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

Lista de ilustrações

Lista de tabelas

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.

2.3 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 e comunicação com APIs. As principais são:

- **Streamlit** — construção da interface web interativa e controle de navegação entre páginas;
- **Earth Engine API** — acesso direto aos dados do Google Earth Engine;
- **geemap e folium** — visualização de mapas interativos e integração com camadas geográficas;
- **geobr** — obtenção de limites administrativos brasileiros (estados e municípios);
- **pandas e numpy** — manipulação e análise de séries temporais;
- **matplotlib e plotly** — criação de gráficos e séries interativas;

- **json e os** — gerenciamento de autenticação e credenciais do GEE;
- **datetime** — controle temporal das consultas e agregação de dados.

Essas bibliotecas foram escolhidas por sua estabilidade, documentação atualizada e compatibilidade com o ambiente *Streamlit Cloud*, permitindo que o aplicativo opere de forma segura, rápida e em multiplataforma.

2.4 Funcionalidades Principais

A plataforma Clima-Cast-Crepaldi oferece quatro módulos interativos, acessíveis por meio do menu lateral:

1. **Mapas Interativos e Estáticos:** permitem a visualização das variáveis meteorológicas selecionadas do ERA5-Land, processadas no GEE, para áreas definidas (estado, município, polígono ou círculo), com escala de cores, legenda e opções de exportação;
2. **Séries Temporais:** geram gráficos de evolução das variáveis meteorológicas nas mesmas áreas definidas e com opção de exportação;
3. **Exportação de Resultados:** disponibiliza a extração de séries temporais e mapas em formatos de planilha (CSV e XLSX) e imagem (PNG e JPG);
4. **Página Sobre:** apresenta informações institucionais, créditos de desenvolvimento e uma síntese da plataforma.

2.5 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*.

2.6 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.

2.7 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	ERA5-Land (ECMWF) e GEE
GEE	ECMWF/ERA5_LAND/DAILY_AGGR
Principais Bibliotecas	pandas, numpy, geemap, geobr, plotly
Ambiente de Deploy	Streamlit Cloud (GitHub Integration)
Autenticação	Service Account (secrets.toml)
Formato de Saída	CSV, XLSX, PNG e JPG
Tempo médio de resposta	10 a 20 segundos (consulta GEE)

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/

charts_visualizer.py
gee_handler.py
map_visualizer.py
main.py
ui.py
utils.py

logo.png
municipios_ibge.json

packages.txt
requirements.txt
runtime.txt

sobre.docx
```

Para ilustrar a implementação dessa estrutura, a Figura 1 apresenta a visualização da árvore de diretórios dentro do ambiente de desenvolvimento integrado (IDE) Visual Studio Code. Note a separação clara entre os scripts de aplicação (`.py`), os arquivos de configuração de ambiente (`requirements.txt`, `packages.txt`) e os recursos estáticos.

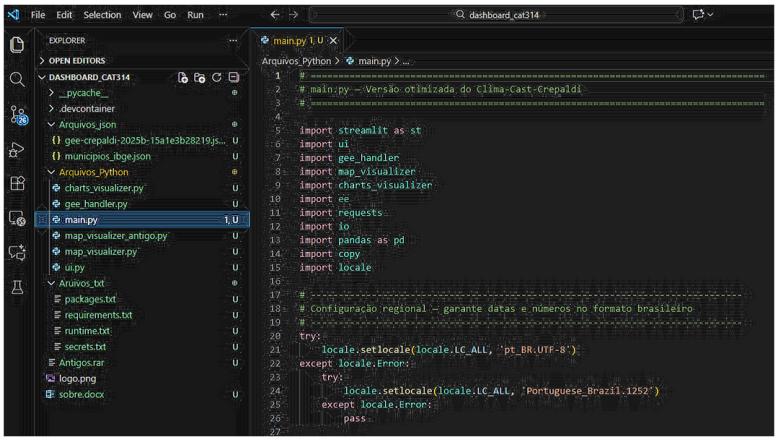

Figura 1 – Visualização da hierarquia de arquivos e módulos do projeto no ambiente de desenvolvimento (VS Code).

3.2 Descrição dos Principais Arquivos e Módulos

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

3.2.0.1 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.

3.2.1 `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.

3.2.2 `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*.

3.2.3 `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 `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 2 ilustra a relação entre os módulos principais e o fluxo de dados dentro da plataforma.

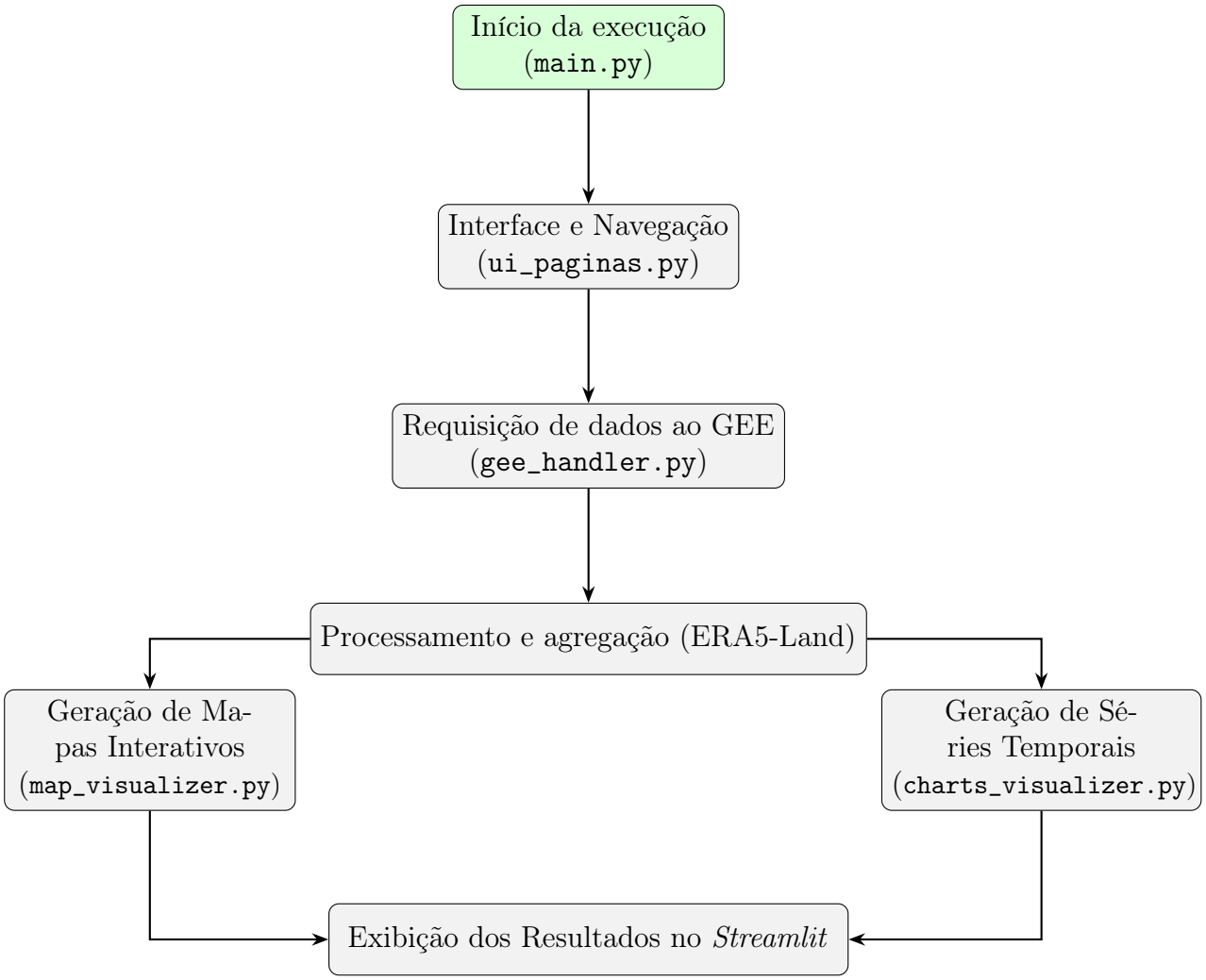

Figura 2 – Fluxograma funcional da Plataforma Interativa Clima-Cast-Crepaldi.

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

1. O arquivo `main.py` gerencia o fluxo de execução e interface;
2. As consultas são processadas pelo módulo `gee_handler.py`;
3. Resultados enviados simultaneamente ao `map_visualizer.py` e ao `charts_visualizer.py`;
4. As funções de interface em `ui.py` exibem os resultados no navegador.

Essa estrutura modular garante independência entre seus componentes, facilitando atualizações, adição de novas variáveis e manutenção futura da plataforma.

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 o fluxo ilustrado na Figura 2, partindo do arquivo principal `main.py`, que faz chamadas sequenciais aos demais módulos. O código utiliza boas práticas de programação, como modularização, tratamento de exceções e funções documentadas.

A seguir, descreve-se o fluxo operacional básico:

1. O usuário acessa o aplicativo via navegador web.
2. O arquivo `main.py` é carregado e inicializa o ambiente *Streamlit*.
3. O módulo `gee_handler.py` autentica a conexão com o Google Earth Engine.
4. O usuário seleciona a área de interesse (estado, município, polígono ou círculo).
5. O módulo `gee_handler.py` consulta as variáveis desejadas do `ERA5-Land`.
6. Os resultados são enviados aos módulos de visualização:
    - `map_visualizer.py` — gera mapas interativos e estáticos;
    - `charts_visualizer.py` — produz séries temporais.
7. A interface `ui.py` organiza os resultados em abas e gráficos dentro do *Streamlit*.

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.

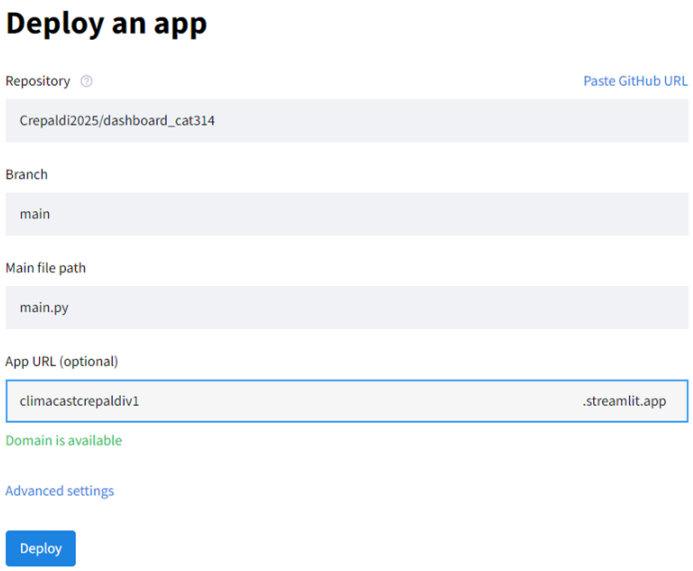

Figura 3 – Tela de configuração de deploy no *Streamlit Cloud*, destacando a integração com o repositório *GitHub* e a definição dos parâmetros de execução.

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 4.

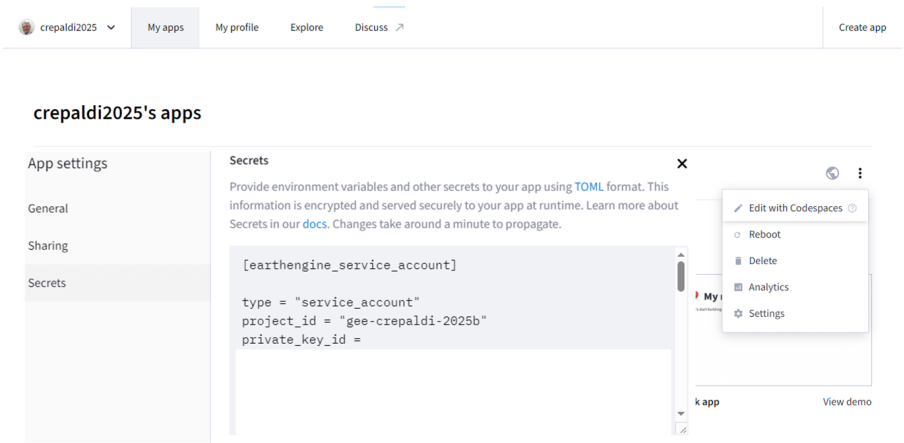

Figura 4 – Interface de gerenciamento de Secrets no Streamlit Cloud, onde as credenciais da Service Account devem ser armazenadas em formato TOML.

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 3 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 5 mostra a tela inicial da plataforma hospedada na nuvem, com o menu lateral de variáveis, período de análise e seleção da área de interesse.

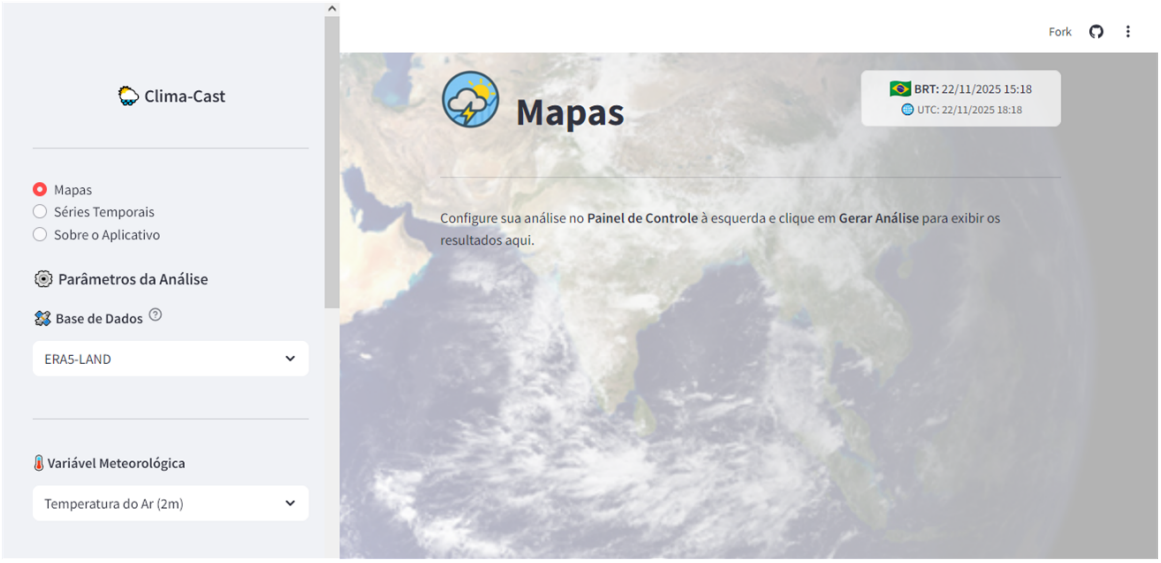

Figura 5 – Interface da Plataforma Clima-Cast-Crepaldi após o *deploy* no *Streamlit Cloud*.

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

# --------------------------------------
# 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, evitando 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`.

Essas versões foram testadas e garantem a execução estável tanto em ambiente local quanto na plataforma em nuvem.

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 6 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`.

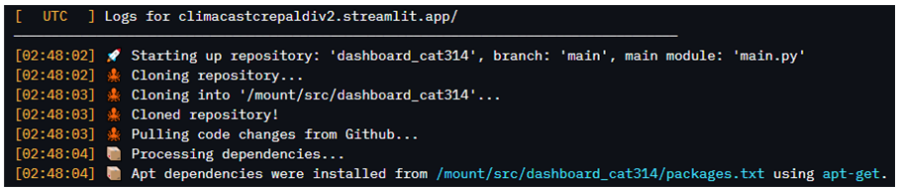

Figura 6 – Console de logs do *Streamlit Cloud* detalhando a inicialização do ambiente, clonagem do repositório e instalação de pacotes do sistema.

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 foram realizados para assegurar que cada módulo executasse sua função conforme o projeto.

- **Autenticação GEE:** verificação da função `inicializar_gee()` para garantir conexão segura via *Service Account*.
- **Consulta de dados:** teste da função `get_time_series_data()` utilizando áreas-piloto nos estados de Minas Gerais e São Paulo.
- **Renderização de mapas:** inspeção visual dos mapas produzidos por `map_visualizer.py`, avaliando escala, colorbar e coerência espacial.
- **Séries temporais:** validação das curvas geradas em `charts_visualizer.py`, comparando resultados com planilhas-referência exportadas do GEE.
- **Interface usuário:** teste de navegação e responsividade dos elementos *Streamlit* em diferentes navegadores (Edge, Firefox).

5.2 Testes de Desempenho

Os testes de desempenho avaliaram o tempo de resposta e o uso de recursos no ambiente de execução em nuvem.

- Tempo médio de inicialização da aplicação: 14 ± 2 s.
- Tempo médio para geração de mapas interativos: 6 ± 1 s.
- Tempo médio para geração de séries temporais diárias: 4 ± 1 s.

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.

5.3 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.

5.4 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/>.
- 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>.
- 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>.
- 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>.