UNIVERSIDADE FEDERAL FLUMINENSE
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA AGRÍCOLA E MEIO AMBIENTE
PROGRAMA DE PÓS-GRADUAÇÃO LATO SENSU RESIDÊNCIA EM PRÁTICAS AGRÍCOLAS, ASSISTÊNCIA TÉCNICA E EXTENSÃO RURAL

DANIEL ALVARES DA SILVA

CLIMATER: UMA APLICAÇÃO NO TELEGRAM PARA VISUALIZAÇÃO DE VARIÁVEIS CLIMÁTICAS E SÉRIES TEMPORAIS

Niterói - RJ
2022

DANIEL ALVARES DA SILVA

CLIMATER: UMA APLICAÇÃO NO TELEGRAM PARA VISUALIZAÇÃO DE VARIÁVEIS CLIMÁTICAS E SÉRIES TEMPORAIS

Trabalho de conclusão de curso apresentado ao Curso de Pós-graduação Lato Sensu Curso de Residência em Práticas Agrícolas, Assistência Técnica e Extensão Rural, da Universidade Federal Fluminense, como requisito parcial à obtenção do título de Especialista em Práticas Agrícolas, Assistência Técnica e Extensão Rural.

Orientador: Prof. Dr. Marcio Cataldi

Niterói, RJ
2022

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

S586c Silva, Daniel Alvares da
CLIMATER: uma aplicação no Telegram para visualização de variáveis climáticas e séries temporais / Daniel Alvares da Silva. - 2022.
119 f.: il.

Orientador: Marcio Cataldi.
Monografia (residência)-Universidade Federal Fluminense, Escola de Engenharia, Niterói, 2022.

1. Agrometeorologia. 2. Agricultura. 3. Produção intelectual. I. Cataldi, Marcio, orientador. II. Universidade Federal Fluminense. Escola de Engenharia. III. Título.

CDD - XXX

Bibliotecário responsável: Debora do Nascimento - CRB7/6368

DANIEL ALVARES DA SILVA

CLIMATER: UMA APLICAÇÃO NO TELEGRAM PARA VISUALIZAÇÃO DE VARIÁVEIS CLIMÁTICAS E SÉRIES TEMPORAIS

Trabalho de conclusão de curso apresentado ao Curso de Pós-graduação Lato Sensu Curso de Residência em Práticas Agrícolas, Assistência Técnica e Extensão Rural, da Universidade Federal Fluminense, como requisito parcial à obtenção do título de Especialista em Práticas Agrícolas, Assistência Técnica e Extensão Rural.

Aprovada em 11 de novembro de 2022.

BANCA EXAMINADORA

Documento assinado digitalmente
gov.br MARCIO CATALDI
Data: 16/11/2022 16:40:01-0300
Verifique em https://verificador.iti.br

Prof. Dr. Marcio Cataldi (Orientador)

UFF – Universidade Federal Fluminense

Documento assinado digitalmente
gov.br LEONARDO DA SILVA HAMACHER
Data: 16/11/2022 11:29:20-0300
Verifique em https://verificador.iti.br

Prof. Dr. Leonardo da Silva Hamacher

UFF – Universidade Federal Fluminense

Dra. Carla Bento da Silva

Secretaria de Saúde de Seropédica-RJ

Niterói, RJ
2022

AGRADECIMENTOS

Agradeço a Deus, por ser meu Norte e meu escudo nos momentos mais difíceis.

Agradeço a minha família, meus pais e irmãos, por se doarem e estarem presentes em todos os desafios da minha vida.

Agradeço meus amigos Jhonny Gontijo, Caroline Santos e Carla Bento por todo o companheirismo.

Agradeço meus mestres da UFF, por transmitirem seu conhecimento e se fazerem presentes no amadurecimento deste projeto, principalmente meu orientador, Marcio Cataldi, me possibilitando enorme desenvolvimento profissional.

Agradeço a Universidade Federal Fluminense, Fundação Euclides da Cunha, Ministério da Agricultura, Pecuária e Abastecimento e Secretaria de Agricultura Familiar e Cooperativismo pela realização do curso de especialização e todo apoio financeiro.

Muito obrigado!

RESUMO

Este trabalho buscou desenvolver um Mínimo Produto Viável, apresentando a proposta de um *bot de Telegram* para servir de interface de visualização para variáveis climáticas, suas médias e séries temporais, facilitando o acesso e interpretação desses dados para o agricultor. O *bot* ofertará dados das estações automáticas disponibilizadas pelo INMET para o estado do Rio de Janeiro. Os dados exibidos na sua interface poderão ser requisitados primariamente através de coordenadas geográficas e, também a partir de métodos alternativos como através da referência do CEP, utilizando API (em português, Interface de Programação de Aplicação), e se comunicando com serviços como Correios para converter tal informação em coordenada geográfica de referência, baseada em região e suas subdivisões. Ainda é possível dispor de informações meteorológicas baseadas na sede de cada município, baseado em informações georreferenciadas informadas pelo governo e entidades, propiciando o cálculo para se conhecer as estações meteorológicas mais próximas do usuário.

PALAVRAS–CHAVE: smartphone, agrometeorologia, agricultura

ABSTRACT

The purpose of this work was to develop a Minimum Viable Product, presenting the proposal of a Telegram bot to serve as a visualization interface for climate variables, their average values and time series, facilitating the access and interpretation of these data for the small farmer. The bot will offer data from the automatic weather stations provided by INMET for the state of Rio de Janeiro. The data displayed on its interface can be requested primarily through geographic coordinates and also through alternative methods such as the CEP reference, using API (Application Programming Interface), and communicating with services such as Correios to convert such information into a geographic coordinate of reference, based on region and its subdivisions. It is still possible to have meteorological information based on the headquarters of each municipality, based on georeferenced information informed by the government and entities, providing the calculation to know the weather stations closest to the user.

KEYWORDS: smartphone, agrometeorology, agriculture

SUMÁRIO

1 INTRODUÇÃO

A agricultura brasileira na busca por alcançar sustentabilidade, deverá ser pautada por dois caminhos que por fim se complementam: a introdução de inovações tecnológicas e o rápido crescimento da agricultura orgânica. A competição natural do mercado provocou essa mudança de atitude entre empresas e produtores, os quais se alinharam às necessidades do mercado internacional (KITAMURA, 2018).

Kitamura (2018), generaliza o conceito de tecnologias abordadas tornando-se evidente a necessidade de ir além da ideia de fertilizantes, agroquímicos, técnicas de cultivo e manejo integrado. As tecnologias já se fazem presentes na rotina de campo de produtores com acesso tecnológico e poder aquisitivo. Tais abordagens e mudanças ocorrem através de estímulos do mercado. Os profissionais que atuam no agro compreendem a necessidade da busca por sustentabilidade, e que tal objetivo é catalisado pela adoção de tecnologias, gerando o desenvolvimento de novas técnicas, que viabilizam o aumento da produtividade agrícola, ao mesmo tempo que aprimora o emprego de insumos agrícolas.

Verificaram-se nos últimos anos grandes avanços tecnológicos abordando sensores, circuitos integrados e comunicações sem fio. Tais tecnologias podem ser usadas amplamente no monitoramento ambiental e agricultura de precisão (SANTOS et al., 2010).

O crescimento e desenvolvimento técnico da agricultura familiar no país proporciona geração de renda, alimentação e preservação ambiental. Entretanto, a adoção tecnológica para viabilizar tais processos ainda é fraca (CUNHA, K. C. B. DA; ROCHA, 2016). No sentido da evolução, as tecnologias nos propiciaram o desenvolvimento dos saberes relativos à natureza, compreendendo seus processos físicos e biológicos (MACHADO et al., 2007), assim como viabilizam a pesquisa e desenvolvimento de aplicações agrometeorológicas de baixo custo (FISHER; GOULD, 2012).

Após a Revolução Industrial, observaram-se alterações no clima global, sendo notável a acentuada mudança ambiental, onde as estações do ano não

mais refletem suas características intrínsecas (SILVA, 2013). Tendo em vista as mudanças climáticas e como isso reflete na agricultura e economia em si, fato ainda mais acentuado nos países pobres em baixas latitudes (WEART et al., 2014), compreendemos a necessidade de modernizar técnicas, assim como aprimorar o uso de recursos naturais. E nesse sentido somos capazes de reproduzir técnicas e aplicar tecnologias de baixo custo para pequenos produtores rurais, situados em situações periféricas.

E o Brasil possuindo milhões de dispositivos digitais, sendo 242 milhões de *smartphones* contabilizados em 2022 (MEIRELLES, 2022), reforça-se a idéia de apresentar aplicações dentro do contexto da agricultura e meteorologia que estejam melhor integradas à essas tecnologias, assim como em sinergia aos hábitos do brasileiro, exigindo menor conhecimento técnico para o seu manuseio e acesso à informação, ganhando força também por agregar inúmeras funções de diversos dispositivos, substituindo um amplo espectro de produtos (FERREIRA; RUFFONI; CARVALHO, 2018). Sendo assim é possível avaliar que existem elementos essenciais para ampliar o acesso de informações agrometeorológicas para pequenos produtores, em sinergia com a evolução dos dispositivos digitais em geral, o crescimento na capacidade de adquirir, armazenar e comunicar dados, além de compreender a importância da agricultura como setor estratégico, a aplicação dessas tecnologias tem capacidade para impulsionar seu desenvolvimento (DRUCKER et al., 2017).

2 OBJETIVOS

2.1 OBJETIVO GERAL

Desenvolver um programa classificado como robô, chamado popularmente pelo termo *bot,* para o aplicativo de comunicação Telegram, se constituindo um MVP, ou seja, Mínimo Produto Viável, servindo de interface para viabilizar a visualização e interpretação de variáveis climáticas, médias e séries temporais, através da requisição e armazenamento de dados climáticos.

2.2 OBJETIVOS ESPECÍFICOS

Para que o bot Climater apresente funcionalidade e aplicabilidade nesse contexto, os seguintes objetivos deverão ser desenvolvidos e entregues:

- Criar funções para captura de dados meteorológicos das estações automáticas do INMET através de sua API;
- Georreferenciar as estações automáticas do estado do Rio de Janeiro;
- Georreferenciar as sedes dos municípios do estado do Rio de Janeiro;
- Desenvolver função para conversão de CEP (Código de Endereçamento Postal) para coordenada geográfica de referência, integrando diversas APIs;
- Desenvolver função para converter de Coordenadas Geográficas GMS para GD, facilitando cálculos e conversões internas;
- Elaborar mapas exibindo as estações meteorológicas e pontos de referência de interesse do usuário;
- Calcular distância entre a estação meteorológica e pontos de referência de interesse do usuário com intuito de prover dados relevantes;
- Criar funções para facilitar a interpretação de dados climáticos dentro do aplicativo do Telegram;
- Disponibilizar código aberto em repositório público e instruções para a implantação da aplicação.

3 JUSTIFICATIVA

Compreende-se que as informações agroclimáticas disponibilizadas atualmente nos meios informatizados no contexto nacional não oferecem interfaces de fácil e ágil acesso, criando a necessidade do desenvolvimento de novas soluções capazes de universalizar o seu acesso e interpretação. Tais pontos se agravam com limitações tecnológicas e financeiras, vividas por pequenos produtores, não dispondo de adequações voltadas para sistemas móveis, como celulares no geral, e dessa maneira a aplicação elaborada neste trabalho apresenta-se como uma proposta para democratizar o acesso de dados agroclimáticos.

4 REVISÃO BIBLIOGRÁFICA E FUNDAMENTAÇÃO TEÓRICA

4.1 METEOROLOGIA

Meteorologia é a ciência que busca compreender os processos físicos, químicos e dinâmicos da atmosfera, assim como suas interações com as camadas que compõem o planeta Terra (YNOUE et al., 2017). “O sistema climático terrestre é influenciado por uma complexa combinação de fatores, que envolvem desde a dinâmica até a composição química atmosférica” (GÓMEZ et al., 2018).

O estado da atmosfera é definido como tempo atmosférico e este estado é descrito pelas variáveis temperatura do ar, pressão atmosférica, umidade, nebulosidade, precipitação, visibilidade e vento (YNOUE et al., 2017).

O tempo é a condição da atmosfera em dado local e momento (ACKERMAN; KNOX, 2013). Destaca-se a necessidade de distinguir clima e tempo, descrevendo tempo como o estado atual da atmosfera, num dado local, num dado instante e seus efeitos sobre a vida e atividade humana (VIEIRA; PICULLI, 2009). Já o clima pode ser descrito como fenômenos meteorológicos que ocorreram num determinado local e período de tempo, sendo possível termos diversos tipos de tempo num mesmo dia (PEREIRA; ANGELOCCI; SENTELHAS, 2007).

A influência do Sol é descrita como a principal fonte do clima em nosso planeta, dado que apenas uma das metades do planeta Terra é aquecida pelo

Sol, enquanto o resto do planeta segue o período noturno, provocando aquecimento desigual da superfície terrestre. Com diferenças de temperaturas temos a definição do clima no planeta: ventos, nuvens, assim como precipitação. Já estações do ano e padrões climáticos sazonais estão relacionados a inclinação e distância de partes do planeta em relação ao Sol (ACKERMAN; KNOX, 2013).

4.1.1 AGROMETEOROLOGIA

A meteorologia possui divisões especializadas tendo como critério a influência das condições atmosféricas sobre as atividades exercidas pelo homem e, nesse contexto, sob o desenvolvimento da prática agrícola temos a agrometeorologia (PEREIRA; ANGELOCCI; SENTELHAS, 2007).

As condições climáticas apontam a atividade agrícola ideal para dado local, determinando o nível de produtividade, assim como influencia práticas e manejos nas atividades agrícolas. Embora o homem não seja capaz de interferir no tempo e clima, ele é capaz de adequar as práticas agrícolas (PEREIRA; ANGELOCCI; SENTELHAS, 2007).

Como pontuam diversos autores e estudos, é notório a importância do clima e suas variações para o estabelecimento da produção agrícola, implicando em níveis de produtividade, norteando a adoção de práticas e manejos (MONTEIRO, 2009), ficando evidente que agricultura é a atividade que mais depende do tempo e do clima. As condições atmosféricas afetam desde o preparo do solo para plantio, até o momento da colheita, além de influenciar no pós-colheita (PEREIRA; ANGELOCCI; SENTELHAS, 2007), pois as condições de tempo e clima atuam na limitação da expressão do potencial genético das culturas agrícolas (MARTORANO et al., 2017).

Nesse sentido, por compreender a magnitude desses eventos climáticos, suas variações em regiões produtivas, é possível determinar e organizar a agricultura, delimitando atividades específicas, minimizando impactos e maximizando fatores que favorecem o desenvolvimento vegetal (PEREIRA; ANGELOCCI; SENTELHAS, 2007), e assim temos base teórica, modelos e

equações que viabilizam estimar com certa exatidão a produtividade e qualidade de produtos agrícolas em dada área e região (PICINI et al., 1999).

4.2 ELEMENTOS E FATORES CLIMÁTICOS

Elementos são as variáveis que irão caracterizar o estado da atmosfera, sendo composto por: temperatura, pressão, umidade relativa, precipitação, velocidade e direção dos ventos, assim como radiação solar (PEREIRA; ANGELOCCI; SENTELHAS, 2007).

Já os fatores climáticos ou meteorológicos são considerados como os agentes causais que condicionam os elementos climáticos e estes são: latitude, altitude, continentalidade/oceanalidade e tipo de corrente oceânica (PEREIRA; ANGELOCCI; SENTELHAS, 2007).

4.2.1 TEMPERATURA

Sob exposição da energia solar, a temperatura do ar na Terra, assim como sua variação é dado pelo movimento de translação e rotação da Terra, também sendo influenciado pela latitude, altitude, proximidade de corpos d'água, circulações oceânicas e atmosféricas (YNOUE et al., 2017). A temperatura do ar é a medida de calor armazenado nele, dada em nosso país em graus Celsius e medida por termômetros (VIEIRA; PICULLI, 2009).

A temperatura do ar ocorre principalmente em função do transporte de calor, do aquecimento da superfície pela radiação solar, sendo que esse transporte de calor ocorre através de dois processos: condução molecular e difusão turbulenta (PEREIRA; ANGELOCCI; SENTELHAS, 2007).

Pereira *et al*., (2007) descreve que a condução molecular do transporte de calor é o processo de troca mais lento, se dando pelo contato direto entre as moléculas de ar, restrito a finas camadas de ar próxima a superfície aquecida.

Já a difusão turbulenta é o processo mais rápido, já que as massas de ar aquecidas estão em movimento convectivo de maneira desordenada, transportando calor, vapor d'água e partículas como poeira, para outras camadas (PEREIRA; ANGELOCCI; SENTELHAS, 2007).

A temperatura do ar tem papel fundamental nas atividades agrícolas, exercendo influência sobre diversas funções das plantas, como fotossíntese, sua respiração e transpiração, agindo sobre o desenvolvimento vegetal e seus estádios (LUCCHESI, 1987), onde a exposição à temperatura média acima de 25° C durante o período de desenvolvimento vegetativo impacta negativamente a floração e fertilização na maioria das culturas agrícolas (KAZANDJIEV et al., 2019).

Para fins meteorológicos e climatológicos, a temperatura é medida sob uma condição de referência, permitindo comparação entre locais (PEREIRA; ANGELOCCI; SENTELHAS, 2007).

A estrutura para tal finalidade é definida dentro de uma área plana e gramada, onde temos instrumentos posicionados a uma altura de 1,5m distante do solo, abrigados da incidência de radiação solar, mas que ainda assim permita livre passagem ar (PEREIRA; ANGELOCCI; SENTELHAS, 2007).

Dentro desses abrigos teremos termômetros que irão atuar na medição de temperatura do ar (PEREIRA; ANGELOCCI; SENTELHAS, 2007).

Os termômetros são divididos conforme o princípio físico que rege o seu funcionamento: dilatação de líquido, dilatação de sólido, pares termoelétricos, resistência elétrica e radiação, sendo empregado nas estações automáticas termômetros do tipo termopares e termistores (PEREIRA; ANGELOCCI; SENTELHAS, 2007).

Na agrometeorologia a temperatura é trabalhada através das seguintes variáveis derivadas: temperatura máxima e mínima, assim como a temperatura média do ar (CHACON, 2019), sendo expressa em graus Celsius, onde se baseia nos pontos de fusão da água, respectivamente 0° C, e em seu ponto de ebulição, 100° C, definindo 1 grau como a centésima parte entre esses pontos (JIMENEZ-CARBALLO, 2018).

4.2.2 PRESSÃO ATMOSFÉRICA

De maneira simplificada, podemos definir a pressão atmosférica como a força que a nossa atmosfera exerce sobre a superfície do planeta Terra (PINHEIRO DE ASSIS et al., 2016). A força ou pressão exercida por essa coluna vertical de ar é igual em todas as direções (BROCK; RICHARDSON, 2001), também relacionado ao peso molecular do ar (OLDANI, 2020).

A incidência e absorção de maneira distinta dos raios solares na superfície do planeta resulta em diferentes pressões (PEREIRA; ANGELOCCI; SENTELHAS, 2007), pois com o aquecimento da superfície e das massas de ar, temos a expansão volumétrica do ar, tornando-o menos denso, exercendo menos força sobre a superfície, ao contrário das massas de ar que se resfriam (PINHEIRO DE ASSIS et al., 2016).

A determinação da pressão atmosférica pode ser realizada pelo uso do instrumento chamado de barômetro, sendo o mais comum o barômetro de mercúrio (BROCK; RICHARDSON, 2001).

O trabalho do italiano Evangelista Torricelli demonstra o seu funcionamento, onde se assume a densidade do mercúrio e a aceleração da gravidade como constantes (PINHEIRO DE ASSIS et al., 2016). Nesse estudo Torricelli apresenta uma coluna com 76 cm Hg para pressão atmosférica ao nível do mar, entendendo-se a pressão exercida (DE ARAUJO ELIAS et al., 2014) e na meteorologia a unidade adotada é a hectopascal, expressando a pressão exercida por uma massa de 1.013 gramas sobre superfície de 1 cm² (OLDANI, 2020), sendo o valor padrão da pressão atmosférica ao nível médio do mar igual a 1.013,25 hPa. Apesar de ser expressa em hectopascal, é comum ver a adoção da unidade milibar, utilizada por centros operacionais e trabalhos da área meteorológica (YNOUE et al., 2017).

Apesar da simplicidade, na atualidade são mais comuns o emprego de sensores (PEREIRA; ANGELOCCI; SENTELHAS, 2007) ou barômetros/barógrafos do tipo aneróide, invenção do francês Lucien Vidie (RIBEIRO, 2014), que consiste de uma cápsula hermética com um diafragma metálico flexível, com uma mola interna. Tal mecanismo comprime-se com o

aumento de pressão e se expande quando a pressão diminui, resultando na leitura/registro da pressão (DE OLIVEIRA et al., 2012).

Na agrometeorologia a pressão é trabalhada através das seguintes variáveis derivadas: pressão atmosférica efetiva, qual reflete a pressão do ar num dado momento, e a pressão atmosférica corrigida ao nível do mar, que é uma correção realizada tendo como parâmetro a elevação que a estação apresenta desde o nível do mar (CHACON, 2019).

4.2.3 UMIDADE DO AR

A água é o único elemento que se encontra nos três estados físicos, não sendo incomum encontrar gelo, água e vapor dentro de nuvens, com sua distribuição variando de maneira espacial e temporal na atmosfera (PEREIRA; ANGELOCCI; SENTELHAS, 2007).

O vapor d'água está presente em nossa atmosfera e desempenha importante papel de maneira física, química e biológica (OLIVEIRA, 2018), sendo para o meteorologista o constituinte atmosférico mais importante (VIEIRA; PICULLI, 2009). O vapor d'água é um termorregulador da atmosfera, agindo sobre as variações acentuadas de temperatura do ar (PEREIRA; ANGELOCCI; SENTELHAS, 2007), assim como principal agente responsável por absorver seletivamente a radiação solar (VIEIRA; PICULLI, 2009).

A umidade relativa do ar pode ser apurada através do uso do instrumento chamado de psicrômetro, sendo possível relacionar a diferença de temperatura entre os bulbos do psicrômetro com a umidade relativa do ar (NEVES et al., 2015). A temperatura e umidade relativa do ar estão intimamente relacionadas, sendo que quando a temperatura tende ao valor mínimo para um local, a umidade relativa do ar tenderá à saturação, configurando comportamento inverso à temperatura do local (PEREIRA; ANGELOCCI; SENTELHAS, 2007).

Existem ainda trabalhos em que os autores citam outros métodos, mais atuais e mais precisos, de modo que a umidade do ar afeta e provoca alterações nas características elétricas de componentes, ressaltando a necessidade do uso de abrigo para todos, independente das condições ambientais às quais os

dispositivos estarão expostos (NEVES et al., 2015; PEREIRA; ANGELOCCI; SENTELHAS, 2007).

Na agrometeorologia a umidade relativa do ar é trabalhada através das seguintes variáveis derivadas: umidade relativa, umidade relativa média, umidade relativa máxima diária, umidade relativa mínima diária (CHACON, 2019).

4.2.4 PRECIPITAÇÃO

O retorno do vapor d'água a superfície ocorre através de diversas formas de precipitação, sendo as chuvas a precipitação mais comum, sendo mensurada a respeito de volume, distribuição temporal e espacial através do emprego de pluviômetros (CHACON, 2019).

A formação de chuvas é um processo que vai além da condensação da água na atmosfera. As gotículas em suspensão, elementos de nuvem, se juntam umas às outras, formando gotas maiores, chamadas de elementos de precipitação, assim rompendo a suspensão exercida pela força térmica (PEREIRA; ANGELOCCI; SENTELHAS, 2007).

Estimar a precipitação com exatidão é primordial para o desenvolvimento e manutenção social e econômica (COSTA et al., 2016; FIRPO, 2012; SILVA-FUZZO; ROCHA, 2016), assim como para a prevenção de acidentes e catástrofes provocados por extremos climáticos (DE SOUSA et al., 2015; HONG; KIM; JEONG, 2018; LOPEZ; VILLARUZ, 2016; ROBERTSON; SHRESTHA; WANG, 2013; TWARDOSZ; CEBULSKA; WALANUS, 2016), sendo nos trópicos a principal forma que a água retorna para a superfície terrestre após processos de evaporação e condensação, completando seu ciclo hidrológico (PEREIRA; ANGELOCCI; SENTELHAS, 2007). Firpo (2012) ainda ressalta a importância dos efeitos do fenômeno El Niño/Oscilação Sul devido impactos de escala global, interferindo no regime pluviométrico e térmico, evidenciando a necessidade de estudos que abordem localmente suas influências.

A precipitação em territórios extensos não é contabilizada de maneira adequada devido a distribuição insuficiente de estações meteorológicas,

afetando análises do escoamento superficial, déficit hídrico e balanço de energia (PEREIRA et al., 2013), sendo recomendado a expansão da disponibilidade de dados, assim como o foco na qualidade dessas informações para aumentar a confiabilidade dos resultados de futuras análises (COSTA et al., 2012). Soma-se ainda aos complicadores citados a condição de funcionamento dos pluviômetros e pluviógrafos (OLIVEIRA JÚNIOR et al., 2014).

O monitoramento da chuva poderá ser feito através do emprego de pluviômetros, convencionais ou automáticos. Os convencionais medem a altura de chuva total num dado período de tempo, já os automáticos registram as variações, armazenando as informações em formato digital (BLAINSKI; GARBOSSA; ANTUNES, 2012).

Na agrometeorologia a precipitação é trabalhada através da altura de precipitado (em milímetros), num período definido, como hora, dia, mês, ano (CHACON, 2019).

4.2.5 RADIAÇÃO SOLAR

A agrometeorologia aborda primariamente as condições meteorológicas nas camadas mais baixas da atmosfera, onde a conversão de energia da radiação solar na superfície do solo e plantas atua mais diretamente (SEEMANN et al., 1979).

O Sol emite radiação eletromagnética proveniente da ação do dínamo solar, sendo a principal fonte de energia do sistema terrestre (GÓMEZ et al., 2018). Temperatura do ar e a radiação solar estão relacionadas, influenciando o desenvolvimento de plantas, apresentando complexa influência no crescimento, desenvolvimento e produção (ARAÚJO et al., 2018; REIS et al., 2012), também sendo considerado o motor por trás do funcionamento do ciclo hidrológico, responsável por converter água no estado líquido para vapor através de processos de evaporação e evapotranspiração (CHACON, 2019) e também sendo responsável por quase todos os processos físicos, químicos, biológicos e bioquímicos que ocorrem no sistema terra-atmosfera (BERUSKI; PEREIRA; SENTELHAS, 2015; CHACON, 2019; GÓMEZ et al., 2018).

Da energia emitida pelo Sol, 41% estão compreendidos entre 400 e 700 nanômetros, a faixa denominada como espectro visível, importante por permitir a visão, assim como a fotossíntese (AHMAD et al., 2017; OLIVEIRA, 2018). Já as emissões no espectro Ultravioleta variam de 3-4%, em 200-300 nanômetros até 100% próximo à linha de emissão *Lyman-alpha*, sendo toda variação do fluxo magnético solar atuante nas alterações do clima da Terra (GÓMEZ et al., 2018).

Estudos realizados com as culturas do trigo e grão de bico apresentam a importância da irradiação no desenvolvimento e produção vegetal, sendo observados alteração em aspectos fisiológicos, como condutância estomática e taxa de fotossíntese no caso de irradiação reduzida, em razão de poluição (MINA et al., 2015), assim como varia ao longo do dia e também em função da nebulosidade, afetando a produção bruta de matéria seca (PEREIRA; ANGELOCCI; SENTELHAS, 2007).

A radiação solar que chega na superfície terrestre é constituído em sua maioria por ondas curtas, sendo sua distribuição espacial e estacional grande influenciador dos fenômenos meteorológicos (PEREIRA; ANGELOCCI; SENTELHAS, 2007).

Apesar da quantidade de energia que chega à superfície ser determinada pela radiação incidente e atmosférica, a energia que é armazenada no sistema é determinada pelo tipo de cobertura, variando com o coeficiente de refletividade (PEREIRA; ANGELOCCI; SENTELHAS, 2007).

Na agrometeorologia a radiação solar é observada e trabalhada através da irradiância, que expressa a energia por unidade de tempo e área, como MJ/m^2 (CHACON, 2019).

4.2.6 NEBULOSIDADE

As nuvens são agregados de pequeninas gotículas de água, cristais de gelo, assim como sua mistura, posicionados acima da superfície da Terra (AHMAD et al., 2017).

4.2.7 VENTOS

O vento é um importante e complexo elemento do clima, influenciando em diversos fenômenos, como erosão de solos, dispersão de poluentes e sementes, assim como a produção de energia eólica. Na meteorologia sua velocidade e direção está elencado entre as principais variáveis usadas para descrever a atmosfera terrestre (MEDEIROS et al., 2020).

Ventos são deslocamentos de ar no sentido horizontal, que ocorrem em decorrência de gradientes de pressão atmosférica (PEREIRA; ANGELOCCI; SENTELHAS, 2007), sendo que para fins meteorológicos é ignorado seu componente vertical, considerando-o um vetor de duas dimensões, que desempenha importante papel no desenvolvimento das culturas, afetando a evapotranspiração (AHMAD et al., 2017).

4.3 ESTAÇÕES METEOROLÓGICAS

Estações meteorológicas são instalações em terra ou mar, munida de instrumentos para observar condições atmosféricas, fornecendo informações para a previsão do clima, permitindo estudos sobre o tempo e clima (LOPEZ; VILLARUZ, 2016).

O monitoramento do clima é uma das maiores prioridades de agências nacionais e instituições de pesquisa, dado que sua observação é de extrema importância para diversas atividades socioeconômicas, norteando decisões (LAGOUVARDOS et al., 2017).

As estações meteorológicas são usadas na coleta de dados do clima (TENZIN et al., 2017), possibilitando a medição de parâmetros atmosféricos como radiação solar, temperatura do ar, umidade relativa, velocidade e direção dos ventos, pressão atmosférica e volume de chuva precipitada (MESTRE et al., 2015).

No Brasil, segundo relato de cientista responsável pela direção do Observatório do Rio de Janeiro em 1917, o nosso país contava com 222 estações meteorológicas espalhadas em seu território naquele período, configurando uma malha de 1 estação meteorológica para cada 38 mil quilômetros quadrados (BARBOZA, 2006). Apenas o INMET e CPTEC possuem cobertura nacional, entretanto temos escassez de dados na porção oeste do país, com menor densidade de estações meteorológicas no interior do país (ALENCAR et al., 2016), e hoje, através da colaboração de diversas entidades, temos a concepção do sistema Agritempo, que oferece dados meteorológicos de uma rede com mais 1.400 estações, somando mais de 60 milhões de registros diários para dados referente à monitoramento e previsão (VICENTE; RODRIGUES, 2014), ofertando 1 ponto de coleta de dados e monitoramento meteorológico a cada 6 mil quilômetros quadrados do território brasileiro. Entretanto, apesar do crescente número de estações meteorológicas mantidas por instituições privadas, públicas e voluntários da sociedade, ainda sofremos com grandes problemas na qualidade desses dados, assim como falta de manutenção de equipamentos, situação agravada nas condições da pandemia da COVID-19 e falta de novos concursos para renovar o corpo técnico das entidades responsáveis por esses sistemas (ALVIM, 2021). Ainda sobre problemas, um estudo comparativo sobre estações automáticas e convencionais do Brasil exalta a questão de erros sistêmicos ocasionados por interrupção na coleta de dados, assim como problemas com sensores, erros de leitura do observador, entre outros, alterando os níveis médios de distorção dos dados (LUCAS et al., 2010).

Apesar da disponibilidade de estações automáticas e convencionais em todo o território nacional, ainda carecemos de dados, sua qualidade e representatividade em escala micro (SILVA et al., 2015), onde informações descritas pelo zoneamento agroclimático podem não ser tão representativas devido influência do relevo, nesses casos sendo ainda necessário a interpretação do topoclima, pois a configuração do relevo pode ocasionar diferenças térmicas e exposição à radiação solar, provocando efeitos como acúmulo de ar frio em certas regiões (MONTEIRO, 2009).

As estações são classificadas de acordo com sua finalidade, pelo sistema de coleta de dados, assim como pela complexidade e número de elementos meteorológicos observados (PEREIRA; ANGELOCCI; SENTELHAS, 2007). Em relação à finalidade, temos:

- Estações Sinóticas: vinculadas ao sistema nacional e mundial de previsão de tempo, com observações em horários definidos para envio rápido de informações para órgãos responsáveis.
- Estações Climatológicas: são estações que tem como função caracterizar o clima de uma região, sendo a estação sinótica também considerada climatológica.
- Estações Aeronáuticas: tem como função a coleta de informações para oferecer segurança nas operações em aeroportos.
- Estações Agrometeorológicas: trabalham na obtenção de dados que norteiam as atividades agrícolas, incluindo temperatura do solo e evaporação.
- Postos Pluviométricos: tem o intuito de coletar dados a respeito de chuvas para o manejo de recursos hídricos.

Já em relação ao número de elementos observados, temos:

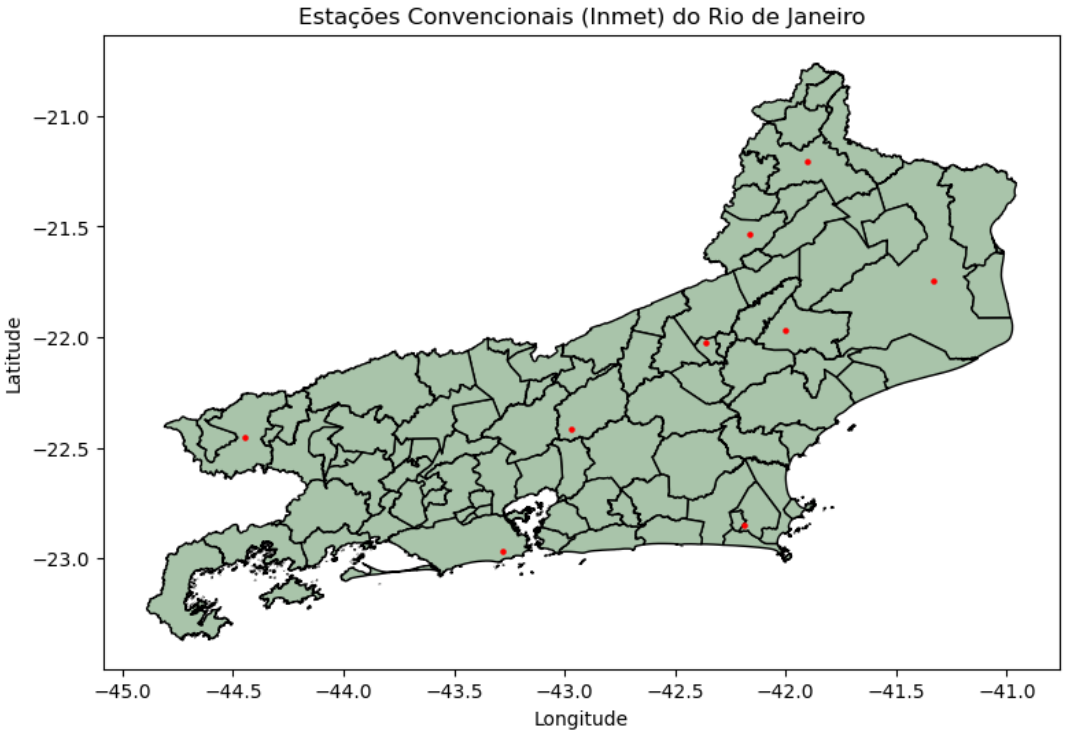
- Estações de Primeira Classe: possui instrumentos abordando todos os elementos meteorológicos.
- Estações de Segunda Classe: diferem das estações de primeira classe por não contemplarem pressão atmosférica, velocidade e direção dos ventos e irradiância solar global, entretanto possibilitam caracterizar os principais elementos para fins agrícolas.
- Estações de Terceira Classe: também conhecidas como estações termo-puvliométricas, por abordarem a temperatura do ar e precipitação na forma de chuva, comum em propriedades agrícolas para monitorar o balanço hídrico do solo.

4.3.1 ESTAÇÕES METEOROLÓGICAS CONVENCIONAIS

Esse tipo de estação meteorológica (EMC), requer a presença diária de um observador para realizar a coleta de dados. Seus equipamentos são de leitura direta (PEREIRA; ANGELOCCI; SENTELHAS, 2007), ou com sistema de princípio mecânico de registro (MENDES REIS; GONÇALVES LOPES; OLIVEIRA, 2015; PEREIRA; ANGELOCCI; SENTELHAS, 2007), e devem ser visitadas pelo menos uma vez por dia para a realização de medições (PERAZZI et al., 2021).

Nos dias de hoje, temos a disposição de 9 estações meteorológicas convencionais operacionais administradas pelo INMET para o estado do Rio de janeiro (INMET, 2022a), apresentadas na figura 1.

Figura 1: Mapa do estado do Rio de Janeiro e suas Estações Meteorológicas Convencionais



Fonte: Autor (2022)

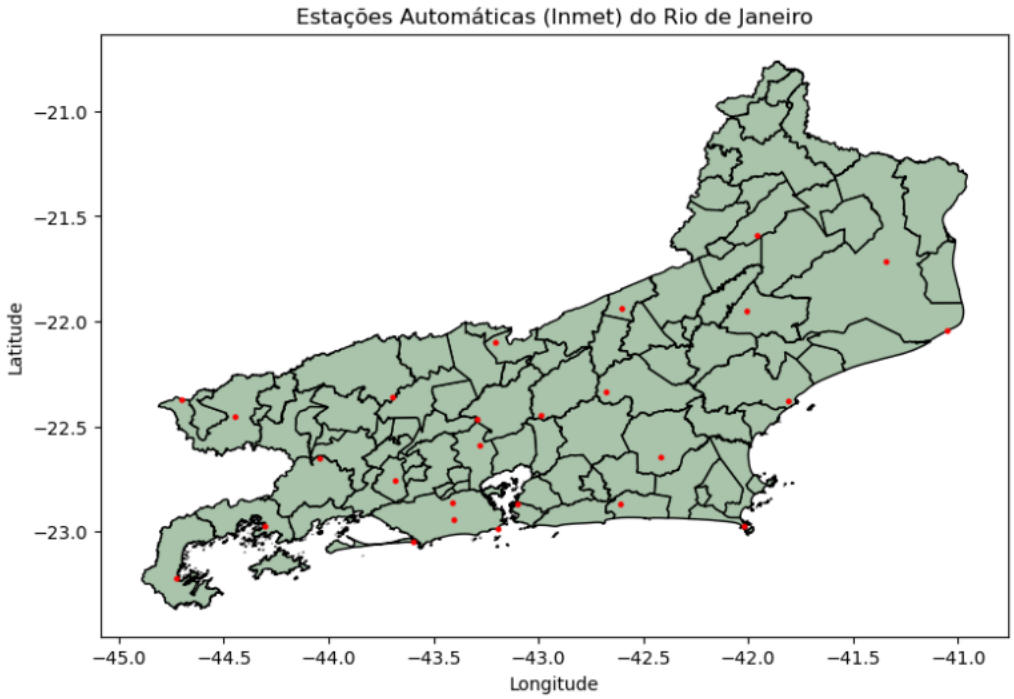
4.3.2 ESTAÇÕES METEOROLÓGICAS AUTOMÁTICAS

Já as estações automáticas (EMA), em comparação com as convencionais, são totalmente automatizadas, operando em função de sensores que permitem emitir sinais elétricos, registrados por um sistema de aquisição de dados conhecido como *datalogger*, o que possibilita seu armazenamento e futuro processamento informatizado dos dados (PEREIRA; ANGELOCCI;

SENTELHAS, 2007), sendo que seu poder de processamento e capacidade de armazenamento varia, devendo ser escolhido de acordo com volume de informações a ser armazenadas, funções usadas, sensores e envio de dados (BLAINSKI; GARBOSSA; ANTUNES, 2012).

Nos dias de hoje, temos a disposição de 26 estações meteorológicas automáticas operacionais (figura 2) administradas pelo INMET para o estado do Rio de janeiro (INMET, 2022b), sendo que as primeiras estações automáticas foram introduzidas pelo INMET no ano 2000, inicialmente substituindo estações convencionais em determinados locais, sendo que na época da produção da obra citada, o INMET contava com 450 estações meteorológicas automáticas distribuídas pelo território nacional (LUCAS et al., 2010).

Figura 2: Mapa do estado do Rio de Janeiro e suas Estações Meteorológicas Automáticas



Fonte: Autor (2022)

Tais estações vem ganhando cada vez mais espaço e importância no meio agrícola, adequando o manejo de maneira precisa, de acordo com as variações do clima, maximizando resultados com distúrbios mínimos (TENZIN et al., 2017), nesse contexto fazemos melhor uso da água na irrigação, aplicação de defensivos, previsão de eventos como risco de incêndio e geadas, entre muitos outros (MENDES REIS; GONÇALVES LOPES; OLIVEIRA, 2015).

4.4 INFORMAÇÕES ESPACIAIS

Durante a evolução da humanidade, é possível ver a importância do uso de informações espaciais em diversos setores, desde as navegações até atividades comerciais e demarcações de terras (WEBER et al., 1999), e hoje, com o avanço das tecnologias e meios informatizados, temos sua aplicação para resolução de problemas, impulsionando setores estratégicos (DRUCKER et al., 2017), assim como tornando atividades mais eficientes no que tange o uso de recursos naturais (BOLFE, 2006) e conservação do meio ambiente (NAKAMURA, 2010).

4.4.1 SISTEMA DE INFORMAÇÕES GEOGRÁFICAS

Outrora limitada por condições analógicas, temos condições para o surgimento e desenvolvimento do geoprocessamento com a evolução de conhecimentos nas áreas matemáticas e estatísticas, práticas para obtenção de dados, assim como avanços tecnológicos da informática na década de 70 (BOLFE, 2006).

Sistemas de informações geográficas (SIG ou, em inglês, GIS), propiciam a criação, manipulação e análise de dados, assim como conectar tais informações a um mapa, como uma interface para georreferencias, servindo de ferramenta para diversas aplicações, assim como para desenvolvimento de estudos (ESRI, 2022), através da combinação de mapas, dados e instrumentos para análise. Sendo assim, podemos compreender e evidenciar esse tipo de tecnologia e sistemas traduzidos em ferramentas e aplicações como produtos como Google Maps e Earth, entre outras, se tornando cada vez mais integradas e inseridas no uso cotidiano de pessoas comuns (ELLIOTT, 2014). Tais sistemas chegaram no Brasil no início da década de 80, com a vinda de pesquisador canadense, Dr. Roger Tomlison, responsável pela criação do primeiro SIG (BOLFE, 2006).

4.4.2 SISTEMAS DE REFERÊNCIA DE COORDENADAS

Conhecido pela sigla em inglês CRS (*Coordinate Reference Systems*), é uma metodologia para definir a localização de um objeto no espaço (JANSSEN, 2009), existindo diversos padrões, compreendendo informações como: projeção, que é a representação bidimensional do globo, elipse, que é a forma da Terra, e

por fim, o datum, que é responsável por permitir a fixação de pontos de acordo com suas coordenadas, as origens e direção dos eixos. Evidencia-se a necessidade da conversão para um CRS comum quando se tem intenção de comparar conjuntos de dados, tornando tal análise viável (LOPES; DELBEM; SOUSA, 2021).

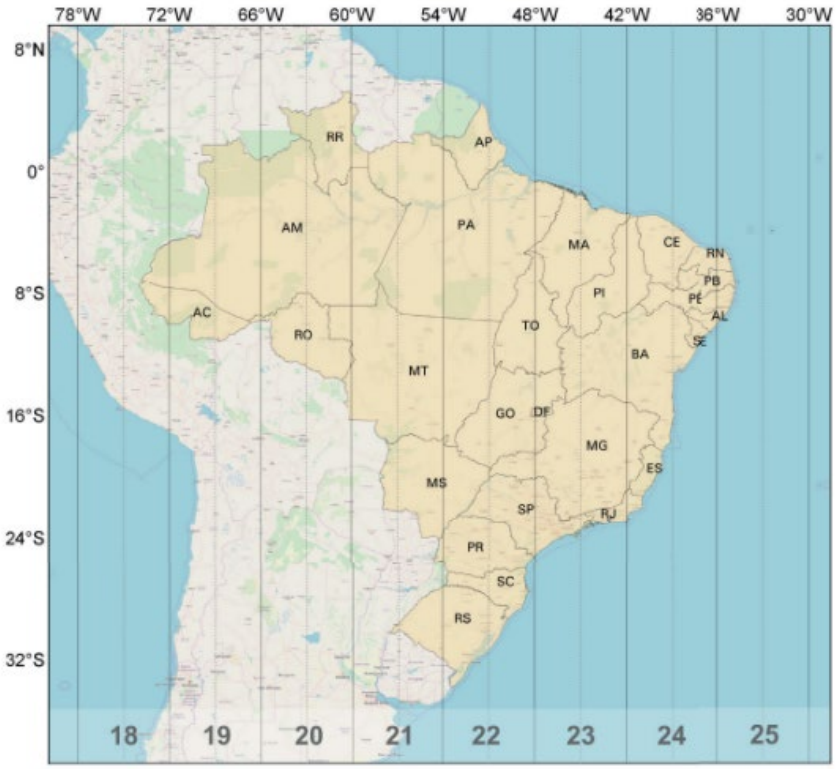
4.4.3 PROJEÇÕES

Regularmente precisamos expressar posições sobre superfícies planas, seguindo uma grade de coordenadas, isto é, num plano cartesiano de duas dimensões, adequando pontos espaciais em três dimensões através de sistemas matemáticos (JANSSEN, 2009). As projeções buscam representar a totalidade ou parte da Terra, sendo que cada modelo apresenta vantagens e desvantagens, cabendo a escolha ser baseada com a finalidade de reduzir as distorções (IBGE, 2019), sendo impossível converter uma superfície que possui três dimensões numa superfície em duas dimensões, com isso se desenvolveu diversas projeções de mapa com intuito de atender propriedades cartográficas (JANSSEN, 2009).

No Brasil temos a adoção da projeção UTM (Universal Transversa de Mercator), usado pela Diretoria de Serviço Geográfico e pelo IBGE como padrão (IBGE, 2019), amplamente utilizado em todo o mundo (JANSSEN, 2009).

Este sistema de projeção é classificado como cilíndrico conforme, e tem como característica preservar ângulos, entretanto altera áreas com distorções que não ultrapassam 0,5%, além de dividir a Terra em 60 fusos de 6° de amplitude longitudinal (HAMADA; DO VALLE GONÇALVES, 2007), sendo o Brasil dividido por 8 fusos (figura 3), entre os fusos 18 a 25 (IBGE, 2019), tendo como aplicação mapeamento básico em grandes e médias escalas, assim como adotado em cartas topográficas (HAMADA; DO VALLE GONÇALVES, 2007).

Figura 3: Fusos UTM no Brasil



Fonte: IBGE (2019)

4.4.4 GPS

GPS é a sigla de Global Positioning System, um sistema de posicionamento, navegação e mensuração de tempo, desenvolvido pelo departamento de defesa dos Estados Unidos, financiado pela Força Aérea do país, que opera um conjunto de satélites e sistema de monitoramento em terra ao redor do mundo (MCNEFF, 2002).

O funcionamento do GPS apresenta cobertura global, atingindo o número de 24 satélites inseridos no sistema no ano de 1995, tendo sido projeto inicialmente com o propósito de ampliar o poderio militar dos Estados Unidos e aliados (ZANOTTA; CAPPELLETTO; MATSUOKA, 2011).

Seu funcionamento básico pode ser descrito da seguinte maneira (MCNEFF, 2002):

1. O sistema tem suas operações baseadas na triangulação de sinais emitidos pela rede de satélites.
2. A triangulação é realizada com o sistema apurando distâncias usando tempos de viagem de sinal de rádio emitido.

3. Cada satélite compartilha sua exata posição e tempo preciso da transmissão.
4. O usuário do GPS recebe o sinal de cada satélite e grava suas posições e o momento do recebimento do sinal.
5. O receptor calcula a posição das distâncias apuradas e por fim temos as coordenadas e tempo preciso utilizando o sinal obtido de quatro satélites.

Inicialmente, apesar da precisão e acesso à tecnologia, existiam limitações impostas pelo departamento de defesa americano, motivado por razões de segurança nacional, segregando a precisão do sinal, originando dois tipos de serviço de posicionamento: o SPS (Standard Positioning System) e o preciso PPS (Precise Positioning System), que era voltado para uso militar. Essa limitação da precisão do sinal foi descontinuada no ano 2000 (ZANOTTA; CAPPELLETTO; MATSUOKA, 2011).

4.5 PROGRAMAÇÃO

Em programação, serão fundamentados tópicos relacionados com a aplicação apresentada nesta obra, em especial o Python, linguagem de programação utilizada, assim as principais bibliotecas responsáveis pelo funcionamento da aplicação CLIMATER.

4.5.1 BOTS

Bot pode ser definido como um programa autônomo e reativo, que funciona de maneira independente e contínua, adaptando suas ações de acordo com o contexto de operação (TSVETKOVA et al., 2017). Compreendendo isso, podemos estabelecer funções para a disseminação automática de informação, viabilizando um meio eficiente de alcançar pessoas e grupos com pouco esforço (STIEGLITZ et al., 2017).

Sendo a tecnologia parte fundamental em nossas vidas nos dias de hoje (DE OLIVEIRA; SANTOS; NETO, 2016), temos a integração dessas entidades no nosso cotidiano, que foram visionadas para agir e pensar como humanos, indo além da automação de tarefas que seriam resolvidas por pessoas, servindo na resolução intelectual de tarefas complexas (LEBEUF, 2018).

4.5.2 PYTHON

Python é uma linguagem de programação interpretada, interativa e orientada a objetos, que incorpora módulos, exceções, tipagem dinâmica, tipos de dados dinâmicos e classes. Além de suportar vários paradigmas da programação além da programação orientada a objetos, como programação procedural e funcional (PYTHON, 2022). Sendo desenvolvida por Guido Von Rossum, possui sintaxe clara e objetiva, sendo facilmente compreendida por iniciantes na programação (PESENTE et al., 2016). Devido sua simplicidade, tais características se evidenciam quando comparados com os desafios enfrentados para implementar o mesmo tipo de solução em outras linguagens de programação adotadas na indústria (HAN et al., 2021).

O Python possui uma biblioteca básica, ou seja, um leque de ferramentas, instruções, funções, que se propõem em resolver uma diversidade de questões, desde manipulação de textos, na forma de *strings*, interação com protocolos de internet, engenharia de software, assim como interfaces de sistemas operacionais, possibilitando manipulação de diretórios e arquivos (PYTHON, 2022). E devido suas características, Python hoje em dia é a linguagem de programação mais popular, adotada por aproximadamente 80% dos programadores do mundo, sendo utilizada em diversos projetos de *big techs* como Intel, Facebook, Spotify e Netflix. Em seguida, no *rank* de popularidade, temos Java e Javascript (GEEKSFORGEEKS, 2022). Python é também a segunda linguagem de programação com maior demanda no mercado, sendo usada para a construção de aplicações robustas para *back end*, ciência de dados e desenvolvimento de programas no geral, sendo considerada uma linguagem de propósito generalista (BERKELEY, 2022).

4.5.3 AMBIENTES DE DESENVOLVIMENTO

São programas, que se assemelham à primeira vista com editores de texto, que à princípio podem ser usados para programar, entretanto os ambientes de desenvolvimento integrado se destacam para esse propósito, por serem munidos de ferramentas e funções específicas para programação, tornando o processo mais ágil, assim como apontando possíveis erros de lógica

(RED HAT, 2019), reforçando aplicação e adoção de boas práticas de desenvolvimento de programas.

4.5.3.1 JUPYTER NOTEBOOK

É um ambiente de desenvolvimento, de código aberto e gratuito, que funciona como um caderno de anotações virtual dentro do navegador, incorporando um ambiente de desenvolvimento, que fornece suporte para rodar códigos, apresentar dados, visualizações, detalhando o processo de pesquisa e estudo, ao mesmo tempo que estimula um processo de desenvolvimento sequencial, interativo, descritivo e visual (RANDLES et al., 2017). Dessa forma, o Jupyter Notebook serviu como ambiente primário para prototipagem das funções iniciais deste projeto.

4.5.3.2 VISUAL STUDIO CODE

É um ambiente de desenvolvimento, de código aberto e gratuito, nos moldes tradicionais de editor de códigos, desenvolvido pela Microsoft.

4.5.4 BIBLIOTECAS

Bibliotecas podem ser entendidas como uma interface de funções que facilitam a execução de tarefas específicas dentro de algoritmos.

4.5.4.1 PANDAS

Seu nome tem origem no termo em inglês *panel data*, que se refere a conjuntos de dados multidimensionais abordados na estatística e econometria (MCKINNEY, 2011).

O Python enfrentou diversas barreiras na sua adoção dentro das mais diversas carreiras da área de exatas, devido ausência de suporte para questões relacionadas a estatística, matemática e análise de dados, mas no ano de 2008 temos o início do desenvolvimento da biblioteca Pandas, acabando os problemas de manipulação de metadados, sendo metadados rótulos que podemos dar para informações (MCKINNEY, 2011).

Sendo desenvolvida inicialmente para a área de análise de dados financeiros, seus criadores esperam que suas ferramentas possibilitem tornar o Python mais atrativo para a área científica, servindo de ambiente para

computação estatística prática para o público acadêmico, assim como para a indústria (MCKINNEY, 2011).

4.5.4.2 GEOPANDAS

É uma biblioteca para Python de código aberto que serve como uma evolução do Pandas, possibilitando operações de bases de dados geoespaciais, manipulando tipos geométricos (LOPES; DELBEM; SOUSA, 2021).

4.5.4.2.1 SHAPE FILE

É um formato de dado, não topológico para a base de dados geoespaciais e vetoriais, constituindo uma coleção de arquivos, contendo alguns que são obrigatórios para o funcionamento básico de um *shape file*, que são os arquivos com extensão .shp, .shx e .dbf, cada um atendendo um propósito, sendo o primeiro, responsável por armazenar informações da geometria, o segundo serve como índice, viabilizando buscas dos dados, e o terceiro constituí os atributos em colunas para cada objeto geométrico descrito (LOPES; DELBEM; SOUSA, 2021). Tais arquivos são manipulados dentro do GeoPandas.

4.5.4.3 MATPLOTLIB

É mais uma biblioteca do Python de código aberto, para plotagem de gráficos de qualidade, construído de forma que possa criar facilmente elaborados gráficos (ARI; USTAZHANOV, 2014). Matplotlib segue a filosofia de que deve ser possível criar gráficos com poucas linhas de código, sem a necessidade de instanciar objetos, definir propriedades, entre outras especificidades (BARRETT et al., 2005) e dessa forma estabelece ambiente interativo de pesquisa e desenvolvimento, sendo acessível para todos os níveis de programadores (MCKINNEY, 2011). Bibliotecas especializadas em visualização de dados, como Matplotlib, tem grande importância na confecção de trabalhos acadêmicos e propagação de conhecimento científico (WASKOM, 2021).

4.5.4.4 AIOGRAM

Aiogram é uma biblioteca do Python voltada para criação de *bots* para o aplicativo Telegram, que se destaca pela simplificação do processo para criação desses *bots*, tornando o processo mais ágil, além de contar interface de

comunicação totalmente assíncrona, provendo alta performance em procedimentos de leitura e escrita de arquivos, assim como sua transmissão via Internet, otimização de encadeamento de processos e eventos, entre outras vantagens.

4.5.4.5 PYCEP-CORREIOS

A biblioteca PyCEPCorreios é uma simples biblioteca criada por Michell Stuttgart, atuando como uma API para busca de CEP integrado aos serviços dos Correios, ViaCEP e ApiCEP. O emprego dessa biblioteca foi essencial para a oferta de método de referência alternativo, possibilitando a conversão do CEP em coordenada geográfica.

4.6 TELEGRAM

O Telegram é um aplicativo gratuito de comunicação e possui grande influência com código fonte aberto (SUTIKNO et al., 2016), uma plataforma de rede social que facilita a comunicação entre pessoas, servindo também como um meio colaborativo de produção (KAUR SWARAN SINGH et al., 2020), contando com 100 milhões de usuários ao redor do mundo, segundo seus desenvolvedores (SETIAJI; PAPUTUNGAN, 2018).

No momento que temos o crescimento da adoção de *smartphones*, tais aplicativos como o Telegram surgem como uma necessidade (SUTIKNO et al., 2016), possibilitando que seus usuários obtenham informações sobre diversos aspectos dentro de segundos, aposentando tecnologias de mensagens de texto como o SMS (SETIAJI; PAPUTUNGAN, 2018). Tais aplicativos vão além da simples troca de mensagens na forma de texto, contemplando formatos como chamadas telefônicas, chamadas de vídeo, compartilhamento de arquivos, conversas em grupos de usuários e contato com pessoas estrangeiras de maneira simples (SUTIKNO et al., 2016), se destacando de demais aplicativos similares por oferecer encriptação de dados, assim como a possibilidade de criar bots, munido de diversas funções programadas baseada na documentação oferecida para sua API (DE OLIVEIRA; SANTOS; NETO, 2016).

Compartilhando de muitas similaridades com seu grande rival, o WhatsApp, entretanto se destaca por características de segurança, anonimato e

privacidade, viabilizando a comunicação sem a necessidade de exposição do número telefônico dos usuários, apenas dependendo do compartilhamento do nome de usuário para ser alcançado por outras pessoas (SUTIKNO et al., 2016).

4.6.1 BOTS NO CONTEXTO DO TELEGRAM

Os bots foram uma das últimas funcionalidades apresentadas pelos desenvolvedores desse aplicativo. Através da interface do aplicativo o usuário tem a capacidade de interagir com o *bot* com comandos pré-estabelecidos, que resultarão no atendimento das necessidades do usuário, travestida de uma habitual conversa entre amigos (SETIAJI; PAPUTUNGAN, 2018). O ato de terceirizarmos tarefas complexas, que tomam tempo, podem ser abordadas e resolvidas por esses programas, pois é crescente a demanda por conveniência e mecanismos que poupem tempo (KLAUS; ZAICHKOWSKY, 2020).

As funcionalidades dos bots no Telegram podem se tornar mais elaboradas e naturais aos usuários com a adição de mecanismos de processamento de linguagem natural, que é um campo da inteligência artificial que trabalha a interpretação e geração de textos (GADELHA, 2019), funcionando através de métodos que trabalham a coocorrência de palavras (CAMBRIA; WHITE, 2014).

Em trabalhos recentes, vemos o uso de bots para compartilhamento e disseminação de informações sobre tempo (KATARINE et al., 2019), temos outra obra que atua com o foco em tornar acessível a disseminação de informação de uma estação hidrometeorológica mantida pela FATEC Jahu, em Jaú, no estado de São Paulo (TERSI et al., 2016), temos também trabalho que apresenta a aplicação de um *chatbot* para redes sociais que atua na área de classificação de soja para sojicultores (BARBOSA et al., 2020), sendo *chatbot* o emprego de bots dentro do contexto de interfaces de comunicação, como programas e aplicativos de conversa, como WhatsApp ou Telegram.

Apesar de inúmeras aplicações criadas com *chatbots*, não se encontra registros de obras que tratem dos pontos explicitados neste trabalho.

4.7 PRODUTO MÍNIMO VIÁVEL

Produto Mínimo Viável, ou a sigla do inglês MVP, como é popularmente chamado no meio de desenvolvimento de programas, é a versão mais simples de um produto que está sendo concebido, provendo condições para que a aplicação proposta esteja em uso em curto tempo e possa ser validada enquanto negócio (CAROLI, 2015).

5 METODOLOGIA

Para o desenvolvimento deste trabalho, após ser definido a idéia do trabalho para oferecer a visualização de dados climáticos, suas médias e séries temporais de maneira simples e fácil, foram traçados os elementos necessários para viabilizar o funcionamento da aplicação e como esse processo ocorreu.

5.1 ABORDAGEM DO PROBLEMA

Após análise do problema abordado, compreendeu-se a necessidade de tratar todas as requisições de dados pautadas em pontos de referências, sendo esses pontos posicionados sobre o plano, de maneira geograficamente localizável, ou seja, pontos georreferenciados, para então poder compreender a relação de distância entre o ponto de referência e a posição das EMAs dentro dos limites do estado do Rio de Janeiro, criando então a validação necessária para ofertar os dados adequados segundo o ponto de interesse do usuário.

Havendo explicitado seu conceito básico de funcionando, compreendeu-se a necessidade de obter as seguintes informações para a construção do MVP:

- Dados meteorológicos das estações automáticas do Rio de Janeiro na base de dados do INMET.
- Obter dados geoespaciais do Rio de Janeiro na base de dados do IBGE.

A posse desses dados já atende todos os aspectos técnicos para a concepção de seu mecanismo básico de funcionamento, entretanto concluiu-se que possíveis usuários não saberão como obter as coordenadas geográficas de seu ponto de interesse. Com isso estabeleceu-se a necessidade básica de ofertar meios alternativos para viabilizar a consulta desses dados climáticos, sendo então apontadas a adoção de georreferenciamento da sede dos municípios do

estado e a conversão do CEP (código postal) em coordenadas geográficas. A solução criada para converter o CEP num ponto georreferenciado constará nos anexos desta obra. Com isso totalizamos 4 métodos para viabilizar a consulta de dados climáticos e suas séries temporais.

A localização das 92 sedes de municípios do Rio de Janeiro foi obtida no site da Fundação CEPERJ, contando com as coordenadas latitude e longitude, seguem o padrão Grau-Minuto-Segundo. Tais dados estão disponíveis em: http://arquivos.proderj.rj.gov.br/sefaz_ceperj_imagens/Arquivos_Ceperj/ceep/dados-estatisticos/series-historicas/excel/cogeo/Tab%202.0.0.34.html.

5.2 DADOS METEOROLÓGICOS

Para se obter os dados climáticos, delimitou-se a atuação da aplicação para atender apenas as estações meteorológicas automáticas, configurando o conjunto de dados que deveriam ser extraídos do INMET.

5.2.1 CATÁLOGO DE ESTAÇÕES METEOROLÓGICAS

Após estabelecer a exclusiva oferta de dados das estações automáticas, foi possível criar a listagem das estações que estariam disponíveis no projeto. Abaixo encontram-se as 26 estações meteorológicas automáticas abordadas no trabalho mantidas pelo INMET. Estas são:

1. A601 (Seropédica-Ecologia Agrícola)
2. A602 (Rio de Janeiro-Marambaia)
3. A603 (Duque de Caxias-Xerém)
4. A604 (Cambuci)
5. A606 (Arraial do Cabo)
6. A607 (Campos dos Goytacazes)
7. A608 (Macaé)
8. A609 (Resende)
9. A610 (Petrópolis-Pico do Couto)
10. A611 (Valença)
11. A618 (Teresópolis-Parque Nacional)
12. A619 (Paraty)
13. A620 (Campos dos Goytacazes-São Tomé)
14. A621 (Rio de Janeiro-Vila Militar)

15. A624 (Nova Friburgo-Salinas)
16. A625 (Três Rios)
17. A626 (Rio Claro)
18. A627 (Niterói)
19. A628 (Angra dos Reis)
20. A629 (Carmo)
21. A630 (Santa Maria Madalena)
22. A635 (Itatiaia-Agulhas Negras)
23. A636 (Rio de Janeiro-Jacarepaguá)
24. A652 (Rio de Janeiro-Forte de Copacabana)
25. A659 (Silva Jardim)
26. A667(Saquarema-Sampaio Correa).

O INMET mantém um catálogo atualizado e georreferenciado de todas as suas estações meteorológicas, sendo no caso deste trabalho as estações meteorológicas automáticas, que poderão ser tratadas pela sigla EMA em partes do trabalho. Este catálogo está disponível em: https://portal.inmet.gov.br/paginas/catalogoaut#.

Os dados podem ser baixados no formato CSV (comma-separated values, ou seja, valores separados por vírgula), contendo informações de todas as EMAs operantes em território nacional, sua situação operacional, latitude, longitude, altitude e a data que a estação foi instalada, além de seu nome, geralmente sendo o nome do município onde se encontra, e por fim, a unidade federativa. Existe outro catálogo, que aborda as estações meteorológicas do tipo convencional, entretanto não serão abordadas neste trabalho.

Em posse desse catálogo foi possível observar quais estações estavam operantes e presentes no Rio de Janeiro e suas posições segundo coordenadas geográficas. Essas informações serão úteis para todas as etapas de validação durante as interações com usuário do *bot*.

5.2.2 SÉRIE TEMPORAL

Série temporal é a simples disposição de dados ao longo de um dado período, sendo indexados por data, horário ou ambos.

Para este trabalho definiu-se uma série temporal de dados climáticos contemplando o período entre o ano de 2017 até o dia 31 de agosto de 2022, que é a última data de dados liberados publicamente para download pelo sistema do INMET.

Nisso se obteve arquivos únicos para cada ano dentro desse período, para cada estação meteorológica abordada, que foram concatenados gerando um único arquivo, com o intuito de facilitar operações gerais necessárias para a plotagem. Além disso, é preciso destacar que tais séries temporais ainda levam em conta a data de instalação das EMAs, dado que algumas delas foram instaladas após o ano de 2017, ofertando uma série temporal mais curta. Essa informação também é tratada na validação das requisições realizadas pelos usuários, ou seja, a informação apenas será retornada na forma de informação para o usuário se o período requisitado pelo usuário for válido para a estação abordada em dada consulta.

5.2.3 OBTENÇÃO DE DADOS METEOROLÓGICOS

Esses dados foram então obtidos através de uma simples função, que se comunica com os servidores do INMET através da sua API pública, requisitando os dados diários registrados das EMAs abordadas, coletando tais informações segmentadas para cada ano que contempla o período de 2017 até agosto de 2022, apresentando as seguintes variáveis climáticas:

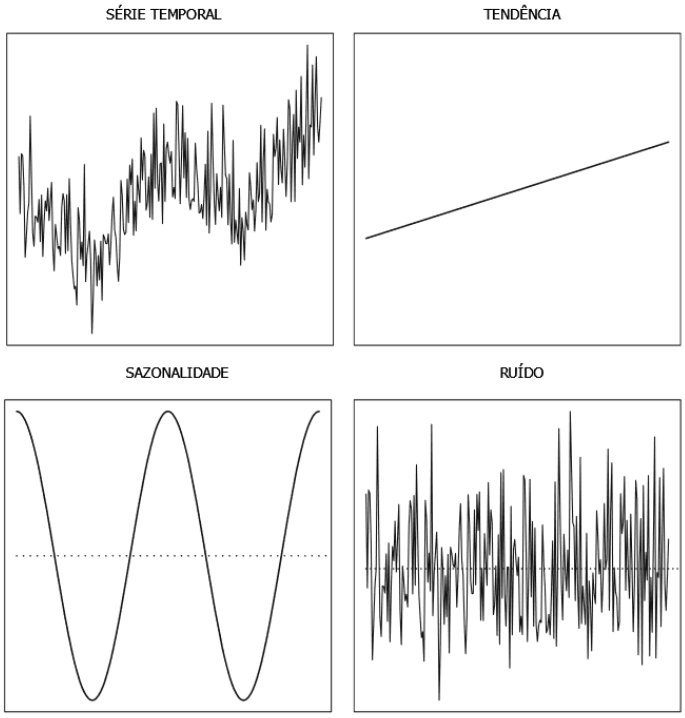
- Temperatura mínima (°C)
- Temperatura média (°C)
- Temperatura máxima (°C)
- Precipitação total (mm)
- Umidade relativa do ar, média (%)
- Umidade relativa do ar, mínima (%)
- Velocidade média do vento (m/s)

As informações da API do INMET para dados diários estão disponíveis em: https://portal.inmet.gov.br/manual/manual-de-uso-da-api-estações.

5.2.3.1 TENDÊNCIA DOS DADOS EM SÉRIES TEMPORAIS

Séries temporais podem ser decompostas em três partes, que são: tendência, sazonalidade e ruído (figura 4). A tendência pode representar a melhora ou piora gradual de um determinado dado. Já a sazonalidade se refere à repetição de fenômenos numa determinada frequência temporal (BARNETT; DOBSON, 2010). Dispor da tendência e sazonalidade dos dados climáticos agregaria fortalecendo as análises dos dados obtidos nas consultas realizadas no CLIMATER. Entretanto, os dados faltantes das estações meteorológicas automáticas (apresentados na tabela 1) impedem a decomposição das séries temporais por limitações de bibliotecas do Python conhecidas e adotadas no projeto. Com isso temos duas opções, preencher as falhas imputando dados através do emprego de bibliotecas do Python especializadas ou aplicar o cálculo da média móvel para a série temporal.

Figura 4: Decompondo série temporal

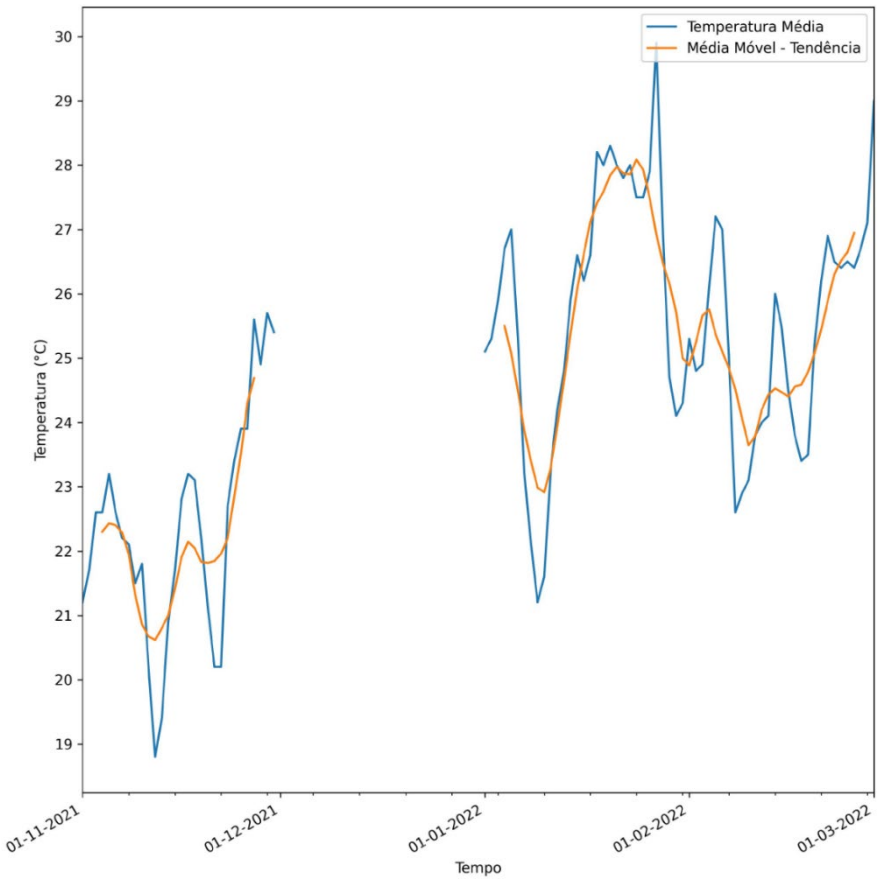


Fonte: Adaptação (BARNETT; DOBSON, 2010)

Levando em conta limitação técnica e o aspecto da produção de um MVP, empregou-se a média móvel simples, tendo como base para cálculo 7 dias, com o objetivo de facilitar a compreensão da dinâmica das variáveis climáticas

exploradas dentro do CLIMATER (figura 5). Dessa forma contamos com um simples mecanismo que evidencia a tendência das observações (HANSUN, 2013).

Figura 5: Média móvel, falhas e tendência de dados climáticos no CLIMATER



Fonte: Autor (2022)

Todas as estações meteorológicas abordadas apresentam falhas nos dados registrados para o período contemplado no projeto, iniciando em 2017 até agosto de 2022, evidenciando um problema de falta de manutenção observado em todo o país (ALVIM, 2021), como no caso da estação A628, localizada em Angra dos Reis (figura 5), possivelmente ocasionadas por interrupção do funcionamento de sensores. Com isso temos cálculos o porcentual de falhas dentro das séries temporais disponibilizadas neste MVP (tabela 1).

Tabela 1: Dados Faltantes das EMAs do Rio de Janeiro

Código	Temp. Mínima	Temp. Média	Temp. Máxima	Precipitação	Umid. Mínima	Umid. Média	Velocidade Média Ventos
A601	0	0,14	0	2,85	0	0	0
A602	0,43	3,19	0,43	0,43	7,64	7,73	20,73
A603	0,05	0,24	0,05	0	1,16	13,44	0
A604	4,88	12,32	4,88	8,6	11,79	12,52	16,72
A606	2,85	3,24	2,95	5,51	9,71	23,3	7,73
A607	2,85	3,14	2,9	3,19	2,8	2,9	3
A608	5,12	5,9	5,17	5,75	2,56	4,3	5,36
A609	3,33	4,16	3,62	4,01	2,13	3	3,91
A610	25,08	32,91	25,52	31,56	20,01	31,22	47,8
A611	5,12	6,86	5,12	9,38	6,19	6,43	6,57
A618	1,64	6,14	1,64	1,84	5,41	9,28	6,04
A619	10,49	14,5	10,54	16,53	11,99	14,5	14,5
A620	14,06	15,03	14,06	14,98	5,99	21,51	14,35
A621	4,01	4,01	4,01	4,2	0,14	3	4,01
A624	3,48	5,36	3,48	12,13	72,02	4,88	82,79
A625	3,38	3,77	3,43	3,67	2,46	3,24	4,3
A626	1,98	8,6	2,08	4,25	4,3	5,17	20,44
A627	4,43	11,51	4,63	5,69	9,26	12,83	11,77
A628	14,61	28,03	14,67	26,39	15,32	17,88	26,72
A629	0,63	0,7	0,7	0,7	12,24	12,59	0,77
A630	9,39	18	9,39	9,74	10,8	24,06	17,85
A635	4,05	4,38	3,67	4,54	8,26	8,81	24,36
A636	2,06	2,43	1,95	2,43	0,16	1,84	2,11
A652	1,16	1,55	1,16	1,4	0,58	1,06	1,26
A659	1,98	2,32	2,03	2,27	0,05	1,5	2,03
A667	10,58	12,03	10,63	11,84	1,4	6,19	11,45

Fonte: Autor (2022)

5.3 DADOS GEOESPACIAIS

Os dados geoespaciais do *bot* CLIMATER são compostos basicamente por dois tipos malhas. A primeira malha tem em sua composição os 92 municípios do estado do Rio de Janeiro, nisso nós temos os seus limites e coordenadas. Já a segunda malha refere-se aos limites do estado do Rio de Janeiro, que servirá para validação de coordenadas informadas pelo usuário, retornando casos positivos de coordenadas informadas regulares para as regiões abordadas no projeto.

5.3.1 PADRONIZAÇÃO DO CRS

Todos os dados geoespaciais do projeto são convertidos para um CRS em comum, como pautado por (LOPES; DELBEM; SOUSA, 2021), com isso, o CRS foi definido da seguinte maneira: projeção UTM estabelecendo a representação bidimensional do globo terrestre, compreendendo a zona 23, no hemisfério Sul, definição do formato da Terra segundo o padrão elipsóide GRS80 e unidades como quilômetros.

5.3.2 DADOS GEOESPACIAIS DOS MUNICÍPIOS

Tais dados geoespaciais dos municípios foram obtidos no repositório público do IBGE, estando disponível para download no seguinte link: https://geoftp.ibge.gov.br/organizacao_do_territorio/malhas_territoriais/malhas_municipais/municipio_2021/UFs/RJ/RJ_Municipios_2021.zip.

5.3.3 DADOS GEOESPACIAIS DO RIO DE JANEIRO

Já os dados geoespaciais para o estado do Rio de Janeiro, também foram obtidos no repositório público do IBGE, na mesma pasta que se encontram as malhas para municípios, suas meso e microrregiões. Os arquivos para o Estado do Rio de Janeiro usados podem ser encontrados no seguinte link: https://geoftp.ibge.gov.br/organizacao_do_territorio/malhas_territoriais/malhas_municipais/municipio_2021/UFs/RJ/RJ_UF_2021.zip.

5.4 DESENVOLVIMENTO DO BOT

Todo o projeto foi desenvolvido em Python, em sua versão 3.7, que através de uma diversidade de bibliotecas, estabelece a comunicação com seus usuários através do aplicativo de conversas Telegram, ofertando dados de cunho climático. Todos os arquivos que fazem parte do bot estão hospedados em repositório do Github, disponível em: https://github.com/dan-alvares/CLIMATERBOT.

5.4.1 CRIAÇÃO DO BOT

Para se criar qualquer *bot* no Telegram, antes de tudo, é preciso registrá-lo através do serviço oficial do Telegram conhecido por BotFather. Este pode ser acessado diretamente pelo aplicativo do Telegram, buscando na lista de contatos pelo nome “BotFather” e iniciar uma conversa com este bot oficial. Nele o usuário deverá clicar em iniciar ou “/start”, então em seguida em “/newbot” para criar um novo bot, para em seguida dar um nome para o seu bot. É importante ressaltar que o nome de todos os bots devem terminar em “bot”, por exemplo, CLIMATERBOT.

Após a criação do bot você será apresentado com um código chamado de *token*, sendo este responsável pela comunicação entre o código escrito em Python e a HTTP API mantida pelo Telegram. Este código é único por usuário e

deverá ser mantido em segredo. Tal *token* deverá ser copiado e salvo no arquivo mantido no diretório raiz do *bot* chamado de config.py. Neste arquivo existe a declaração da seguinte variável "*token_bot*" seguida do sinal de igual e aspas. O token deverá ser copiado e colado entre as aspas, mantendo as aspas, e o arquivo deverá ser salvo em seguida.

Neste mesmo *bot* de serviço fornecido pelo Telegram é possível editar algumas informações, como a descrição do seu bot, um avatar, entre outros aspectos que não serão abordados por não impactarem a concepção do MVP.

5.4.2 DEPENDÊNCIAS DO PROJETO

Essa listagem de bibliotecas sustentam todas as etapas do projeto, garantindo seu funcionamento básico. Tal lista foi obtida com a execução do comando *pip freeze* num ambiente de desenvolvimento integrado. Tal comando exportará os detalhes do ambiente virtual configurado para a produção deste projeto, resultando as dependências descritas abaixo.

- aiogram==2.22.2
- anyio==3.5.0
- beautifulsoup4==4.11.1
- descartes==1.1.0
- geopandas==0.6.1
- geopy==2.2.0
- matplotlib==3.5.2
- matplotlib-inline==0.1.6
- numpy==1.21.5
- pandas==1.3.5
- pillow==9.2.0
- pycep-correios==5.0.0
- python-telegram-bot==13.14
- requests==2.28.1
- scipy==1.7.3
- Shapely==1.6.4

Após a instalação do Python, bastará executar o seguinte comando no terminal de uma IDE: “pip install -r requirements.txt” sem as aspas. Dessa forma teremos a instalação de todos os pacotes e bibliotecas necessárias para rodar o bot CLIMATER.

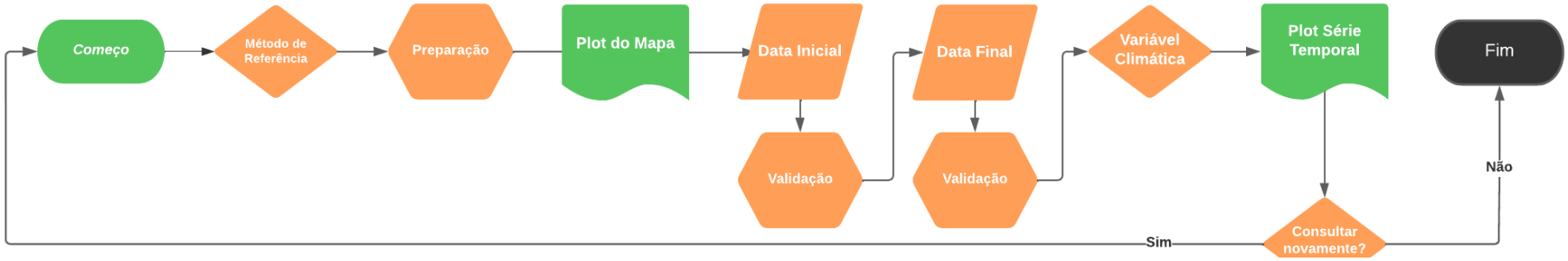
5.4.3 IMPLANTAÇÃO DO BOT

Em seguida o código-fonte do bot deverá ser clonado do repositório, bastando baixar seu conteúdo compactado, disponível em: https://github.com/dan-alvares/CLIMATERBOT/archive/refs/heads/main.zip.
Esse arquivo deverá ser descompactado, para então definir o *token* de acesso, já descrito nessa obra. Bastando então abrir a linha de comando e usar o comando “python climater_bot.py” e se tudo tiver sido configurado como descrito você terá o *bot* respondendo aos comandos dentro do Telegram em seguida.

5.5 FLUXOGRAMA DE PROCESSOS

Em posse dos dados georreferenciados das estações meteorológicas e das malhas geoespaciais na forma de objetos dentro do GeoPandas, é facilmente construído a relação entre os pontos plotados sobre o mapa do estado do Rio de Janeiro, abrindo espaço para a definição do fluxograma básico de processos do *bot,* detalhado na figura 6. O fluxograma básico irá nortear o desenvolvimento da aplicação e definição das rotinas e chamadas de funções para as sucessivas validações, que por fim culminam na plotagem da série temporal, num período estabelecido pelo usuário, dentro das limitações de funcionamento da estação meteorológica, assim como seu início de operação.

Figura 6: Fluxograma básico

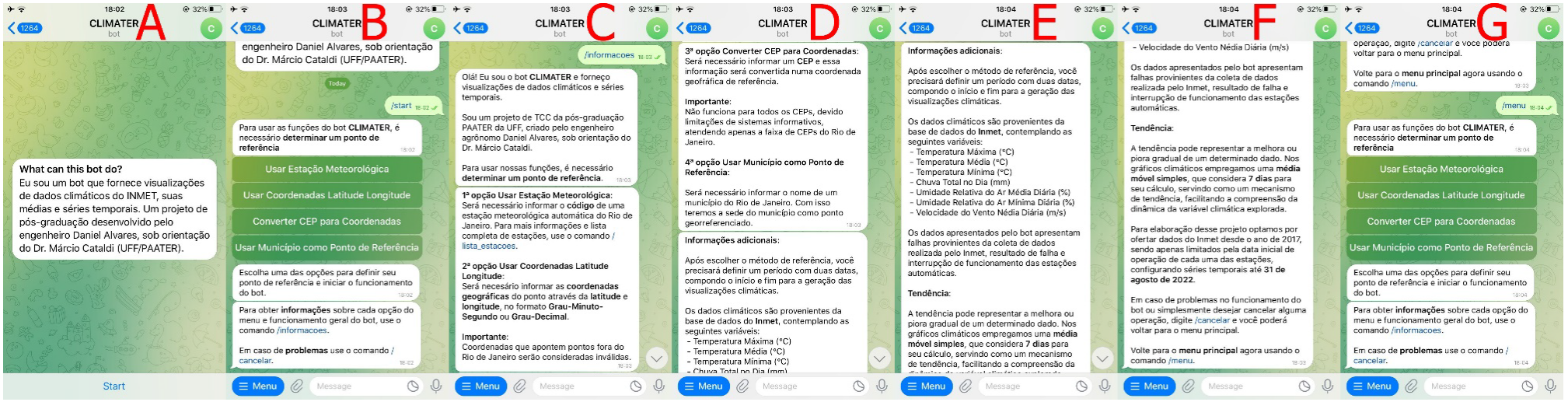


Fonte: Autor (2022)

5.6 CLIMATER NA PRÁTICA

No primeiro contato com o bot CLIMATER, o usuário será cumprimentado com uma breve descrição do projeto, como demonstrado na figura 7 tela A, e após executar o comando “/start” clicando no botão localizado no canto inferior da tela do aplicativo teremos a apresentação do menu principal da aplicação, demonstrado na figura 7 tela B. Neste momento é possível escolher o método que definirá um ponto de referência para requisitar dados climáticos e gerar suas visualizações. Em caso de o usuário ainda possuir dúvidas sobre o funcionamento do *bot*, é possível obter maiores informações, como detalhado na figura 7 telas C, D, E e F. Por fim, o usuário poderá retornar ao menu principal da aplicação usando o comando “/menu”, como na figura 7 tela G.

Figura 7: funcionamento inicial do bot

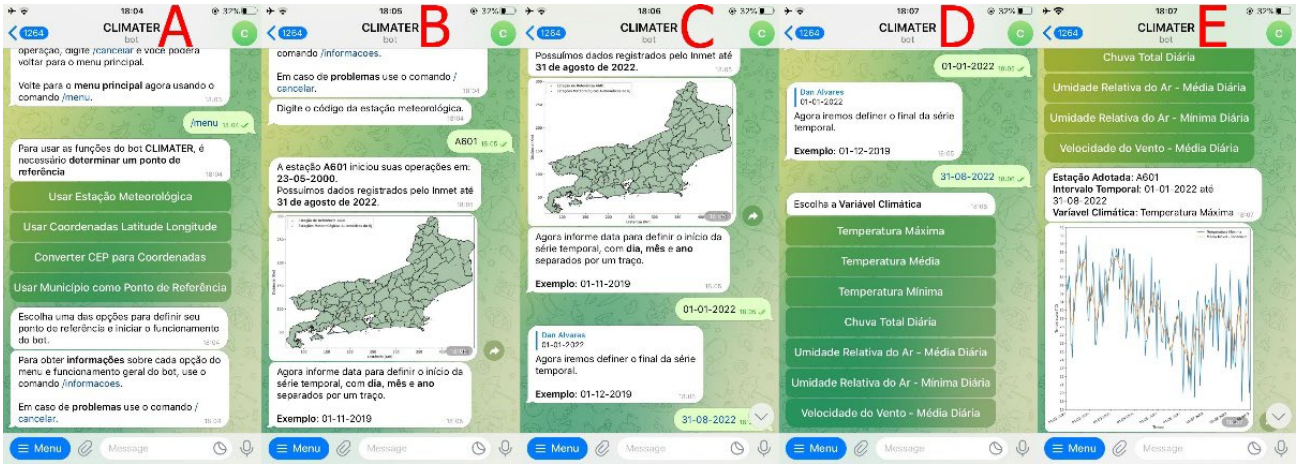


Fonte: Autor (2022)

5.6.1 ESTAÇÃO METEOROLÓGICA COMO REFERÊNCIA

O primeiro método disponibilizado para servir de referência para a obtenção de dados climáticos é “Usar Estação Meteorológica”, como na figura 8 tela A. Nesta opção será necessário informar o código de uma estação automática, adequado para quem já está habituado com o uso do sistema do INMET e conhece as estações adequadas para o caso de interesse. Após informar o código da estação, será exibido o mapa da região, como na figura 8 tela B. Todas as imagens geradas pelo *bot* podem ser exploradas com maiores detalhes, usando a função de *zoom* característica de telas sensíveis ao toque. Em seguida deverá informar data inicial e final, no formato dia, mês e ano, conforme exemplificado na figura 8, nas telas C e D, delimitando a série temporal. Por fim, bastará escolher a variável climática, como na figura 8 tela D, de acordo com as opções exibidas e teremos a plotagem dos dados requisitados, exibidos na figura 8 tela E.

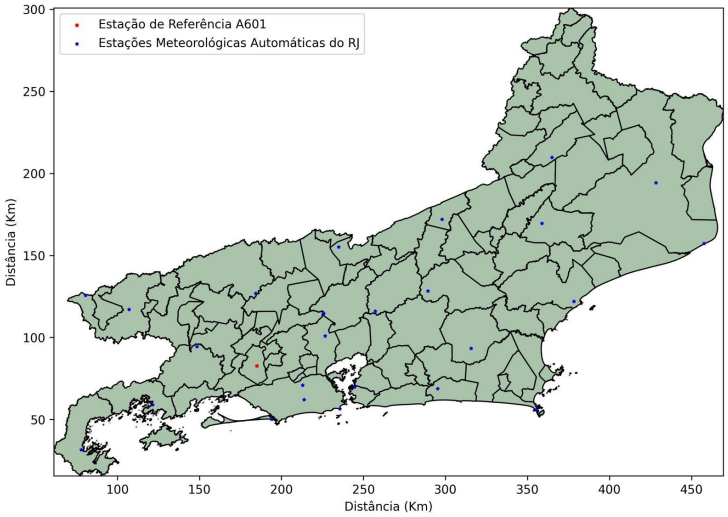
Figura 8: Escolha de EMA como referência



Fonte: Autor (2022)

Na figura 9 é possível conferir a posição da estação meteorológica automática A601 escolhida no exemplo apresentado, localizada em Seropédica, como o ponto vermelho. As demais estações são apresentadas no mapa, em azul. Para fins de cálculo e facilitar a compreensão das distâncias entre pontos, o CRS foi configurado para trabalhar com quilômetros.

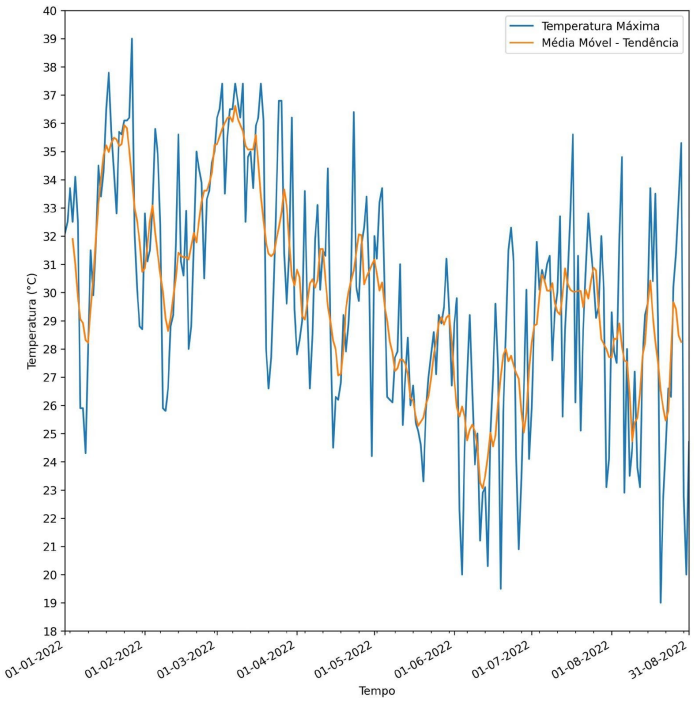
Figura 9: Escolhendo Estação Automática



Fonte: Autor (2022)

No final de todo ciclo operacional do *bot* o usuário será respondido com a plotagem da série temporal explorando a variável climática escolhida, como na figura 8 tela D, destacando o resultado obtido no exemplo na figura 10.

Figura 10: Detalhes do gráfico para Temperatura Máxima da EMA A601



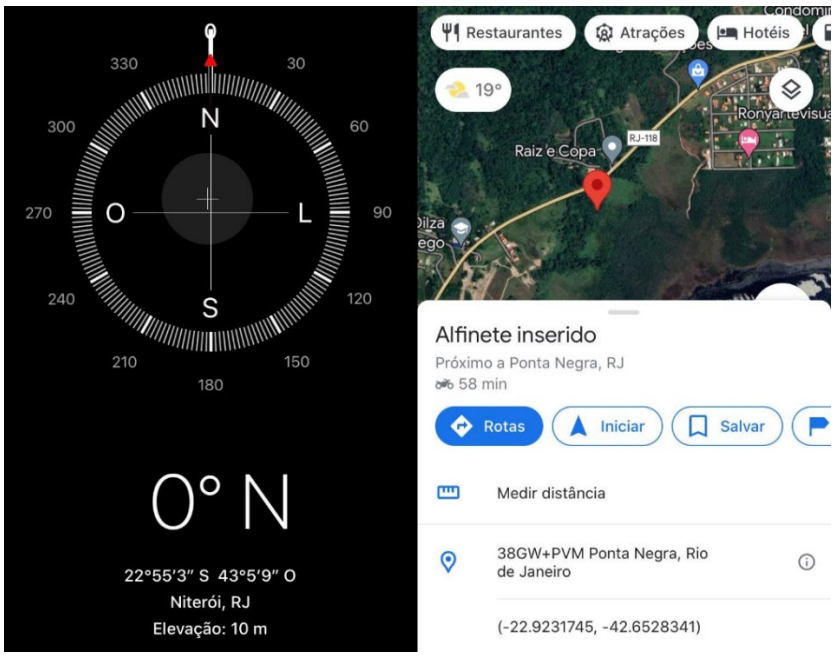
Fonte: Autor (2022)

5.6.2 COORDENADAS GEOGRÁFICAS COMO REFERÊNCIA

O segundo método para definir um ponto de referência para obter dados climáticos é “Usar Coordenadas Latitude Longitude”. Nesta opção o usuário deverá informar as coordenadas geográficas, seja no padrão graus, minutos e segundos ou no padrão graus decimais.

Temos aparelhos de GPS que possibilitam a consulta de coordenadas geográficas para a localização do usuário, entre outras funções. Na indisponibilidade de dispositivos GPS, existem aplicativos que cumprem com certa limitação essas operações, que atuam como bússola e GPS, como o aplicativo Bússola disponível de fábrica para celulares com sistema iOS, demonstrado na figura 11. As informações das coordenadas são exibidas no padrão graus, minutos e segundos. Já em sites e aplicativos de mapas, como Google Maps e similares, temos a adoção do padrão graus decimais, também demonstrado na figura 11. Com a inexistência de um padrão definido em sistemas informatizados e para evitar que usuário precise realizar conversões, foi necessário criar uma função que aceite os dois padrões, adequando as informações para os cálculos e procedimentos necessários para gerar pontos dentro da biblioteca GeoPandas, que trabalha com o padrão graus decimais.

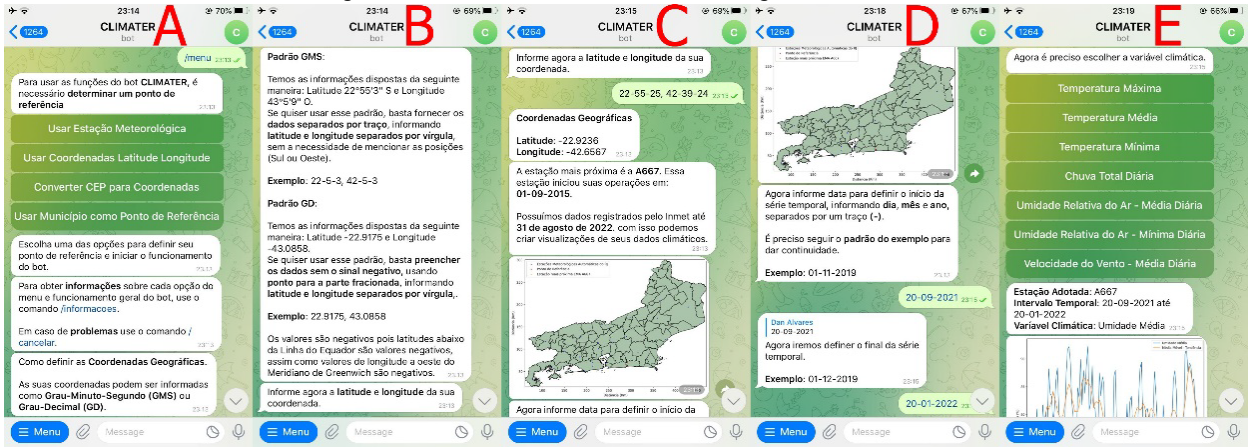
Figura 11: Consultando Coordenadas Geográficas utilizando aplicativos



Fonte: Autor (2022)

Após escolher “Usar Coordenadas Latitude Longitude” (figura 12 tela A), o usuário será informado como preencher as coordenadas do ponto, para os dois padrões (figura 12 tela B). Após informar as suas coordenadas, será possível plotar no mapa o ponto de interesse como referência. Em seguida deverá informar data inicial e final, no formato dia, mês e ano, delimitando a série temporal (exemplificado na figura 12, nas telas C e D). Por fim, bastará escolher a variável climática (figura 12 tela E), de acordo com as opções exibidas e teremos a plotagem dos dados requisitados (figura 12 tela E).

Figura 12: Escolha de Coordenadas Geográficas como referência



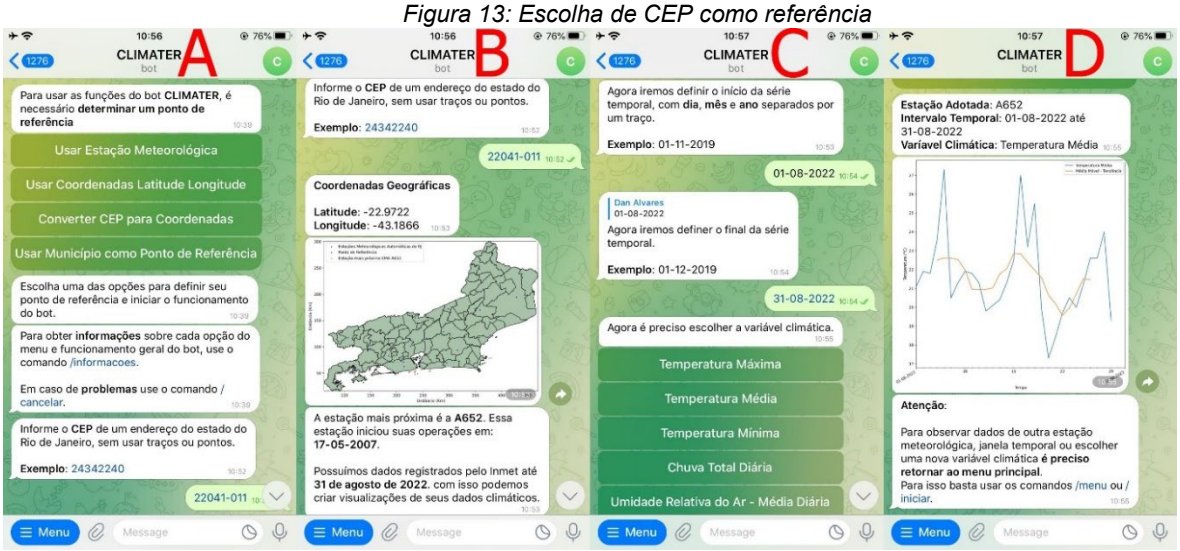
Fonte: Autor (2022)

5.6.3 CÓDIGO POSTAL COMO REFERÊNCIA

Iniciando a apresentação dos métodos alternativos para definição de pontos de referência, temos a possibilidade de uso do CEP, código de endereçamento postal. Contextualizando brevemente, cada número que faz parte dos 8 algarismos do CEP representa uma informação, como região do país, sub-região, setor, subsetor e suas divisões. Dessa forma, configuram-se intervalos de CEP para cada estado brasileiro, seus municípios e bairros, sendo divulgadas pelo serviço Correios (ARANHA, 1997), sendo o Rio de Janeiro compreendido pela faixa de CEP 20000-000 a 28999-999.

Após escolher “Converter CEP para Coordenadas” no menu principal da aplicação e em posse de um CEP como referência (figura 13 tela A), a biblioteca PyCEP do Python resolve as necessidades do projeto, convertendo essa informação em coordenadas (figura 13 tela B), repassando latitude e longitude de um ponto para o GeoPandas, adequando o funcionamento da aplicação conforme exemplificado no fluxograma básico (figura 13 telas C e D).

Figura 13: Escolha de CEP como referência

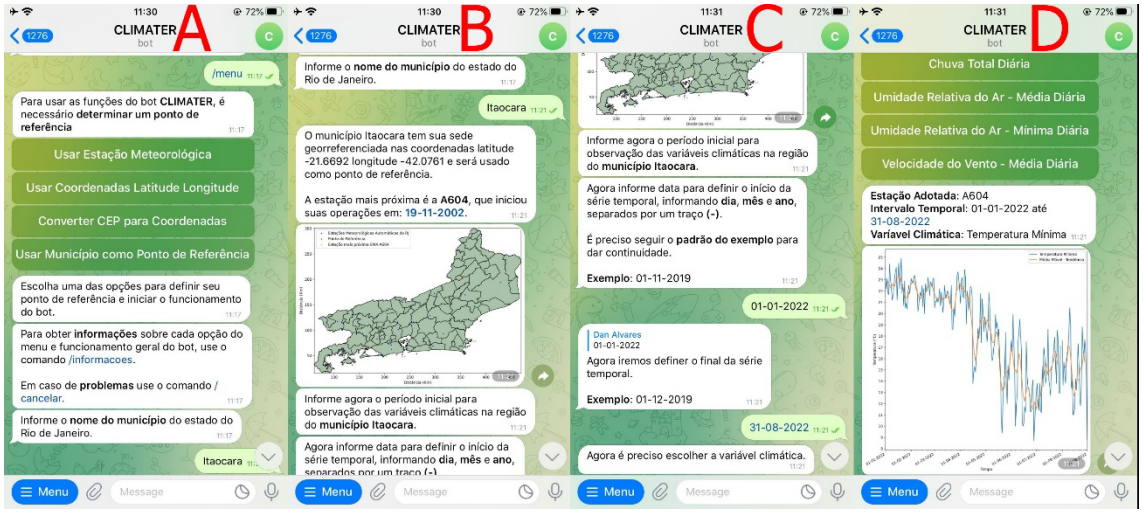


Fonte: Autor (2022)

5.6.4 MUNICÍPIO COMO REFERÊNCIA

Este é o último método disponível no bot CLIMATER, sendo o segundo método alternativo para se definir um ponto de referência. Seu funcionamento se assemelha ao funcionamento da função descrita para a escolha de estação meteorológica como referência, pois para cada local existe seu par de latitude e longitude, nesse caso as sedes dos municípios são georreferenciadas. Logo, após escolher “Usar Município como Ponto de Referência”, será necessário informar o nome de um dos 92 municípios do Rio de Janeiro.

Figura 14: Escolha de Município como referência



Fonte: Autor (2022)

Após informar o nome do município válido (figura 14 tela A), teremos a resposta com o mapa localizando o ponto de interesse no estado do Rio de Janeiro (figura 14 tela B), depois seguindo o fluxograma para as demais etapas, informando datas da série temporal (figura 14 tela C) e variável climática de interesse (figura 14 tela D).

6 RESULTADOS E DISCUSSÕES

Esta parte da obra realizará discussões sobre a aplicação proposta em contraste com a opção de criação de um aplicativo nativo para celulares, os pontos que definiram a escolha entre criar a aplicação no formato de *bot* inserido no WhatsApp ou Telegram, e por fim, comparações destacando as diferenças e vantagens da aplicação proposta em relação ao sistema vigente do INMET.

6.1 BOT OU APLICATIVO – QUAL A MELHOR OPÇÃO?

Inicialmente, quando analisado a possibilidade de criar uma solução para o problema abordado neste trabalho, é natural pensar na criação de aplicativos para celulares. Entretanto, existem diversos fatores que devem ser atendidos para disponibilizar aplicativos nas lojas oficiais das principais marcas de celulares, como Google Play Store para dispositivos Android, ou App Store para dispositivos iOS.

6.1.1 BUROCRACIA

Para as plataformas citadas, existem diretrizes que norteiam como publicar e o que pode ou não constar no seu aplicativo. Os aplicativos na Play Store levam algumas horas para estarem disponíveis para o usuário final se atenderem todas as diretrizes definidas pelo Google.

Já na App Store esse processo é mais demorado e complicado, devendo estar de acordo com as diretrizes da organização. O aplicativo então passa por uma análise de cumprimento de diretrizes, levando aproximadamente três dias para ser aprovado, ocorrem casos dessa avaliação durar duas semanas.

Já o Telegram não estabelece diretrizes para lançamento de *bots*. Os mesmos podem ser criados bastando fazer o uso regular de sua API para funcionar plenamente.

6.1.2 CUSTOS PARA PUBLICAR APLICATIVOS

A Apple cobra 99 dólares por ano para possuir uma licença de desenvolvedor Apple, a qual possibilitará lançar aplicativos em sua plataforma App Store para download de novos usuários.

Google apresenta um plano mais barato para a publicação de aplicativos, cobrando apenas 25 dólares para a publicação do seu primeiro aplicativo, todos os demais não serão cobrados.

Já o lançamento de um *bot* do Telegram é livre de gastos. O único gasto que possa existir é o custeio de servidores dedicados e tais serviços estão disponíveis por preços acessíveis, podendo ainda rodar em máquinas domésticas se a demanda não for intensa.

6.1.3 COMPLEXIDADE DE DESENVOLVIMENTO

Apesar do Python também possuir ferramentas para o desenvolvimento de aplicativos de celular para iOS e Android, existe uma diversidade de implicações técnicas que tornam o processo demasiadamente complexo, tendo em vista os objetivos deste projeto, logo este complexo processo se torna desfavorável no contexto das soluções apresentadas para o CLIMATER *bot*.

6.2 WHATSAPP OU TELEGRAM

Como o custo em si foi elencado como uma barreira para a opção de desenvolvimento de aplicativos, será mantida neste tópico, pois existe a cobrança para uso da API do WhatsApp, ocorrendo após a ocorrência de mil conversas dentro do mês. Além do custo, tal interface para criação do *bot* dentro do WhatsApp está disponível apenas para o serviço da linha Business, não havendo uma API pública e gratuita como disponibilizada para o Telegram. Dessa forma o Telegram consagra-se como a melhor opção para concepção deste MVP.

6.3 COMPARATIVO COM INTERFACES DO INMET

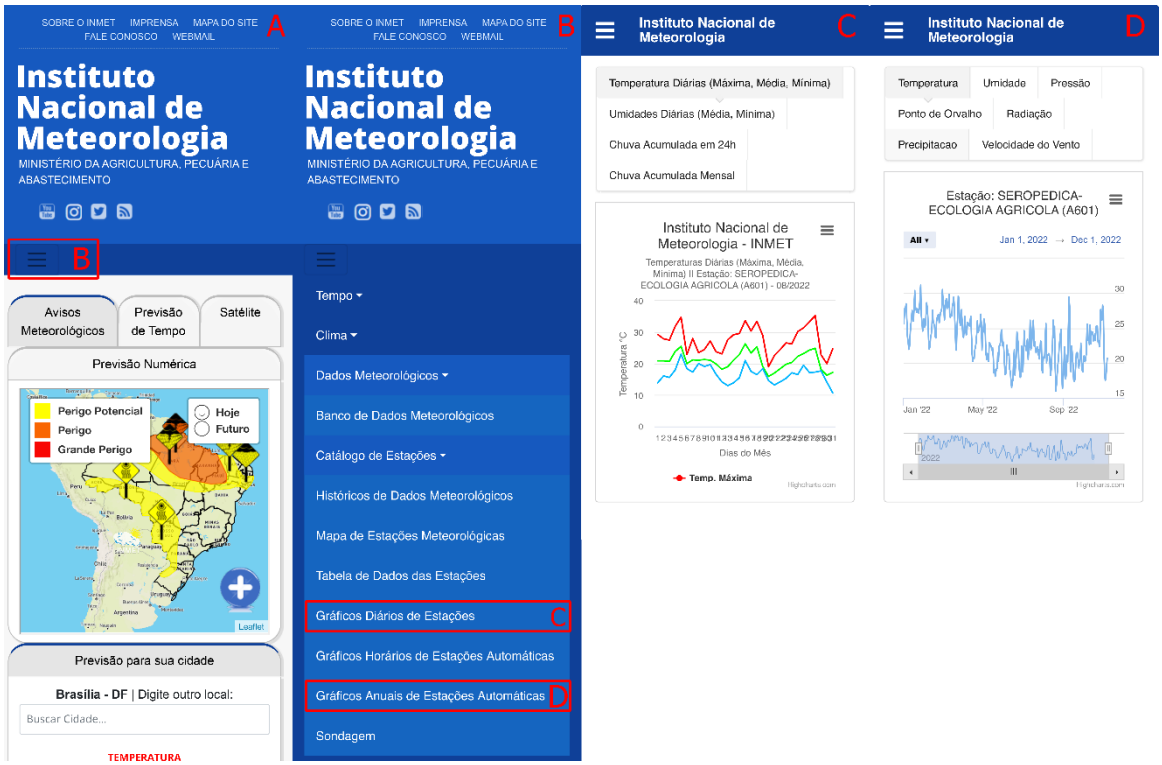
A interface do site do Inmet possui excesso de informações, poluindo sua comunicação, ainda agravada pela falta de contraste de cores na posição onde o menu foi inserido (LEONARDI, 2015) no *site* do INMET (figura 15 tela A). Essa característica atrapalha o foco do usuário, o acesso ao seu menu (figura 15 tela

A, destaque B), tornando dificultosa a obtenção das informações climáticas como apresentadas neste MVP.

Além disso, os dados mantidos pelo INMET em seu *site* são ofertados apenas na frequência de dias dentro de um mês específico, ao acessar os Gráficos Diários de Estações (figura 15 tela C). Já em Gráficos Anuais de Estações Automáticas (figura 15 tela D) é possível alterar a frequência desses dados e janela temporal visualizada, entretanto existem falhas para esta opção, observadas no seu uso no celular e no computador.

Nessas opções, é preciso selecionar o estado onde o ponto de interesse está localizado, para então escolher a estação meteorológica, seja automática ou convencional, por seu código e local onde está instalada, por exemplo, a EMA A601, está localizada no bairro Ecologia Agrícola em Seropédica, aparecendo na lista como SEROPEDICA-ECOLOGIA AGRICOLA (A601), sendo selecionado no menu no canto esquerdo, destacado em contraste com o fundo do cabeçalho da página (figura 15, telas C e D).

Figura 15: Obtenção de Visualizações de Séries Temporais no INMET no Safari (iPhone 8)



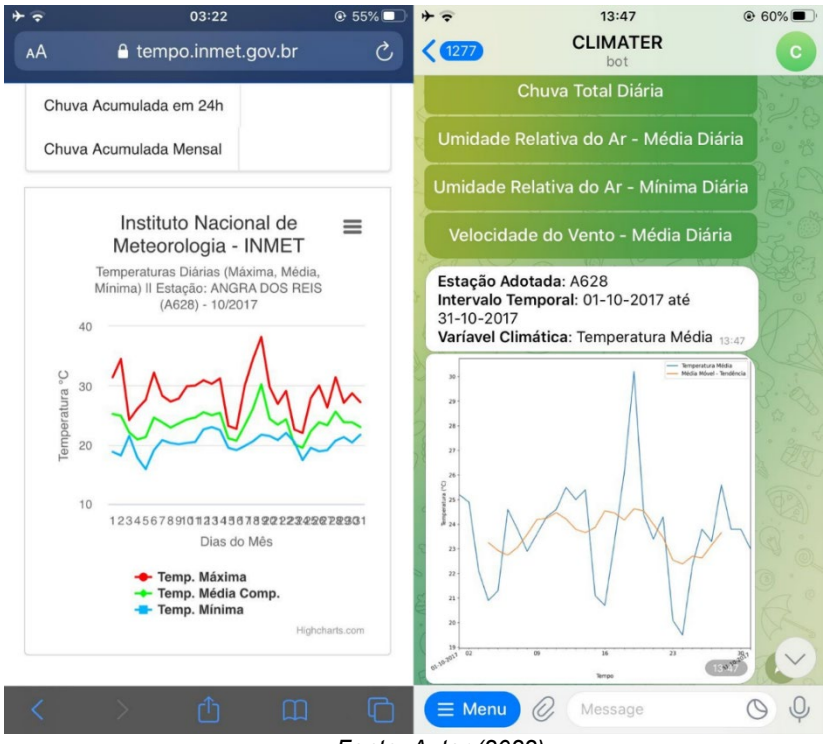
Fonte: Autor (2022)

Tais opções foram desenvolvidas no *bot* CLIMATER, tendo como vantagem a rápida requisição e plotagem de dados na tela, assim como a

resolução de problemas de funcionamento, de ordem de usabilidade. Antes de pautar os problemas resolvidos, é necessário comparar as diferenças iniciais entre a versão do INMET e a versão do *bot* CLIMATER, com o comparativo da visualização de dados de temperatura média da estação meteorológica automática A628 (figura 16), localizada em Angra dos Reis, contemplando o mês de outubro de 2017.

Tratando sobre os problemas encontrados, o primeiro é o truncamento das datas no eixo X do gráfico gerado, que pode ser observado na figura 16, sendo resolvido apenas se a imagem gerada for baixada (figura 17), apresentando as datas espaçadas adequadamente. Agora observando a geração de gráficos anuais, não é possível a geração de gráficos que contemplem qualquer ano inteiro, de 1º de janeiro até 31 de dezembro, pois a interface do site do INMET impede a definição do dia 31 para o mês de dezembro, testado nas versões para navegadores comuns no PC e na versão de celular no navegador Safari.

Figura 16: Dados da estação A628 do INMET e CLIMATER

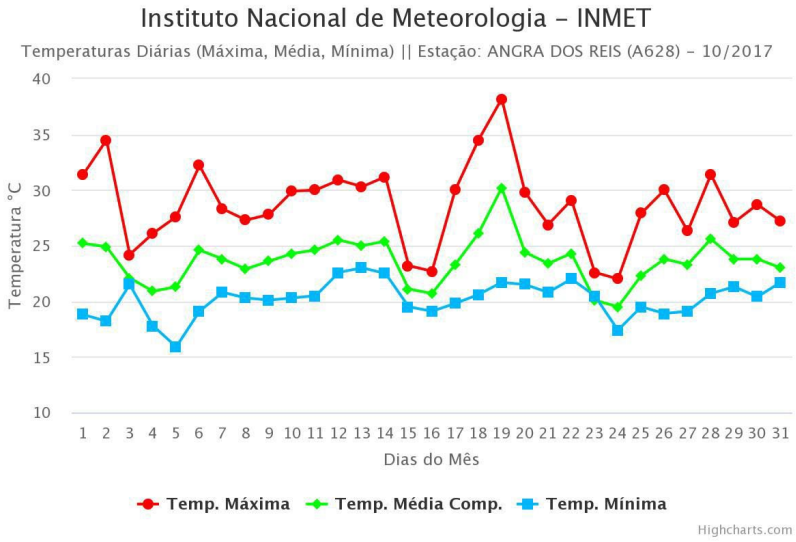


Fonte: Autor (2022)

O gráfico gerado no INMET apresenta as variações de Temperatura Máxima, Temperatura Média e Temperatura Mínima, no mesmo gráfico, causando poluição na visualização. O ideal seria simplificar o gráfico, reduzindo

o número de elementos plotados na imagem, mesmo trabalhando com cores altamente contrastantes. No modelo desenvolvido para o CLIMATER temos a plotagem de apenas uma variante, neste caso a Temperatura Média, seguido de sua Média Móvel para representar a tendência.

Figura 17: Detalhamento de série temporal do INMET



Fonte: Autor (2022)

Entretanto notam-se mais problemas na interface para a visualização de gráficos em Gráficos Anuais de Estações Automáticas, sendo impossível no sistema do INMET requisitar dados de diferentes anos no mesmo gráfico, além da interface de janela móvel entrar em conflito e atrapalhar a manipulação do gráfico em sua interface móvel.

Foram encontrados outros problemas de ordem técnica, de validação e usabilidade, no qual possibilita o usuário requisitar a plotagem de dados para datas que a estação meteorológica sequer existia em operação, retornando a um gráfico vazio (figura 18), desperdiçando tempo e recursos. O CLIMATER introduz etapas de validação para as datas informadas pelo usuário e caso a data seja anterior ao início de operação da estação meteorológica consultada, a data definida como inicial será exatamente sua data de instalação, possibilitando a geração de gráficos, já a data final não deverá ser posterior ao último registro salvo na base de dados, no caso do projeto a data é 31 de agosto de 2022.

Figura 18: Gráfico inválido para EMA A628 no ano de 2016

Fonte: Autor (2022)

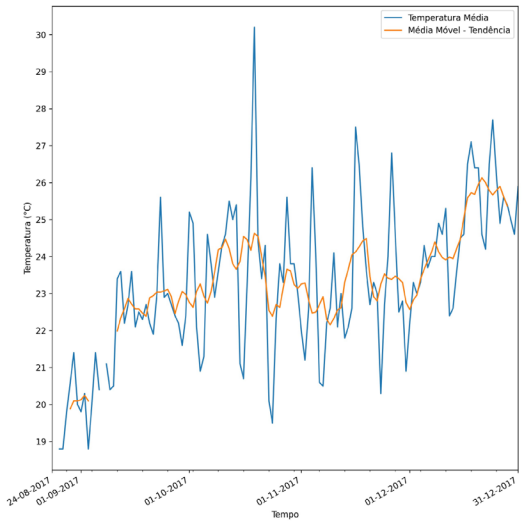
A estação A628, de Angra dos Reis, iniciou operação na data 24/08/2017 (INMET, 2022b), dessa forma só teremos gráficos válidos iniciando nesta data (figura 19).

Figura 19:Gráfico para EMA A628 no ano de 2017

Fonte: Autor (2022)

Abaixo temos exemplo de gráfico plotado com data anterior ao início operacional da estação meteorológica A628, de Angra dos Reis, no *bot* CLIMATER, forçando seu sistema adotar a data inicial de operação.

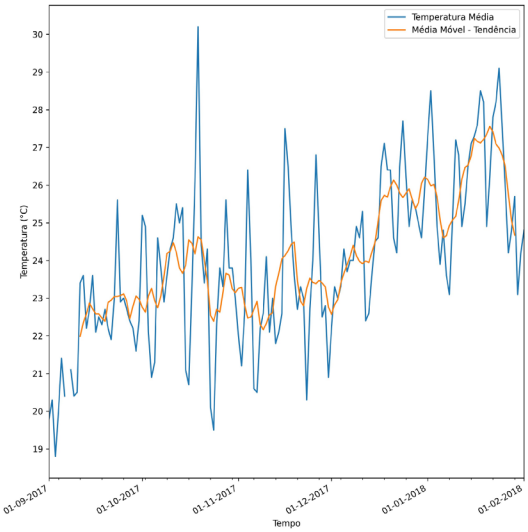
Figura 20: Validação e correção para data antes da geração de gráfico



Fonte: Autor (2022)

Outro ponto relevante é a flexibilidade para manipulação de datas dentro do sistema desenhado para o CLIMATER, possibilitando a geração de gráficos que compreendam anos sequenciais, uma ideia que não pode ser executada dentro do sistema INMET.

Figura 21: Gráfico da estação A628 compreendo datas de 2017-2018



Fonte: Autor (2022)

Havendo explicitado os pontos acima, evidencia-se a flexibilidade na manipulação e plotagem de dados do CLIMATER *bot*, compreendendo uma vantagem de usabilidade em comparação com o sistema adaptado para dispositivos móveis mantido pelo INMET em seu *site* oficial.

Outro ponto importante para a adoção do Telegram como plataforma de desenvolvimento deste projeto é a consistência de interface, se mantendo igual independente do sistema operacional usado nos celulares, seja iOS ou Android, além de dispor de funções nativas de acessibilidade dos celulares. Esse ponto já não é consistente para acesso através de computadores e navegadores comuns, já que a renderização de *sites* varia de acordo com o motor de cada navegador, podendo evidenciar problemas de design, inutilizar certas funções, acesso e leitura de dados, entre outras questões.

7 CONSIDERAÇÕES FINAIS

Conclui-se que foi demonstrada a implantação de um *bot* para dispor de dados climáticos através do Telegram, constituindo de alternativa simples, ágil e com ampla oferta de opções para obtenção de dados climáticos quando comparado diretamente com a alternativa ofertada pela entidade responsável por tais dados, o INMET.

Entende-se que o sistema proposto CLIMATER ainda carece de aprimoramentos, no que tange a qualidade da plotagem dos dados climáticos, principalmente por não ter sido trabalhado os princípios de Gestalt como literaturas especializadas em conceitos de visualização de dados recomendam, tornando o entendimento de gráficos mais fácil e intuito. Além disso, ainda tratando do aspecto ligado ao desenvolvimento, é preciso citar que diversas funções ainda podem ser simplificadas e aprimoradas, assim como refatoração de validações condicionais.

É importante reforçar que, por se tratar de um MVP, delimitou-se a operacionalidade deste projeto para os municípios e limites territoriais do estado do Rio de Janeiro, entretanto é evidente a possibilidade de existir estações meteorológicas nos municípios de estados limítrofes que apresentem dados climáticos que representem melhor as condições climáticas locais do ponto de

interesse informado pelo usuário, como exemplo, dada diversas condições e especificidades de relevo, entre outros aspectos.

Este MVP valida com clareza os objetivos específicos elencados, havendo espaço para sua ampliação para atender o restante do território nacional e demais estações meteorológicas, assim como incorporar no seu funcionamento a oferta de dados das estações convencionais.

8 REFERÊNCIAS

ACKERMAN, S.; KNOX, J. **Meteorology**. [s.l.] Jones & Bartlett Publishers, 2013.

AHMAD, L. et al. **Experimental Agrometeorology: A Practical Manual**. [s.l.] Springer International Publishing, 2017.

ALENCAR, J. R. DE et al. Avaliação dos impactos do uso do Sistema de Monitoramento Agrometeorológico (Agritempo). **Revista de Política Agrícola**, v. 25, n. 1, p. 5–19, 2016.

ALVIM, C. E. **Quase 30% das estações meteorológicas do Brasil estão sem manutenção | Minas Gerais | G1**. Disponível em: <https://g1.globo.com/mg/minas-gerais/noticia/2021/07/19/quase-30percent-das-estacoes-meteorologicas-do-brasil-estao-sem-manutencao.ghtml>. Acesso em: 11 set. 2022.

ARANHA, F. Atlas dos setores postais: uma nova geografia a serviço da empresa. **Revista de Administração de Empresas**, v. 37, n. 3, p. 20–27, set. 1997.

ARAÚJO, R. M. et al. Condições agrometeorológicas para perfilhamento máximo da cultura de cana-de-açúcar em dois ambientes distintos de produção. **Agrometeoros**, v. 25, n. 1, p. 257–264, 29 nov. 2018.

ARI, N.; USTAZHANOV, M. Matplotlib in python. **Proceedings of the 11th International Conference on Electronics, Computer and Computation, ICECCO 2014**, 23 dez. 2014.

BARBOSA, U. C. et al. iGrãos: development of chatbot in social networks for soy classification for soybean farmers. **Research, Society and Development**, v. 9, n. 10, p. e2689108558–e2689108558, 26 set. 2020.

BARBOZA, C. HISTÓRIA DA METEOROLOGIA NO BRASIL (1887-1917) A METEOROLOGIA TELEGRÁFICA. **Congresso Brasileiro de Meteorologia**, 2006.

BARNETT, A. G.; DOBSON, A. J. Decomposing Time Series. p. 93–128,

2010.

BARRETT, P. et al. matplotlib-A Portable Python Plotting Package. **ASP Conference Series**, v. 347, 2005.

BERKELEY. **11 Most In-Demand Programming Languages in 2022 - Berkeley Boot Camps**. Disponível em: <https://bootcamp.berkeley.edu/blog/most-in-demand-programming-languages/>. Acesso em: 11 set. 2022.

BERUSKI, G. C.; PEREIRA, A. B.; SENTELHAS, P. C. Desempenho de diferentes modelos de estimativa da radiação solar global em ponta grossa, PR. **Revista Brasileira de Meteorologia**, v. 30, n. 2, p. 205–213, 2015.

BLAINSKI, É.; GARBOSSA, L. H. P.; ANTUNES, E. N. **Estações hidrometeorológicas automáticas: recomendações técnicas para instalaçãoEpagri**. Florianópolis: Epagri, 2012. Disponível em: <www.epagri.sc.gov.br>. Acesso em: 11 jun. 2021.

BOLFE, E. L. Geotecnologias aplicadas à gestão de recursos naturais. **III Simpósio Regional de Geoprocessamento e Sensoriamento Remoto**, v. 3, 2006.

BROCK, F. V.; RICHARDSON, S. J. **Meteorological measurement systems**. [s.l.] Oxford University Press, 2001.

CAMBRIA, E.; WHITE, B. Jumping NLP curves: A review of natural language processing research. **IEEE Computational Intelligence Magazine**, v. 9, n. 2, p. 48–57, 2014.

CAROLI, P. **Direto ao ponto: criando produtos de forma enxuta.** [s.l.] Casa do Código, 2015.

CHACON, G. **Equipos y sensores de la red de agrometeorologia INIA.** [s.l: s.n.].

COSTA, H. C. et al. Espacialização e Sazonalidade da Precipitação Pluviométrica do Estado de Goiás e Distrito Federal (Seasonality and Spatial Distribution of Rainfall in the State of Goiás and Federal District). **Revista**

Brasileira de Geografia Física, v. 5, n. 1, p. 87, 17 maio 2012.

COSTA, M. DA S. et al. Tendências observadas em extremos de precipitação sobre a região Semiárida do Nordeste do Brasil (Trends observed in precipitation extremes over the semiarid region of Notheast Brazil). **Revista Brasileira de Geografia Física**, v. 8, n. 5, p. 1321-1334–1334, 2016.

CUNHA, K. C. B. DA; ROCHA, R. V. A. Revista Eletrônica Competências Digitais para Agricultura Familiar AUTOMAÇÃO NO PROCESSO DE IRRIGAÇÃO NA AGRICULTURA FAMILIAR COM PLATAFORMA ARDUÍNO AUTOMATION IN IRRIGATION PROCESS IN FAMILY FARM WITH PLATFORM ARDUÍNO. **owl.tupa.unesp.br**, n. 2, p. 62–74, 2016.

DE ARAUJO ELIAS, A. A. et al. ARDWEATHER: UMA ESTAÇÃO METEOROLÓGICA BASEADA NO ARDUINO E EM WEB SERVICES RESTFUL. 2014.

DE OLIVEIRA, E. S. et al. **Medição de Pressão**. [s.l: s.n.].

DE OLIVEIRA, J. C.; SANTOS, D. H.; NETO, M. P. Chatting with Arduino platform through Telegram Bot. **Proceedings of the International Symposium on Consumer Electronics, ISCE**, p. 131–132, 23 dez. 2016.

DE SOUSA, A. M. L. et al. Variabilidade espaço-temporal da precipitação na Amazônia durante eventos enos. 2015.

DRUCKER, D. P. et al. GeoInfo - Infraestrutura de Dados Espaciais Abertos para a Pesquisa Agropecuária GeoInfo-Open Spatial Data Infrastructure for Agricultural Research GeoInfo-Infraestructura de Datos Espaciales Abiertos para la Investigación Agropecuaria. **RECIIS - Revista Eletrônica de Comunicação, Informação e Inovação em Saúde**, v. 11, p. 1–17, nov. 2017.

ELLIOTT, R. Geographic information systems (GIS) and libraries: Concepts, services and resources. **Library Hi Tech News**, v. 31, n. 8, p. 8–11, 30 set. 2014.

ESRI. **O que é GIS? | Tecnologia de Mapeamento de Sistema de Informação Geográfica**. Disponível em: <https://www.esri.com/pt-br/what-is-

gis/overview>. Acesso em: 10 set. 2022.

FERREIRA, J. L.; RUFFONI, J.; CARVALHO, A. M. Dinâmica da difusão de inovações no contexto brasileiro. **Rev. Bras. Inov**, v. 17, n. 1, p. 175–200, 2018.

FIRPO, M. Â. F. **Influências remotas das TSM dos Oceanos Pacífico e Atlântico e da oscilação Antártica na variabilidade climática interanual no Rio Grande do Sul e suas inter-relações**. [s.l: s.n.].

FISHER, D. K.; GOULD, P. J. Open-Source Hardware Is a Low-Cost Alternative for Scientific Instrumentation and Research. **Modern Instrumentation**, v. 1, p. 8–20, 2012.

GADELHA, I. B. L. **O uso de chatbots no atendimento de clientes de revenda por catálogo**. [s.l.] Universidade Federal do Pará, 28 maio 2019.

GEEKSFORGEEKS. **Top 10 Programming Languages to Learn in 2022**. Disponível em: <https://www.geeksforgeeks.org/top-10-programming-languages-to-learn-in-2022/>. Acesso em: 11 set. 2022.

GÓMEZ, J. M. R. et al. A irradiância solar: conceitos básicos. **Revista Brasileira de Ensino de Física**, v. 40, n. 3, 26 mar. 2018.

HAMADA, E.; DO VALLE GONÇALVES, R. R. **Introdução ao geoprocessamento: princípios básicos e aplicação.** Jaguariúna: Embrapa Meio Ambiente, 2007.

HAN, X. et al. Technology Adoption in Dependency Networks: A Study of the Python Programming Language. 2021.

HANSUN, S. A new approach of moving average method in time series analysis. **2013 International Conference on New Media Studies, CoNMedia 2013**, 2013.

HONG, M.; KIM, J.; JEONG, S. Rainfall intensity-duration thresholds for landslide prediction in South Korea by considering the effects of antecedent rainfall. **Landslides**, v. 15, n. 3, p. 523–534, 1 mar. 2018.

IBGE. **Acesso e uso de dados geoespaciais**. Rio de Janeiro: IBGE, 2019.

INMET. **CATÁLOGO DE ESTAÇÕES CONVENCIONAIS**. Disponível em: <https://portal.inmet.gov.br/paginas/catalogoman>. Acesso em: 11 set. 2022a.

INMET. **CATÁLOGO DE ESTAÇÕES AUTOMÁTICAS**. Disponível em: <https://portal.inmet.gov.br/paginas/catalogoaut>. Acesso em: 11 set. 2022b.

JANSSEN, V. Understanding Coordinate Reference Systems, Datums and Transformations. v. 5, n. 4, p. 41–53, 2009.

JIMENEZ-CARBALLO, C. A. **TEMPERATURA Y EXPANSIÓN TÉRMICA**. [s.l: s.n.].

KATARINE, A. et al. Desenvolvimento de um Chatbot para Compartilhamento e Disseminação de Informações do Tempo. **Anais do Workshop de Computação Aplicada à Gestão do Meio Ambiente e Recursos Naturais (WCAMA)**, p. 125–134, 4 jul. 2019.

KAUR SWARAN SINGH, C. et al. RETHINKING ENGLISH LANGUAGE TEACHING THROUGH TELEGRAM, WHATSAPP, GOOGLE CLASSROOM AND ZOOM. **Systematic Reviews in Pharmacy**, v. 11, n. 11, p. 45–54, 2020.

KAZANDJIEV, V. et al. **Contemporary agrometeorological research - Opportunity for modern agriculture in conditions of climatic anomalies and changes**. AIP Conference Proceedings. **Anais**...American Institute of Physics Inc., 26 fev. 2019

KITAMURA, P. C. Agricultura sustentável no Brasil: avanços e perspectivas. 2018.

KLAUS, P.; ZAICHKOWSKY, J. AI voice bots: a services marketing research agenda. **Journal of Services Marketing**, v. 34, n. 3, p. 389–398, 19 jun. 2020.

LAGOUVARDOS, K. et al. The automatic weather stations NOANN network of the National Observatory of Athens: operation and database. **Geoscience Data Journal**, v. 4, n. 1, p. 4–16, 1 jun. 2017.

LEBEUF, C. R. A taxonomy of software bots: towards a deeper understanding of software bot characteristics. 2018.

LEONARDI, J. J. **Importância da engenharia de usabilidade na computação**. [s.l.] Faculdade de Tecnologia de Americana, 8 dez. 2015.

LOPES, G.; DELBEM, A.; SOUSA, J. Introdução à Análise de Dados Geoespaciais com Python. **Minicursos do XIV Encontro Unificado de Computação do Piauí (ENUCOMPI) e XI Simpósio de Sistemas de Informação (SINFO)**, p. 82–106, 23 nov. 2021.

LOPEZ, J. C. B.; VILLARUZ, H. M. **Low-cost weather monitoring system with online logging and data visualization**. 8th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management, HNICEM 2015. **Anais**...Institute of Electrical and Electronics Engineers Inc., 25 jan. 2016

LUCAS, E. W. M. et al. COMPARATIVE ANALYSIS OF OBSERVED METEOROLOGICAL DATA IN THE CONVENTIONAL AND SURFACE AUTOMATIC STATION AT BRAZILIAN NATIONAL INSTITUTE OF METEOROLOGY. **INMET**, v. 174, 2010.

LUCCHESI, A. A. **Fatores da produção vegetal**. [s.l: s.n.].

MACHADO, B. et al. Aplicação das Técnicas de Mineração de Dados como Complemento às Previsões Estocásticas Univariadas de Vazão Natural: Estudo de Caso para a Bacia do Rio Iguaçu. **Revista Brasileira de Recursos Hídricos**, v. 12, n. 3, p. 83–92, 2007.

MARTORANO, L. G. et al. Climate conditions in the eastern amazon: Rainfall variability in Belem and indicative of soil water deficit. **African Journal of Agricultural Research**, v. 12, n. 21, p. 1801–1810, 25 maio 2017.

MCKINNEY, W. Pandas: a Foundational Python Library for Data Analysis and Statistics. v. 14, n. 9, p. 1–99, 2011.

MCNEFF, J. G. The global positioning system. **IEEE Transactions on Microwave Theory and Techniques**, v. 50, n. 3, p. 645–652, mar. 2002.

MEDEIROS, R. M. DE et al. Characterization of the Average Wind Speed in the Municipaly of Teresina-PI. **Journal of Experimental Agriculture International**, v. 42, n. 7, p. 94–101, 28 ago. 2020.

MEIRELLES, F. S. Pesquisa do Uso da TI-Tecnologia de Informação nas Empresas, FGVcia. 2022.

MENDES REIS, M.; GONÇALVES LOPES, E. M.; OLIVEIRA, F. G. **COMPARAÇÃO DE DADOS METEOROLOGICOS OBTIDOS POR ESTAÇÕES METEOROLOGICAS CONVENCIONAL E AUTOMÁTICA**. [s.l: s.n.].

MESTRE, G. et al. An intelligent weather station. **Sensors (Switzerland)**, v. 15, n. 12, p. 31005–31022, 10 dez. 2015.

MINA, U. et al. Response of wheat and chickpea cultivars to reduced levels of solar irradiance. **Journal of Agrometeorology**, v. 17, n. 2, p. 165, 2015.

MONTEIRO, J. E. **Agrometeorologia dos Cultivos O fator meteorológico na produção agrícola**. [s.l.] INMET, 2009.

NAKAMURA, E. T. **Infraestrutura de Dados Espaciais em Unidades de Conservação: uma proposta para disseminação da informação geográfica do Parque Estadual de Intervales-SP** . São Paulo: [s.n.].

NEVES, G. A. R. et al. Desenvolvimento e Calibração de um Termohigrômetro para uso em Pesquisas de Micrometeorologia, Agrometeorologia e Clima (Development and calibration of a Thermohygrometer for use in Research Micrometeorology, Agrometeorology and Climatological). **Revista Brasileira de Geografia Física**, v. 8, n. 1, p. 136, 23 set. 2015.

OLDANI, J. **La meteorología**. [s.l.] Parkstone International, 2020.

OLIVEIRA, G. F. DE. **Sistema de aquisição de dados agrometeorológicos utilizando estações de sensores sem fio**. [s.l.] Universidade de Passo Fundo, 14 dez. 2018. Disponível em: <http://repositorio.upf.br/handle/riupf/1700>. Acesso em: 14 maio. 2021.

OLIVEIRA JÚNIOR, J. F. et al. Análise da Precipitação e sua Relação com

Sistemas Meteorológicos em Seropédica, Rio de Janeiro. **Floresta e Ambiente**, v. 21, p. 140–149, 2014.

PERAZZI, P. R. et al. O Tradicional ou o Moderno? Uma Visão da Informação da Rede de Estações Meteorológicas Brasileiras The Traditional or the Modern? A Vision of the Information from the Brazilian Weather Stations Network. 2021.

PEREIRA, A. R.; ANGELOCCI, L. R.; SENTELHAS, P. C. **Agrometeorologia: Fundamentos e Aplicações Práticas**. [s.l: s.n.].

PEREIRA, G. et al. **Avaliação dos Dados de Precipitação Estimados pelo Satélite TRMM para o Brasil**. [s.l: s.n.].

PESENTE, G. M. et al. UMA ABORDAGEM DE ENSINO DE PROGRAMAÇÃO DE COMPUTADORES UTILIZANDO SCRATCH E PYTHON. 2016.

PICINI, A. G. et al. Desenvolvimento e teste de modelos agrometeorológicos para a estimativa de produtividade do cafeeiro. **Bragantia**, v. 58, p. 157–170, 1999.

PINHEIRO DE ASSIS, J. et al. Distribuições de probabilidade para séries históricas mensais de pressão atmosférica no município de Mossoró-RN Probability distributions for historic series of monthly atmospheric pressure in Mossoró-RN. **Revista Verde de Agroecologia e Desenvolvimento Sustentável, ISSN-e 1981-8203, Vol. 11, Nº. 3, 2016, págs. 135-142**, v. 11, n. 3, p. 135–142, 2016.

PYTHON. **What is Python?** Disponível em: <https://docs.python.org/3/faq/general.html#what-is-python>. Acesso em: 11 set. 2022.

RANDLES, B. M. et al. Using the Jupyter Notebook as a Tool for Open Science: An Empirical Study. **Proceedings of the ACM/IEEE Joint Conference on Digital Libraries**, 25 jul. 2017.

RED HAT. **O que é IDE? Ambiente de Desenvolvimento Integrado**.

Disponível em: <https://www.redhat.com/pt-br/topics/middleware/what-is-ide>. Acesso em: 26 out. 2022.

REIS, L. S. et al. Componentes da radiação solar em cultivo de tomate sob condições de ambiente protegido. **Revista Brasileira de Engenharia Agricola e Ambiental**, v. 16, n. 7, p. 739–744, 2012.

RIBEIRO, D. Barómetro. 2014.

ROBERTSON, D. E.; SHRESTHA, D. L.; WANG, Q. J. Post-processing rainfall forecasts from numerical weather prediction models for short-term streamflow forecasting. **Hydrology and Earth System Sciences**, v. 17, n. 9, p. 3587–3603, 2013.

SANTOS, I. M. et al. **VISÃO GERAL DA APLICABILIDADE DE REDES DE SENSORES SEM FIO NO MONITORAMENTO AGRÍCOLA NO ESTADO DE MATO GROSSO**. [s.l: s.n.].

SEEMANN, J. et al. **Agrometeorology**. [s.l.] Springer Science & Business Media, 1979.

SETIAJI, H.; PAPUTUNGAN, I. V. Design of Telegram Bots for Campus Information Sharing. 2018.

SILVA-FUZZO, D. F.; ROCHA, J. V. VALIDAÇÃO DOS DADOS DE PRECIPITAÇÃO ESTIMADOS PELO TRMM, PARA O ESTADO DO PARANÁ, E SUA CONTRIBUIÇÃO AO MONITORAMENTO AGROMETEOROLÓGICO | Silva-Fuzzo | Formação (Online). 2016.

SILVA, M. S. DA, 1980-. **MARCEL SALVIONI DA SILVA REDE DE SENSORES SEM FIO DE BAIXO CUSTO PARA MONITORAMENTO AMBIENTAL**. [s.l.] [s.n.], 2013. Disponível em: <http://repositorio.unicamp.br/jspui/handle/REPOSIP/259030>. Acesso em: 19 nov. 2020.

SILVA, R. B. D. et al. Estações meteorológicas de código aberto: Um projeto de pesquisa e desenvolvimento tecnológico. **Revista brasileira de ensino de Física**, v. 37, 2015.

STIEGLITZ, S. et al. Do Social Bots Dream of Electric Sheep? A Categorisation of Social Media Bot Accounts. **Proceedings of the 28th Australasian Conference on Information Systems, ACIS 2017**, 11 out. 2017.

SUTIKNO, T. et al. WhatsApp, Viber and Telegram: which is the Best for Instant Messaging? **International Journal of Electrical and Computer Engineering (IJECE)**, v. 6, n. 3, p. 909–914, 2016.

TENZIN, S. et al. **Low cost weather station for climate-smart agriculture**. 2017 9th International Conference on Knowledge and Smart Technology: Crunching Information of Everything, KST 2017. **Anais**...Institute of Electrical and Electronics Engineers Inc., 23 mar. 2017

TERSI, M. C. P. et al. Implementação de um atendente virtual para uma estação hidrometeorológica utilizando AIML. **Revista Internacional de Tecnología, Ciencia y Sociedad**, v. 5, n. 1, 2016.

TSVETKOVA, M. et al. Even good bots fight: The case of Wikipedia. **PLOS ONE**, v. 12, n. 2, p. e0171774, 1 fev. 2017.

TWARDOSZ, R.; CEBULSKA, M.; WALANUS, A. Anomalously heavy monthly and seasonal precipitation in the Polish Carpathian Mountains and their foreland during the years 1881–2010. **Theoretical and Applied Climatology**, v. 126, n. 1–2, p. 323–337, 1 out. 2016.

VICENTE, M.; RODRIGUES, N. **Delegação de pesquisadores visita Embrapa**. Disponível em: <https://www.embrapa.br/busca-de-noticias?p_p_id=buscanoticia_WAR_pcebusca6_1portlet&p_p_lifecycle=0&p_p_state=pop_up&p_p_mode=view&p_p_col_id=column-1&p_p_col_count=1&_buscanoticia_WAR_pcebusca6_1portlet_groupId=1355145&_buscanoticia_WAR_pcebusca6_1portlet_articleId=2383371&_buscanoticia_WAR_pcebusca6_1portlet_viewMode=print>. Acesso em: 10 set. 2022.

VIEIRA, L.; PICULLI, F. J. **Meteorologia e Climatologia Agrícola**, 2009.

WASKOM, M. L. seaborn: statistical data visualization. **Journal of Open Source Software**, v. 6, n. 60, p. 3021, 2021.

WEART, S. et al. Climate change 2014: impacts, adaptation, and vulnerability – IPCC WGII AR5 summary for policymakers. **Proceedings of the National Academy of Sciences of the United States of America**, v. 111, n. SUPPL. 2, p. 9340–9345, 2014.

WEBER, E. et al. **Qualidade de Dados Geoespaciais**. Porto Alegre: [s.n.].

YNOUE, R. Y. et al. **Meteorologia: noções básicas**. [s.l: s.n.].

ZANOTTA, D. C.; CAPPELLETTO, E.; MATSUOKA, M. T. O GPS: unindo ciência e tecnologia em aulas de física. **Revista Brasileira de Ensino de Física**, v. 33, n. 2, p. 2313, abr. 2011.

ANEXO 1 – CÓDIGO DO BOT CLIMATER

```
from aiogram import Bot, Dispatcher, executor, types

from aiogram.types import InlineKeyboardMarkup, InlineKeyboardButton, ReplyKeyboardMarkup, KeyboardButton, ReplyKeyboardRemove, CallbackQuery

from aiogram.types import Message

from aiogram.dispatcher import FSMContext

from aiogram.dispatcher.filters.state import State, StatesGroup

from aiogram.contrib.fsm_storage.memory import MemoryStorage

import aiogram.utils.markdown as md

from telegram import ParseMode

from aiogram.dispatcher.filters import Text

import pandas as pd

import numpy as np

import matplotlib

matplotlib.use('agg')

import matplotlib.pyplot as plt

import matplotlib.dates as mdates

import warnings

warnings.simplefilter(action='ignore', category=FutureWarning)


import config # chaves particulares

import utils  # funções geradas nos notebooks para obtenção e manipulação de dados geoespaciais e climáticos
```

```
storage = MemoryStorage()

bot = Bot(token = config.token_bot)

dp = Dispatcher(bot, storage=storage)


class EMA_sessao(StatesGroup):
    ema = State()
    ema_inicio_periodo = State()
    ema_fim_periodo = State()
    ema_var_climatica = State()


class COORD_sessao(StatesGroup):
    coordenadas = State()
    coord_ema = State()
    coord_inicio_periodo = State()
    coord_fim_periodo = State()
    coord_var_climatica = State()


class MUNICIPIO_sessao(StatesGroup):
    municipio = State()
    municipio_inicio_periodo = State()
    municipio_fim_periodo = State()
    municipio_var_climatica = State()
```

```
class CEP_sessao(StatesGroup):
    cep = State()
    cep_inicio_periodo = State()
    cep_fim_periodo = State()
    cep_var_climatica = State()

botao_ema = InlineKeyboardButton(text='Usar Estação Meteorológica', callback_data='informa_ema')
botao_latlong = InlineKeyboardButton(text='Usar Coordenadas Latitude Longitude', callback_data='informa_latlong')
botao_cep = InlineKeyboardButton(text='Converter CEP para Coordenadas', callback_data='informa_cep')
botao_municipio = InlineKeyboardButton(text='Usar Município como Ponto de Referência', callback_data='informa_municipio')

# Insere botões definidos no Menu Inicial do bot
menu_inicial = InlineKeyboardMarkup(resize_keyboard=True).row(botao_ema).row(botao_latlong).row(botao_cep).row(botao_municipio)

botao_temp_min = InlineKeyboardButton(text='Temperatura Mínima', callback_data='var_temp_min')
botao_temp_med = InlineKeyboardButton(text='Temperatura Média', callback_data='var_temp_med')
```

```
botao_temp_max = InlineKeyboardButton(text='Temperatura Máxima', callback_data='var_temp_max')

botao_chuva = InlineKeyboardButton(text='Chuva Total Diária', callback_data='var_chuva')

botao_umid_min = InlineKeyboardButton(text='Umidade Relativa do Ar - Mínima Diária', callback_data='var_umid_min')

botao_umid_med = InlineKeyboardButton(text='Umidade Relativa do Ar - Média Diária', callback_data='var_umid_med')

botao_vento = InlineKeyboardButton(text='Velocidade do Vento - Média Diária', callback_data='var_vento_med')


menu_vars_climaticas = InlineKeyboardMarkup(resize_keyboard=True).row(botao_temp_max).row(botao_temp_med).row(botao_temp_min).row(botao_chuva).row(botao_umid_med).row(botao_umid_min).row(botao_vento)


@dp.callback_query_handler(lambda c: c.data == 'menu_principal')

@dp.message_handler(state='*', commands=['start', 'iniciar', 'menu', 'menu_principal'])

@dp.message_handler(Text(equals=['start', 'iniciar', 'menu', 'menu_principal'], ignore_case=True), state='*')

async def inicio(msg: Message):

    await bot.send_message(msg.chat.id, utils.msg_inicial, reply_markup=menu_inicial, parse_mode='HTML')

    await bot.send_message(msg.chat.id, 'Escolha uma das opções para definir seu ponto de referência e iniciar o funcionamento do bot.', parse_mode='HTML')
```

```
    await bot.send_message(msg.chat.id, 'Para obter <b>informações</b> sobre cada opção do menu e funcionamento geral do bot, use o comando /informacoes.\n\nEm caso de <b>problemas</b> use o comando /cancelar.', parse_mode='HTML')


@dp.message_handler(commands=['informacoes'])
async def info_menu(msg: Message):
    await bot.send_message(msg.chat.id, utils.msg_inicial, parse_mode='HTML')
    await bot.send_message(msg.chat.id, '<b>1ª opção Usar Estação Meteorológica</b>:\nSerá necessário informar o <b>código</b> de uma estação meteorológica automática do Rio de Janeiro. Para mais informações e lista completa de estações, use o comando /lista_estacoes.\n\n<b>2ª opção Usar Coordenadas Latitude Longitude</b>:\nSerá necesário informar as <b>coordenadas geográficas</b> do ponto através da <b>latitude</b> e <b>longitude</b>, no formato <b>Grau-Minuto-Segundo</b> ou <b>Grau-Decimal</b>.\n\n<b>Importante</b>:\nCoordenadas que apontem pontos fora do Rio de Janeiro serão consideradas inválidas.', parse_mode='HTML')
    await bot.send_message(msg.chat.id, '<b>3ª opção Converter CEP para Coordenadas</b>:\nSerá necessário informar um <b>CEP</b> e essa informação será convertida numa coordenada geofráfica de referência.\n\n<b>Importante</b>:\nNão funciona para todos os CEPs, devido limitações de sistemas informativos, atendendo apenas a faixa de CEPs do Rio de Janeiro.\n\n<b>4ª opção Usar Município como Ponto de Referência</b>:\n\nSerá necessário informar o nome de um município do Rio de Janeiro. Com isso teremos a sede do município como ponto georreferenciado.', parse_mode='HTML')
    await bot.send_message(msg.chat.id, '<b>Informações adicionais</b>:\n\nApós escolher o método de referência, você precisará definir um período com duas datas, compondo o início e fim para a geração das visualizações climáticas.\n\nOs dados climáticos são provenientes da base de
```

dados do Inmet, contemplando as seguintes variáveis:\n - Temperatura Máxima\n - Temperatura Média\n - Temperatura Mínima\n - Chuva Total no Dia\n - Umidade Relativa do Ar Média Diária\n - Umidade Relativa do Ar Mínima Diária\n - Velocidade do Vento Nédia Diária\n\nOs dados apresentados pelo bot apresentam falhas provinientes da coleta de dados realizada pelo Inmet, resultado de falha e interrupção de funcionamento das estações automáticas.\nNos gráficos climáticos empregamos uma média móvel que considera 7 dias anteriores.\n\nPara elaboração desse projeto optamos por ofertar dados do Inmet desde o ano de 2017, sendo apenas limitados pela data inicial de operação de cada uma das estações, configurando séries temporais até 31 de agosto de 2022.\n\nEm caso de problemas no funcionamento do bot ou simplesmente desejar cancelar alguma operação, digite /cancelar e você poderá voltar para o menu principal.\n\nVolte para o menu principal agora usando o comando /menu.', parse_mode='HTML')

@dp.message_handler(state='*', commands=['lista_estacoes'])

@dp.message_handler(Text(equals='lista_estacoes', ignore_case=True), state='*')

async def lista_estacoes(msg: Message, state: FSMContext):

    dados_sessao = await state.get_state()

    if dados_sessao is None:

        return await bot.send_message(msg.chat.id, f'Estações Meteorológicas Automáticas:\nSegue a listagem das estações disponíveis no projeto: {utils.info_estacoes_disponiveis}.\n\nVolte para o menu principal agora usando o comando /menu.', parse_mode='HTML')

    await bot.send_message(msg.chat.id, f'Estações Meteorológicas Automáticas:\nSegue a listagem das estações disponíveis no projeto: {utils.info_estacoes_disponiveis}.\n\nVolte para o menu principal agora usando o comando /menu.', parse_mode='HTML')

```
    await msg.reply('Volte para o <b>menu principal</b> agora usando o comando /menu.', parse_mode='HTML')

    await state.finish()


@dp.message_handler(state='*', commands=['cancelar', 'reiniciar', 'restart', 'voltar'])

@dp.message_handler(Text(equals='cancelar', ignore_case=True), state='*')

async def cancela_operacoes(msg: Message, state: FSMContext):

    # Possibilita cancelar operações e reseta o estado da sessão atual

    dados_sessao = await state.get_state()

    if dados_sessao is None:

        return msg.reply('Nada para ser cancelado.')

    await msg.reply('Cancelando operações!\n\nVolte para o <b>menu principal</b> agora usando o comando /menu.', parse_mode='HTML')

    await state.finish()


#####################################################################
#####################

#                    DEFINE ESTAÇÕES METEOROLÓGICAS                 #

#####################################################################
#####################


@dp.callback_query_handler(lambda c: c.data == 'informa_ema')

async def escolhe_ema(call: CallbackQuery):
```

```
    await EMA_sessao.ema.set()

    await bot.send_message(call.message.chat.id, 'Digite o código da estação meteorológica.')


@dp.message_handler(state=EMA_sessao.ema)

async def valida_ema(msg: Message, state: FSMContext):

    async with state.proxy() as dados_sessao:

        if msg.text.strip().upper() in utils.estacoes: # valida o código inserido e registra nos dados da sessão ativa, mesmo que se digitado com letras minúsculas

            dados_sessao['ema'] = msg.text.strip().upper()

            await bot.send_message(msg.chat.id, f'A estação <b>{dados_sessao["ema"]}</b> iniciou suas operações em: <b>{utils.trata_data_completa(utils.inicio_operacao[dados_sessao["ema"]])}</b>.\nPossuímos dados registrados pelo Inmet até <b>31 de agosto de 2022</b>.', parse_mode='HTML')

            await bot.send_photo(msg.chat.id, utils.plotar_mapa_estacao(msg.chat.id, dados_sessao['ema']))

            await bot.send_message(msg.chat.id, 'Agora informe data para definir o início da série temporal, com <b>dia</b>, <b>mês</b> e <b>ano</b> separados por um traço.\n\n<b>Exemplo</b>: 01-11-2019', parse_mode='HTML')

            await EMA_sessao.next()

        else:

            await msg.reply('<b>Código inválido</b>.\nInforme um código válido.\n\nSe não souber os códigos, use outro método ou confira os códigos das estações meteorológicas automáticas através do comando /lista_estacoes.', parse_mode='HTML')
```

```
# Valida entrada e formato de data

@dp.message_handler(lambda msg: not utils.valida_data(msg.text), state=EMA_sessao.ema_inicio_periodo)

async def valida_ema_inicio_periodo(msg: Message):

    return await msg.reply('Você precisa definir uma data informando <b>dia</b>, <b>mês</b> e <b>ano</b>, separados por um traço <b>(-)</b>.\n\nÉ preciso seguir o <b>padrão do exemplo</b> para dar continuidade.\n\n<b>Exemplo</b>: 01-11-2019', parse_mode='HTML')


@dp.message_handler(state=EMA_sessao.ema_inicio_periodo)

async def define_ema_periodo_inicial(msg: Message, state: FSMContext):

    async with state.proxy() as dados_sessao:

        await state.update_data(ema_inicio_periodo=utils.valida_inicio_operacao(dados_sessao["ema"], msg.text))

        await msg.reply('Agora iremos definer o final da série temporal.\n\n<b>Exemplo</b>: 01-12-2019', parse_mode='HTML')

        await EMA_sessao.next()


# Valida entrada e formato de data

@dp.message_handler(lambda msg: not utils.valida_data(msg.text), state=EMA_sessao.ema_fim_periodo)

async def valida_ema_fim_periodo(msg: Message):

    return await msg.reply('Você precisa definir a data informando <b>dia</b>, <b>mês</b> e <b>ano</b>, separados por um traço <b>(-)</b>.\n\nÉ preciso
```

seguir o padrão do exemplo para dar continuidade.\n\nExemplo: 01-12-2019', parse_mode='HTML')

@dp.message_handler(state=EMA_sessao.ema_fim_periodo)

async def define_ema_fim_periodo(msg: Message, state: FSMContext):

    await bot.send_message(msg.chat.id, 'Escolha a Variável Climática', reply_markup=menu_vars_climaticas, parse_mode='HTML')

    async with state.proxy() as dados_sessao:

        await state.update_data(ema_fim_periodo=utils.valida_fim_operacao(dados_sessao['ema_inicio_periodo'], msg.text))

        await EMA_sessao.next()

@dp.callback_query_handler(state=EMA_sessao.ema_var_climatica)

async def define_ema_var_climatica(call: CallbackQuery, state: FSMContext):

    async with state.proxy() as dados_sessao:

dados_sessao['ema_var_climatica']=utils.opcoes_inline_vars_climaticas[call.data]

        await bot.send_message(call.message.chat.id,

            f"Estação Adotada: {dados_sessao['ema']}\nIntervalo Temporal: {dados_sessao['ema_inicio_periodo']} até {dados_sessao['ema_fim_periodo']}\nVaríavel Climática: {utils.vars_climaticas[dados_sessao['ema_var_climatica']]}", parse_mode='HTML')

```
    await bot.send_photo(call.message.chat.id, utils.plotar_dados_estacao(call.message.chat.id, dados_sessao['ema'], dados_sessao['ema_var_climatica'], dados_sessao['ema_inicio_periodo'], dados_sessao['ema_fim_periodo']))

    await bot.send_message(call.message.chat.id, '<b>Atenção</b>:\n\nPara observar dados de outra estação meteorológica, janela temporal ou escolher uma nova variável climátic <b>é preciso retornar ao menu principal</b>.\nPara isso basta usar os comandos /menu ou /iniciar.', parse_mode='HTML')

    utils.limpa_dados_gerados()

    await state.finish()



#####################################################################
#####################
#                  COORDENADAS GEOGRÁFICAS                  #
#####################################################################
#####################


@dp.callback_query_handler(lambda c: c.data == 'informa_latlong')

async def latlong(call: CallbackQuery):

  await bot.send_message(call.message.chat.id,'Como definir as <b>Coordenadas Geográficas</b>.\n\nAs suas coordenadas podem ser informadas como <b>Grau-Minuto-Segundo (GMS)</b> ou <b>Grau-Decimal (GD)</b>.', parse_mode='HTML')

  await bot.send_message(call.message.chat.id, "<b>Padrão GMS</b>:\n\nTemos as informações dispostas da seguinte maneira: Latitude 22°55'3" S e Longitude 43°5'9" O.\nSe quiser usar esse padrão, basta fornecer
```

os dados separados por traço, informando latitude e longitude separados por vírgula, sem a necessidade de mencionar as posições (Sul ou Oeste).\n\nExemplo: 22-5-3, 42-5-3\n\nPadrão GD:\n\nTemos as informações dispostas da seguinte maneira: Latitude -22.9175 e Longitude -43.0858.\nSe quiser usar esse padrão, basta preencher os dados sem o sinal negativo, usando ponto para a parte fracionada, informando latitude e longitude separados por vírgula,.\n\nExemplo: 22.9175, 43.0858\n\nOs valores são negativos pois latitudes abaixo da Linha do Equador são valores negativos, assim como valores de longitude a oeste do Meridiano de Greenwich são negativos.", parse_mode='HTML')

await bot.send_message(call.message.chat.id, 'Informe agora a latitude e longitude da sua coordenada.', parse_mode='HTML')

await COORD_sessao.next()

@dp.message_handler(lambda msg: not utils.valida_coordenadas(msg.text), state=COORD_sessao.coordenadas)

async def valida_coordenadas(msg: Message):

await msg.reply('As coordenadas informadas não são válidas.\n\nInforme coordenadas válidas contendo latitude e longitude conforme os exemplos citados.', parse_mode='HTML')

return await bot.send_message(msg.chat.id, 'Reveja as instruções assim como as coordenadas para o seu ponto de referência. Lembre-se que temos outros métodos para definir pontos de referência para obtenção de dados climáticos.')

@dp.message_handler(state=COORD_sessao.coordenadas)

async def processa_coordenadas(msg: Message, state: FSMContext):

```
    async with state.proxy() as dados_sessao:
        dados_sessao['coordenadas'] = msg.text
        coordenadas_dividida = dados_sessao['coordenadas'].split(',')
        latitude = utils.converte_gms_gd(coordenadas_dividida[0].strip())
        longitude = utils.converte_gms_gd(coordenadas_dividida[1].strip())
        dados_sessao['coord_ema'] = utils.busca_estacao_usando_coordenadas(latitude, longitude)
        await COORD_sessao.next()
        await bot.send_message(msg.chat.id, f'<b>Coordenadas Geográficas</b>\n\n<b>Latitude</b>: {latitude}\n<b>Longitude</b>: {longitude}', parse_mode='HTML')
        await bot.send_message(msg.chat.id, f'A estação mais próxima é a <b>{dados_sessao["coord_ema"]}</b>. Essa estação iniciou suas operações em: <b>{utils.trata_data_completa(utils.inicio_operacao[dados_sessao["coord_ema"]])}</b>.\n\nPossuímos dados registrados pelo Inmet até <b>31 de agosto de 2022</b>. com isso podemos criar visualizações de seus dados climáticos.', parse_mode='HTML')
        await bot.send_photo(msg.chat.id, utils.plota_mapa_coord(msg.chat.id, latitude, longitude))
        await COORD_sessao.next()
        await bot.send_message(msg.chat.id, 'Agora informe data para definir o início da série temporal, informando <b>dia</b>, <b>mês</b> e <b>ano</b>, separados por um traço <b>(-)</b>.\n\nÉ preciso seguir o <b>padrão do exemplo</b> para dar continuidade.\n\n<b>Exemplo</b>: 01-11-2019', parse_mode='HTML')
```

```
@dp.message_handler(lambda msg: not utils.valida_data(msg.text), state=COORD_sessao.coord_inicio_periodo)

async def valida_periodo_inicial_coord(msg: Message):

    return await msg.reply('Você precisa definir a data informando mês e ano, separados por um traço (-).\n\nÉ preciso seguir o <b>padrão do exemplo</b> para dar continuidade.\n\nExemplo: 01-12-2019')


@dp.message_handler(state=COORD_sessao.coord_inicio_periodo)

async def define_periodo_inicial_coord(msg: Message, state: FSMContext):

    async with state.proxy() as dados_sessao:

        await state.update_data(coord_inicio_periodo=utils.valida_inicio_operacao(dados_sessao["coord_ema"], msg.text))

        await msg.reply('Agora iremos definer o final da série temporal.\n\n<b>Exemplo</b>: 01-12-2019', parse_mode='HTML')

        await COORD_sessao.next()


# Valida entrada e formato de data

@dp.message_handler(lambda msg: not utils.valida_data(msg.text), state=COORD_sessao.coord_fim_periodo)

async def valida_fim_periodo_coord(msg: Message):

    return await msg.reply('Você precisa definir a data informando <b>dia</b>, <b>mês</b> e <b>ano</b>, separados por um traço <b>(-)</b>.\n\nÉ preciso seguir o <b>padrão do exemplo</b> para dar continuidade.\n\n<b>Exemplo</b>: 01-12-2019', parse_mode='HTML')
```

```
@dp.message_handler(state=COORD_sessao.coord_fim_periodo)
async def define_periodo_final_coord(msg: Message, state: FSMContext):
    await bot.send_message(msg.chat.id, 'Agora é preciso escolher a variável climática.', reply_markup=menu_vars_climaticas)
    async with state.proxy() as dados_sessao:
        await state.update_data(coord_fim_periodo=utils.valida_fim_operacao(dados_sessao['coord_inicio_periodo'], msg.text))
        await COORD_sessao.next()

@dp.callback_query_handler(state=COORD_sessao.coord_var_climatica)
async def define_var_climatica(call: CallbackQuery, state: FSMContext):
    async with state.proxy() as dados_sessao:

dados_sessao['coord_var_climatica']=utils.opcoes_inline_vars_climaticas[call.data]

        await bot.send_message(call.message.chat.id,
            f"<b>Estação Adotada</b>: {dados_sessao['coord_ema']}\n<b>Intervalo Temporal</b>: {dados_sessao['coord_inicio_periodo']} até {dados_sessao['coord_fim_periodo']}\n<b>Varíavel Climática</b>: {utils.vars_climaticas[dados_sessao['coord_var_climatica']]}",
parse_mode='HTML')
        await bot.send_photo(call.message.chat.id, utils.plotar_dados_estacao(call.message.chat.id, dados_sessao['coord_ema'],
```

```
dados_sessao['coord_var_climatica'], dados_sessao['coord_inicio_periodo'], dados_sessao['coord_fim_periodo']))

    await bot.send_message(call.message.chat.id, '<b>Atenção</b>:\n\nPara observar dados de outra estação meteorológica, janela temporal ou escolher uma nova variável climátic <b>é preciso retornar ao menu principal</b>.\nPara isso basta usar os comandos /menu ou /iniciar.', parse_mode='HTML')

    utils.limpa_dados_gerados()

    await state.finish()


#####################################################################
######################

#                    DEFINE MUNICÍPIO                    #

#####################################################################
######################


@dp.callback_query_handler(lambda c: c.data == 'informa_municipio')

async def municipio(call: CallbackQuery):

  await MUNICIPIO_sessao.municipio.set()

  await bot.send_message(call.message.chat.id, 'Informe o <b>nome do município</b> do estado do Rio de Janeiro.', parse_mode='HTML')


@dp.message_handler(lambda msg: not msg.text in utils.municipios_rj, state=MUNICIPIO_sessao.municipio)

async def processa_municipio_invalido(msg: Message):
```

return await msg.reply(f'O município "{msg.text}" é inválido. Não deve ser um município do Rio de Janeiro ou foi escrito incorretamente.\n\nInforme o nome do município novamente.', parse_mode='HTML')

@dp.message_handler(lambda msg: msg.text in utils.municipios_rj, state=MUNICIPIO_sessao.municipio)

async def processa_municipio(msg: Message, state: FSMContext):

await MUNICIPIO_sessao.next()

await state.update_data(municipio=msg.text)

estacao_adotada = utils.descobrir_ema_com_municipio(msg.text)

latitude_municipio = utils.coordenada_municipio(msg.text, "latitude")

longitude_municipio = utils.coordenada_municipio(msg.text, "longitude")

await bot.send_message(msg.chat.id, f'O município {msg.text} tem sua sede georreferenciada nas coordenadas latitude {latitude_municipio} longitude {longitude_municipio} e será usado como ponto de referência.\n\nA estação mais próxima é a {estacao_adotada}, que iniciou suas operações em: {utils.trata_data_completa(utils.inicio_operacao[estacao_adotada])}.', parse_mode='HTML')

await bot.send_photo(msg.chat.id, utils.plota_mapa_coord(msg.chat.id, latitude_municipio, longitude_municipio))

await bot.send_message(msg.chat.id, f'Informe agora o período inicial para observação das variáveis climáticas na região do município {msg.text}.', parse_mode='HTML')

await bot.send_message(msg.chat.id, 'Agora informe data para definir o início da série temporal, informando dia, mês e ano,

separados por um traço (-).\n\nÉ preciso seguir o padrão do exemplo para dar continuidade.\n\nExemplo: 01-11-2019', parse_mode='HTML')

@dp.message_handler(lambda msg: not utils.valida_data(msg.text), state=MUNICIPIO_sessao.municipio_inicio_periodo)

async def valida_municipio_inicio_periodo(msg: Message):

return await msg.reply('Você precisa definir a data informando dia, mês e ano, separados por um traço (-).\n\nÉ preciso seguir o padrão do exemplo para dar continuidade.\n\nExemplo: 01-11-2019', parse_mode='HTML')

@dp.message_handler(state=MUNICIPIO_sessao.municipio_inicio_periodo)

async def define_periodo_inicial_municipio(msg: Message, state: FSMContext):

async with state.proxy() as dados_sessao:

await state.update_data(municipio_inicio_periodo=utils.valida_inicio_operacao(utils.descobrir_ema_com_municipio(dados_sessao["municipio"]), msg.text))

await msg.reply('Agora iremos definer o final da série temporal.\n\nExemplo: 01-12-2019', parse_mode='HTML')

await MUNICIPIO_sessao.next()

@dp.message_handler(lambda msg: not utils.valida_data(msg.text), state=MUNICIPIO_sessao.municipio_fim_periodo)

async def valida_municipio_fim_periodo(msg: Message):

```
    return await msg.reply('Você precisa definir a data informando <b>dia</b>, <b>mês</b> e <b>ano</b>, separados por um traço <b>(-)</b>.\n\nÉ preciso seguir o <b>padrão do exemplo</b> para dar continuidade.\n\n<b>Exemplo</b>: 01-12-2019', parse_mode='HTML')


@dp.message_handler(state=MUNICIPIO_sessao.municipio_fim_periodo)
async def define_periodo_final_municipio(msg: Message, state: FSMContext):
    async with state.proxy() as dados_sessao:
        await state.update_data(municipio_fim_periodo=utils.valida_fim_operacao(dados_sessao['municipio_inicio_periodo'], msg.text))
        await bot.send_message(msg.chat.id, 'Agora é preciso escolher a variável climática.', reply_markup=menu_vars_climaticas)
        await MUNICIPIO_sessao.next()


@dp.callback_query_handler(state=MUNICIPIO_sessao.municipio_var_climatica)
async def define_var_climatica(call: CallbackQuery, state: FSMContext):
    async with state.proxy() as dados_sessao:

dados_sessao['municipio_var_climatica']=utils.opcoes_inline_vars_climaticas[call.data]
        estacao_adotada = utils.descobrir_ema_com_municipio(dados_sessao['municipio'])
        await bot.send_message(call.message.chat.id,
```

```
        f"<b>Estação Adotada</b>: {estacao_adotada}\n<b>Intervalo Temporal</b>: {dados_sessao['municipio_inicio_periodo']} até {dados_sessao['municipio_fim_periodo']}\n<b>Varíavel Climática</b>: {utils.vars_climaticas[dados_sessao['municipio_var_climatica']]}", parse_mode='HTML')


    await bot.send_photo(call.message.chat.id, utils.plotar_dados_estacao(call.message.chat.id, estacao_adotada, dados_sessao['municipio_var_climatica'], dados_sessao['municipio_inicio_periodo'], dados_sessao['municipio_fim_periodo']))

    await bot.send_message(call.message.chat.id, '<b>Atenção</b>:\n\nPara observar dados de outra estação meteorológica, janela temporal ou escolher uma nova variável climátic <b>é preciso retornar ao menu principal</b>.\nPara isso basta usar os comandos /menu ou /iniciar.', parse_mode='HTML')

    utils.limpa_dados_gerados()

    await state.finish()


#####################################################################
#####################

#                   DEFINE CEP                    #

#####################################################################
#####################


@dp.callback_query_handler(lambda c: c.data == 'informa_cep')

async def cep(call: CallbackQuery):

  await CEP_sessao.cep.set()
```

```
    await bot.send_message(call.message.chat.id, 'Informe o <b>CEP</b> de um endereço do estado do Rio de Janeiro, sem usar traços ou pontos.\n\n<b>Exemplo</b>: 24342240', parse_mode='HTML')


@dp.message_handler(lambda msg: not utils.valida_cep(msg.text), state=CEP_sessao.cep)
async def processa_cep_invalido(msg: Message):
    return await msg.reply(f'<b>CEP informado "{msg.text}"</b> é inválido ou não é encontrado no sistema.\n\nVerifique o CEP, confira se foi digitado corretamente e tente novamente ou considere outra opção para buscar dados climáticos no /menu_principal.', parse_mode='HTML')


@dp.message_handler(state=CEP_sessao.cep)
async def registra_cep(msg: Message, state: FSMContext):
    await CEP_sessao.next()
    await state.update_data(cep=(msg.text).strip().replace('-', ''))
    cep_df = utils.geolocaliza_endereco(msg.text)
    latitude_cep = round(cep_df.latitude[0], 4)
    longitude_cep = round(cep_df.longitude[0], 4)
    cep_ema = utils.busca_estacao_usando_coordenadas(latitude_cep, longitude_cep)
    await bot.send_message(msg.chat.id, f'<b>Coordenadas Geográficas</b>\n\n<b>Latitude</b>: {latitude_cep}\n<b>Longitude</b>: {longitude_cep}', parse_mode='HTML')
    await bot.send_photo(msg.chat.id, utils.plota_mapa_coord(msg.chat.id, latitude_cep, longitude_cep))
```

```
    await bot.send_message(msg.chat.id, f'A estação mais próxima é a <b>{cep_ema}</b>. Essa estação iniciou suas operações em: <b>{utils.trata_data_completa(utils.inicio_operacao[cep_ema])}</b>.\n\nPossuímos dados registrados pelo Inmet até <b>31 de agosto de 2022</b>. com isso podemos criar visualizações de seus dados climáticos.', parse_mode='HTML')

    await bot.send_message(msg.chat.id, 'Agora iremos definir o início da série temporal, com <b>dia</b>, <b>mês</b> e <b>ano</b> separados por um traço.\n\n<b>Exemplo</b>: 01-11-2019', parse_mode='HTML')


@dp.message_handler(lambda msg: not utils.valida_data(msg.text), state=CEP_sessao.cep_inicio_periodo)

async def valida_inicio_periodo_cep(msg: Message):

    return await msg.reply('Você precisa definir a data informando <b>dia</b>, <b>mês</b> e <b>ano</b>, separados por um traço <b>(-)</b>.\n\nÉ preciso seguir o <b>padrão do exemplo</b> para dar continuidade.\n\n<b>Exemplo</b>: 01-11-2019', parse_mode='HTML')


@dp.message_handler(state=CEP_sessao.cep_inicio_periodo)

async def define_periodo_inicial_cep(msg: Message, state: FSMContext):

    async with state.proxy() as dados_sessao:

        cep = dados_sessao['cep']

        cep_df = utils.geolocaliza_endereco(cep)

        latitude_cep = round(cep_df.latitude[0], 4)

        longitude_cep = round(cep_df.longitude[0], 4)

        cep_ema = utils.busca_estacao_usando_coordenadas(latitude_cep, longitude_cep)
```

```
    await
state.update_data(cep_inicio_periodo=utils.valida_inicio_operacao(cep_ema,
msg.text))

    await msg.reply('Agora iremos definer o final da série
temporal.\n\n<b>Exemplo</b>: 01-12-2019', parse_mode='HTML')

    await CEP_sessao.next()


@dp.message_handler(lambda msg: not utils.valida_data(msg.text),
state=CEP_sessao.cep_fim_periodo)

async def valida_fim_periodo_cep(msg: Message):

  return await msg.reply('Você precisa definir a data informando <b>dia</b>,
<b>mês</b> e <b>ano</b>, separados por um traço <b>(-)</b>.\n\nÉ preciso
seguir o <b>padrão do exemplo</b> para dar continuidade.\n\n<b>Exemplo</b>:
01-12-2019', parse_mode='HTML')


@dp.message_handler(state=CEP_sessao.cep_fim_periodo)

async def define_periodo_inicial_ema(msg: Message, state: FSMContext):

  await bot.send_message(msg.chat.id, 'Agora é preciso escolher a variável
climática.', reply_markup=menu_vars_climaticas)

  async with state.proxy() as dados_sessao:

    await
state.update_data(cep_fim_periodo=utils.valida_fim_operacao(dados_sessao['c
ep_inicio_periodo'], msg.text))

    await CEP_sessao.next()


@dp.callback_query_handler(state=CEP_sessao.cep_var_climatica)
```

```
async def define_var_climatica(call: CallbackQuery, state: FSMContext):
    async with state.proxy() as dados_sessao:

dados_sessao['cep_var_climatica']=utils.opcoes_inline_vars_climaticas[call.dat
a]
        cep = dados_sessao['cep']
        cep_df = utils.geolocaliza_endereco(cep)
        latitude_cep = round(cep_df.latitude[0], 4)
        longitude_cep = round(cep_df.longitude[0], 4)
        cep_ema = utils.busca_estacao_usando_coordenadas(latitude_cep,
longitude_cep)
        await bot.send_message(call.message.chat.id,
            f"<b>Estação Adotada</b>: {cep_ema}\n<b>Intervalo Temporal</b>:
{dados_sessao['cep_inicio_periodo']} até
{dados_sessao['cep_fim_periodo']}\n<b>Varíavel Climática</b>:
{utils.vars_climaticas[dados_sessao['cep_var_climatica']]}",
parse_mode='HTML')
        await bot.send_photo(call.message.chat.id,
utils.plotar_dados_estacao(call.message.chat.id, cep_ema,
dados_sessao['cep_var_climatica'], dados_sessao['cep_inicio_periodo'],
dados_sessao['cep_fim_periodo']))
        await bot.send_message(call.message.chat.id, '<b>Atenção</b>:\n\nPara
observar dados de outra estação meteorológica, janela temporal ou escolher
uma nova variável climática<b> é preciso retornar ao menu principal</b>.\nPara
isso basta usar os comandos /menu ou /iniciar.', parse_mode='HTML')
        utils.limpa_dados_gerados()
        await state.finish()
```

```
print('>>>>>>>>>RODANDO<<<<<<<<<')


if __name__ == '__main__':
    executor.start_polling(dp, skip_updates=True)
```

ANEXO 2 – CÓDIGO DAS FUNÇÕES USADAS NO PROGRAMA

```
from operator import contains

import pandas as pd

import geopandas as gpd

import matplotlib

matplotlib.use('agg')

import matplotlib.pyplot as plt

import matplotlib.ticker as plticker

import matplotlib.dates as mdates

from shapely.geometry import Point

from geopy.geocoders import Nominatim

from geopy.extra.rate_limiter import RateLimiter

import datetime

import pycep_correios

import os

import glob

import warnings

import numpy as np

warnings.simplefilter(action='ignore', category=FutureWarning)

import config

from io import BytesIO

from PIL import Image
```

```
mapa_estado_rj = gpd.read_file('./dados_geo/RJ_UF_2021.shp', encoding='utf-8').to_crs('+proj=utm +zone=23 +south +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=km +no_defs')


def converte_gms_gd(latlong):
    if latlong[2] == '.':
        coordenada_convertida = float(latlong) * -1
    else:
        info = str(latlong)
        direcao = -1 # multiplicará o valor da coordenada, adequando ao sistema
        conversao = info.replace('-',' ') # substitui os hífens entre os valores de grau, minuto, segundo para serem manipulados e organizados
        conversao = conversao.split() # segmenta a string anterior pelos espaços
        # conversao_direcao = conversao.pop() # remove o último segmento da coordenada, que trata da posição/direção no globo N-S-L-O
        grau = int(conversao[0])
        minuto = int(conversao[1])
        segundo = int(conversao[2])
        coordenada_convertida_gd = grau + minuto /60.0+ segundo/3600.0
        coordenada_convertida = coordenada_convertida_gd * direcao
    return round(coordenada_convertida, 4)


def valida_coordenadas(coordenadas):
    coordenadas_dividida = coordenadas.split(',')
```

```
    latitude = converte_gms_gd(coordenadas_dividida[0].strip())
    longitude = converte_gms_gd(coordenadas_dividida[1].strip())
    coordenadas_gd = {'latitude': [latitude], 'longitude': [longitude]}
    coordenadas_gd = pd.DataFrame(data = coordenadas_gd)
    coordenadas_gd = gpd.GeoDataFrame(data = coordenadas_gd, geometry = gpd.points_from_xy(coordenadas_gd.longitude, coordenadas_gd.latitude), crs = crs)
    coordenadas_gd = coordenadas_gd.to_crs('+proj=utm +zone=23 +south +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=km +no_defs')
    if mapa_estado_rj.contains(coordenadas_gd)[0]:
        return True
    else:
        return False

crs = {'proj': 'latlong', 'ellps': 'WGS84', 'datum': 'WGS84', 'no_defs': True}
'''
Definindo pontos das estações automáticas abordadas.
'''
estacoes = gpd.read_file('./dados_geo/estacoes_rj_geoloc.shp')
estacoes = estacoes.to_crs('+proj=utm +zone=23 +south +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=km +no_defs')

'''
```

Definindo mapa base para o Rio de Janeiro que será usado em todos os plots do projeto.

'''

mapa_rj = gpd.read_file('./dados_geo/RJ_Municipios_2021.shp', encoding='utf-8')

mapa_rj = mapa_rj.to_crs('+proj=utm +zone=23 +south +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=km +no_defs')

def obter_endereco(cep):

    '''

    Recebe um CEP e retorna endereço formatado.

    '''

    endereco = pycep_correios.get_address_from_cep(cep, webservice=pycep_correios.WebService.CORREIOS)

    return endereco['logradouro'] + ", " + endereco['bairro'] + ", " + endereco['cidade'] + " - " + endereco['uf']

def geolocaliza_endereco(cep_referencia):

    '''

    Recebe um CEP de referência e retorna objeto geodataframe contendo endereço e coordenadas para o ponto.

    Será preciso aplicar CRS compatível para plotar o ponto do CEP sobre mapa.

    '''

    cep_geolocalizado = pd.DataFrame({'CEP': [str(cep_referencia)]})

```
    geolocator = Nominatim(user_agent = config.user_agent) # user_agent definido no arquivo config.py

    geocode = RateLimiter(geolocator.geocode, min_delay_seconds=1)

    cep_geolocalizado['endereco'] = cep_geolocalizado['CEP'].apply(obter_endereco).apply(geocode)

    cep_geolocalizado['latitude'] = cep_geolocalizado['endereco'].apply(lambda loc: loc.latitude if loc else None)

    cep_geolocalizado['longitude'] = cep_geolocalizado['endereco'].apply(lambda loc: loc.longitude if loc else None)

    cep_geolocalizado['geometry'] = [Point(x) for x in zip(cep_geolocalizado.longitude, cep_geolocalizado.latitude)]

    return cep_geolocalizado


def valida_cep(informar_cep):

    # faixa de CEP RJ 20000000 até 28999999, antes do intervalo são CEPs de SP, após são CEPs do ES

    cep_inicio = 20000000

    cep_fim = 28999999

    if cep_inicio <= int(informar_cep.strip().replace('-', '')) <= cep_fim: # valida cep dentro do intervalo permitido para o RJ

        try: # através da api, checa se o mesmo retorna dados geográficos válidos

            cep = pd.DataFrame({'CEP': [informar_cep.strip().replace('-', '')]}) # cria dataframe para conter dados gerados pela API

            geolocator = Nominatim(user_agent = config.user_agent) # user_agent definido no arquivo config.py

            geocode = RateLimiter(geolocator.geocode, min_delay_seconds=1)
```

```
            cep['endereco'] = cep['CEP'].apply(obter_endereco).apply(geocode)
            cep['latitude'] = cep['endereco'].apply(lambda loc: loc.latitude if loc else None)
            cep['longitude'] = cep['endereco'].apply(lambda loc: loc.longitude if loc else None)
            cep['geometry'] = [Point(x) for x in zip(cep.longitude, cep.latitude)]
            return True
        except:
            return False
    else:
        return False


def plotar_mapa_estacao(userid, ema):
    estacoes = gpd.read_file('./dados_geo/estacoes_rj_geoloc.shp')
    estacoes = estacoes.to_crs('+proj=utm +zone=23 +south +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=km +no_defs')
    ponto = estacoes[estacoes['codigo'] == ema]
    demais_estacoes = estacoes[estacoes['codigo'] != ema]
    base = mapa_rj.plot(color='#a9c4aa', edgecolor='#000', figsize=(10,10))
    ponto.plot(ax = base, markersize = 5, marker='x', color = 'red', label = f'Estação de Referência {ema}')
    demais_estacoes.plot(ax = base, markersize=5, marker = '*', color='blue', label = 'Estações Meteorológicas Automáticas do RJ')
```

```
    base.set_xlabel('Distância (Km)')
    base.set_ylabel('Distância (Km)')
    plt.legend(loc = 'upper left')
    locs,labels = plt.xticks()
    plt.xticks(locs, map(lambda x: "%g" % x, locs - 450))
    locs, labels = plt.yticks()
    plt.yticks(locs, map(lambda y: "%g" % y, locs - 7400))
    bytes_img = BytesIO()
    plt.savefig(bytes_img, format='png', dpi=300, bbox_inches='tight')
    bytes_img.seek(0)
    bytes_img.name = f'{str(userid)}_{ema}_plotado'
    return bytes_img


def plotar_mapa_cep(cep_geolocalizado):
    '''
    Recebe CEP já geolocalizado e plota sobre mapa na forma de ponto, na cor vermelha, sinalizando o ponto de interesse do usuário.
    '''
    ponto_cep = gpd.GeoDataFrame(cep_geolocalizado, crs = crs).to_crs('+proj=utm +zone=23 +south +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=km +no_defs')
    base = mapa_rj.plot(color='#a9c4aa', edgecolor='#000', figsize=(10,10))
    estacoes.plot(ax = base, markersize=5, marker = '*', color='blue', label = 'Estações Automáticas')
```

```
    ponto_cep.plot(ax = base, markersize = 5, marker = 'x', color = 'red', label = 'Local de Referência')

    base.set_title(f'Localização para o CEP {str(cep_geolocalizado)}')

    base.set_xlabel('Distância (Km)')

    base.set_ylabel('Distância (Km)')

    plt.legend(loc = 'upper left')

    plt.savefig(f'imgs_plots/{str(cep_geolocalizado)}_plotado.png', bbox_inches='tight')


def obter_estacao_proxima(ref):

    '''

    Passa o DF com o CEP geolocalizado, para se então obter a estação meteorológica mais próxima.

    '''

    estacoes['distancia_pont_estacao'] = estacoes.geometry.apply(lambda x: ref.distance(x).min())

    estacao_proxima = estacoes['distancia_pont_estacao'].min()

    estacao = estacoes[estacoes.distancia_pont_estacao == estacao_proxima]

    estacao_proxima = str(list(estacao.local)[0], list(estacao.codigo)[0])

    return estacao_proxima


# Gera mapa com a coordenada geográfica GD apontando a posição do ponto e EMA mais próxima

def plota_mapa_coord(userid, latitude, longitude):
```

```
    coordenadas_gd = {'latitude': [latitude], 'longitude': [longitude]}

    coordenadas_gd = pd.DataFrame(data = coordenadas_gd)

    coordenadas_gd = gpd.GeoDataFrame(data = coordenadas_gd, geometry = gpd.points_from_xy(coordenadas_gd.longitude, coordenadas_gd.latitude), crs = crs)

    coordenadas_gd = coordenadas_gd.to_crs('+proj=utm +zone=23 +south +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=km +no_defs')


    estacoes = gpd.read_file('./dados_geo/estacoes_rj_geoloc.shp')

    estacoes = estacoes.to_crs('+proj=utm +zone=23 +south +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=km +no_defs')

    estacoes['ema_distancia_min'] = estacoes.geometry.apply(lambda x: coordenadas_gd.distance(x).min())

    estacao_proxima = estacoes['ema_distancia_min'].min()

    estacao = estacoes[estacoes.ema_distancia_min == estacao_proxima]


    mapa_rj = gpd.read_file('./dados_geo/RJ_Municipios_2021.shp', encoding='utf-8')

    mapa_rj = mapa_rj.to_crs('+proj=utm +zone=23 +south +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=km +no_defs')


    # carrega mapa base do RJ com geopandas

    base = mapa_rj.plot(color ='#a9c4aa', edgecolor = '#000', figsize = (10,10))

    # carrega as posições das EMAs sobre o mapa do RJ
```

```
    estacoes.plot(ax = base, markersize = 5, marker = '*', color = 'blue', label = 'Estações Meteorológicas Automáticas do RJ')

    # posiciona o ponto de referência de acordo com a coordenada geográfica informada

    coordenadas_gd.plot(ax = base, markersize=5, marker = 'x', color = 'red', label = 'Ponto de Referência')

    # posiciona o ponto no mapa da estação mais próxima do ponto de referência

    estacao.plot(ax = base, markersize=5, marker = 'x', color = 'orange', label = f'Estação mais próxima EMA {list(estacao.codigo)[0]}')


    base.set_xlabel('Distância (Km)')

    base.set_ylabel('Distância (Km)')

    plt.legend(loc = 'upper left')

    locs,labels = plt.xticks()

    plt.xticks(locs, map(lambda x: "%g" % x, locs - 450))

    locs, labels = plt.yticks()

    plt.yticks(locs, map(lambda y: "%g" % y, locs - 7400))

    bytes_img = BytesIO()

    plt.savefig(bytes_img, format='png', dpi=300, bbox_inches='tight')

    bytes_img.seek(0)

    bytes_img.name = f'imgs_plots/{str(userid)}_latlong.png'

    return bytes_img

    # return list(estacao.codigo)[0]
```

```
def busca_estacao_usando_coordenadas(latitude, longitude):

    coordenadas_gd = {'latitude': [latitude], 'longitude': [longitude]}

    coordenadas_gd = pd.DataFrame(data = coordenadas_gd)

    coordenadas_gd = gpd.GeoDataFrame(data = coordenadas_gd, geometry = gpd.points_from_xy(coordenadas_gd.longitude, coordenadas_gd.latitude), crs = crs)

    coordenadas_gd = coordenadas_gd.to_crs('+proj=utm +zone=23 +south +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=km +no_defs')

    estacoes = gpd.read_file('./dados_geo/estacoes_rj_geoloc.shp')

    estacoes = estacoes.to_crs('+proj=utm +zone=23 +south +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=km +no_defs')

    estacoes['ema_distancia_min'] = estacoes.geometry.apply(lambda x: coordenadas_gd.distance(x).min())

    estacao_proxima = estacoes['ema_distancia_min'].min()

    estacao = estacoes[estacoes.ema_distancia_min == estacao_proxima]

    return list(estacao.codigo)[0]
```

```
estacoes = ['A601', 'A602', 'A603', 'A604', 'A606', 'A607', 'A608', 'A609', 'A610', 'A611', 'A618', 'A619', 'A620', 'A621', 'A624', 'A625', 'A626', 'A627', 'A628', 'A629', 'A630', 'A635', 'A636', 'A652', 'A659', 'A667']
```

```
info_estacoes_disponiveis = 'A601, A602, A603, A604, A606, A607, A608, A609, A610, A611, A618, A619, A620, A621, A624, A625, A626, A627, A628, A629, A630, A635, A636, A652, A659, A667'
```

```
inicio_operacao = {
    'A601': '2000-05-23',
    'A602': '2002-11-07',
    'A603': '2002-10-20',
    'A604': '2002-11-19',
    'A606': '2006-09-21',
    'A607': '2006-09-24',
    'A608': '2006-09-21',
    'A609': '2006-09-28',
    'A610': '2006-10-21',
    'A611': '2006-09-26',
    'A618': '2006-10-31',
    'A619': '2006-11-18',
    'A620': '2008-06-12',
    'A621': '2007-04-12',
    'A624': '2010-09-17',
    'A625': '2016-06-07',
    'A626': '2016-06-02',
    'A627': '2018-07-12',
    'A628': '2017-08-24',
    'A629': '2018-10-10',
    'A630': '2018-10-15',
    'A635': '2017-08-31',
```

```
    'A636': '2017-08-09',
    'A652': '2007-05-17',
    'A659': '2015-07-27',
    'A667': '2015-09-01'}

def ajusta_data_plot(dia_mes_ano):
    tratando_data = dia_mes_ano.replace('-',' ')
    dividindo_data = tratando_data.split()
    dia = dividindo_data[0]
    mes = dividindo_data[1]
    ano = dividindo_data[2]
    ano_mes_dia = f'{ano}-{mes}-{dia}'
    return ano_mes_dia

def trata_data_completa(data_completa):
    tratando_data = data_completa.replace('-',' ')
    dividindo_data = tratando_data.split()
    ano = dividindo_data[0]
    mes = dividindo_data[1]
    dia = dividindo_data[2]
    return f'{dia}-{mes}-{ano}'
```

```
# Recebe datas separadas por traço, separa elementos da data e reorganiza na string
def trata_data_mes_ano(data_completa):
    tratando_data = data_completa.strip().replace('-',' ')
    dividindo_data = tratando_data.split()
    ano = dividindo_data[0]
    mes = dividindo_data[1]
    return f'{mes}-{ano}'


def valida_data(data_completa):
    data_certa = None
    try:
        separa_dia_mes_ano = data_completa.replace('-', ' ').split()
        ano = int(separa_dia_mes_ano[2])
        mes = int(separa_dia_mes_ano[1])
        dia = int(separa_dia_mes_ano[0])
        if len(data_completa.replace('-','')) != 8 or not data_completa.replace('-','').replace('/','').isdigit():
            data_certa = False
            return data_certa
        else:
            try:
                nova_data = datetime.datetime(ano, mes, dia)
```

```
            data_certa = True
            return data_certa
        except:
            data_certa = False
            return data_certa
    except:
        return False


def valida_inicio_operacao(ema, dia_mes_ano):
    # testes A601 início operacional em 23-05-2000
    formato_data = '%d-%m-%Y'
    data_inicio_operacional = datetime.datetime.strptime(trata_data_completa(inicio_operacao[ema]), formato_data)
    data_inicial_proposta = datetime.datetime.strptime(dia_mes_ano.strip(), formato_data)
    data_inicial_limite = datetime.datetime(2017, 1, 1)
    data_final_limite = datetime.datetime(2022, 8, 31)
    if (data_inicial_limite < data_inicio_operacional < data_final_limite) or (data_inicial_limite < data_inicial_proposta < data_final_limite): # data de início de op da ema e data proposta estão contidas no intervalo init e fim
        if data_inicial_proposta > data_inicio_operacional: # se a data proposta ocorrer após o início de operação da ema, retorna a data proposta
            return dia_mes_ano
        else: # caso contrário, retorna a data de início de operação da ema
```

```
            return trata_data_completa(inicio_operacao[ema])
    elif data_inicio_operacional < data_inicial_limite:
        return '01-01-2017'
    elif data_inicial_proposta < data_inicial_limite:
        return '01-01-2017'


def valida_fim_operacao(dia_mes_ano_inicial_adotado, dia_mes_ano_final_proposto): # chegando nessa etapa de validação, a data inicial proposta já foi validada e está contida no intervalo de limites inicial e final
    formato_data = '%d-%m-%Y'
    data_inicial_adotada = datetime.datetime.strptime(dia_mes_ano_inicial_adotado, formato_data)
    data_final_proposta = datetime.datetime.strptime(dia_mes_ano_final_proposto.strip(), formato_data)
    data_inicial_limite = datetime.datetime(2017, 1, 1)
    data_final_limite = datetime.datetime(2022, 8, 31)
    if (data_inicial_limite < data_final_proposta < data_final_limite) and (data_final_proposta > data_inicial_adotada): # caso a data final proposta esteja no intervalo limite e for menor que a data inicial proposta, retorna a data final proposta
        return dia_mes_ano_final_proposto
    elif (data_inicial_limite < data_final_proposta < data_final_limite) and (data_inicial_adotada > data_final_proposta): # caso esteja dentro do intervalo, mas for menor que a data inicial adotada, retorna data final limite
        return '31-08-2022'
```

```
    elif (data_final_proposta > data_inicial_adotada) and (data_final_proposta ==
data_final_limite):
        return '31-08-2022'
    elif data_final_proposta < data_inicial_limite:
        return '31-08-2022'


opcoes_inline_vars_climaticas = {
        'var_temp_min': 'TEMP_MIN',
        'var_temp_med': 'TEMP_MED',
        'var_temp_max': 'TEMP_MAX',
        'var_chuva': 'CHUVA',
        'var_umid_min': 'UMID_MIN',
        'var_umid_med': 'UMID_MED',
        'var_vento_med': 'VEL_VENTO_MED'}


vars_climaticas = {
    'TEMP_MIN': 'Temperatura Mínima',
    'TEMP_MED': 'Temperatura Média',
    'TEMP_MAX': 'Temperatura Máxima',
    'CHUVA': 'Chuva',
    'UMID_MIN': 'Umidade Mínima',
    'UMID_MED': 'Umidade Média',
    'VEL_VENTO_MED': 'Velocidade Média do Vento'}
```

```
def plotar_dados_estacao(userid, estacao, variavel, inicio, fim = '31-08-2022'):
    ema = pd.read_csv(f'./dados_estacoes/{estacao}/{estacao}.csv', parse_dates=['DT_MEDICAO'])
    ema.set_index('DT_MEDICAO', inplace = True)
    inicio = ajusta_data_plot(inicio)
    fim = ajusta_data_plot(fim)
    vars_climaticas = {
        'TEMP_MIN': 'Temperatura Mínima',
        'TEMP_MED': 'Temperatura Média',
        'TEMP_MAX': 'Temperatura Máxima',
        'CHUVA': 'Chuva',
        'UMID_MIN': 'Umidade Mínima',
        'UMID_MED': 'Umidade Média',
        'VEL_VENTO_MED': 'Velocidade Média do Vento'}

    rotulo_variavel_climatica = vars_climaticas[variavel]
    fig, ax = plt.subplots(figsize=(10, 10))
    if 'TEMP' in variavel:
        loc = plticker.MultipleLocator(base=1.0)
        ax.yaxis.set_major_locator(loc)
    elif 'CHUVA' in variavel:
        loc = plticker.MultipleLocator(base=10.0)
```

```
        ax.yaxis.set_major_locator(loc)
    elif 'UMID' in variavel:
        loc = plticker.MultipleLocator(base=5.0)
        ax.yaxis.set_major_locator(loc)
    elif 'VENTO' in variavel:
        loc = plticker.MultipleLocator(base=1.0)
        ax.yaxis.set_major_locator(loc)


    ax = ema[f'{variavel}'][f'{inicio}':f'{fim}'].plot(figsize=(10,10), label = rotulo_variavel_climatica)
    media_movel = ema[f'{variavel}'][f'{inicio}':f'{fim}'].rolling(7).mean()
    media_movel.plot(label = 'Média Móvel')
    plt.legend(loc = 'upper right')
    ax.set_xlabel('Tempo')
    unidade = {
        'TEMP_MIN': 'Temperatura (°C)',
        'TEMP_MED': 'Temperatura (°C)',
        'TEMP_MAX': 'Temperatura (°C)',
        'CHUVA': 'Precipitação Total (mm)',
        'UMID_MIN': 'Umidade Relativa (%)',
        'UMID_MED': 'Umidade Relativa (%)',
        'VEL_VENTO_MED': 'Velocidade do Vento (m/s)'
    }
```

```
    ax.set_ylabel(f'{unidade[variavel]}')
    ax.xaxis.set_major_formatter(mdates.DateFormatter('%d-%m-%Y'))
    for label in ax.get_xticklabels(which='major'):
        label.set(rotation=30, horizontalalignment='right')
    ax.set_xlim(left=inicio, right=fim)
    # plt.savefig(f'imgs_plots/{nome_arquivo}.png', dpi=300, bbox_inches = "tight")
    bytes_img = BytesIO()
    plt.savefig(bytes_img, format='png', dpi=300, bbox_inches='tight')
    bytes_img.seek(0)
    bytes_img.name = f'{userid}_{estacao}_{variavel}'
    return bytes_img
```

```
municipios_rj = ['Angra dos Reis', 'Aperibé', 'Araruama', 'Areal', 'Armação dos Búzios', 'Arraial do Cabo', 'Barra do Piraí', 'Barra Mansa', 'Belford Roxo', 'Bom Jardim', 'Bom Jesus do Itabapoana', 'Cabo Frio', 'Cachoeiras de Macacu', 'Cambuci', 'Campos dos Goytacazes', 'Cantagalo', 'Carapebus', 'Cardoso Moreira', 'Carmo', 'Casimiro de Abreu', 'Comendador Levy Gasparian', 'Conceição de Macabu', 'Cordeiro', 'Duas Barras', 'Duque de Caxias', 'Engenheiro Paulo de Frontin', 'Guapimirim', 'Iguaba Grande', 'Itaboraí', 'Itaguaí', 'Italva', 'Itaocara', 'Itaperuna', 'Itatiaia', 'Japeri', 'Laje do Muriaé', 'Macaé', 'Macuco', 'Magé', 'Mangaratiba', 'Maricá', 'Mendes', 'Mesquita', 'Miguel Pereira', 'Miracema', 'Natividade', 'Nilópolis', 'Niterói', 'Nova Friburgo', 'Nova Iguaçu', 'Paracambi', 'Paraíba do Sul', 'Paraty', 'Paty do Alferes', 'Petrópolis', 'Pinheiral', 'Piraí', 'Porciúncula', 'Porto Real', 'Quatis', 'Queimados', 'Quissamã', 'Resende', 'Rio Bonito', 'Rio Claro', 'Rio das Flores', 'Rio das Ostras', 'Rio de Janeiro', 'Santa
```

```
Maria Madalena', 'Santo Antônio de Pádua', 'São Fidélis', 'São Francisco de Itabapoana', 'São Gonçalo', 'São João da Barra', 'São João de Meriti', 'São José de Ubá', 'São José do Vale do Rio Preto', 'São Pedro da Aldeia', 'São Sebastião do Alto', 'Sapucaia', 'Saquarema', 'Seropédica', 'Silva Jardim', 'Sumidouro', 'Tanguá', 'Teresópolis', 'Trajano de Moraes', 'Três Rios', 'Valença', 'Varre-Sai', 'Vassouras', 'Volta Redonda']


def coordenada_municipio(municipio, coordenada):
    municipios_georef = pd.read_csv('./dados_geo/municipios_georef.csv')
    return round(list(municipios_georef[municipios_georef.municipio==municipio][coordenada])[0], 4)


def descobrir_ema_com_municipio(nome_municipio):
    latitude_municipio = coordenada_municipio(nome_municipio, 'latitude')
    longitude_municipio = coordenada_municipio(nome_municipio, 'longitude')
    estacao = busca_estacao_usando_coordenadas(latitude_municipio, longitude_municipio)
    return estacao


def limpa_dados_gerados():
    plt.close('all')
    arquivos = glob.glob('imgs_plots/*')
    for arquivo in arquivos:
        os.remove(arquivo)
```

msg_inicial = "Olá! Eu sou o bot CLIMATER e forneço visualizações de dados climáticos e séries temporais.\n\nSou um projeto de TCC da pós-graduação PAATER da UFF, criado pelo engenheiro agrônomo Daniel Alvares, sob orientação do Dr. Márcio Cataldi.\n\nPara usar nossas funções, é necessário determinar um ponto de referência."