Paulo César Crepaldi

Manual Técnico da Plataforma Interativa Clima-Cast-Crepaldi

Itajubá - MG
2025

Paulo César Crepaldi

Manual Técnico da Plataforma Interativa Clima-Cast-Crepaldi

Universidade Federal de Itajubá – UNIFEI
Instituto de Recursos Naturais (IRN)

Docente: Prof. Dr. Enrique Vieira Mattos

Itajubá - MG
2025

Sumário

Sumário 2

Lista de ilustrações 4

Lista de tabelas 4

1 INTRODUÇÃO 5
1.1 Nome do Projeto Básico 6
1.2 Objetivo 6

2 DESCRIÇÃO DO PRODUTO 7
2.1 Arquitetura Geral do Sistema 7
2.2 Fontes de Dados e Datasets Utilizados 7
2.2.1 Perfis verticais para o módulo Skew-T 8
2.2.2 Dados vetoriais e geometrias de recorte 9
2.2.3 Perfis verticais para o módulo Skew-T 9
2.2.4 Bibliotecas e Tecnologias Empregadas 9
2.2.5 Funcionalidades Principais 10
2.3 Ambiente Computacional 11
2.4 Considerações sobre Desempenho 11
2.5 Resumo Técnico 12

3 ESTRUTURA DOS DIRETÓRIOS DO PROJETO 13
3.1 Organização Geral 13
3.2 Descrição dos Principais Arquivos e Módulos 13
3.2.1 main.py 13
3.2.2 gee_handler.py 14
3.2.3 map_visualizer.py 14
3.2.4 charts_visualizer.py 14
3.2.4.1 shapefile_handler.py 15
3.2.4.2 skewt_handler.py 15
3.2.4.3 skewt_visualizer.py 15
3.2.5 ui.py 16
3.3 Arquivos Auxiliares 16
3.4 Estrutura Modular e Interdependência 16

4 ORGANIZAÇÃO DO CÓDIGO E *DEPLOY* DO PRODUTO 19
4.1 Organização Lógica do Código 19

4.2 Controle de Versões e *GitHub* . . . 20
4.3 Autenticação e Configuração do Google Earth Engine . . . 20
4.4 *Deploy* no *Streamlit Cloud* . . . 22
4.5 Resultado do *Deploy* . . . 25
4.6 Arquivo de Dependências (`requirements.txt`) . . . 26
4.7 Monitoramento durante o *deploy* . . . 27
4.8 Disponibilidade do Código-Fonte . . . 27

5 TESTES E VALIDAÇÃO . . . 28
5.1 Testes Funcionais . . . 28
5.1.1 Testes de Desempenho . . . 31
5.2 Testes de Compatibilidade e Portabilidade . . . 32
5.3 Limitações Operacionais . . . 32
5.3.1 Limitações associadas ao modo Shapefile . . . 32
5.3.2 Limitações associadas ao modo Skew-T . . . 33
5.3.3 Limitações associadas a modos comparativos (Múltiplos Mapas e Múltiplas Séries) 33
5.4 Considerações sobre Robustez . . . 34

6 CONSIDERAÇÕES FINAIS . . . 35

Bibliografia Recomendada . . . 36

Lista de ilustrações

Figura 1 – Fluxograma funcional da Plataforma Interativa Clima-Cast-Crepaldi. 17
Figura 2 – Tela de configuração de deploy no *Streamlit Cloud*, destacando a integração com o repositório *GitHub* e a definição dos parâmetros de execução. 23
Figura 3 – Interface de gerenciamento de Secrets no Streamlit Cloud, onde as credenciais da Service Account devem ser armazenadas em formato TOML. 24
Figura 4 – Exemplo de tela inicial após o *deploy* no *Streamlit Cloud*. 25
Figura 5 – Console de logs do *Streamlit Cloud* detalhando a inicialização do ambiente, clonagem do repositório e instalação de pacotes do sistema. 27

Lista de tabelas

Tabela 1 – Resumo técnico da Plataforma Interativa Clima-Cast-Crepaldi. 12
Tabela 2 – Cenários recomendados para testes de desempenho. Preencher os tempos com as medições da versão atual. 31

1 Introdução

O monitoramento e a análise de variáveis meteorológicas tornaram-se ferramentas importantes para o entendimento dos processos atmosféricos e para o apoio à tomada de decisão em diferentes setores, como agricultura, energia e gestão de riscos ambientais. Considerando-se as ciências atmosféricas, a crescente disponibilidade de dados de reanálises globais e o avanço de plataformas computacionais em nuvem — como o *Google Earth Engine (GEE)* — possibilitaram o desenvolvimento de plataformas interativas de visualização e análise em tempo real, com baixo custo computacional e grande potencial para aplicações didáticas.

A Plataforma Interativa Clima-Cast-Crepaldi foi desenvolvida nesse cenário, com o objetivo de integrar dados meteorológicos provenientes do ERA5-Land, uma reanálise de alta resolução mantida pelo *European Centre for Medium-Range Weather Forecasts (ECMWF)*, em um ambiente web dinâmico construído com *Streamlit*. A plataforma oferece acesso intuitivo a mapas e séries temporais de algumas variáveis climáticas (Temperatura do ar, Precipitação, Vento, Umidade Relativa, Radiação Solar, Temperatura do ponto de Orvalho, Temperatura da Superfície e Umidade do solo em diferentes camadas), permitindo análises espaciais e temporais em diferentes escalas — estadual, municipal, por polígono ou círculo — sem a necessidade de processamento local.

O desenvolvimento desta plataforma está vinculado à disciplina *CAT314 – Ferramentas de Previsão de Curtíssimo Prazo (Nowcasting)*, ministrada pelo Instituto de Recursos Naturais (IRN) da Universidade Federal de Itajubá (UNIFEI), sob responsabilidade do Prof. Dr. Enrique Vieira Mattos. O projeto consolida o aprendizado em integração de dados geoespaciais, reanálises climáticas e técnicas de programação aplicadas à meteorologia.

A plataforma foi concebida como uma ferramenta prática de apoio a estudos meteorológicos, fornecendo ao usuário:

- acesso simplificado a dados atmosféricos diários do ERA5-Land;
- geração automática de mapas interativos e estáticos com escalas ajustadas à variável sob análise;
- séries temporais com agregação diária, incluindo informações estatísticas de valores médios, mínimos, máximos, amplitude e desvio padrão;
- interface compatível com o *Google Earth Engine*, permitindo consultas diretas aos seus *datasets*.

A Plataforma Interativa Clima-Cast-Crepaldi representa, portanto, uma aplicação de integração entre a ciência atmosférica e o desenvolvimento computacional, promovendo o uso de tecnologias de código aberto e de acesso aberto, como o *Streamlit* e o *Google Earth Engine*, respectivamente.

1.1 Nome do Projeto Básico

Clima-Cast-Crepaldi
Plataforma Interativa para Análise Meteorológica com Integração ao Google Earth Engine

1.2 Objetivo

Desenvolver uma plataforma web interativa em Python, utilizando o *framework Streamlit*, capaz de integrar e visualizar dados meteorológicos provenientes do ERA5-Land e de arquivos de dados geofísicos hospedadas no *Google Earth Engine*. O objetivo principal é fornecer ferramentas de fácil acesso e manipulação para a análise espacial e temporal de variáveis climáticas.

2 Descrição do Produto

Desenvolvida em linguagem Python e publicada no *framework Streamlit Cloud*, permite a execução remota de consultas, processamento e visualização de dados.

2.1 Arquitetura Geral do Sistema

A arquitetura da plataforma foi projetada segundo uma lógica modular, garantindo, assim, independência entre as principais funções e facilidade de manutenção. Cada módulo executa uma etapa específica do fluxo de dados — desde a autenticação no *Google Earth Engine* até a renderização de mapas e gráficos no ambiente web.

O fluxo operacional é estruturado segundo as principais funções:

1. **Interface e Navegação:** desenvolvida com o *framework Streamlit*, responsável pela exibição dos componentes interativos (mapas, gráficos, menus e filtros).
2. **Processamento e Controle:** contém as funções de manipulação de dados e execução das operações no GEE e no *ERA5-Land*.
3. **Comunicação com o GEE:** responsável por autenticação, requisições e extração de dados do *Google Earth Engine*.
4. **Visualização Geoespacial:** encarregada da conversão dos dados processados em representações visuais interativas, por meio das bibliotecas *geemap* e *folium*.
5. **Exportação e Armazenamento:** destinada à geração de arquivos de saída como, por exemplo, as extensões .PNG e .CSV

2.2 Fontes de Dados e Datasets Utilizados

A principal fonte de dados da plataforma é o conjunto de reanálises ERA5-Land, disponibilizado pelo *European Centre for Medium-Range Weather Forecasts (ECMWF)* e acessado por meio do dataset `ECMWF/ERA5_LAND/DAILY_AGGR`. Esse *dataset* oferece dados diários agregados, eliminando a necessidade de download de dados horários, o que reduz o tempo de execução e o custo computacional.

As variáveis atmosféricas disponíveis atualmente incluem:

- Temperatura média do ar a 2 m (`temperature_2m`);
- Temperatura máxima e mínima diária do ar a 2 m (`temperature_2m_max`, `temperature_2m_min`);
- Temperatura do ponto de orvalho a 2 m (`dewpoint_temperature_2m`);

- Temperatura da superfície ou "Skin Temperature"(`skin_temperature`);
- Precipitação total (`total_precipitation`);
- Radiação solar líquida na superfície (`surface_net_solar_radiation`);
- Componentes U e V do vento a 10 m (`u_component_of_wind_10m`, `v_component_of_wind_10m`);
- Umidade volumétrica do solo em 4 camadas (`volumetric_soil_water_layer_1` a `layer_4`);
- Umidade relativa*.

* A umidade relativa do ar (RH, em %) foi calculada a partir da temperatura do ar a 2 m (T) e da temperatura do ponto de orvalho a 2 m (T_d), ambas fornecidas pelo ERA5-Land em Kelvin. Converte-se inicialmente para Celsius, $T_C = T - 273.15$ e $T_{dC} = T_d - 273.15$. Em seguida, computam-se a pressão de vapor de saturação $e_s(T_C)$ e a pressão de vapor $e(T_{dC})$ por meio da forma de Magnus–Tetens, com constantes $A = 17.67$ e $B = 243.5$ °C:

$$e_s(T_C) = 6.112 \exp\!\left(\frac{A\,T_C}{B + T_C}\right), \qquad e(T_{dC}) = 6.112 \exp\!\left(\frac{A\,T_{dC}}{B + T_{dC}}\right) \ \text{[hPa]}.$$

a umidade relativa resulta então de

$$RH = 100 \times \frac{e(T_{dC})}{e_s(T_C)} = 100 \times \exp\!\left(\frac{A\,T_{dC}}{B + T_{dC}} - \frac{A\,T_C}{B + T_C}\right),$$

limitando-se o valor final ao intervalo físico de 100%.

Esses dados são consultados diretamente do GEE, processados em tempo real e representados graficamente no navegador do usuário.

2.2.1 Perfis verticais para o módulo Skew-T

Além dos dados de superfície e camadas de solo obtidos via Google Earth Engine, a plataforma inclui um módulo de perfil vertical (Skew-T) que consulta dados por níveis de pressão em um serviço externo de meteorologia. O objetivo é obter um conjunto mínimo de variáveis termodinâmicas e do vento ao longo da coluna atmosférica, permitindo a geração do diagrama Skew-T e o cálculo de índices de instabilidade.

O perfil vertical é retornado em níveis padrão de pressão (hPa), tipicamente entre 1000 e 100 hPa, incluindo temperatura e umidade relativa, bem como vento (velocidade e direção), posteriormente convertido em componentes zonal (u) e meridional (v). Por se tratar de uma fonte distinta do ERA5-Land, os resultados do Skew-T devem ser interpretados como uma aproximação de perfil atmosférico modelado para o ponto selecionado.

Observa-se ainda que a disponibilidade de dados verticais pode estar limitada por regras do próprio provedor (por exemplo, data mínima de disponibilidade para níveis de pressão e janelas de retenção para previsões recentes versus previsões históricas).

2.2.2 Dados vetoriais e geometrias de recorte

As geometrias de recorte utilizadas nas análises espaciais e temporais podem ser definidas por: (i) limites administrativos (Estado e Município), (ii) geometrias desenhadas no mapa (Polígono), (iii) geometria circular (Lat/Lon/Raio) e (iv) Shapefile enviado pelo usuário.

No modo Shapefile, o usuário fornece um arquivo compactado (`.zip`) contendo os componentes do Shapefile (por exemplo, `.shp`, `.shx`, `.dbf`). A geometria é lida e padronizada para coordenadas geográficas (EPSG:4326) e pode ser simplificada para reduzir complexidade geométrica, antes de ser convertida para um objeto compatível com o Google Earth Engine, permitindo seu uso nas rotinas de recorte e extração de estatísticas.

2.2.3 Perfis verticais para o módulo Skew-T

Além dos dados de superfície e camadas de solo obtidos via Google Earth Engine, a plataforma inclui um módulo de perfil vertical (Skew-T) que consulta dados por níveis de pressão em um serviço externo de meteorologia. O objetivo é obter um conjunto mínimo de variáveis termodinâmicas e do vento ao longo da coluna atmosférica, permitindo a geração do diagrama Skew-T e o cálculo de índices de instabilidade.

O perfil vertical é retornado em níveis padrão de pressão (hPa), tipicamente entre 1000 e 100 hPa, incluindo temperatura e umidade relativa, bem como vento (velocidade e direção), posteriormente convertido em componentes zonal (u) e meridional (v). Por se tratar de uma fonte distinta do ERA5-Land, os resultados do Skew-T devem ser interpretados como uma aproximação de perfil atmosférico modelado para o ponto selecionado.

Observa-se ainda que a disponibilidade de dados verticais pode estar limitada por regras do próprio provedor (por exemplo, data mínima de disponibilidade para níveis de pressão e janelas de retenção para previsões recentes versus previsões históricas).

No modo Shapefile, o usuário fornece um arquivo compactado (`.zip`) contendo os componentes do Shapefile (por exemplo, `.shp`, `.shx`, `.dbf`). A geometria é lida e padronizada para coordenadas geográficas (EPSG:4326) e pode ser simplificada para reduzir complexidade geométrica, antes de ser convertida para um objeto compatível com o Google Earth Engine, permitindo seu uso nas rotinas de recorte e extração de estatísticas.

2.2.4 Bibliotecas e Tecnologias Empregadas

O desenvolvimento da plataforma utiliza um conjunto integrado de bibliotecas de código aberto, que viabilizam a manipulação de dados geoespaciais, visualização interativa, exportação de resultados e comunicação com APIs. As principais são:

- **Streamlit** — construção da interface web interativa e controle de navegação entre páginas;
- **Google Earth Engine API (earthengine-api)** — autenticação e acesso direto aos dados do Google Earth Engine;

- **geemap, folium e streamlit-folium** — visualização de mapas interativos, integração de camadas geográficas e renderização no ambiente Streamlit;
- **geobr** — obtenção de limites administrativos brasileiros (estados e municípios);
- **geopandas e pyproj** — leitura, validação e reprojeção de dados vetoriais (incluindo Shapefiles), além de suporte a transformações de coordenadas;
- **pandas e numpy** — manipulação, agregação e análise de séries temporais;
- **matplotlib e plotly** — criação de gráficos, séries temporais interativas e figuras para exportação;
- **MetPy** — geração do diagrama Skew-T, cálculo de índices termodinâmicos e manipulação de unidades físicas;
- **requests** — comunicação com serviços externos via HTTP (por exemplo, consulta de perfis verticais e serviços auxiliares);
- **openpyxl** — exportação de resultados tabulares para planilhas no formato `.xlsx`;
- **python-docx, pypandoc e pandoc** — leitura e conversão de documentos utilizados na seção "Sobre";
- **Bibliotecas padrão do Python (json, os, datetime, tempfile, zipfile, io, math, re)** — autenticação, gerenciamento de arquivos temporários, manipulação de arquivos compactados, controle temporal e rotinas auxiliares.

Essas bibliotecas foram escolhidas por sua estabilidade, compatibilidade com o ambiente Streamlit Cloud e ampla adoção em aplicações científicas e geoespaciais, permitindo que o aplicativo opere de forma segura, rápida e multiplataforma.

2.2.5 Funcionalidades Principais

A plataforma Clima-Cast-Crepaldi oferece modos de análise espacial, temporal e vertical, acessíveis por meio do menu lateral *(sidebar)*.

1. **Mapas**: gera mapas para uma única variável meteorológica (ERA5-Land/GEE), em uma área e intervalo de datas definidos pelo usuário.
2. **Múltiplos Mapas**: gera painéis estáticos para comparação simultânea de até quatro variáveis, permitindo análise comparativa (ex.: precipitação vs. umidade).
3. **Sobreposição (Camadas)**: compara duas variáveis no mesmo mapa por meio de transparência ou modo de "cortina" (*split map*), facilitando a análise de padrões espaciais correlatos.

4. **Shapefile**: permite o envio de um Shapefile compactado (.zip) para definição de uma geometria personalizada (ex.: bacia, fazenda, área de estudo). O arquivo é processado, reprojetado para EPSG:4326 e a geometria é simplificada antes de ser convertida para uso no Google Earth Engine, reduzindo risco de falhas por alta complexidade geométrica.

5. **Séries Temporais**: produz gráficos interativos de evolução temporal de uma variável meteorológica, para a mesma geometria definida na análise espacial.

6. **Múltiplas Séries**: plota séries temporais comparativas (até quatro variáveis), favorecendo identificação de correlações e defasagens temporais.

7. **Skew-T (Sondagem)**: gera perfis verticais (diagrama Skew-T) e índices termodinâmicos a partir de dados atmosféricos obtidos via Open-Meteo (GFS/Seamless e Historical Forecast). Observa-se que perfis verticais (pressão/níveis) estão disponíveis apenas a partir de 23/03/2021 (limitação do GFS).

8. **Sobre o Aplicativo**: apresenta informações institucionais, créditos de desenvolvimento e síntese da plataforma.

Exportação de resultados. As rotinas de exportação (tabelas e figuras) são disponibilizadas de forma integrada aos modos de análise, permitindo ao usuário salvar produtos gerados (ex.: gráficos e tabelas em formatos compatíveis) conforme as opções exibidas na interface.

2.3 Ambiente Computacional

O desenvolvimento e a execução local foram realizados em ambiente *Python 3.12* com IDE *Visual Studio Code*, e o *deploy* (Implantação de um aplicativo *Streamlit*: refere-se ao processo de tornar o código Python e a aplicação web gerada por ele acessível ao público ou a outros usuários pela internet). Foi efetuado em *Streamlit Cloud*, serviço gratuito de hospedagem e execução em nuvem. As dependências do projeto são especificadas no arquivo `requirements.txt`, garantindo reprodutibilidade e compatibilidade entre ambientes.

O uso da autenticação via *Service Account* do Google garante acesso seguro ao GEE e independência de tokens locais, sendo as chaves privadas armazenadas no arquivo `secrets.toml` do ambiente *Streamlit*.

2.4 Considerações sobre Desempenho

Para reduzir a latência de execução, o código emprega mecanismos de:

- **Cache inteligente** (`@st.cache_data`) para consultas repetidas ao GEE;
- **Limitação de área** (recortes espaciais) para acelerar a renderização de mapas;

- **Dados diários agregados** (dataset `DAILY_AGGR`) para evitar sobrecarga no carregamento de séries horárias.

Essas estratégias resultam em tempos médios de resposta inferiores a 10 segundos para séries temporais e a 20 segundos para mapas de médias mensais, conforme medições realizadas no ambiente de teste.

2.5 Resumo Técnico

A Tabela 1 sintetiza os principais componentes técnicos da plataforma.

Tabela 1 – Resumo técnico da Plataforma Interativa Clima-Cast-Crepaldi.

Componente	**Descrição Técnica**
Linguagem	Python 3.12
Framework Web	Streamlit 1.45
Fonte de dados (reanálise)	ERA5-Land (ECMWF) via Google Earth Engine (GEE)
Dataset no GEE	ECMWF/ERA5_LAND/DAILY_AGGR
Fonte de dados (perfil vertical)	Serviço externo para perfis por níveis de pressão (módulo Skew-T), com endpoints de previsão e histórico conforme disponibilidade do provedor
Geometrias de recorte	Estado, Município, Polígono (desenho no mapa), Círculo (Lat/Lon/Raio) e Shapefile (.zip)
Recursos principais	Mapas interativos e estáticos; múltiplos mapas (até 4 variáveis); sobreposição por camadas; séries temporais; múltiplas séries (até 4 variáveis); Skew-T
Principais bibliotecas	pandas, numpy, geemap, folium, geobr, geopandas, pyproj, plotly, matplotlib, MetPy, requests
Ambiente de deploy	Streamlit Cloud (integração com GitHub)
Autenticação	Service Account (secrets.toml) para acesso ao GEE
Formatos de saída	CSV, XLSX, PNG e JPG
Tempo médio de resposta	10 a 20 s (consultas ao GEE; varia com área, período e conectividade)

3 Estrutura dos Diretórios do Projeto

O código-fonte está organizado no repositório *GitHub* `dashboard_cat314`, o qual contém todos os módulos Python, dependências e arquivos de configuração necessários para execução local e em nuvem. A estrutura segue um padrão modular que separa as funções de interface, processamento, visualização e integração com o *Google Earth Engine*.

3.1 Organização Geral

A Listagem 3.1 apresenta a hierarquia dos diretórios e arquivos do projeto.

Listagem 3.1 – Estrutura de diretórios do repositório `dashboard_cat314`.

```
dashboard_cat314/
.devcontainer/
main.py
ui.py
gee_handler.py
map_visualizer.py
charts_visualizer.py
shapefile_handler.py
skewt_handler.py
skewt_visualizer.py
utils.py
logo.png
municipios_ibge.json
packages.txt
requirements.txt
runtime.txt
sobre.docx
```

3.2 Descrição dos Principais Arquivos e Módulos

A seguir descrevem-se as funções e responsabilidades de cada arquivo do projeto.

3.2.1 main.py

Arquivo de inicialização principal da plataforma. É responsável pelo ponto de entrada da, definição das configurações globais da página e orquestração do fluxo de dados.

Este script executa a função `run_full_analysis()`, que centraliza as chamadas aos demais módulos na seguinte ordem:

1. Configuração da página (`st.set_page_config`);

2. Renderização do menu lateral (via `ui.py`);
3. Autenticação e consulta de dados (via `gee_handler.py`);
4. Exibição condicional dos resultados (mapas ou gráficos) conforme a escolha do usuário.

3.2.2 `gee_handler.py`

Módulo responsável pela comunicação direta com o *Google Earth Engine*. Contém as funções de autenticação, requisição e extração de dados do *datset* `ECMWF/ERA5_LAND/DAILY_AGGR`. Também executa o pré-processamento das variáveis, recorte espacial e agregação temporal.

Funções principais:

- `inicializar_gee()` — autentica a plataforma com o GEE usando credenciais do tipo *Service Account*;
- `get_time_series_data()` — extrai séries temporais médias para a geometria definida pelo usuário;
- `get_image_collection()` — retorna imagens do ERA5-Land para o intervalo de datas especificado;
- `converter_para_pandas()` — converte os resultados em *DataFrame* para posterior plotagem.

3.2.3 `map_visualizer.py`

Responsável pela criação dos mapas interativos e estáticos de visualização das variáveis meteorológicas. Utiliza as bibliotecas *geemap*, *folium* e *matplotlib* para gerar camadas raster coloridas com legendas personalizadas.

Principais componentes:

- `display_map()` — gera o mapa principal de visualização;
- `gerar_colormap()` — cria a escala de cores (colorbar) com valores discretos;
- `st_folium()` — integra o mapa com a interface *Streamlit*.

3.2.4 `charts_visualizer.py`

Módulo encarregado de criar gráficos interativos de séries temporais. Utiliza *plotly* e *matplotlib* para gerar visualizações dinâmicas temporais.

Funções principais:

- `_create_chart_figure` — Cria a figura do gráfico de linha interativo com estilo detalhado.;

- `display_time_series_chart` — Exibe um gráfico de série temporal interativo e uma explicação de seus controles.

3.2.4.1 shapefile_handler.py

Módulo responsável pelo processamento de Shapefiles personalizados enviados pelo usuário. O arquivo recebido deve estar compactado (formato `.zip`) contendo os componentes do Shapefile. O módulo extrai o conteúdo em diretório temporário, valida a existência do arquivo `.shp`, realiza a leitura com *GeoPandas*, reprojeta para `EPSG:4326` quando necessário e aplica simplificação geométrica para reduzir a complexidade e evitar falhas de execução durante o recorte e processamento no Google Earth Engine. Ao final, converte a geometria para `ee.Geometry` (`Polygon` ou `MultiPolygon`) e cria um `ee.Feature` com rótulo genérico para uso na aplicação.

Função principal:

- `process_uploaded_shapefile(uploaded_file)` — retorna `(ee_geom, ee_feat)` para integração com as rotinas de recorte e visualização.

3.2.4.2 skewt_handler.py

Módulo responsável pela aquisição de perfis verticais por níveis de pressão para construção do diagrama Skew-T. Implementa a normalização de data e hora (UTC), a seleção automática do endpoint (previsão recente ou previsão histórica) e o cache temporário dos resultados para reduzir chamadas repetidas ao serviço externo. Os dados são retornados como *DataFrame* contendo níveis padrão de pressão e variáveis termodinâmicas e do vento, incluindo componentes zonal (u) e meridional (v) calculadas a partir de velocidade e direção do vento.

Função principal:

- `get_vertical_profile_data(lat, lon, date_obj, hour)` — retorna *DataFrame* com as colunas `pressure`, `temperature`, `relative_humidity`, `u_component`, `v_component`. Registra metadados em `df.attrs` (fonte e data efetivamente retornada).

3.2.4.3 skewt_visualizer.py

Módulo encarregado da renderização do diagrama Skew-T e da apresentação de índices termodinâmicos associados. A rotina realiza interpolação vertical para aumentar a resolução do perfil, converte as variáveis para unidades físicas compatíveis, plota temperatura e ponto de orvalho, e adiciona a visualização do vento por barbelas. O módulo também calcula e exibe índices como LCL, LFC (estimado por busca do primeiro nível acima do LCL em que a parcela torna-se mais quente que o ambiente), CAPE, CIN, EL, Lifted Index, K-Index e Água Precipitável, além de permitir a exportação do gráfico em formato `PNG`.

Função principal:

- `render_skewt_plot(df, lat, lon, date, hour)` — plota o diagrama e exibe os índices; requer a biblioteca `MetPy` instalada.

3.2.5 `ui.py`

Gerencia as páginas da interface do usuário, incluindo a renderização da página inicial, o módulo de mapas, as séries temporais e a página "Sobre". Possui estrutura baseada em funções *Streamlit* com uso de `st.tabs()`, `st.markdown()` e `st.columns()`.

3.2.6 Módulo de Suporte Temporal (`utils.py`)

O arquivo `utils.py` reúne as funções auxiliares empregadas em diferentes partes da plataforma, com foco na manipulação de períodos de análise definidos pelo usuário. O módulo implementa rotinas de conversão de nomes de meses em valores numéricos, gerenciamento de datas e delimitação de intervalos temporais de consulta.

A função principal, `get_date_range()`, retorna as datas de início e término da análise com base no tipo de período selecionado na interface interativa (*Personalizado*, *Mensal* ou *Anual*). Para o modo mensal, utiliza-se o dicionário `MESES_PARA_NUMEROS`, que mapeia os nomes dos meses em português para seus correspondentes numéricos (1 a 12), garantindo compatibilidade com o módulo `calendar` da biblioteca padrão do Python.

3.3 Arquivos Auxiliares

- `requirements.txt`: lista todas as dependências necessárias para execução da plataforma, incluindo versões exatas das bibliotecas Python.
- `secrets.toml`: arquivo de configuração privado (não incluído no *GitHub*) que contém as credenciais do *Service Account* do Google Earth Engine.
- `sobre.docx`: documentação simplificada da plataforma.

3.4 Estrutura Modular e Interdependência

A Figura 1 ilustra a relação entre os módulos principais e o fluxo de dados dentro da plataforma.

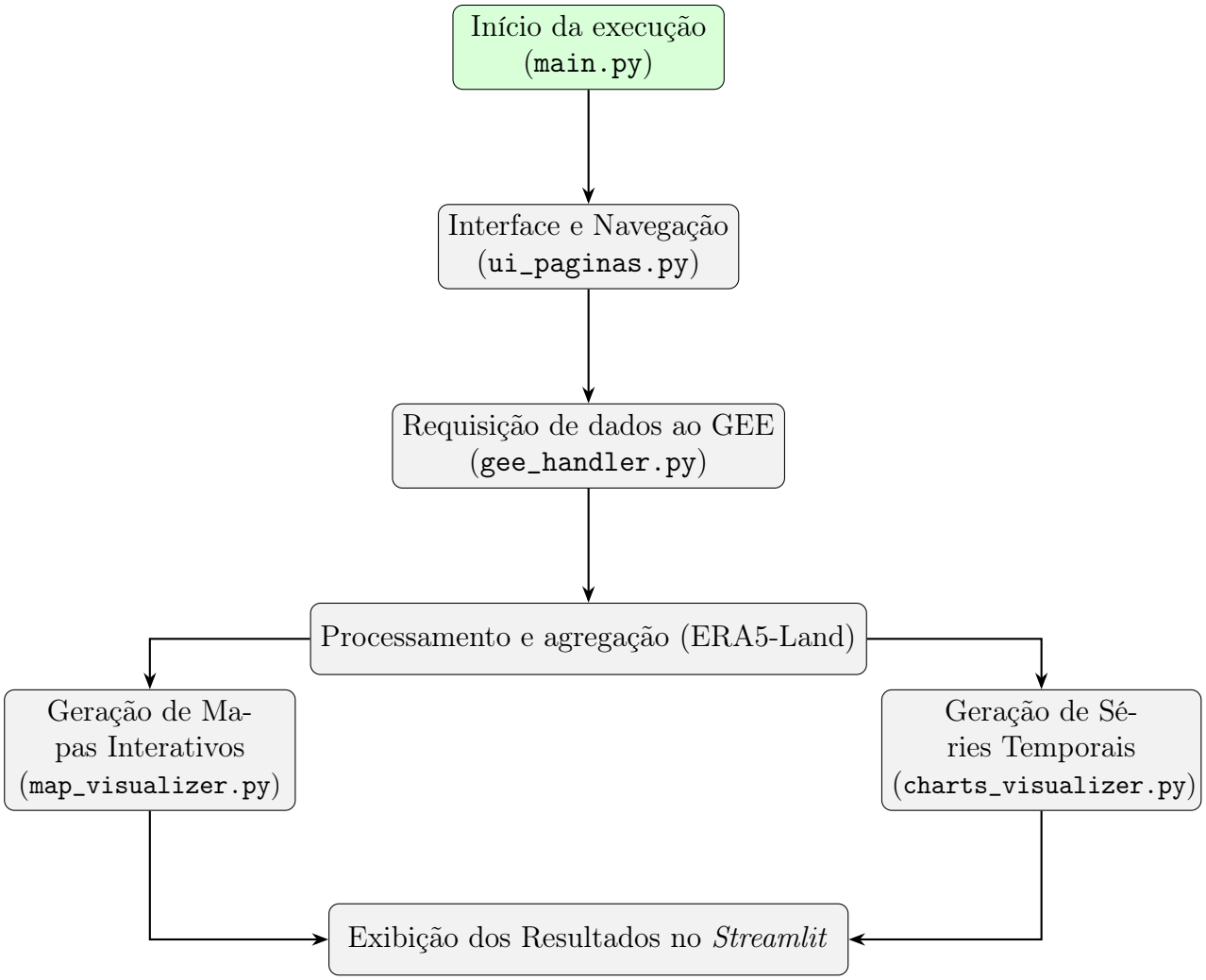

Figura 1 – Fluxograma funcional da Plataforma Interativa Clima-Cast-Crepaldi.

O diagrama demonstra que a comunicação entre os módulos segue um fluxo hierárquico e extensível:

1. **Coordenação:** o arquivo `main.py` inicializa a aplicação e coordena o fluxo geral de execução.
2. **Interface e controle de parâmetros:** o módulo `ui.py` centraliza a navegação entre os modos (Mapas, Múltiplos Mapas, Sobreposição/Camadas, Séries Temporais, Múltiplas Séries, Shapefile e Skew-T), além de coletar os parâmetros de consulta (variáveis, período e área de interesse).
3. **Construção da geometria de análise:** a área de interesse pode ser definida por Estado, Município, Polígono desenhado, Círculo (Lat/Lon/Raio) ou por Shapefile. No caso de Shapefile, o módulo `shapefile_handler.py` processa o arquivo enviado, padroniza a projeção e converte a geometria para uso no recorte espacial.
4. **Consultas e processamento no GEE (ERA5-Land):** o módulo `gee_handler.py` executa autenticação, recorte espacial e agregação temporal sobre o dataset, retornando produtos adequados para mapas e para séries temporais, conforme o modo selecionado.

5. **Visualização geoespacial:** o módulo `map_visualizer.py` renderiza mapas interativos e estáticos. Este módulo atende tanto o modo de mapa único quanto os modos de comparação (múltiplos mapas e camadas).

6. **Visualização temporal:** o módulo `charts_visualizer.py` gera séries temporais interativas e estatísticas associadas, incluindo o modo de comparação com múltiplas variáveis (até quatro séries no mesmo gráfico).

7. **Perfil vertical (Skew-T):** para o modo Skew-T, o módulo `skewt_handler.py` obtém o perfil por níveis de pressão e organiza as variáveis em estrutura tabular; em seguida, o módulo `skewt_visualizer.py` realiza a plotagem do diagrama e calcula índices termodinâmicos associados.

8. **Rotinas auxiliares:** o arquivo `utils.py` concentra funções de suporte (datas, intervalos, validações e utilidades) reutilizadas entre os módulos.

Essa estrutura modular mantém o acoplamento baixo entre componentes, favorecendo manutenção, depuração e evolução incremental do sistema (por exemplo, inclusão de novos modos de análise, novas variáveis ou novas fontes de dados).

4 Organização do Código e *Deploy* do Produto

A Plataforma foi desenvolvida com uma arquitetura modular, de modo que cada script desempenha uma função específica dentro do fluxo de execução. Essa organização facilita a manutenção, a depuração e a inserção de novas variáveis meteorológicas. O código foi versionado no *GitHub*, garantindo rastreabilidade e integração direta com o ambiente de *deploy Streamlit Cloud*.

4.1 Organização Lógica do Código

O processo de execução do aplicativo segue uma arquitetura modular, com dois fluxos operacionais principais: (i) o fluxo de análise espacial e temporal baseado no ERA5-Land via Google Earth Engine (GEE), utilizado para mapas e séries temporais; e (ii) o fluxo de perfil vertical (Skew-T), que consulta dados por níveis de pressão em um serviço externo para geração do diagrama e índices termodinâmicos.

A seguir, descreve-se o fluxo operacional de forma resumida:

1. O usuário acessa o aplicativo via navegador web.
2. O arquivo `main.py` é carregado e inicializa o ambiente Streamlit, definindo as configurações globais da página e importando os módulos necessários.
3. O módulo `ui.py` renderiza o menu lateral e captura os parâmetros definidos pelo usuário, incluindo o modo de análise (Mapas, Múltiplos Mapas, Camadas, Séries Temporais, Múltiplas Séries, Shapefile ou Skew-T), variáveis, período e a área de interesse.
4. **Fluxo ERA5-Land/GEE (mapas e séries):**
   a) O módulo `gee_handler.py` executa a autenticação e prepara a conexão com o Google Earth Engine.
   b) A geometria de análise é definida conforme a opção selecionada pelo usuário: Estado, Município, Polígono, Círculo ou Shapefile. No modo Shapefile, o módulo `shapefile_handler.py` processa o arquivo enviado, padroniza a geometria e a converte para um formato compatível com o recorte no GEE.
   c) O módulo `gee_handler.py` realiza as consultas ao ERA5-Land e executa o recorte espacial e a agregação temporal conforme o intervalo de datas selecionado, retornando os resultados em estruturas adequadas para visualização e exportação.
   d) Os resultados são enviados aos módulos de visualização:
      - `map_visualizer.py` — gera mapas interativos e estáticos, incluindo comparação por múltiplos mapas e sobreposição por camadas;

- `charts_visualizer.py` — produz séries temporais interativas, incluindo comparação com múltiplas séries (até quatro variáveis no mesmo gráfico).

5. **Fluxo Skew-T (perfil vertical):**

   a) O módulo `skewt_handler.py` consulta o perfil por níveis de pressão para a latitude/-longitude e data/hora selecionadas, organizando as variáveis em estrutura tabular.

   b) O módulo `skewt_visualizer.py` realiza a plotagem do diagrama Skew-T e calcula índices termodinâmicos associados, disponibilizando a figura para visualização e exportação.

6. O módulo `utils.py` provê funções auxiliares compartilhadas, como manipulação de datas e rotinas de suporte empregadas em diferentes partes do aplicativo.

7. Por fim, os resultados (mapas, séries temporais e/ou Skew-T) são exibidos na interface do Streamlit e podem ser exportados conforme as opções disponibilizadas em cada modo.

4.2 Controle de Versões e *GitHub*

O repositório do projeto, denominado `dashboard_cat314`, foi hospedado no *GitHub* para controle de versão. O *GitHub* permite a adição de novas versões dos códigos python, registro de commits e integração contínua com a nuvem.

A estrutura do repositório é composta pelos seguintes elementos principais:

- **Branch principal:** `main`;
- **Repositório:** <https://github.com/Crepaldi2025/dashboard_cat314>;
- **Ambiente de desenvolvimento:** *Visual Studio Code* (Python 3.12);
- **Integração:** *Streamlit Cloud* via *GitHub* (*Deploy* automático a cada commit).

A cada atualização do código local, o commit é sincronizado com o repositório remoto, permitindo o redeploy automático na nuvem sem necessidade de reconfiguração manual.

4.3 Autenticação e Configuração do Google Earth Engine

O acesso aos dados do GEE é realizado por meio de uma *Service Account*, criada dentro do console do *Google Cloud Platform (GCP)*. O par de chaves da conta de serviço é armazenado com segurança no arquivo `secrets.toml`, localizado no ambiente do *Streamlit Cloud*. Esse método elimina a necessidade de autenticação interativa, tornando o processo automatizado e reprodutível.

O arquivo `secrets.toml` contém o seguinte formato simplificado:

Listagem 4.1 – Estrutura do arquivo `secrets.toml` para autenticação com o GEE.

```
[earthengine_service_account]
type = "service_account"
project_id = "gee-crepaldi-2025b"
private_key_id = "(sua␣private␣Key␣id)"
private_key = """-----BEGIN␣PRIVATE␣KEY-----
(sua␣private␣Key)
-----END␣PRIVATE␣KEY-----"""
client_email = "gee-service@gee-crepaldi-2025b.iam.gserviceaccount.com"
client_id = "(seu␣Client␣id)"
auth_uri = "https://accounts.google.com/o/oauth2/auth"
token_uri = "https://oauth2.googleapis.com/token"
auth_provider_x509_cert_url = "https://www.googleapis.com/oauth2/v1/certs"
client_x509_cert_url = "https://www.googleapis.com/robot/v1/metadata/x509/
    ↪ gee-service%40gee-crepaldi-2025b.iam.gserviceaccount.com"
universe_domain = "googleapis.com"
```

A função `inicializar_gee()` (localizada em `gee_handler.py`) executa a autenticação automática:

Listagem 4.2 – Função de inicialização do GEE na plataforma.

```
def inicializar_gee():
    """Inicializa a API do Google Earth Engine."""
    try:
        ee.Image.constant(0).getInfo()
    except ee.EEException:
        if "earthengine_service_account" in st.secrets:
            service_account = st.secrets["earthengine_service_account"]["
                ↪ client_email"]
            private_key = st.secrets["earthengine_service_account"]["
                ↪ private_key"]
            credentials = ee.ServiceAccountCredentials(service_account,
                ↪ key_data=private_key)
            ee.Initialize(credentials)
            st.info("Conectado ao Google Earth Engine (Service Account).")
        else:
            ee.Initialize()
```

4.4 *Deploy* no *Streamlit Cloud*

A publicação do projeto foi realizada na plataforma *Streamlit Cloud*, que permite a execução de aplicações Python diretamente a partir de repositórios no *GitHub*, garantindo integração contínua, reprodutibilidade e acesso público. O processo completo de *deploy* do produto foi estruturado em três etapas principais:

1. preparação do repositório *GitHub* contendo o código-fonte, arquivos de configuração e dependências;
2. configuração de credenciais seguras do Google Earth Engine (GEE) utilizando uma *Service Account*;
3. criação e execução do *deploy* no ambiente do *Streamlit Cloud*.

A seguir descrevem-se todas as etapas necessárias para a publicação e manutenção do sistema na nuvem.

1. Criar conta no *Streamlit Cloud*

O primeiro passo consiste em acessar o serviço em:

`https://streamlit.io/`

e criar uma conta utilizando *GitHub* ou Google. O *Streamlit Cloud* integra-se nativamente ao *GitHub*, facilitando o *deploy* automático a partir do repositório do projeto.

2. Iniciar o processo de criação da aplicação

   Após autenticação, na página inicial do *Streamlit Cloud*, selecione o botão:

   `Create app`

   localizado no canto superior direito da interface da plataforma.

3. Selecionar a origem do código

   Na janela seguinte, escolha a opção:

   Deploy `a public app from` *GitHub*

   Esta opção permite vincular diretamente o repositório do projeto ao ambiente de *deploy*.

4. Configurar os parâmetros do *deploy*

   O usuário deverá preencher os seguintes campos:

   - **Repository:** selecionar o repositório *GitHub* contendo o código-fonte do projeto, por exemplo: `Crepaldi2025/dashboard_cat314`;
   - **Branch:** selecionar a *branch* principal do repositório, geralmente `main`;
   - **Main file path:** informar o arquivo principal da aplicação, neste projeto: `main.py`;
   - **App URL:** definir o endereço público que será utilizado para acessar o dashboard.

Esse fluxo garante a reprodutibilidade científica, a rastreabilidade das versões e a atualização automática do Clima-Cast-Crepaldi sempre que o código é aprimorado.

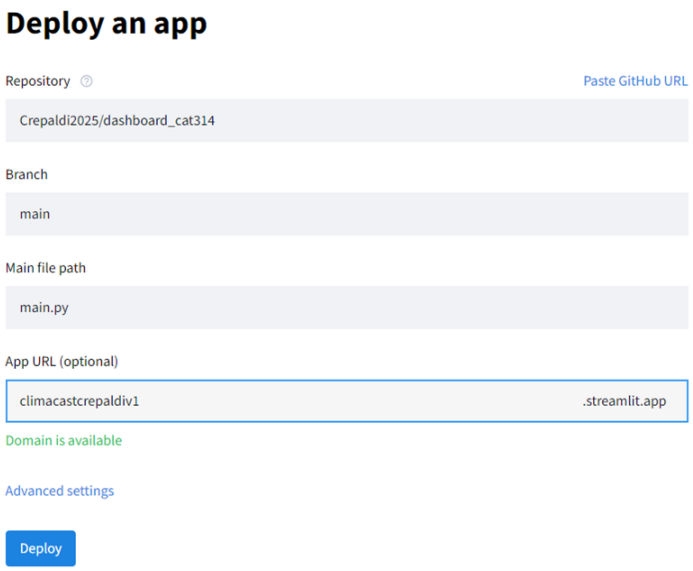

Figura 2 – Tela de configuração de deploy no *Streamlit Cloud*, destacando a integração com o repositório *GitHub* e a definição dos parâmetros de execução.

5. Inserir credenciais do Google Earth Engine

Como a plataforma utiliza dados do ERA5-Land acessados diretamente via Google Earth Engine, é necessário configurar o arquivo `secrets.toml` no ambiente do *Streamlit Cloud*.

No painel do aplicativo, acesse:

`Settings` → `Secrets`

e insira as credenciais no formato apresentado na listagem 4.1

Esse procedimento garante:

- autenticação segura e automatizada no GEE;
- funcionamento pleno do aplicativo na nuvem;
- independência de autenticação manual no navegador.

Para garantir a segurança das chaves de acesso, o *Streamlit Cloud* não utiliza o arquivo local. Em vez disso, o conteúdo deve ser inserido no gerenciador de segredos da plataforma, conforme demonstrado na Figura 3.

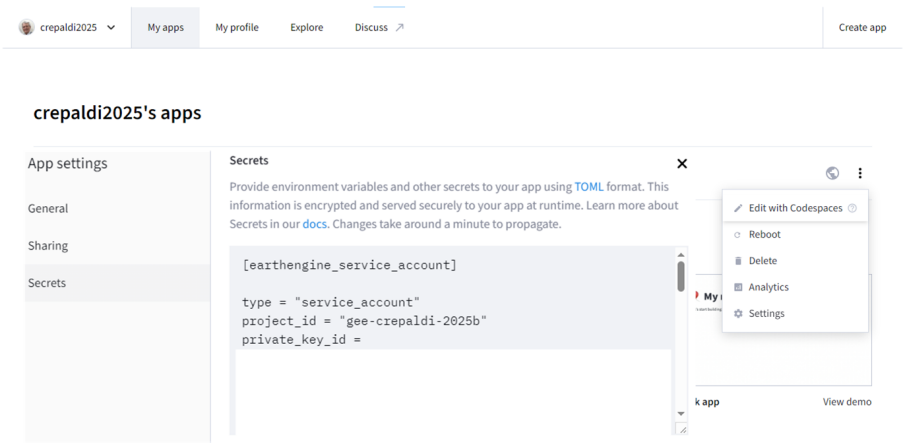

Figura 3 – Interface de gerenciamento de Secrets no Streamlit Cloud, onde as credenciais da Service Account devem ser armazenadas em formato TOML.

6. Executar o *deploy*

Após concluir todas as configurações, clique no botão:

Deploy

O *Streamlit Cloud* executará automaticamente:

1. instalação das dependências definidas no arquivo `requirements.txt`;
2. inicialização da aplicação e carregamento dos módulos Python;

3. verificação da comunicação com o Google Earth Engine;

4. publicação pública do endereço final da aplicação.

A Figura 2 ilustra a interface de configuração do *Streamlit Cloud*. Nela, observa-se o preenchimento dos campos essenciais: a seleção do repositório *GitHub* conectado, a definição da *branch* de produção e o caminho para o arquivo principal da aplicação.

7. Integração Contínua (CI/CD)

A integração contínua (CI) e o *deploy* contínuo (CD) são práticas recomendadas em projetos científicos e computacionais que automatizam a atualização e republicação do aplicativo. O *Streamlit Cloud* implementa automaticamente esse mecanismo. Assim, a cada novo *commit* enviado ao *GitHub*:

- o serviço reconstrói o ambiente da aplicação;
- instala novamente as dependências;
- reinicia o aplicativo;
- publica a nova versão sem necessidade de intervenção manual.

4.5 Resultado do *Deploy*

A Figura 4 mostra a tela inicial da plataforma hospedada na nuvem, com o menu lateral de variáveis, período de análise e seleção da área de interesse.

Figura 4 – Exemplo de tela inicial após o *deploy* no *Streamlit Cloud*.

4.6 Arquivo de Dependências (`requirements.txt`)

O arquivo `requirements.txt` contém a lista de bibliotecas e suas versões exatas, garantindo compatibilidade entre o ambiente local e a plataforma em nuvem. A versão abaixo foi ajustada para o ambiente Python 3.12, assegurando estabilidade com o *Streamlit Cloud* e integração com o *Google Earth Engine*.

Listagem 4.3 – Conteúdo do arquivo `requirements.txt`.

```
# ======================================
# Dependencias principais
# ======================================
numpy==1.26.4
pandas==2.2.3
matplotlib==3.8.4
plotly==5.23.0

# --------------------------------------
# Frameworks de interface e visualizacao
# --------------------------------------
streamlit==1.45.1
streamlit-folium==0.22.0
folium==0.17.0
geemap==0.35.3

# --------------------------------------
# Geoprocessamento e acesso aos dados
# --------------------------------------
geopandas==1.0.1
earthengine-api==1.5.11
geobr==0.2.2
pyproj==3.6.1

# --------------------------------------
# Termodinamica e Skew-T
# --------------------------------------
metpy==1.6.3

# --------------------------------------
# Entrada, saida e documentacao
# --------------------------------------
openpyxl==3.1.5
python-docx==1.2.0
pypandoc==1.15
pandoc==2.4
requests==2.32.5
setuptools==75.1.0
```

A definição precisa das versões permite que o *deploy* ocorra de forma reprodutível, evi-

tando conflitos entre dependências no ambiente de execução. A atualização de bibliotecas como `geemap` e `earthengine-api` foi realizada para garantir total compatibilidade com o *Google Earth Engine* e com o sistema de autenticação via `Service Account`.

Observação: o modo **Skew-T** requer a biblioteca `MetPy` para geração do diagrama e cálculo de índices termodinâmicos. Caso essa dependência não esteja instalada ou falhe no ambiente de execução, o módulo Skew-T não poderá ser renderizado, mas os demais modos do aplicativo (mapas e séries temporais via GEE/ERA5-Land) permanecem operacionais. Recomenda-se validar a instalação das dependências a partir do `requirements.txt` e realizar um teste mínimo do Skew-T após o deploy.

4.7 Monitoramento durante o *deploy*

O desempenho da aplicação foi monitorada durante o *deploy* utilizando o próprio log do *Streamlit Cloud*. O tempo médio de inicialização após o *deploy* é de aproximadamente 15 segundos.

A Figura 5 exibe o console de registros (*logs*) durante o processo de inicialização do container na nuvem. A imagem comprova a integração contínua com o GitHub, evidenciada pelas etapas de clonagem do repositório e, principalmente, a instalação automatizada das dependências definidas no arquivo `requirements.txt`.

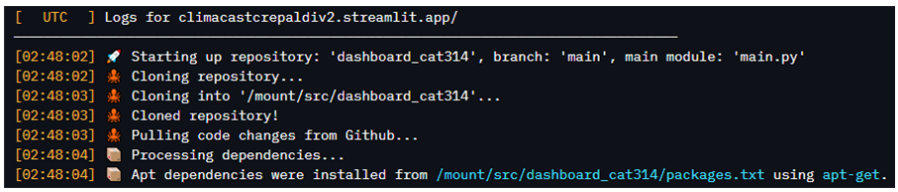

Figura 5 – Console de logs do *Streamlit Cloud* detalhando a inicialização do ambiente, clonagem do repositório e instalação de pacotes do sistema.

4.8 Disponibilidade do Código-Fonte

Os códigos-fonte completos da **Plataforma Interativa Clima-Cast-Crepaldi** estão disponíveis publicamente no repositório *GitHub*:

<https://github.com/Crepaldi2025/dashboard_cat314>

O repositório contém todos os módulos Python utilizados na aplicação, bem como arquivos de configuração e dependências.

5 Testes e Validação

A etapa de testes e validação teve como objetivo verificar a integridade funcional da plataforma, o desempenho no ambiente em nuvem e a consistência das informações obtidas a partir do *Google Earth Engine (GEE)* e da base de dados *ERA5-Land*. Foram aplicados testes unitários nos módulos individuais, testes integrados de comunicação entre funções e testes de desempenho no *Streamlit Cloud*.

5.1 Testes Funcionais

Os testes funcionais verificam se cada modo do aplicativo executa o fluxo completo esperado (seleção de parâmetros, aquisição de dados, geração do produto visual e exportação), incluindo tratamento de exceções e mensagens ao usuário. Recomenda-se executar os testes com áreas pequenas e períodos curtos inicialmente, aumentando progressivamente a complexidade.

T1 — Inicialização e navegação

Procedimento:

- Abrir o aplicativo e verificar carregamento da página inicial.
- Alternar entre as opções do menu lateral (Mapas, Múltiplos Mapas, Camadas, Séries, Múltiplas Séries, Shapefile e Skew-T).

Critério de aceitação: a navegação ocorre sem erros, mantendo estado mínimo necessário e exibindo os controles do modo selecionado.

T2 — Mapas (modo simples)

Procedimento:

- Selecionar uma área de interesse e um período curto.
- Escolher uma variável e gerar o mapa.

Critério de aceitação: o mapa é exibido corretamente e os controles de visualização funcionam (zoom, arraste, camadas base quando aplicável).

T3 — Múltiplos Mapas (até 4 variáveis)

Procedimento:

- Selecionar 2, 3 e 4 variáveis no modo **Múltiplos Mapas**.

- Gerar o painel e verificar consistência espacial (mesma área e período para todos os mapas).

Critério de aceitação: o painel é gerado sem sobreposição indevida de elementos e cada mapa corresponde à variável selecionada.

T4 — Sobreposição (Camadas)

Procedimento:

- Selecionar duas variáveis no modo **Camadas**.
- Ajustar transparência e/ou controle de comparação (quando disponível).

Critério de aceitação: as duas camadas são exibidas corretamente e o controle de comparação responde em tempo real.

T5 — Séries Temporais (modo simples)

Procedimento:

- Selecionar uma área de interesse e um período.
- Escolher uma variável e gerar a série temporal.

Critério de aceitação: o gráfico é exibido e permite inspeção dos valores (interação do cursor), com estatísticas calculadas quando configuradas.

T6 — Múltiplas Séries (até 4 variáveis)

Procedimento:

- Selecionar 2, 3 e 4 variáveis no modo **Múltiplas Séries**.
- Verificar legibilidade (legenda, escalas e identificação das curvas).

Critério de aceitação: todas as séries são plotadas no mesmo gráfico, correspondendo às variáveis selecionadas, sem falhas de renderização.

T7 — Shapefile (geometria personalizada)

Procedimento:

- Enviar um arquivo `.zip` contendo um Shapefile válido e selecionar o modo Shapefile como área de interesse.
- Gerar um mapa e uma série temporal utilizando essa geometria.
- Executar testes de robustez:

  - **Arquivo inválido:** `.zip` sem `.shp`;
  - **Geometria complexa:** arquivo com muitos vértices;
  - **Multiparte:** `MultiPolygon` (mais de um polígono).

Critério de aceitação: Shapefile válido resulta em recorte funcional; arquivos inválidos geram mensagens claras; geometrias complexas não causam travamento e permanecem utilizáveis.

T8 — Skew-T (perfil vertical)

Procedimento:

- Informar latitude/longitude e data/hora (UTC) e gerar o diagrama Skew-T.
- Verificar a presença de: temperatura, ponto de orvalho e vento (barbelas).
- Testar datas/horas fora da disponibilidade do provedor e indisponibilidade temporária de rede.

Critério de aceitação: o diagrama é gerado quando há dados disponíveis; quando não houver, o aplicativo informa claramente a limitação e não interrompe a execução geral.

T9 — Exportação de resultados

Procedimento:

- Exportar produtos gerados em cada modo:
  - Mapas: `PNG`/`JPG`;
  - Séries: `CSV`/`XLSX` (quando disponível) e imagem do gráfico;
  - Skew-T: `PNG`.

Critério de aceitação: arquivos exportados abrem corretamente e refletem o conteúdo exibido na tela.

T10 — Mensagens, validações e falhas controladas

Procedimento:

- Executar o aplicativo com parâmetros incompletos (sem área, sem variável, sem período).
- Simular falhas comuns: ausência de credenciais, falha de conexão com serviços e timeouts.

Critério de aceitação: o aplicativo bloqueia ações inválidas com mensagens claras e trata falhas externas sem encerrar a sessão do usuário.

5.1.1 Testes de Desempenho

Os testes de desempenho avaliam latência de resposta, consumo de memória e estabilidade de execução no ambiente de nuvem, contemplando os modos tradicionais (mapas e séries) e as novas funcionalidades (múltiplos mapas, múltiplas séries, Shapefile e Skew-T). Recomenda-se executar as medições com rede estável e registrar os resultados em ao menos três execuções por cenário, reportando média e desvio padrão.

Métricas observadas

- **Tempo de inicialização:** tempo desde o carregamento da página até a interface estar pronta para interação.
- **Tempo de geração:** tempo entre acionar a execução (botão/ação) e a exibição completa do produto (mapa, painel, gráfico ou Skew-T).
- **Uso de memória:** pico aproximado durante a execução dos cenários mais pesados.
- **Efeito do cache:** comparação entre a primeira execução e execuções repetidas com os mesmos parâmetros.

Cenários de teste

Tabela 2 – Cenários recomendados para testes de desempenho. Preencher os tempos com as medições da versão atual.

Cenário	**Descrição**	**Métrica**	**Resultado**
D1	Inicialização do app	Tempo (s)	_____
D2	Mapa simples (1 variável)	Tempo (s)	_____
D3	Múltiplos mapas (2 a 4 variáveis)	Tempo (s)	_____
D4	Sobreposição (2 camadas)	Tempo (s)	_____
D5	Série temporal (1 variável)	Tempo (s)	_____
D6	Múltiplas séries (2 a 4 variáveis)	Tempo (s)	_____
D7	Shapefile (pequeno, poucos vértices)	Tempo (s)	_____
D8	Shapefile (complexo, muitos vértices)	Tempo (s) + estabilidade	_____
D9	Skew-T (perfil vertical)	Tempo (s)	_____
D10	Cache (repetir D2, D5 e D9)	Redução (%)	_____

Procedimento de medição

1. Executar o aplicativo no Streamlit Cloud e abrir o console de logs para acompanhar o comportamento geral.

2. Para cada cenário (D1 a D9), realizar três execuções independentes, registrando os tempos observados.

3. Repetir os cenários D2, D5 e D9 com parâmetros idênticos, para quantificar o ganho do cache.

4. Para Shapefile, executar separadamente com um arquivo simples e outro com alta complexidade geométrica, verificando se há travamentos, timeouts ou falhas de renderização.

Critérios de aceitação (metas operacionais)

- O aplicativo deve permanecer responsivo durante a geração de mapas, séries e Skew-T, sem travamentos.
- O modo Shapefile deve processar arquivos simples de forma estável; para geometrias complexas, deve haver degradação controlada (por exemplo, maior latência), sem interromper a aplicação.
- O cache deve reduzir de forma perceptível o tempo em consultas repetidas com os mesmos parâmetros.

A utilização do cache interno (`@st.cache_data`) reduziu em cerca de 70 % o tempo de processamento em consultas repetidas. O consumo de memória durante as operações manteve-se inferior a 250 MB, dentro do limite imposto pelo *Streamlit Cloud*.

Os valores demonstram alta consistência estatística, validando a confiabilidade da base ERA5-Land para fins acadêmicos e de pesquisa.

5.2 Testes de Compatibilidade e Portabilidade

A aplicação foi testada nos sistemas operacionais Windows 10 e Windows 11, com navegadores Edge e Firefox. Em todos os casos, a renderização dos mapas e das séries temporais ocorreu sem falhas. O *deploy* em diferentes resoluções (1366×768 a 1920×1080) manteve o layout estável, demonstrando a responsividade da interface.

5.3 Limitações Operacionais

Esta seção registra limitações técnicas observáveis na versão atual do aplicativo, com o objetivo de orientar manutenção, operação e evolução da plataforma.

5.3.1 Limitações associadas ao modo Shapefile

- **Complexidade geométrica:** geometrias com muitos vértices, polígonos muito detalhados ou arquivos com múltiplas feições podem aumentar significativamente o tempo de processamento, elevar o consumo de memória e ocasionar falhas por timeout. Por esse

motivo, a aplicação pode aplicar simplificação geométrica antes do uso no recorte e na agregação no GEE.

- **Validação do arquivo:** o modo Shapefile requer um arquivo compactado (`.zip`) contendo os componentes mínimos do Shapefile. Arquivos incompletos ou corrompidos devem resultar em mensagens de erro e interrupção apenas do fluxo do Shapefile, sem afetar outros modos.
- **Sistema de referência de coordenadas:** Shapefiles em projeções distintas são reprojetados para coordenadas geográficas (EPSG:4326). Entretanto, inconsistências de metadados de projeção podem exigir correção do arquivo de origem.

5.3.2 Limitações associadas ao modo Skew-T

- **Fonte externa:** o Skew-T utiliza dados por níveis de pressão obtidos de um serviço externo (via HTTP). Assim, o modo pode sofrer degradação temporária por indisponibilidade do provedor, instabilidade de rede ou limites de taxa de requisição.
- **Disponibilidade temporal:** perfis verticais podem não estar disponíveis para todo o histórico de datas. O aplicativo deve informar ao usuário quando uma data/hora não puder ser atendida, evitando falhas não tratadas.
- **Compatibilidade e dependências:** o Skew-T depende da biblioteca `MetPy`. Falhas de instalação ou incompatibilidades de versão podem impedir a renderização do diagrama, sem impactar os modos baseados em GEE.
- **Interpretação física:** o perfil retornado deve ser interpretado como uma aproximação modelada para o ponto e horário selecionados. Valores e índices podem diferir de radiossondagens observacionais e de reanálises completas, especialmente em condições convectivas locais.

5.3.3 Limitações associadas a modos comparativos (Múltiplos Mapas e Múltiplas Séries)

- **Comparabilidade entre variáveis:** variáveis distintas podem possuir escalas e unidades diferentes; em gráficos comparativos, recomenda-se validar legibilidade e evitar interpretações indevidas quando múltiplas grandezas são exibidas simultaneamente.
- **Custo computacional:** selecionar até quatro variáveis em modos comparativos tende a aumentar o tempo de processamento e renderização, especialmente para áreas grandes e períodos longos.

5.4 Considerações sobre Robustez

Os resultados dos testes indicam que a plataforma é robusta, apresentando:

- desempenho estável e tempos de resposta adequados;
- comunicação consistente com o GEE;
- ausência de erros críticos ou travamentos;
- compatibilidade multiplataforma.

Esses resultados asseguram que a **Plataforma Interativa Clima-Cast-Crepaldi** atende plenamente aos requisitos de desempenho, confiabilidade e usabilidade definidos no projeto.

6 Considerações Finais

O presente documento apresentou o desenvolvimento completo da Plataforma Interativa Clima-Cast-Crepaldi, desde a concepção e arquitetura até a validação funcional e a publicação em ambiente de nuvem. O projeto foi idealizado com o propósito de integrar dados meteorológicos provenientes do *ERA5-Land* e do *Google Earth Engine (GEE)* em um ambiente interativo, gratuito e de fácil utilização, contribuindo para a disseminação de informações climáticas e o apoio a atividades de ensino e pesquisa.

A implementação em *Python*, utilizando o *framework Streamlit*, possibilitou o desenvolvimento de uma aplicação leve e modular, com visualização interativa de mapas, séries temporais de variáveis meteorológicas. A integração com o GEE permitiu o acesso direto a reanálises de alta resolução espacial, eliminando a necessidade de download local de dados e reduzindo significativamente o tempo de processamento.

O conjunto de testes realizados confirmou a confiabilidade e a robustez da plataforma. Os tempos médios de resposta permaneceram dentro dos limites aceitáveis. A interface, por sua vez, mostrou-se responsiva e funcional em diferentes navegadores, consolidando o caráter multiplataforma da aplicação.

A Plataforma Clima-Cast-Crepaldi representa uma contribuição acadêmica ao integrar ciência atmosférica, programação científica e computação em nuvem. Sua modularidade permite expansões futuras, tais como:

- inclusão de novas variáveis meteorológicas e índices derivados;
- integração com outros modelos numéricos;
- uso de algoritmos de *machine learning* para detecção de anomalias;
- criação de um painel de comparação entre observações de estações locais e reanálises;
- implementação de um módulo de exportação de relatórios automáticos em PDF.

A continuidade do projeto poderá estender seu uso para atividades de pesquisa, ensino e divulgação científica, fortalecendo a interdisciplinaridade entre a Meteorologia, a Computação e a Engenharia de Software. Além disso, o código aberto hospedado no *GitHub* permite sua reprodução e aprimoramento por outros estudantes e pesquisadores, promovendo o caráter colaborativo e educativo que motivou sua criação.

Acredita-se que o desenvolvimento do Clima-Cast-Crepaldi alcançou os seus objetivos, demonstrando a viabilidade e eficiência de soluções computacionais de baixo custo aplicadas à análise e visualização de dados meteorológicos.

Bibliografia Recomendada

- ECMWF – European Centre for Medium-Range Weather Forecasts. **ERA5-Land (Reanalysis for Land Applications)**. Reading, 2023. Disponível em: <https://cds.climate.copernicus.eu/>.
- GILLIES, S. et al. *shapely/shapely: 2.1.1.* Zenodo, 2025. DOI: <https://doi.org/10.5281/zenodo.15463269>. Acesso em: 14 dez. 2025.
- GORELICK, N. et al. **Google Earth Engine: Planetary-scale geospatial analysis for everyone**. *Remote Sensing of Environment*, v. 202, p. 18–27, 2017. DOI: <https://doi.org/10.1016/j.rse.2017.06.031>.
- HERSBACH, H. et al. **The ERA5 Global Reanalysis**. *Quarterly Journal of the Royal Meteorological Society*, v. 146, p. 1999–2049, 2020. DOI: <https://doi.org/10.1002/qj.3803>.
- HUNTER, J. D. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, v. 9, n. 3, p. 90–95, 2007. DOI: <https://doi.org/10.1109/MCSE.2007.55>.
- IBGE – Instituto Brasileiro de Geografia e Estatística. **Base Cartográfica Contínua do Brasil**. Rio de Janeiro, 2024. Disponível em: <https://www.ibge.gov.br>.
- JORDAHL, K. et al. *geopandas/geopandas: v1.1.1.* Zenodo, 2025. DOI: <https://doi.org/10.5281/zenodo.15750510>. Acesso em: 14 dez. 2025.
- MAY, R. M. et al. MetPy: A Meteorological Python Library for Data Analysis and Visualization. *Bulletin of the American Meteorological Society*, v. 103, n. 10, p. E2273–E2284, 2022. DOI: <https://doi.org/10.1175/BAMS-D-21-0125.1>.
- MUÑOZ-SABATER, J. et al. **ERA5-Land: A state-of-the-art global reanalysis dataset for land applications**. *Earth System Science Data*, v. 13, p. 4349–4383, 2021. DOI: <https://doi.org/10.5194/essd-13-4349-2021>.
- NOAA/NCEP. *NCEP Data Products: GFS and GDAS*. Atualizado em 06 fev. 2024. Disponível em: <https://www.nco.ncep.noaa.gov/pmb/products/gfs/>. Acesso em: 14 dez. 2025.
- OPEN-METEO. *Historical Forecast API Documentation*. 2025. Disponível em: <https://open-meteo.com/en/docs/historical-forecast-api>. Acesso em: 14 dez. 2025.
- PLOTLY. *Plotly.py*. Zenodo, 2024. DOI: <https://doi.org/10.5281/zenodo.14503524>. Acesso em: 14 dez. 2025.

- SNOW, A. D. et al. *pyproj4/pyproj: 3.7.0 Release*. Zenodo, 2024. DOI: <https://doi.org/10.5281/zenodo.13864781>. Acesso em: 14 dez. 2025.
- STREAMLIT INC. **Streamlit: The fastest way to build data apps**. 2025. Disponível em: <https://streamlit.io/>.
- THE PANDAS DEVELOPMENT TEAM. pandas-dev/pandas: Pandas (Version 1.0.0). *Zenodo*, 2020. DOI: <https://doi.org/10.5281/zenodo.3509134>.
- WILSON, G. et al. **Good enough practices in scientific computing**. *PLoS Computational Biology*, v. 13, n. 6, e1005510, 2017. DOI: <https://doi.org/10.1371/journal.pcbi.1005510>.
- WU, Q. **geemap: A Python package for interactive mapping with Google Earth Engine**. *The Journal of Open Source Software*, v. 5, n. 51, p. 2305, 2020. DOI: <https://doi.org/10.21105/joss.02305>.
- ZWITCH, R. **streamlit-folium: Streamlit component for Folium maps**. 2022. Disponível em: <https://github.com/randyzwitch/streamlit-folium>.