atual, o plugin oferece suporte adequado apenas aos dados disponibilizados pelas estações automáticas. Isso se deve à maior facilidade de obtenção desses dados, além do intervalo de coleta de dados ser maior do que o das estações convencionais. Além disso, as estações automáticas fornecem mais informações climáticas em comparação com as convencionais. No entanto, existem variáveis climáticas que estão presentes em ambas as estações ou apenas nas convencionais. Portanto, uma implementação futura para receber dados dessas estações pode se tornar útil, uma vez que aumentaria a variedade de dados e informações disponíveis de acordo com a disponibilidade em ambas as estações.

6.3 DELIMITAR POR ESTADOS OU ÁREA DE INTERESSE.

Se o usuário deseja obter informações de determinadas estações em um intervalo pré-estabelecido, atualmente é necessário fazer o download de todas as estações com dados disponibilizados durante o período desejado. No entanto, uma alternativa para facilitar o acesso às informações desejadas seria delimitar as estações de acordo com o estado ou área de interesse do usuário. Dessa forma, apenas as estações que se encontram dentro da área estabelecida seriam retornadas, simplificando assim o processo de obtenção dos dados desejados.

6.4 BUFFERS

Outra implementação que poderia ser feita para auxiliar na delimitação por estado ou áreas seria a implementação de buffers. Essa ferramenta seria bastante útil para elaborar interpolações das variações climáticas. Isso se deve ao fato de que, para realizar uma interpolação que se aproxime da realidade, é fundamental ter informações tanto dentro quanto fora da área de interesse. A inclusão de buffers permitiria a coleta de dados além da área delimitada, aumentando assim a precisão na interpolação e garantindo uma maior confiabilidade dos resultados.

6.5 COLABORAÇÃO COM INMET

Um dos problemas que contribui para a demora no processamento dos arquivos é a necessidade de abrir todos os arquivos das estações a cada ano de interesse. Uma possível solução para esse problema seria uma colaboração em conjunto com o INMET, que poderia permitir a filtragem mais rápida dos dados por região e até mesmo fornecer informações georreferenciadas de cada estação. Essas informações já estão disponíveis no próprio site do INMET, porém, atualmente é impossível extrair esses dados através de códigos sem o

consentimento do INMET. Embora uma parceria desse tipo não seja viável no momento, é possível que, no futuro, essa colaboração se torne realidade, o que possibilitaria trabalhar com os dados disponibilizados de forma mais prática e rápida.

REFERÊNCIAS

BATISTA, Antonio Cartos. **Detecção de incêndios florestais por satélites**. Floresta, v. 34, n. 2, 2004. Disponível em: https://revistas.ufpr.br/floresta/article/download/2402/2010. Acesso em: 30 mai. 2023.

CAELUN. **Python e orientação a objetos**. Disponível em: https://www.caelum.com.br/apostila/apostila-python-orientacao-a-objetos.pdf. Acesso em 12 de abr. 2024

DATE, C. J. **Introdução a Sistemas de Bancos de Dados**. GEN LTC, 8ª edição. 2004. Acesso em: 13 de abril de 2024

DEPPE, Flavio et al. **Comparação de índice de risco de incêndio florestal com focos de calor no estado do Paraná.** Floresta, v. 34, n. 2, 2004. Disponível em: Open Journal Systems (ufpr.br). Acesso 29 mai. 2023.

DIETRICH, Jens; HOSKING, John; GILES, Jonathan. **A formal contract language for pluginbased software engineering**. In: 12th IEEE International Conference on Engineering Complex Computer Systems (ICECCS 2007). IEEE, 2007. p. 175-184. Disponível em: A Formal Contract Language for Plugin-based Software Engineering | IEEE Conference Publication | IEEE Xplore. Acesso em: 29 mai. 2023.

FREIRE, Sérgio; CARRÃO, Hugo; CAETANO, Mário R. Produção de cartografia de risco de incêndio florestal com recurso a imagens de satélite e dados auxiliares. Lisboa: IGP, 2002. Disponível em: https://www.researchgate.net/publication/242754146_Producao_de_Cartografia_de_Risco_de_Incendio_Florestal_com_Recurso_a_Imagens_de_Satelite_e_Dados_Auxiliares. Acesso em: 23 de mai. 2023.

GARRIDO, Filipe et al. **On the way for materializing cMOOC requirements: an experience dealing with plugins on Moodle Platform. In**: Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE). 2019. p. 566. Disponível em: On-the-way-for-materializing-cMOOC-requirements-an-experience-dealingwith-plugins-on-Moodle-Platform.pdf (researchgate.net). Acesso em: 30 mai. 2023

GISGeography. **30 Best GIS Software Applications [Rankings]**. [S. l.], 11 mar. 2023. Disponível em: https://gisgeography.com/best-gis-software/ . Acesso em: 1 jun. 2023.

HAMADA, Emília; GHINI, Raquel; GONÇALVES, R.R do V. **Efeito da mudança climática sobre problemas fitossanitários de plantas: metodologia de elaboração de mapas**. 2007. Disponível em: http://www.alice.cnptia.embrapa.br/alice/handle/doc/15932. Acesso em: 29 mai. 2023.

LIU, W. T. H. **Aplicações de Sensoriamento Remoto**. Campo Grande: UNIDERP, 2007. V. 1. 811p. Acesso em: 10 mai. 2023.

MACIEL, Girlene Figueiredo; AZEVEDO, P.V de; ANDRADE JÚNIOR, A.S de. **Impactos do aquecimento global no zoneamento de risco climático da soja no estado do Tocantins**. Revista Engenharia Ambiental, Espírito Santo do Pinhal, v. 6, n. 3, p. 141-154, 2009. Disponível em: https://www.researchgate.net/profile/Aderson-Andrade-Junior/publication/267994781_IMPACTOS_DO_AQUECIMENTO_GLOBAL_NO_ZONEAMENTO_DE_RISCO_CLIMATICO_DA_SOJA_NO_ESTADO_DO_TOCANTINS/links/5473cef20cf245eb436dba0e/IMPACTOS-DO-AQUECIMENTO-GLOBAL-NO-ZONEAMENTO-DE-RISCO-CLIMATICO-DA-SOJA-NO-ESTADO-DO-TOCANTINS.pdf. Acesso em: 29 mai. 2023.

OLIVEIRA, Aristeu Geovani; ASSUNÇÃO, Washington Luiz. **A importância dos dados das variáveis climáticas nas pesquisas em geografia: um estudo de caso empregando a precipitação pluviométrica**. Caminhos de Geografia, v. 10, n. 31, p. 147-157, 2009. Disponível em: https://seer.ufu.br/index.php/caminhosdegeografia/article/download/16078/9057 OLIVEIRA 2009. Acesso em: 28. Mai 2023.

PAPAJORGJI, Petraq. **A plug and play approach for developing environmental models**. Environmental Modelling & Software, v. 20, n. 10, p. 1353-1357, 2005. Disponível em: A plug and play approach for developing environmental models - ScienceDirect. Acesso em: 27 Mai. 2023

PEJOVIĆ, Milutin et al. **Solving a surveying problem by using R and QGIS: Setting out of a land expropriation zone**. Geonauka, v. 2, n. 2, p. 12-18, 2014. Disponível em: https://grafar.grf.bg.ac.rs/bitstream/handle/123456789/626/624.pdf?sequence=1&isAllowed=y. Acesso em: 29 mai. 2023.

PEREIRA, Lívia Maria Pederzini et al. **Análise comparativa de dados meteorológicos obtidos por estação convencional e automática em Londrina-PR**. Semina: Ciências

Agrárias, v. 29, n. 2, p. 299-305, 2008. Disponível em: https://www.redalyc.org/pdf/4457/445744088007.pdf. Acesso em: 13 abril 2024.

ROB, Peter; CORONEL, Carlos. **Sistemas de banco de dados**. **Projeto, implementação e gerenciamento**, 2011. Disponível em: sistemas_de_banco_de_dados-libre.pdf (d1wqtxts1xzle7.cloudfront.net). Acesso em: 13 abril 2024.

ROSA, Roberto. **Geotecnologias na geografia aplicada**. Revista do Departamento de Geografia, v. 16, p. 81-90, 2005. Disponível em: https://www.revistas.usp.br/rdg/article/view/47288/51024. Acesso em: 29 mai. 2023.

ROSA, Roberto. **Introdução ao geoprocessamento**. UFU: Apostila. Uberlândia, 2013. Disponível em: https://edisciplinas.usp.br/pluginfile.php/5551878/mod_resource/content/2/Apostila_Geop_rrosa.pdf. Acesso em: 24 mai. 2023.

SAABITH, Sayeth; VINOTHRAJ, T.; FAREEZ, M. **A review on Python libraries and Ides for Data Science**. Int. J. Res. Eng. Sci, v. 9, n. 11, p. 36-53, 2021. Disponível em: A-Review-on-Python-Libraries-and-IDEs-for-Data-Science.pdf (researchgate.net). Acesso: 15 abril de 2024.

TANNER, Bertrand D. **Automated weather stations**. Remote Sensing Reviews, v. 5, n. 1, p. 73-98, 1990. Disponível em: https://www.tandfonline.com/doi/abs/10.1080/02757259009532123. Acesso em: 25 maio. 2023

TREVISAN, Rodrigo G.; MOLIN, José P. **Sistemas de informação geográfica (sig) para agricultura de precisão**. 2014. Disponível em: https://www.agriculturadeprecisao.org.br/wp-content/uploads/2020/04/BoletimT%C3%A9cnico01-SIG_OUT_2014.pdf. Acesso em: 13 abr. 2024.

TRINDADE, Edgard. **ArcGIS x QGIS: Qual é o Melhor Software de SIG?.** In: ArcGIS x QGIS: Qual é o Melhor Software de SIG?. [S. l.], 6 dez. 2021. Disponível em: https://geosense.net.br/2021/12/06/arcgis-x-qgis-qual-e-o-melhor-software-de-sig/. Acesso em: 1 jun. 2023.

URANO, Laís. **Pycharm ou VSCode**: qual ferramenta escolher para programar em Python?. [*S. l.*], 15 set. 2023. Disponível em: https://www.alura.com.br/artigos/pycharm-vscode-programar-python . Acesso em: 9 abr. 2024.

VETTORAZZI, Carlos A.; FERRAZ, Silvio F. de B. **Uso de sistemas de informações geográficas aplicados à prevenção e combate a incêndios em fragmentos florestais**. Série Técnica IPEF, v. 12, n. 32, p. 111-115, 1998. Disponível em: https://www.ipef.br/PUBLICACOES/stecnica/nr32/cap11.pdf. Acesso em: 29 mai. 2023.

APÊNDICE

```
# -*- coding: utf-8 -*-
"""
/***************************************************************************
 DadosMeteorologicos
                                 A QGIS plugin
 Transforme dados meteorológicos em informações espaciais de maneira eficaz e faça análises geoespaciais com facilidade. Ideal para tomada de decisões informadas relacionadas ao clima e ao tempo.
 Generated by Plugin Builder: http://g-sherman.github.io/Qgis-Plugin-Builder/
                             -------------------
        begin                : 2023-10-23
        git sha              : $Format:%H$
        copyright            : (C) 2023 by Matheus Albuquerque de Melo
        email                : matheusmello0@outlook.com

 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/
"""
from qgis.PyQt.QtCore import QVariant
from qgis.core import QgsFeature, QgsField, QgsFields, QgsPoint, QgsVectorLayer, QgsVectorFileWriter
from PyQt5.QtWidgets import QFileDialog
from qgis.PyQt.QtCore import QSettings, QTranslator, QCoreApplication
```

```
from qgis.PyQt.QtGui import QIcon
from qgis.PyQt.QtWidgets import QAction, QFileDialog
from qgis.core import *
from PyQt5.QtCore import QTime, Qt, QDateTime, QDate
from PyQt5.QtWidgets import QMessageBox
from .resources import *
from .INMET_dialog import DadosMeteorologicosDialog
import os.path
import processing
import sys, os
from osgeo import ogr
from PyQt5.QtWidgets import QDialog, QDateTimeEdit
import pandas as pd
from datetime import datetime, timedelta
import geopandas as gpd
from shapely.geometry import Point
import pyproj
import sys
from PyQt5.QtWidgets import QDialog, QApplication
from PyQt5.uic import loadUi
from PyQt5.QtCore import QThread, pyqtSignal
from pathlib import Path
import requests
import zipfile
import os

class BackgroundTask(QThread):
    progress_updated = pyqtSignal(int)
    finished = pyqtSignal()
    def __init__(self, parent=None):
        super().__init__(parent)
    def run(self):
        total_steps = 100
class DadosMeteorologicos:
```

```
"""QGIS Plugin Implementation."""
def __init__(self, iface):
    """Constructor.
    :param iface: An interface instance that will be passed to this class
        which provides the hook by which you can manipulate the QGIS
        application at run time.
    :type iface: QgsInterface
    """
    # Save reference to the QGIS interface
    self.iface = iface
    # initialize plugin directory
    self.plugin_dir = os.path.dirname(__file__)
    # initialize locale
    locale = QSettings().value('locale/userLocale')[0:2]
    locale_path = os.path.join(
        self.plugin_dir,
        'i18n',
        'DadosMeteorologicos_{}.qm'.format(locale))
    if os.path.exists(locale_path):
        self.translator = QTranslator()
        self.translator.load(locale_path)
        QCoreApplication.installTranslator(self.translator)
    # Declare instance attributes
    self.actions = []
    self.menu = self.tr(u'&INMET GeoSync')
    # Check if plugin was started the first time in current QGIS session
    # Must be set in initGui() to survive plugin reloads
# noinspection PyMethodMayBeStatic
def tr(self, message):
    """Get the translation for a string using Qt translation API.
    We implement this ourselves since we do not inherit QObject.
    :param message: String for translation.
    :type message: str, QString
    :returns: Translated version of message.
```

```
        :rtype: QString
        """
        # noinspection PyTypeChecker,PyArgumentList,PyCallByClass
        return QCoreApplication.translate('DadosMeteorologicos', message)
    def add_action(
        self,
        icon_path,
        text,
        callback,
        enabled_flag=True,
        add_to_menu=True,
        add_to_toolbar=True,
        status_tip=None,
        whats_this=None,
        parent=None):
        """Add a toolbar icon to the toolbar.

        :param icon_path: Path to the icon for this action. Can be a resource
            path (e.g. ':/plugins/foo/bar.png') or a normal file system path.
        :type icon_path: str
        :param text: Text that should be shown in menu items for this action.
        :type text: str
        :param callback: Function to be called when the action is triggered.
        :type callback: function
        :param enabled_flag: A flag indicating if the action should be enabled
            by default. Defaults to True.
        :type enabled_flag: bool
        :param add_to_menu: Flag indicating whether the action should also
            be added to the menu. Defaults to True.
        :type add_to_menu: bool
        :param add_to_toolbar: Flag indicating whether the action should also
            be added to the toolbar. Defaults to True.
        :type add_to_toolbar: bool
        :param status_tip: Optional text to show in a popup when mouse pointer
```

```
            hovers over the action.
        :type status_tip: str
        :param parent: Parent widget for the new action. Defaults None.
        :type parent: QWidget
        :param whats_this: Optional text to show in the status bar when the
            mouse pointer hovers over the action.
        :returns: The action that was created. Note that the action is also
            added to self.actions list.
        :rtype: QAction
        """

        icon = QIcon(icon_path)
        action = QAction(icon, text, parent)
        action.triggered.connect(callback)
        action.setEnabled(enabled_flag)
        if status_tip is not None:
            action.setStatusTip(status_tip)
        if whats_this is not None:
            action.setWhatsThis(whats_this)
        if add_to_toolbar:
            # Adds plugin icon to Plugins toolbar
            self.iface.addToolBarIcon(action)
        if add_to_menu:
            self.iface.addPluginToMenu(
                self.menu,
                action)
        self.actions.append(action)
        return action
    def initGui(self):
        """Create the menu entries and toolbar icons inside the QGIS GUI."""
        icon_path = ':/plugins/INMET/icon.png'
        self.add_action(
            icon_path,
            text=self.tr(u'INMET GeoSync'),
```

```
            callback=self.run,
            parent=self.iface.mainWindow())
        # will be set False in run()
        self.first_start = True


    def unload(self):
        """Removes the plugin menu item and icon from QGIS GUI."""
        for action in self.actions:
            self.iface.removePluginMenu(
                self.tr(u'&INMET GeoSync'),
                action)
            self.iface.removeToolBarIcon(action)
    def pasta(self):
        """Selecionar o diretório aonde está a pasta e subpastas das estações a serem tratadas"""
        pasta_selecionada = QFileDialog.getExistingDirectory(caption="Escolha a pasta dos arquivos",
        options=QFileDialog.ShowDirsOnly)
        self.dlg.caminho_select.setText(pasta_selecionada)
    def definirsaida(self):
        salvar = str(QFileDialog.getSaveFileName(caption="Escolha a pasta dos arquivos",
        filter="Shapefile (*.shp)")[0])
        self.dlg.dir_saida.setText(salvar)
    def validar_horarios(self):
        hora_inicial = self.dlg.hora_inicial.time()
        hora_final = self.dlg.hora_final.time()
        if hora_inicial > hora_final:
            self.dlg.hora_inicial.setTime(hora_final)
            QMessageBox.warning(None, "Aviso", "A hora inicial foi ajustada para não ser posterior ao horário final.")
    def validar_datas(self):
        data_inicial = self.dlg.dt_inicial.date()
        data_final = self.dlg.dt_final.date()
        if data_inicial > data_final:
```

```
            QMessageBox.warning(None, "Aviso", "A data inicial foi ajustada para não ser posterior à data final.")
            self.dlg.dt_inicial.setDate(data_final)
        def percorrer_arquivo(self):
            salvar_dados = pd.DataFrame(columns=['ESTACAO'])
            hora_inicial = self.dlg.hora_inicial.time()
            hora_final = self.dlg.hora_final.time()
            percentual = self.dlg.percentual.value()
            fuso_selecionado = self.dlg.fuso.currentText()
            variavel = self.dlg.variavel.currentText()
            fusos_horarios = ["-12:00 (UTC)", "-11:00 (UTC)", "-10:00 (UTC)", "-09:00 (UTC)", "-08:00 (UTC)", "-07:00 (UTC)",
                              "-06:00 (UTC)", "-05:00 (UTC)", "-04:00 (UTC)", "-03:00 (UTC)", "-02:00 (UTC)", "-01:00 (UTC)",
                              " 00:00 (UTC)", "+01:00 (UTC)", "+02:00 (UTC)", "+03:00 (UTC)", "+04:00 (UTC)", "+05:00 (UTC)",
                              "+06:00 (UTC)", "+07:00 (UTC)", "+08:00 (UTC)", "+09:00 (UTC)", "+10:00 (UTC)", "+11:00 (UTC)",
                              "+12:00 (UTC)"]
            for numero, horario in enumerate(fusos_horarios):
                if horario == fuso_selecionado:
                    ajustar_fuso = (12 - numero) * -1
                    break
            def lista_dados():
                verifica = []
                for index in range(self.dlg.list_dados.count()):
                    item = self.dlg.list_dados.itemText(index)
                    checked = self.dlg.list_dados.itemCheckState(index)
                    if checked == Qt.Checked:
                        verifica.append(item)
                return verifica
            def cabecalho(diretorio):
                col = ["Atributo", "Valor", '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15',
                       '16', '17', '18']
```

```
        header_info = pd.read_csv(diretorio, sep=';', nrows=7, encoding='ISO-8859-1', decimal=',', header=None, names=col)
        deletar = ['2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15', '16', '17', '18']
        header_info = header_info.drop(columns=deletar)
        if header_info.shape[0] != 7:
            return False
        header_info = header_info.T
        header_info.columns = ["REGIAO", "UF", "ESTACAO", "CODIGO", "LATITUDE", "LONGITUDE", "ALTITUDE"]
        header_info = header_info[1:]
        colunas_numericas = ["LATITUDE", "LONGITUDE", "ALTITUDE"]
        header_info[colunas_numericas] = header_info[colunas_numericas].apply(lambda x: x.str.replace(',', '.'))
        return header_info
    def verificar_arquivo(arquivo, fuso=ajustar_fuso):
        informacoes = ["DATA (YYYY-MM-DD)",
            "HORA (UTC)",
            "PRECIPITAÇÃO TOTAL, HORÁRIO (mm)",
            "PRESSAO ATMOSFERICA AO NIVEL DA ESTACAO, HORARIA (mB)",
            "PRESSÃO ATMOSFERICA MAX.NA HORA ANT. (AUT) (mB)",
            "PRESSÃO ATMOSFERICA MIN. NA HORA ANT. (AUT) (mB)",
            "RADIACAO GLOBAL (KJ/m²)",
            "TEMPERATURA DO AR - BULBO SECO, HORARIA (°C)",
            "TEMPERATURA DO PONTO DE ORVALHO (°C)",
            "TEMPERATURA MÁXIMA NA HORA ANT. (AUT) (°C)",
            "TEMPERATURA MÍNIMA NA HORA ANT. (AUT) (°C)",
            "TEMPERATURA ORVALHO MAX. NA HORA ANT. (AUT) (°C)",
            "TEMPERATURA ORVALHO MIN. NA HORA ANT. (AUT) (°C)",
            "UMIDADE REL. MAX. NA HORA ANT. (AUT) (%)",
            "UMIDADE REL. MIN. NA HORA ANT. (AUT) (%)",
            "UMIDADE RELATIVA DO AR, HORARIA (%)",
            "VENTO, DIREÇÃO HORARIA (gr) (° (gr))",
            "VENTO, RAJADA MAXIMA (m/s)",
            "VENTO, VELOCIDADE HORARIA (m/s)"]
```

```
        colunas_mapeadas = {'Data': 'DATA (YYYY-MM-DD)', 'Hora UTC': 'HORA (UTC)',
                            'RADIACAO GLOBAL (Kj/m²)': 'RADIACAO GLOBAL (KJ/m²)'}
        arquivo.rename(columns=colunas_mapeadas, inplace=True)
        colunas_faltantes = [coluna for coluna in informacoes if coluna not in
set(arquivo.columns)]
        if not colunas_faltantes:
            if arquivo.shape[1] == 20:
                del arquivo['Unnamed: 19']
            arquivo.columns = [
                "DATA", "HORA", "PRECIPITAÇÃO TOTAL, HORÁRIO (mm)",
                "PRESSAO ATMOSFERICA AO NIVEL DA ESTACAO, HORARIA (mB)",
                "PRESSÃO ATMOSFERICA MAX.NA HORA ANT. (AUT) (mB)",
                "PRESSÃO ATMOSFERICA MIN. NA HORA ANT. (AUT) (mB)",
                "RADIACAO GLOBAL (KJ/m²)",
                "TEMPERATURA DO AR - BULBO SECO, HORARIA (°C)",
                "TEMPERATURA DO PONTO DE ORVALHO (°C)",
                "TEMPERATURA MÁXIMA NA HORA ANT. (AUT) (°C)",
                "TEMPERATURA MÍNIMA NA HORA ANT. (AUT) (°C)",
                "TEMPERATURA ORVALHO MAX. NA HORA ANT. (AUT) (°C)",
                "TEMPERATURA ORVALHO MIN. NA HORA ANT. (AUT) (°C)",
                "UMIDADE REL. MAX. NA HORA ANT. (AUT) (%)",
                "UMIDADE REL. MIN. NA HORA ANT. (AUT) (%)",
                "UMIDADE RELATIVA DO AR, HORARIA (%)",
                "VENTO, DIREÇÃO HORARIA (gr) (° (gr))",
                "VENTO, RAJADA MAXIMA (m/s)",
                "VENTO, VELOCIDADE HORARIA (m/s)"]

            def padronizar_data(data):
                formatos = ['%d/%m/%Y', '%Y/%m/%d', '%Y-%m-%d', '%d-%m-%Y', '%d-%m-
%Y %H:%M:%S']
                for formato in formatos:
                    try:
                        return datetime.strptime(data, formato).strftime('%Y-%m-%d')
                    except ValueError:
```

```
                pass
            return data
        arquivo['DATA'] = arquivo['DATA'].apply(padronizar_data)
        arquivo['DATA'] = pd.to_datetime(arquivo['DATA'])
        arquivo['HORA'] = [f'{hora[:2]}:00' for hora in arquivo['HORA']]
        arquivo['HORA'] = pd.to_datetime(arquivo['HORA'], format='%H:%M').dt.time
        arquivo.to_csv(r"C:\PyQGIS\antes.csv")
        arquivo['DATA'] = [datetime.combine(data, hora) - timedelta(hours=fuso) for data,
hora
                           in zip(arquivo['DATA'], arquivo['HORA'])]
        arquivo['HORA'] = arquivo['DATA'].dt.time
        arquivo.to_csv(r"C:\PyQGIS\depois.csv")
        arquivo.replace(-9999, None, inplace=True)
        return arquivo
    elif:
        print('Arquivo Alterado!')
def tabela_principal(diretorio, salvar_dados):
    try:
        data_inicial = self.dlg.dt_inicial.dateTime().toString('yyyy-MM-dd hh:mm:ss')
        data_final = self.dlg.dt_final.dateTime().toString('yyyy-MM-dd hh:mm:ss')

        arquivo = pd.read_csv(diretorio, sep=';', header=8, encoding='ISO-8859-1',
decimal=',')
        arquivo = verificar_arquivo(arquivo)
        arquivo = arquivo.loc[(arquivo['DATA'] >= data_inicial) & (arquivo['DATA'] <=
data_final)]
        arquivo = arquivo.loc[(arquivo['HORA'] >= hora_inicial) & (arquivo['HORA'] <=
hora_final)]
        if arquivo.shape[0] != 0:
            info_estacao = cabecalho(diretorio)
            colunas = lista_dados()
            for coluna in colunas:
                num_dados_totais = (arquivo['DATA'].count())
                num_vazios = arquivo[coluna].isnull().sum()
```

```
                num_dados = num_dados_totais - num_vazios
                arquivo[coluna] = arquivo[coluna].astype(float)
                if (num_dados * 100 / num_dados_totais) >= percentual:
                    if variavel == "Máximo":
                        info_estacao[coluna] = arquivo[coluna].max()
                    elif variavel == "Mínimo":
                        info_estacao[coluna] = arquivo[coluna].min()
                    elif variavel == "Média":
                        info_estacao[coluna] = arquivo[coluna].mean()
                    elif variavel == "Soma":
                        info_estacao[coluna] = arquivo[coluna].sum()
                elif:
                    info_estacao[coluna] = None
            arquivo.to_csv(r"C:\PyQGIS\analise.csv", index=False)
            salvar_dados = pd.concat([salvar_dados, info_estacao], ignore_index=True)
            return salvar_dados
        elif:
            return salvar_dados
    except pd.errors.ParserError:
        print(diretorio, '\n')
        print('Erro ao analisar o arquivo CSV')
    except pd.errors.EmptyDataError:
        print(diretorio, '\n')
        print('O arquivo está vazio')
    except Exception as e:
        print(diretorio, '\n')
        print(f'Erro inesperado: {str(e)}')
    return salvar_dados
diretorio_pasta = self.dlg.caminho_select.text()
if self.dlg.download.isChecked():
    data_inicial = self.dlg.dt_inicial.date().year()
    data_final = self.dlg.dt_final.date().year()
    def download_zip_file(url, save_path):
        with requests.get(url, stream=True) as response:
```

```
            response.raise_for_status()
            with open(save_path, "wb") as file:
                for chunk in response.iter_content(chunk_size=8192):
                    file.write(chunk)
        def extract_zip_file(zip_file_path, extraction_path):
            with zipfile.ZipFile(zip_file_path, "r") as zip_ref:
                zip_ref.extractall(extraction_path)
        def download_arquivos_inmet(data_inicial, data_final):
            for ano in range(data_inicial, data_final+1):
                url = f"https://portal.inmet.gov.br/uploads/dadoshistoricos/{ano}.zip"
                save_path = f"{ano}.zip"
                downloads_path = Path.home() / "Downloads"
                extraction_path = downloads_path / "INMET_DOWNLOAD" / f"{ano}"


                if not os.path.exists(extraction_path):
                    try:
                        download_zip_file(url, save_path)
                        extract_zip_file(save_path, extraction_path)
                        print("Arquivo ZIP baixado e extraído com sucesso!")
                    except requests.exceptions.RequestException as e:
                        print(f"Erro ao baixar o arquivo: {e}")
                    except zipfile.BadZipFile:
                        print("Erro ao extrair o arquivo ZIP.")
                    finally:
                        if os.path.exists(save_path):
                            os.remove(save_path)
                elif:
                    print("Os dados já foram baixados e extraídos anteriormente.")
        download_arquivos_inmet(data_inicial, data_final)
        downloads_path = Path.home() / "Downloads"
        diretorio_pasta = downloads_path / "INMET_DOWNLOAD"
    total_steps = sum(1 for _, _, arquivos in os.walk(diretorio_pasta) for arq in arquivos if
arq.endswith('.CSV'))
```

```
barra_progresso = 1
for raiz, pastas, arquivo in os.walk(diretorio_pasta):
    for num_arquivos, arq in enumerate(arquivo):
        barra_progresso += num_arquivos
        self.dlg.progressBar.setValue(barra_progresso * 100 / total_steps)
        self.dlg.progressBar.setValue(barra_progresso * 100 / total_steps)
        if arq.endswith('.CSV'):
            salvar_dados = tabela_principal(os.path.join(raiz, arq), salvar_dados)
self.dlg.progressBar.setValue(100)



coluna_original = ["PRECIPITAÇÃO TOTAL, HORÁRIO (mm)",
    "PRESSAO ATMOSFERICA AO NIVEL DA ESTACAO, HORARIA (mB)",
    "PRESSÃO ATMOSFERICA MAX.NA HORA ANT. (AUT) (mB)",
    "PRESSÃO ATMOSFERICA MIN. NA HORA ANT. (AUT) (mB)",
    "RADIACAO GLOBAL (KJ/m²)",
    "TEMPERATURA DO AR - BULBO SECO, HORARIA (°C)",
    "TEMPERATURA DO PONTO DE ORVALHO (°C)",
    "TEMPERATURA MÁXIMA NA HORA ANT. (AUT) (°C)",
    "TEMPERATURA MÍNIMA NA HORA ANT. (AUT) (°C)",
    "TEMPERATURA ORVALHO MAX. NA HORA ANT. (AUT) (°C)",
    "TEMPERATURA ORVALHO MIN. NA HORA ANT. (AUT) (°C)",
    "UMIDADE REL. MAX. NA HORA ANT. (AUT) (%)",
    "UMIDADE REL. MIN. NA HORA ANT. (AUT) (%)",
    "UMIDADE RELATIVA DO AR, HORARIA (%)",
    "VENTO, DIREÇÃO HORARIA (gr) (° (gr))",
    "VENTO, RAJADA MAXIMA (m/s)",
    "VENTO, VELOCIDADE HORARIA (m/s)"]
coluna_renomeada = ["PRECIPIT", "PRESSAO_EST", "PRESSAO_MAX",
"PRESSAO_MIN",
    "RADIACAO", "TEMP_BULBO", "TEMP_ORVAL", "TEMP_MAX",
"TEMP_MIN", "TEMP_OVMAX",
```

```
        "TEMP_OVMIN", "UMID_MAX", "UMID_MIN", "UMID_RELAT", "VENTO_DIR",
        "VENTO_RAJ", "VENTO_VEL"]
    # ESTACAO REGIAO UF CODIGO (WMO) LATITUDE LONGITUDE ALTITUDE
    for trava, coluna in enumerate(salvar_dados.columns):
      if trava > 6:
        for posicao, verifica_coluna in enumerate(coluna_original):
          if coluna == verifica_coluna:
            salvar_dados = salvar_dados.rename(columns={coluna: coluna_renomeada[posicao]})
    salvar_dados = salvar_dados.dropna(thresh=8)
    colunas_dados = []
    for num_coluna, colunas in enumerate(salvar_dados.columns):
      if num_coluna > 6:
        colunas_dados.append(colunas)
    for var, comando in zip(['Máximo', 'Mínimo', 'Média', 'Soma'], ['max', 'min', 'mean', 'sum']):
      if var == variavel:
        salvar_dados = salvar_dados.groupby(['ESTACAO', 'REGIAO', 'UF', 'CODIGO', 'LATITUDE', 'LONGITUDE',
        'ALTITUDE']).agg({coluna: comando for coluna in colunas_dados}).reset_index()
    salvar_dados['geometry'] = salvar_dados.apply(lambda row: Point(row['LONGITUDE'], row['LATITUDE']), axis=1)
    gdf = gpd.GeoDataFrame(salvar_dados, geometry='geometry')
    gdf.crs = pyproj.CRS('EPSG:4674')
    dir_shapefile = self.dlg.dir_saida.text()
    gdf.to_file(dir_shapefile)
    nome = dir_shapefile[dir_shapefile.rfind("/")+1:dir_shapefile.rfind(".")]
    camada = QgsVectorLayer(dir_shapefile, nome, 'ogr')
    QgsProject.instance().addMapLayer(camada)
  def validar_campos(self):
    caminho_select = self.dlg.caminho_select.text()
    dir_saida = self.dlg.dir_saida.text()
```

```
        verifica = []
        for index in range(self.dlg.list_dados.count()):
            item = self.dlg.list_dados.itemText(index)
            checked = self.dlg.list_dados.itemCheckState(index)
            if checked == Qt.Checked:
                verifica.append(item)
        if not self.dlg.download.isChecked() and not caminho_select:
            QMessageBox.warning(None, "Aviso", "Informe o diretórios dos arquivos ou selecione
a opção 'Download automático'.")
        elif len(verifica) == 0:
            QMessageBox.warning(None, "Aviso", "Escolha pelo menos um tipo de informação.")
        elif not dir_saida:
            QMessageBox.warning(None, "Aviso", "Informe um caminho de saída.")
        elif:
            self.percorrer_arquivo()
    def run(self):
        """Método que executa todo o trabalho real em segundo plano."""
        # Criar uma instância do diálogo de dados meteorológicos
        self.dlg = DadosMeteorologicosDialog()
        # Conectar os sinais do diálogo aos métodos correspondentes
        self.dlg.dt_final.dateChanged.connect(self.validar_datas)
        self.dlg.dt_inicial.dateChanged.connect(self.validar_datas)
        self.dlg.hora_inicial.timeChanged.connect(self.validar_horarios)
        self.dlg.hora_final.timeChanged.connect(self.validar_horarios)
        self.dlg.executar.clicked.connect(self.validar_campos)
        self.dlg.select_dir.clicked.connect(self.pasta)
        self.dlg.salvar_arquivo.clicked.connect(self.definirsaida)
        # Configurar os tempos padrão
        self.dlg.dt_inicial.setTime(QTime(0, 0))
        self.dlg.dt_final.setTime(QTime(23, 0))
        # Criar uma instância da classe BackgroundTask para execução em segundo plano
        self.background_task = BackgroundTask()
        # Conectar sinais emitidos pela tarefa em segundo plano aos slots apropriados
        # self.background_task.progress_updated.connect(self.update_progress)
```

```
# self.background_task.finished.connect(self.task_finished)

# Iniciar a tarefa em segundo plano
# self.background_task.start()
# Exibir o diálogo ao usuário
self.dlg.show()
result = self.dlg.exec_()
```