Laboratorio 4 - Análisis Geoespacial y sensores remotos

Sebastián Juárez - 21471

Juan Pablo Cordón - 21458

Link al github: https://github.com/SebasJuarez/DS-Collection/tree/Lab4

Avances

In [1]:
```python
import rasterio
import numpy as np
import matplotlib.pyplot as plt
from datetime import date
import openeo
```

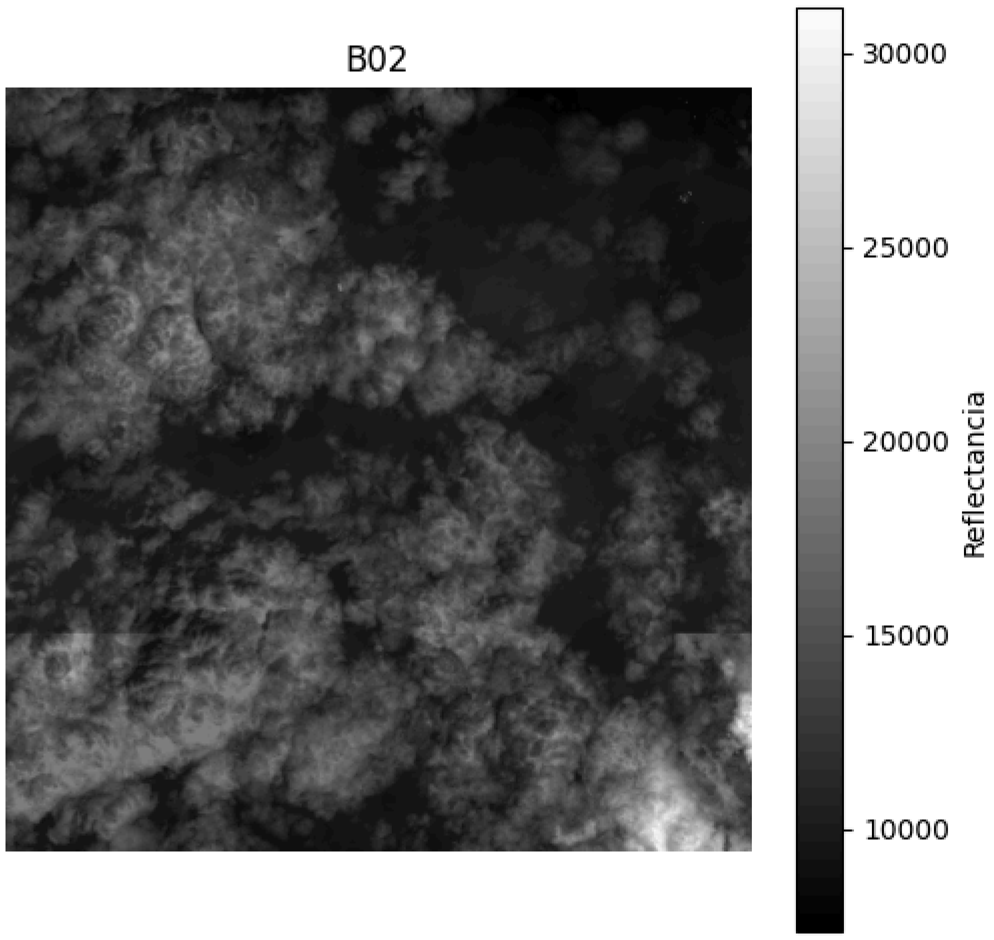
In [ ]:
```python
connection = openeo.connect("https://openeo.dataspace.copernicus.eu").authenticate_
```

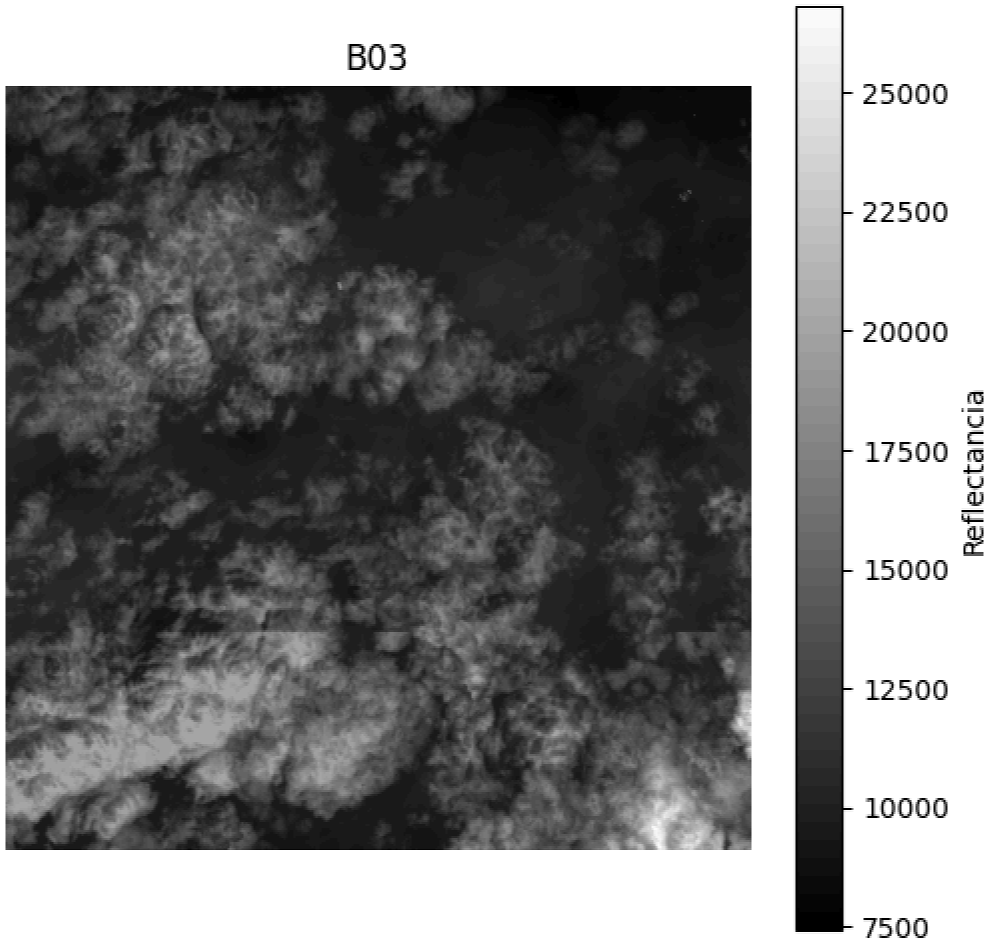
```
Authenticated using refresh token.
```

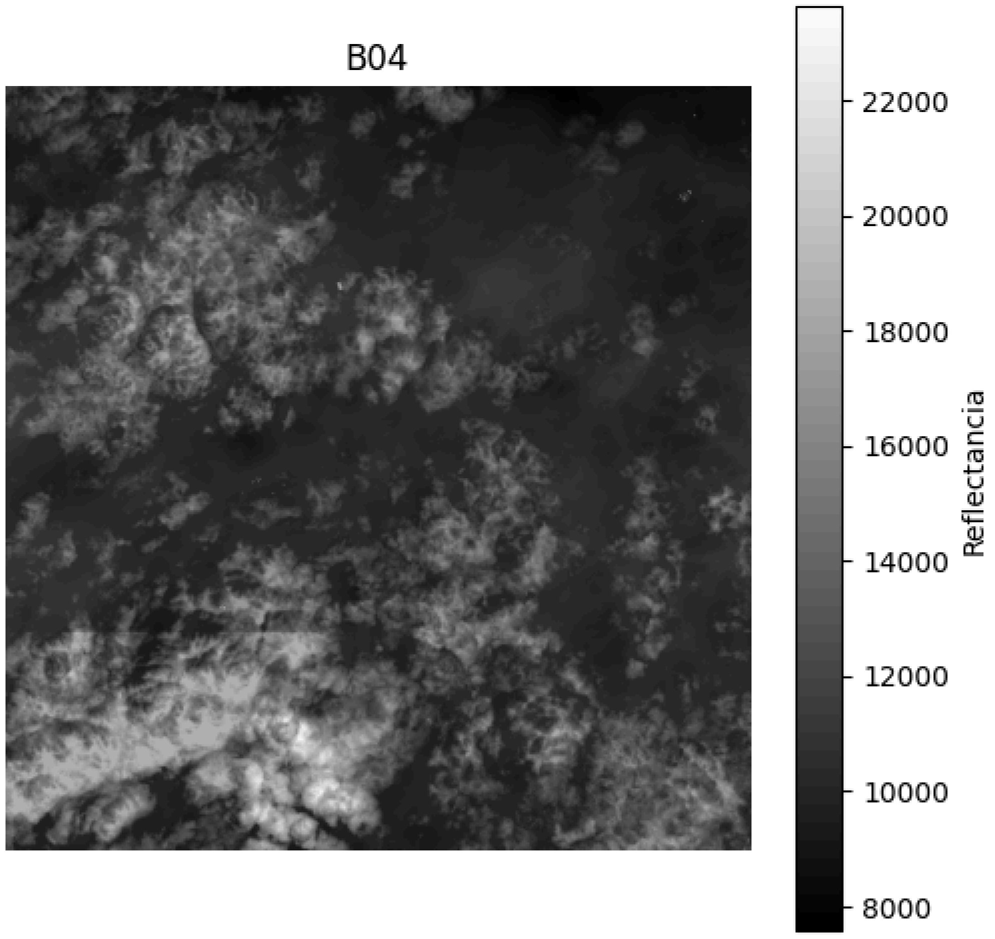
Toma de datos temporales basado en el ejemplo

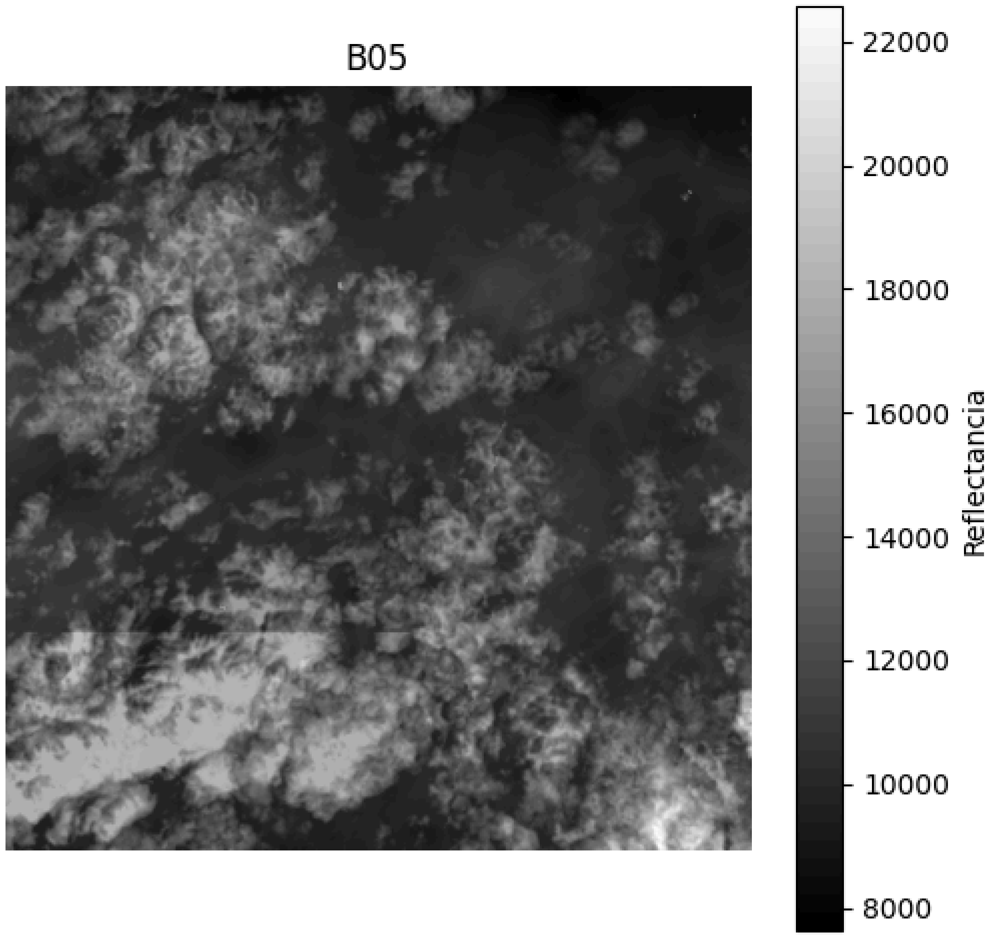
In [ ]:
```python
import tempfile
import os

lago_atitlan = {
    "west": -91.31,
    "east": -91.08,
    "south": 14.60,
    "north": 14.74
}

lago_amatitlan = {
    "west": -90.66,
    "east": -90.58,
    "south": 14.43,
    "north": 14.51
}

fecha_inicio = "2024-01-01"
fecha_fin = "2024-06-30"

cube = connection.load_collection(
    "SENTINEL2_L2A",
    spatial_extent=lago_amatitlan,
    temporal_extent=[fecha_inicio, fecha_fin],
    bands=["B02", "B03", "B04", "B05", "B08", "B11", "B12"]
```

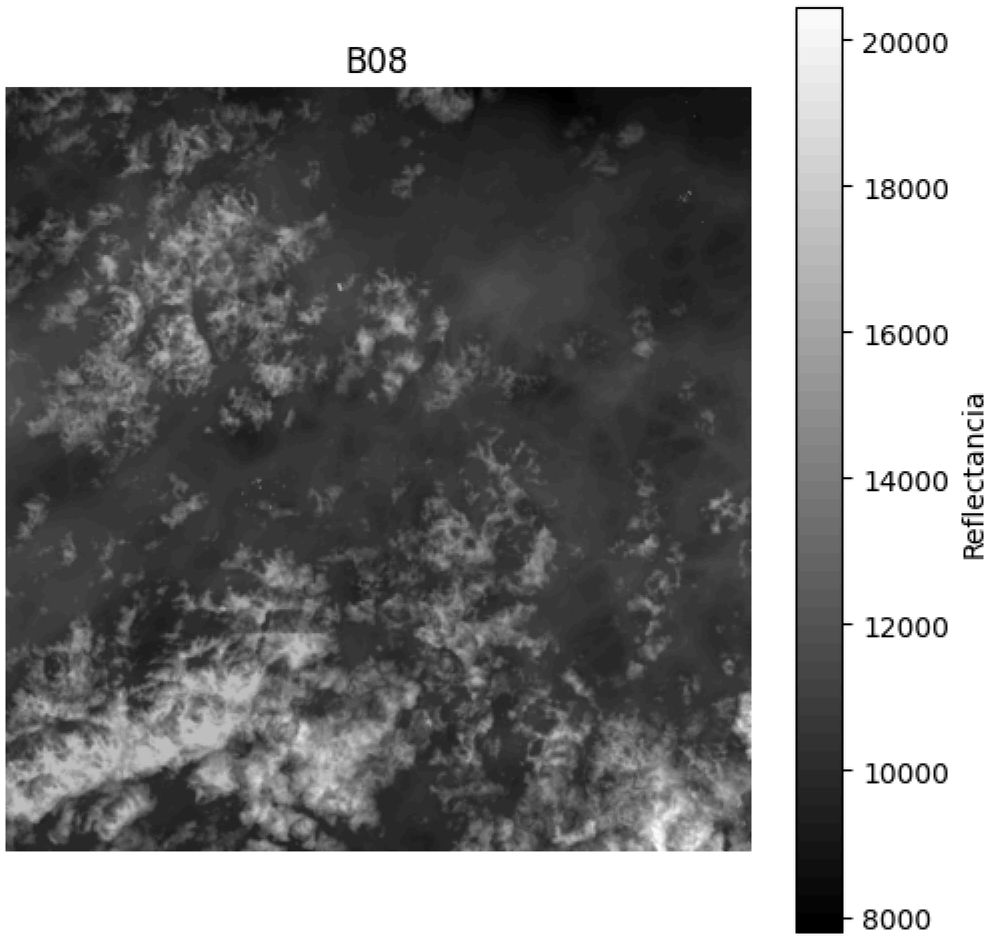
```
).max_time()

fd, temp_path = tempfile.mkstemp(suffix=".tif")
os.close(fd)

try:
    cube.download(temp_path)

    with rasterio.open(temp_path) as src:
        bandas = src.read()
        nombres = src.descriptions if src.descriptions[0] else [f"Banda {i+1}" for
        nodata = src.nodata

    for i in range(bandas.shape[0]):
        plt.figure(figsize=(6, 6))
        img = bandas[i]
        img = np.ma.masked_where(img == nodata, img)
        plt.imshow(img, cmap='gray')
        plt.title(f"{nombres[i]}")
        plt.axis('off')
        plt.colorbar(label="Reflectancia")
        plt.show()

finally:
    os.remove(temp_path)
```

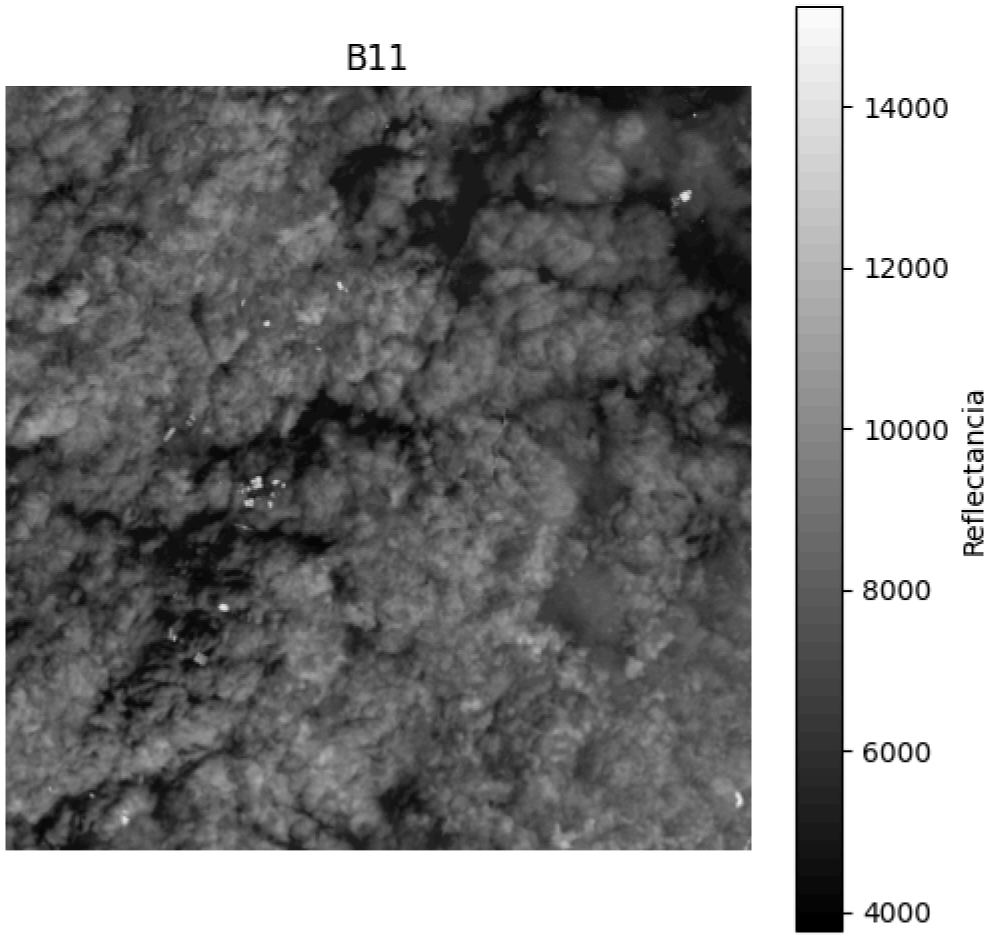


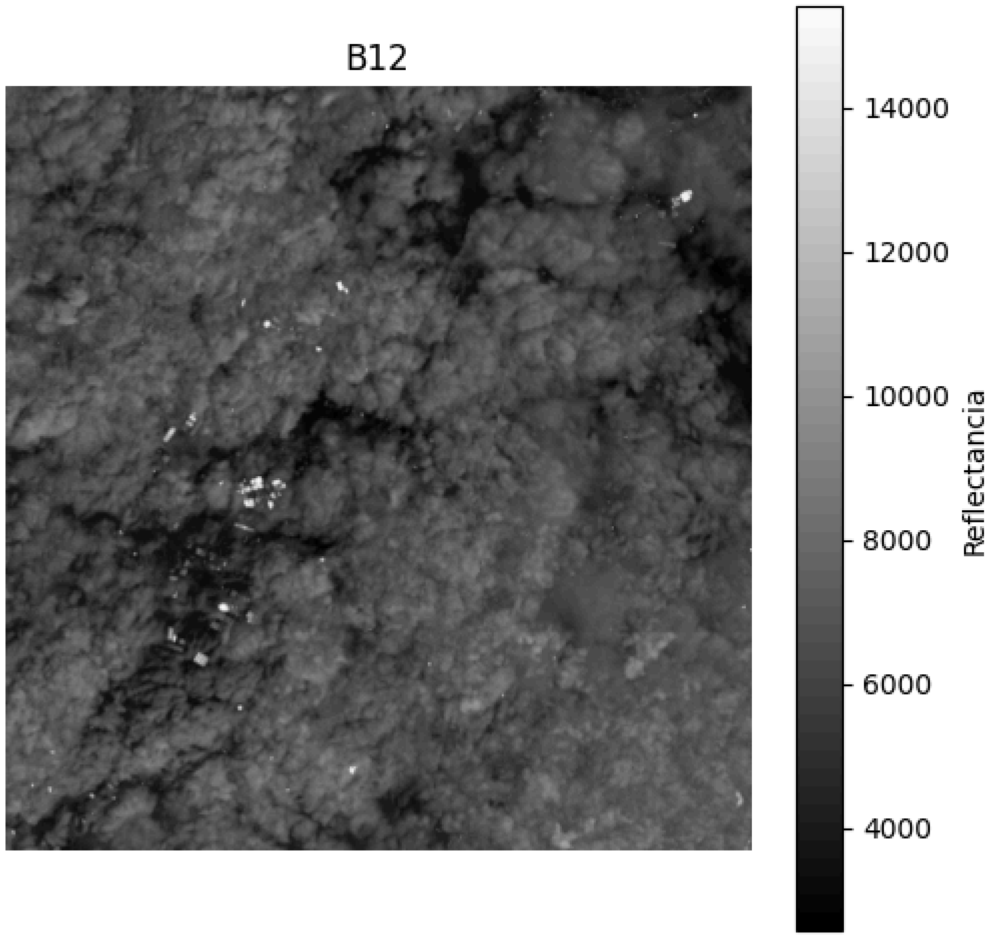

B03
25000
22500
20000
17500
15000
12500
10000
7500
Reflectancia


B04
Reflectancia
22000
20000
18000
16000
14000
12000
10000
8000


B05
22000
20000
18000
16000
14000
12000
10000
8000
Reflectancia


B08
20000
18000
16000
14000
12000
10000
8000
Reflectancia


B11
14000
12000
10000
8000
6000
4000
Reflectancia



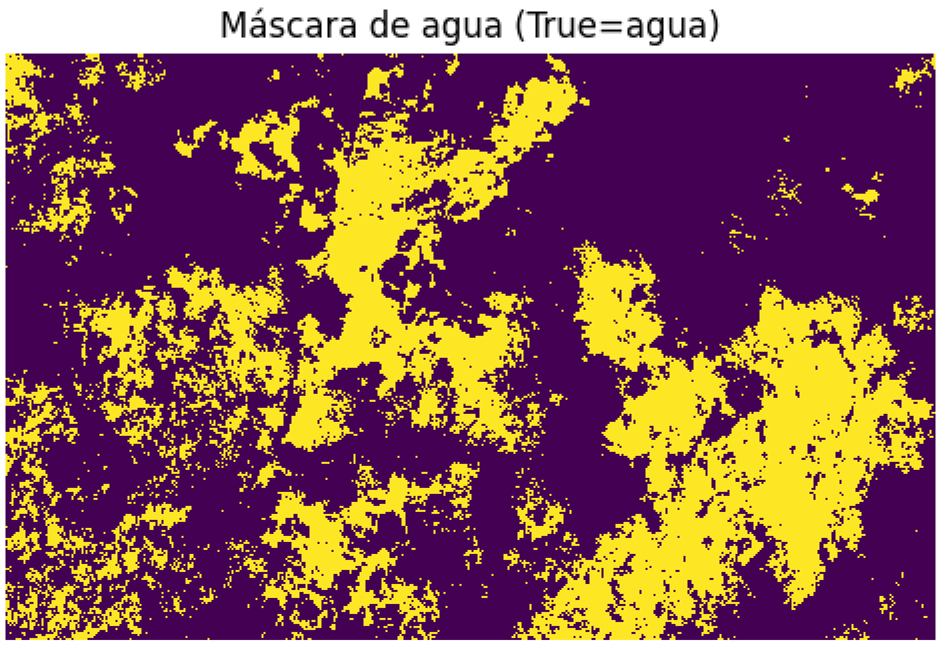
Pruebas codigo Sentinel Hub para Cianobacterias

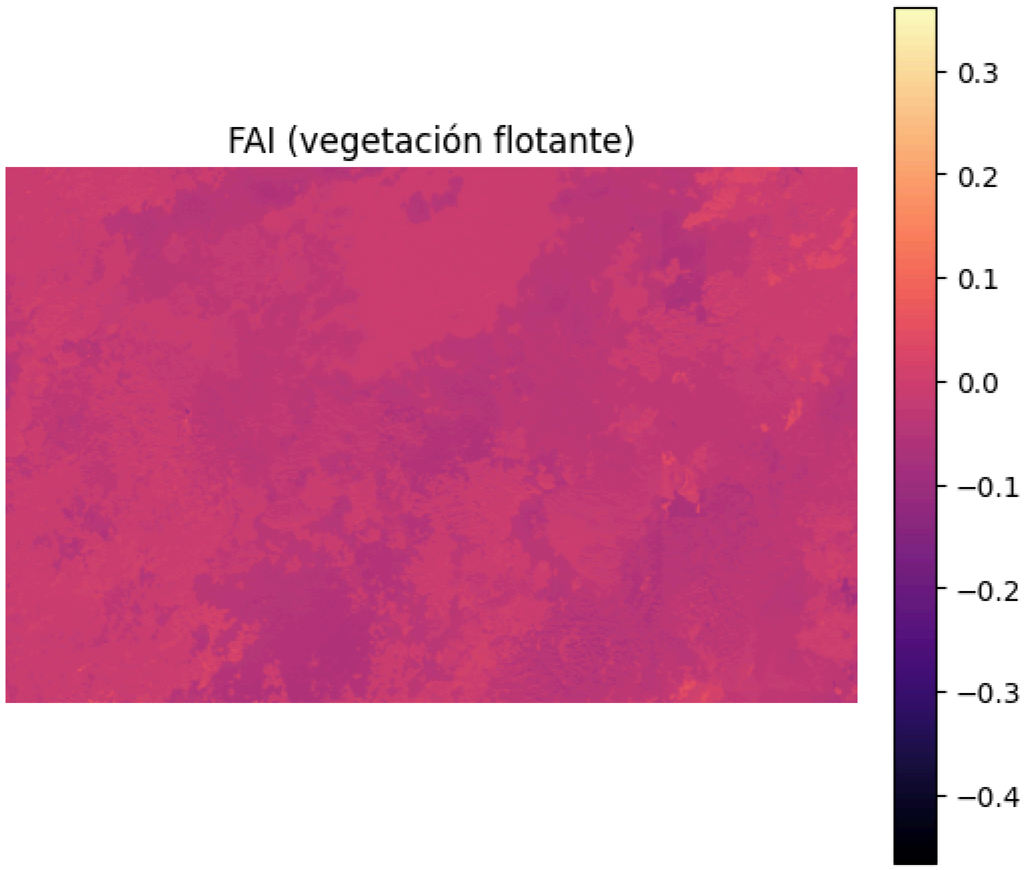
In [17]:
```python
import openeo
import numpy as np
import rasterio
import matplotlib.pyplot as plt
import os
import tempfile

bbox = {
    "west": -91.31,
    "east": -91.08,
    "south": 14.60,
    "north": 14.74
}
fecha_inicio = "2024-01-01"
fecha_fin = "2024-06-30"

MNDWI_threshold = 0.42
NDWI_threshold = 0.40
filter_UABS = True
filter_SSI = False

con = openeo.connect("https://openeo.dataspace.copernicus.eu").authenticate_oidc()
```

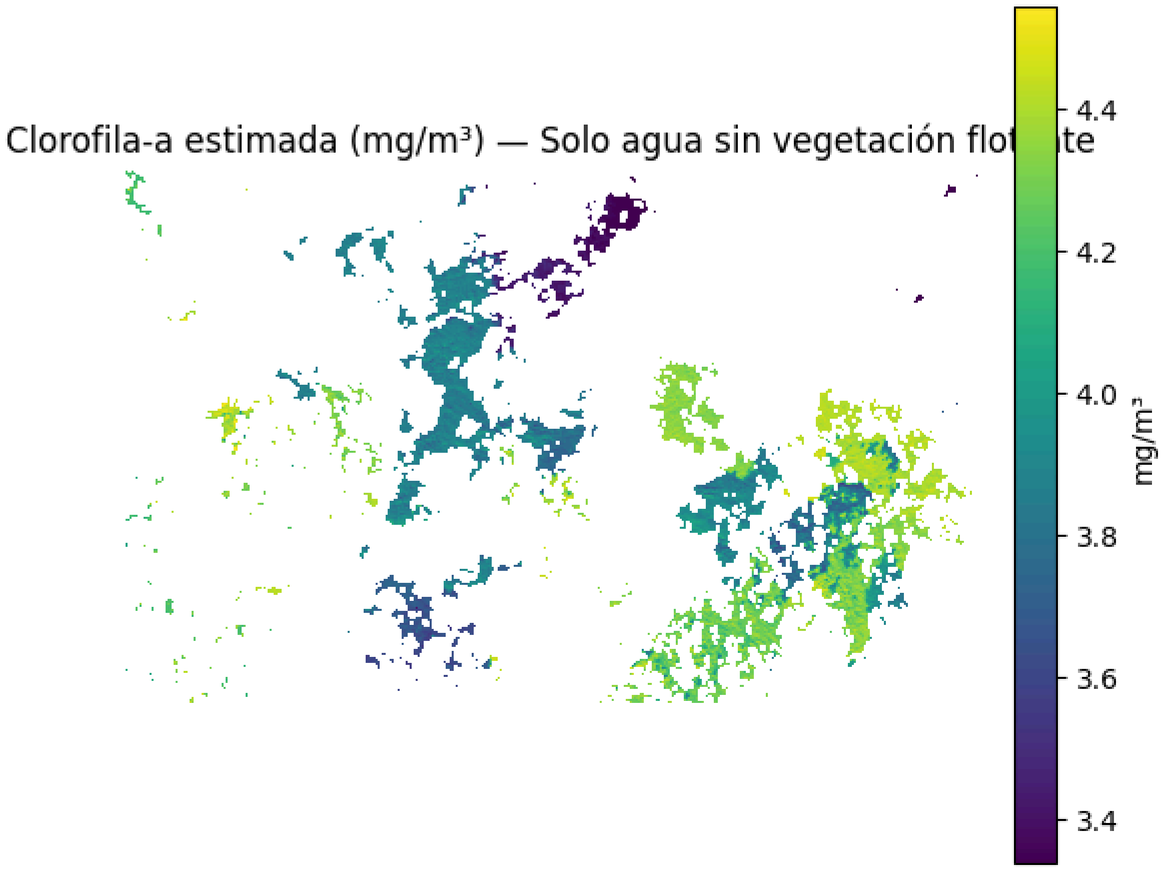
```
bands = ["B02","B03","B04","B05","B07","B8A","B08","B11","B12"]

cube = con.load_collection(
    "SENTINEL2_L2A",
    spatial_extent=bbox,
    temporal_extent=[fecha_inicio, fecha_fin],
    bands=bands
).max_time()

import os, tempfile
fd, temp_path = tempfile.mkstemp(suffix=".tif")
os.close(fd)

try:
    cube.download(temp_path)

    with rasterio.open(temp_path) as src:
        data = src.read()
        desc = src.descriptions
        nodata = src.nodata

    name_to_idx = { (desc[i] if desc and desc[i] else f"Banda {i+1}"): i for i in r

    def get_band(label, fallback_name):
        if desc and any(d for d in desc if d):
            for i, d in enumerate(desc):
                if d and label in d:
                    return data[i].astype("float32")
        i = bands.index(label)
        return data[i].astype("float32")

    B02 = get_band("B02", "B02")
    B03 = get_band("B03", "B03")
    B04 = get_band("B04", "B04")
    B05 = get_band("B05", "B05")
    B07 = get_band("B07", "B07")
    B8A = get_band("B8A", "B8A")
    B08 = get_band("B08", "B08")
    B11 = get_band("B11", "B11")
    B12 = get_band("B12", "B12")

    def mask_nodata(x):
        if nodata is not None:
            x = np.where(x == nodata, np.nan, x)
        return x

    for v in [B02,B03,B04,B05,B07,B8A,B08,B11,B12]:
        pass

    B02 = mask_nodata(B02); B03 = mask_nodata(B03); B04 = mask_nodata(B04)
    B05 = mask_nodata(B05); B07 = mask_nodata(B07); B8A = mask_nodata(B8A)
    B08 = mask_nodata(B08); B11 = mask_nodata(B11); B12 = mask_nodata(B12)

    def maybe_scale(x):
        p = np.nanpercentile(x, 99)
        return x/10000.0 if p > 2 else x
```

```
B02 = maybe_scale(B02); B03 = maybe_scale(B03); B04 = maybe_scale(B04)
B05 = maybe_scale(B05); B07 = maybe_scale(B07); B8A = maybe_scale(B8A)
B08 = maybe_scale(B08); B11 = maybe_scale(B11); B12 = maybe_scale(B12)


def ndvi(nir, r): return (nir - r) / (nir + r + 1e-6)
def mndwi(g, swir1): return (g - swir1) / (g + swir1 + 1e-6)
def ndwi(g, nir): return (g - nir) / (g + nir + 1e-6)
def ndwi_leaves(nir, swir1): return (nir - swir1) / (nir + swir1 + 1e-6)
def aweish(b, g, nir, swir1, swir2): return b + 2.5*g - 1.5*(nir + swir1) - 0.2
def aweinsh(g, nir, swir1): return 4*(g - swir1) - (0.25*nir + 2.75*swir1)
def dbsi(swir1, g, ndvi_): return ((swir1 - g) / (swir1 + g + 1e-6)) - ndvi_

_ndvi = ndvi(B08, B04)
_mndwi = mndwi(B03, B11)
_ndwi = ndwi(B03, B08)
_ndwi_leaves = ndwi_leaves(B08, B11)
_aweish = aweish(B02, B03, B08, B11, B12)
_aweinsh = aweinsh(B03, B08, B11)
_dbsi = dbsi(B11, B03, _ndvi)

water = (
    (_mndwi > MNDWI_threshold) |
    (_ndwi > NDWI_threshold) |
    (_aweinsh > 0.1879) |
    (_aweish > 0.1112) |
    (_ndvi < -0.2) |
    (_ndwi_leaves > 1)
)

if filter_UABS:
    water = np.where((water) & ((_aweinsh <= -0.03) | (_dbsi > 0)), False, wate

FAIv = (B07 - (B04 + (B8A - B04) * (783 - 665) / (865 - 665)))

NDCIv = (B05 - B04) / (B05 + B04 + 1e-6)
chl = 826.57 * (NDCIv**3) - 176.43 * (NDCIv**2) + 19.0 * NDCIv + 4.071

chl_masked = np.where((water) & (FAIv <= 0.08), chl, np.nan)

plt.figure(figsize=(6,6))
plt.imshow(water, interpolation="nearest")
plt.title("Máscara de agua (True=agua)")
plt.axis("off")
plt.show()

plt.figure(figsize=(6,6))
im = plt.imshow(FAIv, cmap="magma")
plt.title("FAI (vegetación flotante)")
plt.axis("off")
plt.colorbar(im, fraction=0.046, pad=0.04)
plt.show()

plt.figure(figsize=(6,6))
im = plt.imshow(chl_masked, cmap="viridis", vmin=np.nanpercentile(chl_masked, 2
    plt.title("Clorofila-a estimada (mg/m³) — Solo agua sin vegetación flotante")
    plt.axis("off")
    plt.colorbar(im, fraction=0.046, pad=0.04, label="mg/m³")
    plt.show()

finally:
    try:
        os.remove(temp_path)
    except Exception:
        pass
```

```
Authenticated using refresh token.
```




FAI (vegetación flotante)
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4


Clorofila-a estimada (mg/m³) — Solo agua sin vegetación flotante
4.4
4.2
4.0
3.8
3.6
3.4
mg/m³