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

Las imagenes, por cuestion de tiempo, se han incluido en documento images.ipynb: https://github.com/SebasJuarez/DS-Collection/blob/Lab4/Images.ipynb

Analisis / primeros pasos de la busqueda de las imagenes

In [2]:
```
import rasterio
import numpy as np
import matplotlib.pyplot as plt
from datetime import date
import openeo
```

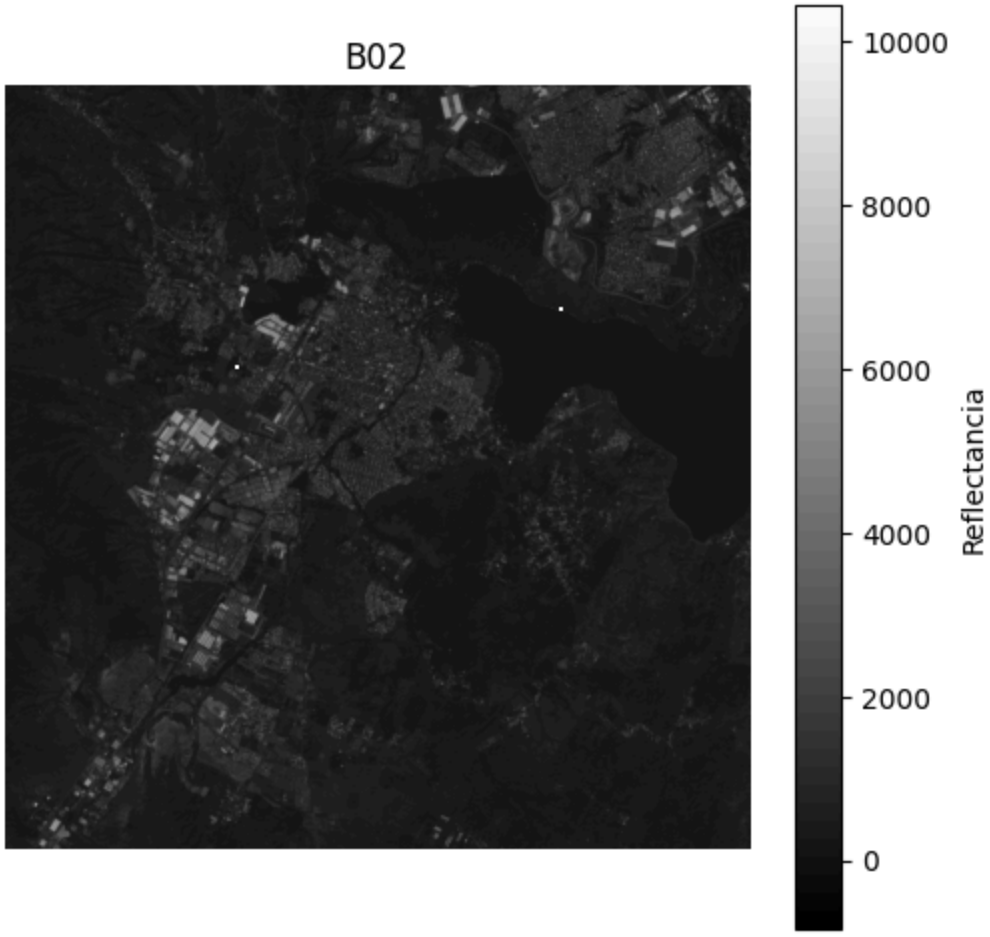
In [3]:
```
connection = openeo.connect("https://openeo.dataspace.copernicus.eu").authenticate_
```

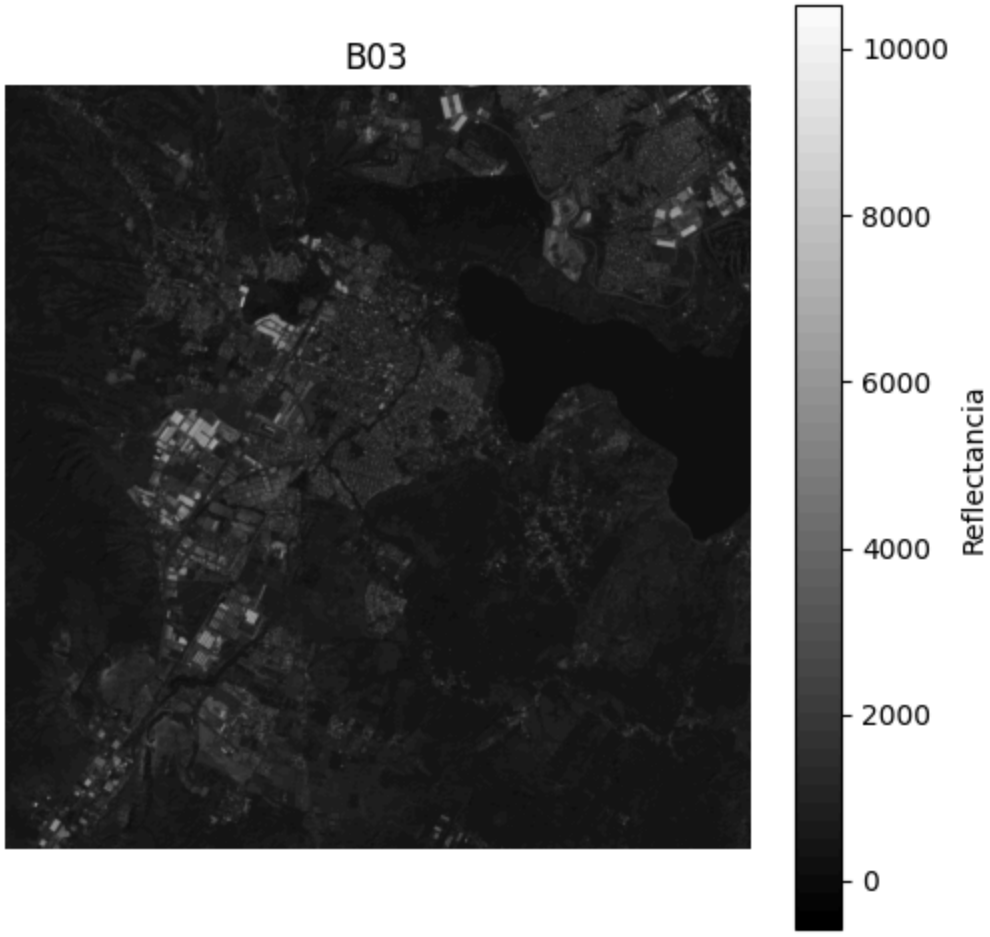
```
Authenticated using refresh token.
```

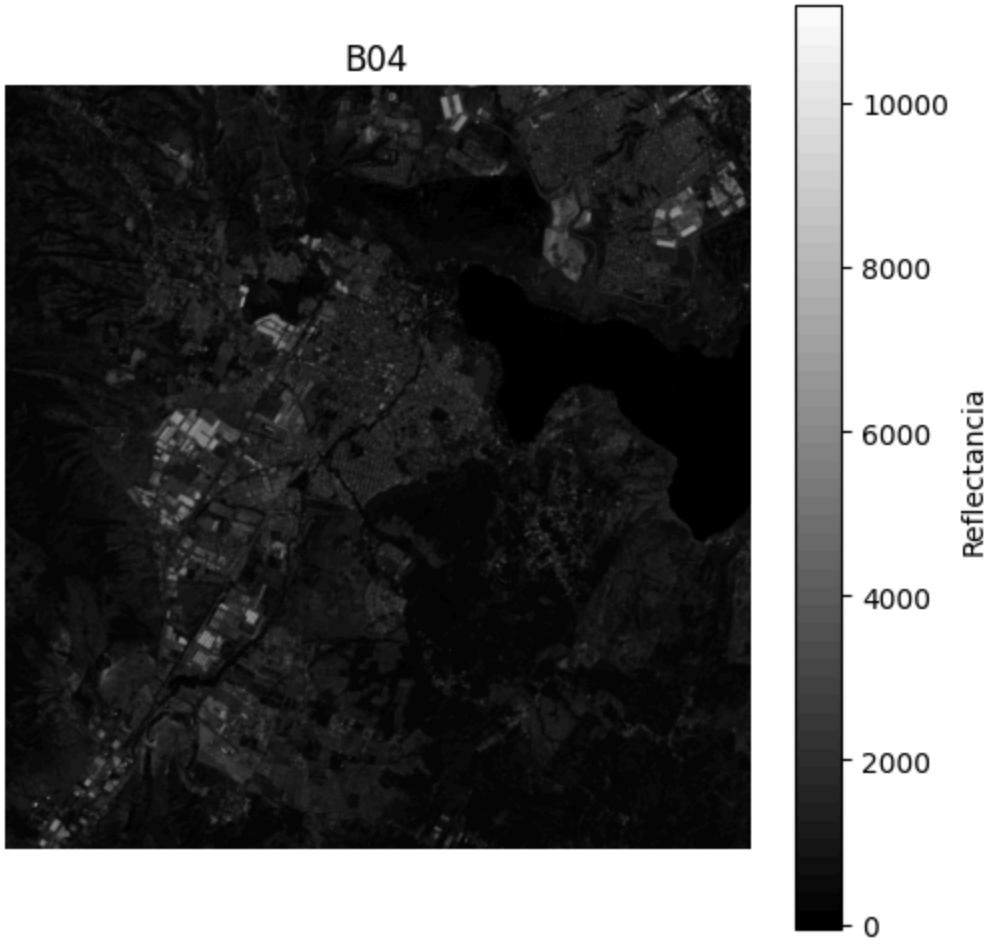
Toma de datos temporales basado en el ejemplo

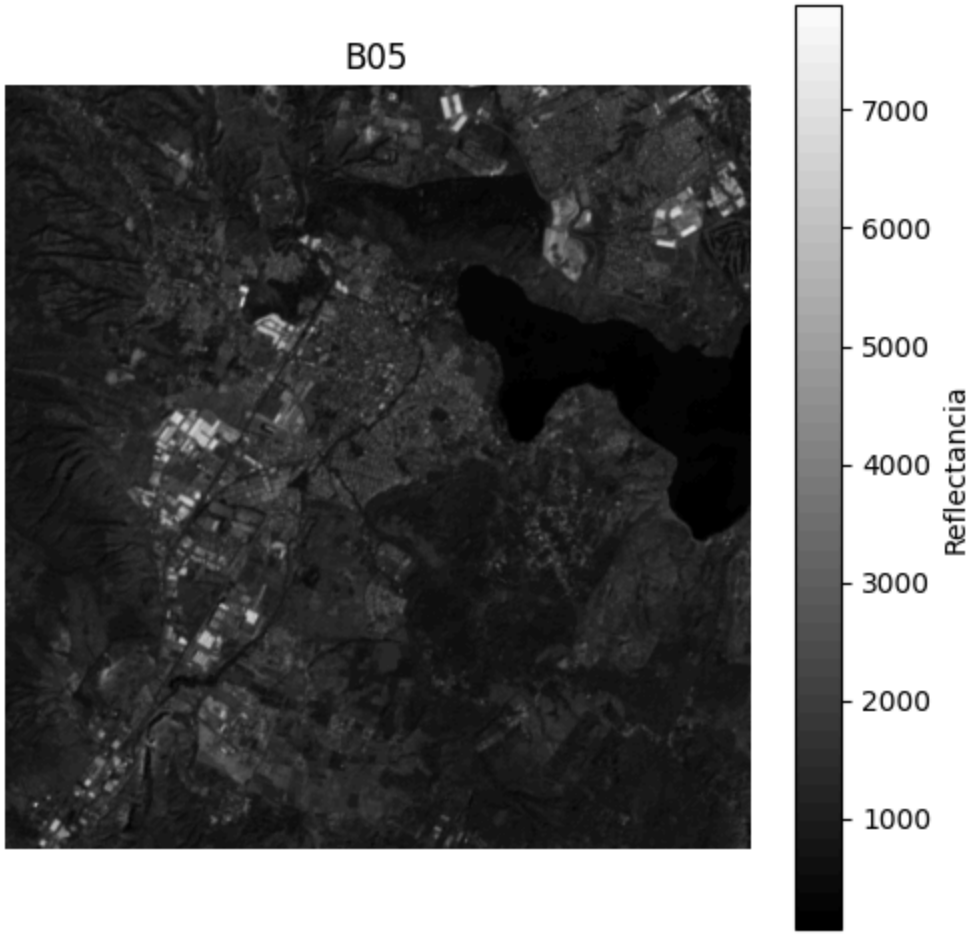
In [4]:
```
import tempfile
import os

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 = "2025-02-07"
fecha_fin = "2025-02-07"
```

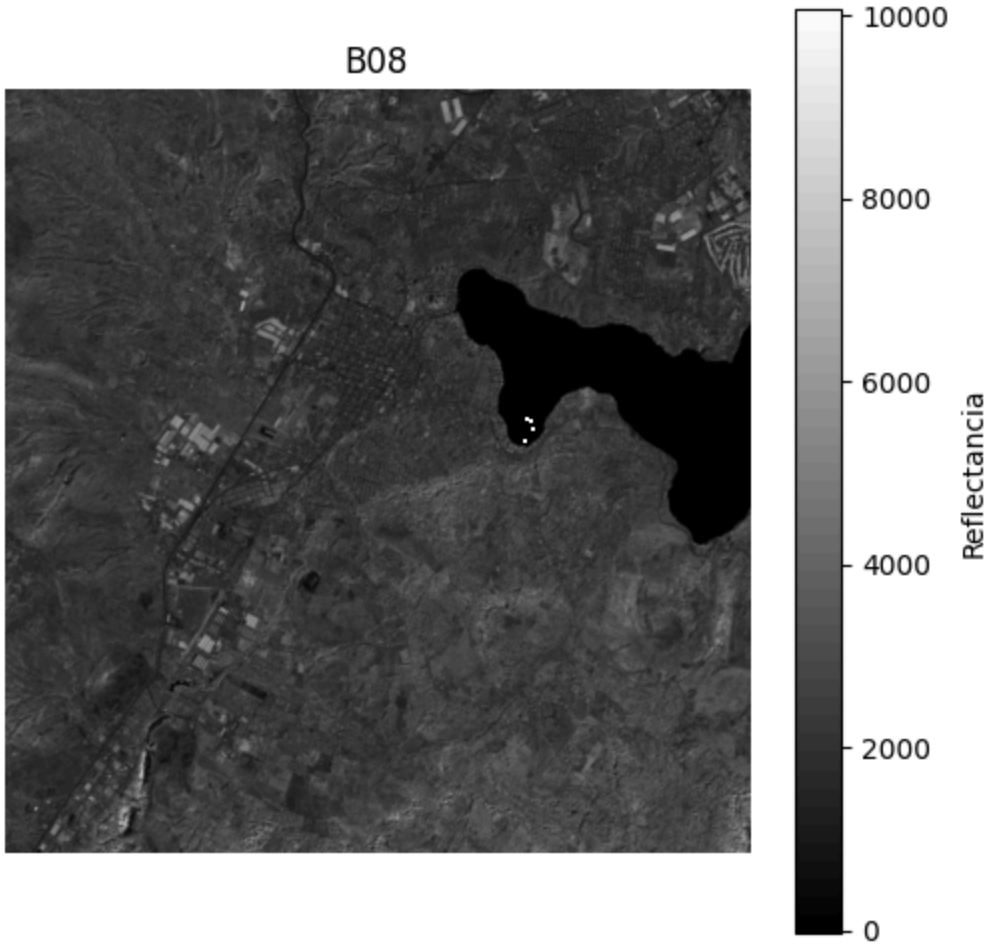
```
cube = connection.load_collection(
    "SENTINEL2_L2A",
    spatial_extent=lago_amatitlan,
    temporal_extent=[fecha_inicio, fecha_fin],
    bands=["B02", "B03", "B04", "B05", "B08", "B11", "B12"]
).max_time()

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)
```

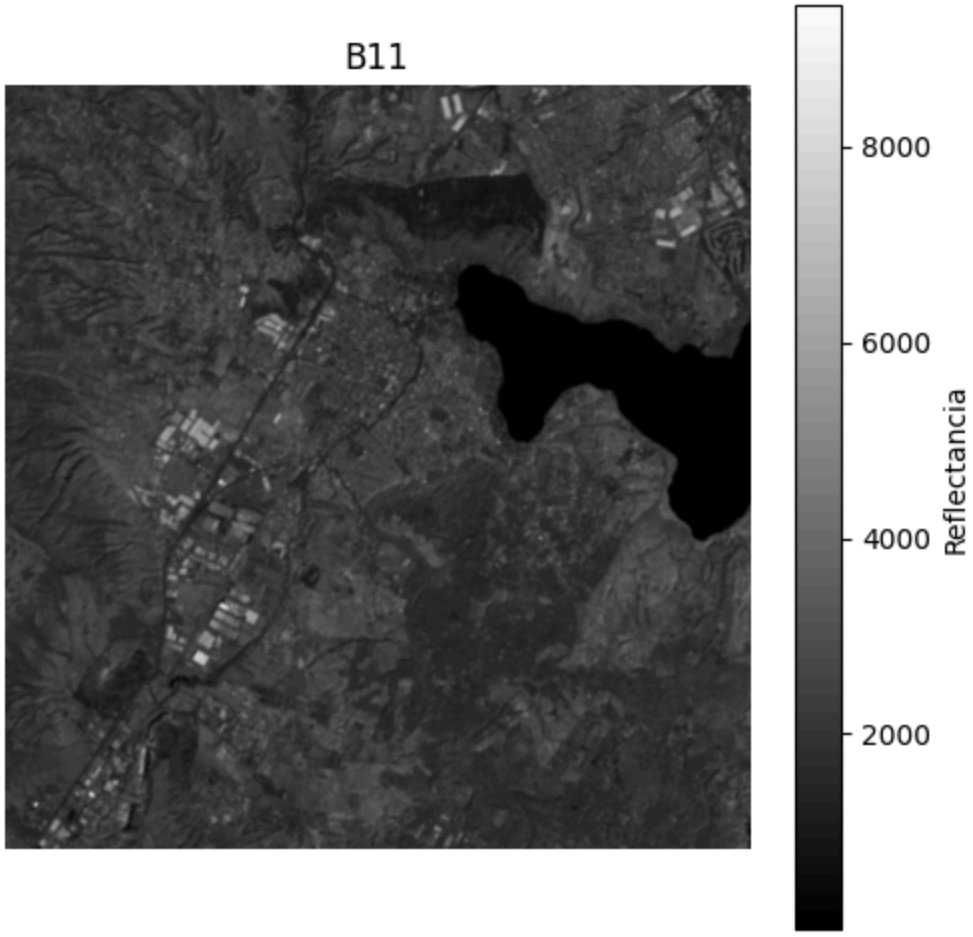

B02
10000
8000
6000
4000
2000
0
Reflectancia

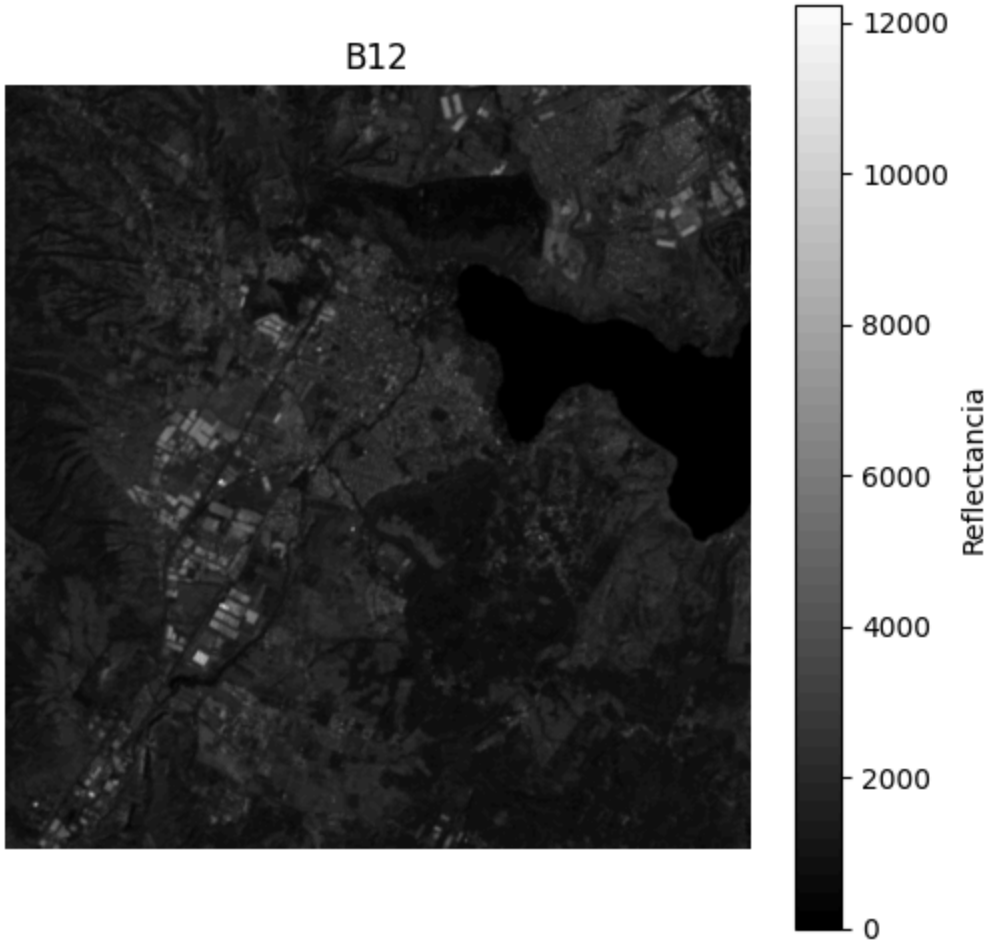

B03
10000
8000
6000
4000
2000
0
Reflectancia


B04
10000
8000
6000
4000
2000
0
Reflectancia


B05
7000
6000
5000
4000
3000
2000
1000
Reflectancia


B08
10000
8000
6000
4000
2000
0
Reflectancia


B11
8000
6000
4000
2000
Reflectancia



Pruebas codigo Sentinel Hub para Cianobacterias

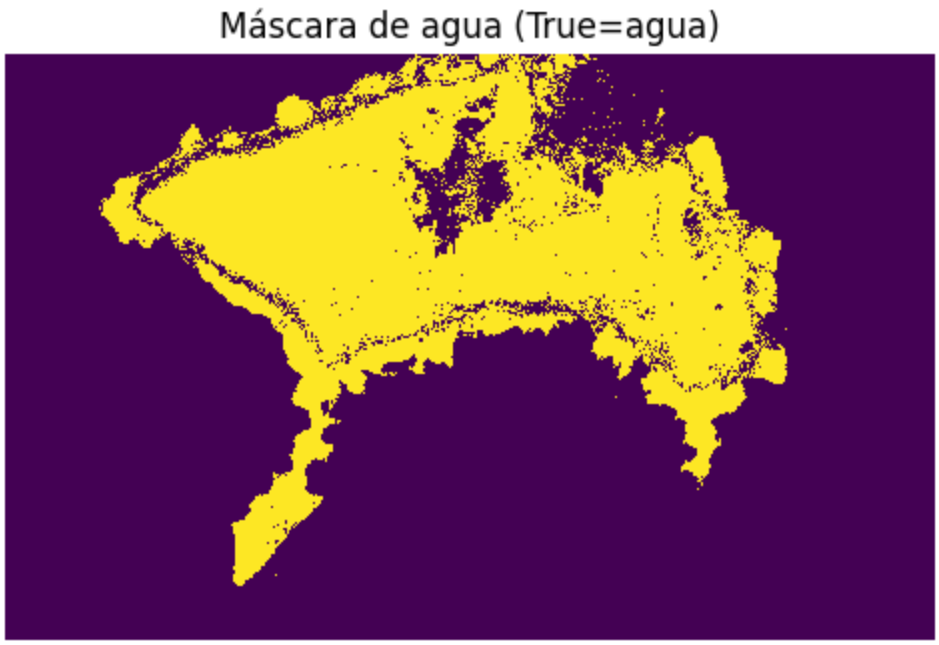
In [5]:
```
import openeo
import numpy as np
import rasterio
import matplotlib.pyplot as plt
import os
import tempfile

bbox = {
    "west": -91.31,
    "east": -91.08,
    "south": 14.60,
    "north": 14.74
}
fecha_inicio = "2025-02-07"
fecha_fin = "2025-02-07"

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

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

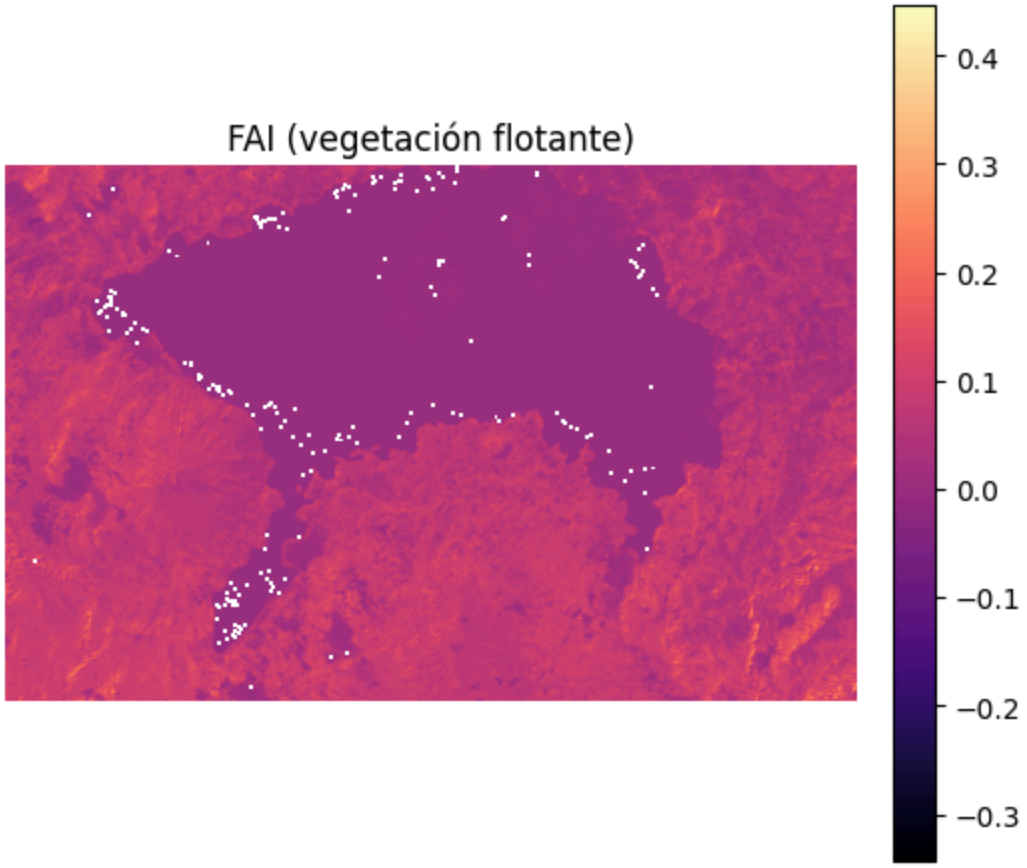
```
bands = ["B02","B03","B04","B05","B07","B8A","B08","B11","B12"]

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
```

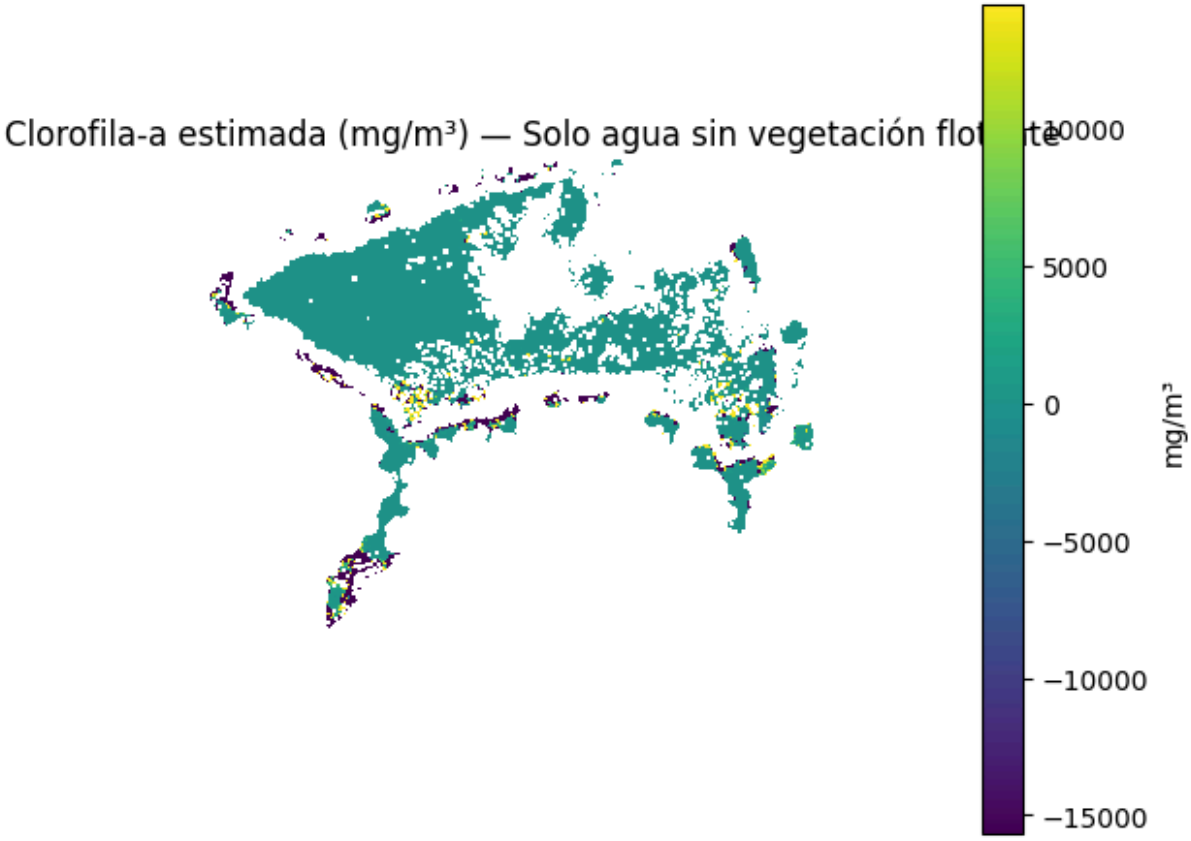
```
B02 = maybe_scale(B02); B03 = maybe_scale(B03); B04 = maybe_scale(B04)
B05 = maybe_scale(B05); B07 = maybe_scale(B07); B8A = maybe_scale(B8A)
B08 = maybe_scale(B08); B11 = maybe_scale(B11); B12 = maybe_scale(B12)


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.
```







Toma de datos crudos, sin imagenes, de los lagos

In [6]:
```
print("=== BLOQUE 1: Configuración ===")

import os, json, tempfile, datetime as dt
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import rasterio

# Coordenadas Lago Atitlán (WGS84)
lago_atitlan = {
    "west": -91.326256,
    "east": -91.071510,
    "south": 14.594800,
    "north": 14.750979,
}


# Fechas “buenas” que compartiste
fechas_buenas = [
    "2025-02-07","2025-02-10","2025-02-25","2025-02-27","2025-03-02",
    "2025-03-04","2025-03-07","2025-03-09","2025-03-12","2025-03-14",
    "2025-03-19","2025-03-22","2025-03-24","2025-03-26","2025-04-03",
    "2025-04-11","2025-04-13","2025-04-15","2025-04-16","2025-04-18",
    "2025-04-28","2025-05-03","2025-05-13","2025-05-28","2025-07-10",
    "2025-07-17","2025-07-20","2025-07-24","2025-08-01"
]

COVERAGE_THRESHOLD = 0.40  # % mínimo del lago cubierto para considerar la fecha
BANDS = ["B02","B03","B04","B05","B07","B8A","B08","B11","B12","SCL"]

print(f"Lago Atitlán bbox: {lago_atitlan}")
print(f"Fechas a procesar: {len(fechas_buenas)}")
print("Listo.")
```

```
=== BLOQUE 1: Configuración ===
Lago Atitlán bbox: {'west': -91.326256, 'east': -91.07151, 'south': 14.5948, 'nort
h': 14.750979}
Fechas a procesar: 29
Listo.
```

In [7]:
```
print("=== BLOQUE 2: Conexión openEO ===")
import openeo
con = openeo.connect("https://openeo.dataspace.copernicus.eu").authenticate_oidc()
print("Conectado a openEO. Autenticación lista.")
```

```
=== BLOQUE 2: Conexión openEO ===
Authenticated using refresh token.
Conectado a openEO. Autenticación lista.
```

In [8]:
```
print("=== BLOQUE 3: Utilidades ===")
from affine import Affine
from pyproj import CRS, Transformer
import numpy as np

def scale01(x):
    # Escala 0..1 si vienen en 0..10000
```

```
    return x/10000.0 if np.nanpercentile(x, 99) > 2 else x

def mask_clouds_with_SCL(arr, scl):
    # SCL: 3 sombra, 7/8/9 nubes, 10 cirrus, 0/1 no data/borde
    bad = np.isin(scl, [0,1,3,7,8,9,10])
    out = arr.astype("float32").copy()
    out[bad] = np.nan
    return out

def water_mask(B02,B03,B04,B08,B11,B12):
    # Basado en script compartido (WBI + filtros UABS)
    ndvi_ = (B08-B04)/(B08+B04+1e-6)
    mndwi = (B03-B11)/(B03+B11+1e-6)
    ndwi_ = (B03-B08)/(B03+B08+1e-6)
    ndwi_leaves = (B08-B11)/(B08+B11+1e-6)
    aweish = B02 + 2.5*B03 - 1.5*(B08+B11) - 0.25*B12
    aweinsh = 4*(B03-B11) - (0.25*B08 + 2.75*B11)
    dbsi = ((B11-B03)/(B11+B03+1e-6)) - ndvi_

    water = (
        (mndwi > 0.42) |
        (ndwi_ > 0.40) |
        (aweinsh > 0.1879) |
        (aweish > 0.1112) |
        (ndvi_ < -0.2) |
        (ndwi_leaves > 1)
    )
    # filtrar urbano/suelo desnudo
    water = np.where((water) & ((aweinsh <= -0.03) | (dbsi > 0)), False, water)
    return water

def FAI(B04, B07, B8A):
    return B07 - (B04 + (B8A - B04) * (783 - 665) / (865 - 665))

def chlorophyll_from_NDCI(B04, B05):
    ndci = (B05 - B04)/(B05 + B04 + 1e-6)
    return 826.57*(ndci**3) - 176.43*(ndci**2) + 19.0*ndci + 4.071

def ndvi(NIR, RED):
    return (NIR - RED) / (NIR + RED + 1e-6)

def ndwi(GREEN, NIR):
    return (GREEN - NIR) / (GREEN + NIR + 1e-6)

def mask_with_bbox_any_crs(arr, transform, raster_crs, bbox_wgs84):
    """
    Recorta 'arr' (en el CRS del raster) con un bbox dado en WGS84.
    Convierte el bbox a CRS del raster y aplica la máscara.
    bbox_wgs84: dict con west/east/south/north (lon/lat).
    """
    if not isinstance(transform, Affine):
        transform = Affine(*transform)

    # grid de coordenadas en CRS del raster (X=Easting o lon; Y=Northing o lat)
    H, W = arr.shape
    yy, xx = np.indices((H, W))
```

```
    X = transform.c + xx*transform.a + yy*transform.b
    Y = transform.f + xx*transform.d + yy*transform.e

    try:
        # Transformar bbox WGS84 -> CRS del raster
        r_crs = CRS.from_user_input(raster_crs) if raster_crs is not None else CRS.
        transformer = Transformer.from_crs(CRS.from_epsg(4326), r_crs, always_xy=Tr
        x_w, y_s = transformer.transform(bbox_wgs84["west"],  bbox_wgs84["south"])
        x_e, y_n = transformer.transform(bbox_wgs84["east"],  bbox_wgs84["north"])

        xmin, xmax = (min(x_w, x_e), max(x_w, x_e))
        ymin, ymax = (min(y_s, y_n), max(y_s, y_n))

        inside = (X >= xmin) & (X <= xmax) & (Y >= ymin) & (Y <= ymax)
        out = arr.copy()
        out[~inside] = np.nan
        return out, inside
    except Exception as e:
        # Fallback: si algo falla, no recortamos
        print(f"[mask_with_bbox_any_crs] aviso: no se recortó por bbox ({e}).")
        return arr.copy(), np.ones_like(arr, dtype=bool)

def percentile_stretch(x, p_lo=2, p_hi=98):
    lo, hi = np.nanpercentile(x, (p_lo, p_hi))
    return np.clip((x - lo) / (hi - lo + 1e-6), 0, 1)

print("Funciones utilitarias cargadas.")
```

```
=== BLOQUE 3: Utilidades ===
Funciones utilitarias cargadas.
```

In [9]:

```
print("=== BLOQUE 4: Descarga por fecha (ventana configurable) ===")
from openeo.processes import median as pmedian, max as pmax

# Configura la ventana temporal y el reductor
VENTANA_DIAS = 1
REDUCTOR = "max"

def get_bands_for_date(date_str):
    """Descarga bandas para 'date_str' con ventana configurable.
       Si VENTANA_DIAS==0 usa solo ese día; si >0 aplica reducción temporal según R
    date = pd.to_datetime(date_str).date()
    if VENTANA_DIAS == 0:
        t_start = t_end = date.isoformat()
    else:
        t_start = (date - dt.timedelta(days=VENTANA_DIAS)).isoformat()
        t_end   = (date + dt.timedelta(days=VENTANA_DIAS)).isoformat()

    print(f"[get_bands_for_date] Fecha {date_str} | ventana: {t_start} → {t_end}")

    cube = con.load_collection(
        "SENTINEL2_L2A",
        spatial_extent=lago_atitlan,
        temporal_extent=[t_start, t_end],
        bands=BANDS
    )
```

```
    # Reducir dimensión temporal solo si hay ventana > 0 y se definió REDUCTOR
    if VENTANA_DIAS > 0 and REDUCTOR is not None:
        if REDUCTOR == "median":
            print(" - Reductor temporal: MEDIANA")
            cube = cube.reduce_dimension(dimension="t", reducer=pmedian)
        elif REDUCTOR == "max":
            print(" - Reductor temporal: MÁXIMO")
            cube = cube.reduce_dimension(dimension="t", reducer=pmax)
        else:
            print(" - Reductor no reconocido; no se aplica reducción temporal.")
    else:
        print(" - Sin reducción temporal (día exacto o REDUCTOR=None).")

    # Descargar a archivo temporal (Windows-safe)
    fd, tmp = tempfile.mkstemp(suffix=".tif"); os.close(fd)
    cube.download(tmp)

    with rasterio.open(tmp) as src:
        data = src.read()             # (nbands, H, W)
        desc = src.descriptions
        nodata = src.nodata
        transform = src.transform
        crs = src.crs                 # <<--- NUEVO
        height, width = src.height, src.width
    try: os.remove(tmp)
    except: pass

    print(f"[get_bands_for_date] raster shape: {data.shape} (H={height}, W={width})")
    print(f"[get_bands_for_date] CRS: {crs}")  # <<--- NUEVO

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

    out = {b: band(b) for b in BANDS}
    out["transform"] = transform
    out["crs"] = crs            # <<--- NUEVO
    out["nodata"] = nodata
    return out

print("Función get_bands_for_date lista.")
```

```
=== BLOQUE 4: Descarga por fecha (ventana configurable) ===
Función get_bands_for_date lista.
```

In [10]:

```
print("=== BLOQUE 5: Pipeline por fecha ===")
rows = []
per_date_maps = {}

for d in fechas_buenas:
    print(f"\n--- Procesando {d} ---")
```

```
    try:
        out = get_bands_for_date(d)
        B02,B03,B04,B05,B07,B8A,B08,B11,B12,SCL = [out[k] for k in BANDS]

        # NODATA -> NaN, escalar 0..1
        if out["nodata"] is not None:
            nod = out["nodata"]
            for k in ["B02","B03","B04","B05","B07","B8A","B08","B11","B12"]:
                x = out[k]
                x[x == nod] = np.nan
                out[k] = x

        B02,B03,B04,B05,B07,B8A,B08,B11,B12 = map(scale01, (B02,B03,B04,B05,B07,B8A

        # Recorte exacto al lago (si hay usando coordenadas) SOLO en B04 para defin
        B04, inside = mask_with_bbox_any_crs(B04, out["transform"], out["crs"], lag

        # Aplicar máscara de nubes (SCL) y recorte al resto (incluye B04 ahora)
        def apply_masks(x):
            # x = mask_clouds_with_SCL(x, SCL)
            x[~inside] = np.nan
            return x

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

        # Agua (WBI + filtros). Calculado con bandas ya enmascaradas.
        water = water_mask(B02,B03,B04,B08,B11,B12)
        water = water & inside

        # Vegetación flotante
        fai = FAI(B04, B07, B8A)
        not_floating = ~(fai > 0.08)

        # Índices
        chl = chlorophyll_from_NDCI(B04, B05)
        ndvi_v = ndvi(B08, B04)
        ndwi_v = ndwi(B03, B08)

        # Aplicar máscaras finales
        chl = np.where(water & not_floating, chl, np.nan)
        ndvi_v = np.where(water, ndvi_v, np.nan)
        ndwi_v = np.where(water, ndwi_v, np.nan)

        # Coberturas
        poly_pixels  = int(np.sum(inside))
        water_pixels = int(np.sum(water))
        valid_pixels = int(np.sum(~np.isnan(chl)))

        coverage_pct_poly  = 0.0 if poly_pixels==0  else valid_pixels / poly_pixels
        coverage_pct_water = 0.0 if water_pixels==0 else valid_pixels / water_pixel

        # Prints de diagnóstico
        nan_ratio_chl  = 1 - (valid_pixels / (water_pixels if water_pixels>0 else 1
        nan_ratio_ndvi = 1 - (np.sum(~np.isnan(ndvi_v)) / (water_pixels if water_pi
```

```
        nan_ratio_ndwi = 1 - (np.sum(~np.isnan(ndwi_v)) / (water_pixels if water_pi

        print(f"[{d}] poly_px={poly_pixels:,} | water_px={water_pixels:,} | "
              f"valid_px={valid_pixels:,} | cov_poly={coverage_pct_poly:.1%} | cov_
        print(f"      NaN ratios → chl:{nan_ratio_chl:.2f}  ndvi:{nan_ratio_ndvi:.2

        rows.append({
            "date": pd.to_datetime(d),
            "coverage_pct": coverage_pct_poly,          # para compatibilidad con e
            "coverage_pct_poly": coverage_pct_poly,
            "coverage_pct_water": coverage_pct_water,
            "chl_mean": float(np.nanmean(chl)),
            "chl_median": float(np.nanmedian(chl)),
            "chl_p95": float(np.nanpercentile(chl, 95)) if np.isfinite(np.nanmean(c
            "ndvi_mean": float(np.nanmean(ndvi_v)),
            "ndwi_mean": float(np.nanmean(ndwi_v)),
            "valid_px": valid_pixels,
            "water_px": water_pixels
        })

        # Guardar mapa espacial acotado a 0–50 mg/m³ (para visual comparación)
        per_date_maps[d] = np.clip(chl, 0, 50)

    except Exception as e:
        print(f"[{d}] ERROR: {e}")
        rows.append({"date": pd.to_datetime(d), "error": str(e)})

print("\nTerminó el loop de fechas.")
```

```
=== BLOQUE 5: Pipeline por fecha ===

--- Procesando 2025-02-07 ---
[get_bands_for_date] Fecha 2025-02-07 | ventana: 2025-02-06 → 2025-02-08
 - Reductor temporal: MÁXIMO
[get_bands_for_date] raster shape: (10, 1751, 2759) (H=1751, W=2759)
[get_bands_for_date] CRS: PROJCS["WGS 84 / UTM zone 15N",GEOGCS["WGS 84",DATUM["Worl
d Geodetic System 1984",SPHEROID["WGS 84",6378137,298.257223563]],PRIMEM["Greenwic
h",0],UNIT["degree",0.0174532925199433,AUTHORITY["EPSG","9122"]]],PROJECTION["Transv
erse_Mercator"],PARAMETER["latitude_of_origin",0],PARAMETER["central_meridian",-93],
PARAMETER["scale_factor",0.9996],PARAMETER["false_easting",500000],PARAMETER["false_
northing",0],UNIT["metre",1,AUTHORITY["EPSG","9001"]],AXIS["Easting",EAST],AXIS["Nor
thing",NORTH]]
[2025-02-07] poly_px=4,777,500 | water_px=982,048 | valid_px=981,862 | cov_poly=20.
6% | cov_water=100.0%
      NaN ratios → chl:0.00  ndvi:0.00  ndwi:0.00

--- Procesando 2025-02-10 ---
[get_bands_for_date] Fecha 2025-02-10 | ventana: 2025-02-09 → 2025-02-11
 - Reductor temporal: MÁXIMO
[get_bands_for_date] raster shape: (10, 1751, 2759) (H=1751, W=2759)
[get_bands_for_date] CRS: PROJCS["WGS 84 / UTM zone 15N",GEOGCS["WGS 84",DATUM["Worl
d Geodetic System 1984",SPHEROID["WGS 84",6378137,298.257223563]],PRIMEM["Greenwic
h",0],UNIT["degree",0.0174532925199433,AUTHORITY["EPSG","9122"]]],PROJECTION["Transv
erse_Mercator"],PARAMETER["latitude_of_origin",0],PARAMETER["central_meridian",-93],
PARAMETER["scale_factor",0.9996],PARAMETER["false_easting",500000],PARAMETER["false_
northing",0],UNIT["metre",1,AUTHORITY["EPSG","9001"]],AXIS["Easting",EAST],AXIS["Nor
thing",NORTH]]
[2025-02-10] poly_px=4,777,500 | water_px=1,258 | valid_px=1,258 | cov_poly=0.0% | c
ov_water=100.0%
      NaN ratios → chl:0.00  ndvi:0.00  ndwi:0.00

--- Procesando 2025-02-25 ---
[get_bands_for_date] Fecha 2025-02-25 | ventana: 2025-02-24 → 2025-02-26
 - Reductor temporal: MÁXIMO
[get_bands_for_date] raster shape: (10, 1751, 2759) (H=1751, W=2759)
[get_bands_for_date] CRS: PROJCS["WGS 84 / UTM zone 15N",GEOGCS["WGS 84",DATUM["Worl
d Geodetic System 1984",SPHEROID["WGS 84",6378137,298.257223563]],PRIMEM["Greenwic
h",0],UNIT["degree",0.0174532925199433,AUTHORITY["EPSG","9122"]]],PROJECTION["Transv
erse_Mercator"],PARAMETER["latitude_of_origin",0],PARAMETER["central_meridian",-93],
PARAMETER["scale_factor",0.9996],PARAMETER["false_easting",500000],PARAMETER["false_
northing",0],UNIT["metre",1,AUTHORITY["EPSG","9001"]],AXIS["Easting",EAST],AXIS["Nor
thing",NORTH]]
[2025-02-25] poly_px=4,777,500 | water_px=236,316 | valid_px=236,316 | cov_poly=4.9%
| cov_water=100.0%
      NaN ratios → chl:0.00  ndvi:0.00  ndwi:0.00

--- Procesando 2025-02-27 ---
[get_bands_for_date] Fecha 2025-02-27 | ventana: 2025-02-26 → 2025-02-28
 - Reductor temporal: MÁXIMO
[get_bands_for_date] raster shape: (10, 1751, 2759) (H=1751, W=2759)
[get_bands_for_date] CRS: PROJCS["WGS 84 / UTM zone 15N",GEOGCS["WGS 84",DATUM["Worl
d Geodetic System 1984",SPHEROID["WGS 84",6378137,298.257223563]],PRIMEM["Greenwic
h",0],UNIT["degree",0.0174532925199433,AUTHORITY["EPSG","9122"]]],PROJECTION["Transv
erse_Mercator"],PARAMETER["latitude_of_origin",0],PARAMETER["central_meridian",-93],
PARAMETER["scale_factor",0.9996],PARAMETER["false_easting",500000],PARAMETER["false_
```

```
northing",0],UNIT["metre",1,AUTHORITY["EPSG","9001"]],AXIS["Easting",EAST],AXIS["Nor
thing",NORTH]]
[2025-02-27] poly_px=4,777,500 | water_px=1,284,646 | valid_px=1,284,579 | cov_poly=
26.9% | cov_water=100.0%
      NaN ratios → chl:0.00  ndvi:0.00  ndwi:0.00

--- Procesando 2025-03-02 ---
[get_bands_for_date] Fecha 2025-03-02 | ventana: 2025-03-01 → 2025-03-03
 - Reductor temporal: MÁXIMO
[get_bands_for_date] raster shape: (10, 1751, 2759) (H=1751, W=2759)
[get_bands_for_date] CRS: PROJCS["WGS 84 / UTM zone 15N",GEOGCS["WGS 84",DATUM["Worl
d Geodetic System 1984",SPHEROID["WGS 84",6378137,298.257223563]],PRIMEM["Greenwic
h",0],UNIT["degree",0.0174532925199433,AUTHORITY["EPSG","9122"]]],PROJECTION["Transv
erse_Mercator"],PARAMETER["latitude_of_origin",0],PARAMETER["central_meridian",-93],
PARAMETER["scale_factor",0.9996],PARAMETER["false_easting",500000],PARAMETER["false_
northing",0],UNIT["metre",1,AUTHORITY["EPSG","9001"]],AXIS["Easting",EAST],AXIS["Nor
thing",NORTH]]
[2025-03-02] poly_px=4,777,500 | water_px=136,144 | valid_px=136,144 | cov_poly=2.8%
| cov_water=100.0%
      NaN ratios → chl:0.00  ndvi:0.00  ndwi:0.00

--- Procesando 2025-03-04 ---
[get_bands_for_date] Fecha 2025-03-04 | ventana: 2025-03-03 → 2025-03-05
 - Reductor temporal: MÁXIMO
[get_bands_for_date] raster shape: (10, 1751, 2759) (H=1751, W=2759)
[get_bands_for_date] CRS: PROJCS["WGS 84 / UTM zone 15N",GEOGCS["WGS 84",DATUM["Worl
d Geodetic System 1984",SPHEROID["WGS 84",6378137,298.257223563]],PRIMEM["Greenwic
h",0],UNIT["degree",0.0174532925199433,AUTHORITY["EPSG","9122"]]],PROJECTION["Transv
erse_Mercator"],PARAMETER["latitude_of_origin",0],PARAMETER["central_meridian",-93],
PARAMETER["scale_factor",0.9996],PARAMETER["false_easting",500000],PARAMETER["false_
northing",0],UNIT["metre",1,AUTHORITY["EPSG","9001"]],AXIS["Easting",EAST],AXIS["Nor
thing",NORTH]]
[2025-03-04] poly_px=4,777,500 | water_px=712,002 | valid_px=711,994 | cov_poly=14.
9% | cov_water=100.0%
      NaN ratios → chl:0.00  ndvi:0.00  ndwi:0.00

--- Procesando 2025-03-07 ---
[get_bands_for_date] Fecha 2025-03-07 | ventana: 2025-03-06 → 2025-03-08
 - Reductor temporal: MÁXIMO
[get_bands_for_date] raster shape: (10, 1751, 2759) (H=1751, W=2759)
[get_bands_for_date] CRS: PROJCS["WGS 84 / UTM zone 15N",GEOGCS["WGS 84",DATUM["Worl
d Geodetic System 1984",SPHEROID["WGS 84",6378137,298.257223563]],PRIMEM["Greenwic
h",0],UNIT["degree",0.0174532925199433,AUTHORITY["EPSG","9122"]]],PROJECTION["Transv
erse_Mercator"],PARAMETER["latitude_of_origin",0],PARAMETER["central_meridian",-93],
PARAMETER["scale_factor",0.9996],PARAMETER["false_easting",500000],PARAMETER["false_
northing",0],UNIT["metre",1,AUTHORITY["EPSG","9001"]],AXIS["Easting",EAST],AXIS["Nor
thing",NORTH]]
[2025-03-07] poly_px=4,777,500 | water_px=15,089 | valid_px=15,089 | cov_poly=0.3% |
cov_water=100.0%
      NaN ratios → chl:0.00  ndvi:0.00  ndwi:0.00

--- Procesando 2025-03-09 ---
[get_bands_for_date] Fecha 2025-03-09 | ventana: 2025-03-08 → 2025-03-10
 - Reductor temporal: MÁXIMO
[get_bands_for_date] raster shape: (10, 1751, 2759) (H=1751, W=2759)
[get_bands_for_date] CRS: PROJCS["WGS 84 / UTM zone 15N",GEOGCS["WGS 84",DATUM["Worl
```

```
d Geodetic System 1984",SPHEROID["WGS 84",6378137,298.257223563]],PRIMEM["Greenwic
h",0],UNIT["degree",0.0174532925199433,AUTHORITY["EPSG","9122"]]],PROJECTION["Transv
erse_Mercator"],PARAMETER["latitude_of_origin",0],PARAMETER["central_meridian",-93],
PARAMETER["scale_factor",0.9996],PARAMETER["false_easting",500000],PARAMETER["false_
northing",0],UNIT["metre",1,AUTHORITY["EPSG","9001"]],AXIS["Easting",EAST],AXIS["Nor
thing",NORTH]]
[2025-03-09] poly_px=4,777,500 | water_px=1,196,953 | valid_px=1,196,953 | cov_poly=
25.1% | cov_water=100.0%
      NaN ratios → chl:0.00  ndvi:0.00  ndwi:0.00

--- Procesando 2025-03-12 ---
[get_bands_for_date] Fecha 2025-03-12 | ventana: 2025-03-11 → 2025-03-13
 - Reductor temporal: MÁXIMO
[get_bands_for_date] raster shape: (10, 1751, 2759) (H=1751, W=2759)
[get_bands_for_date] CRS: PROJCS["WGS 84 / UTM zone 15N",GEOGCS["WGS 84",DATUM["Worl
d Geodetic System 1984",SPHEROID["WGS 84",6378137,298.257223563]],PRIMEM["Greenwic
h",0],UNIT["degree",0.0174532925199433,AUTHORITY["EPSG","9122"]]],PROJECTION["Transv
erse_Mercator"],PARAMETER["latitude_of_origin",0],PARAMETER["central_meridian",-93],
PARAMETER["scale_factor",0.9996],PARAMETER["false_easting",500000],PARAMETER["false_
northing",0],UNIT["metre",1,AUTHORITY["EPSG","9001"]],AXIS["Easting",EAST],AXIS["Nor
thing",NORTH]]
[2025-03-12] poly_px=4,777,500 | water_px=22,168 | valid_px=22,168 | cov_poly=0.5% |
cov_water=100.0%
      NaN ratios → chl:0.00  ndvi:0.00  ndwi:0.00

--- Procesando 2025-03-14 ---
[get_bands_for_date] Fecha 2025-03-14 | ventana: 2025-03-13 → 2025-03-15
 - Reductor temporal: MÁXIMO
[get_bands_for_date] raster shape: (10, 1751, 2759) (H=1751, W=2759)
[get_bands_for_date] CRS: PROJCS["WGS 84 / UTM zone 15N",GEOGCS["WGS 84",DATUM["Worl
d Geodetic System 1984",SPHEROID["WGS 84",6378137,298.257223563]],PRIMEM["Greenwic
h",0],UNIT["degree",0.0174532925199433,AUTHORITY["EPSG","9122"]]],PROJECTION["Transv
erse_Mercator"],PARAMETER["latitude_of_origin",0],PARAMETER["central_meridian",-93],
PARAMETER["scale_factor",0.9996],PARAMETER["false_easting",500000],PARAMETER["false_
northing",0],UNIT["metre",1,AUTHORITY["EPSG","9001"]],AXIS["Easting",EAST],AXIS["Nor
thing",NORTH]]
[2025-03-14] poly_px=4,777,500 | water_px=1,167,356 | valid_px=1,167,355 | cov_poly=
24.4% | cov_water=100.0%
      NaN ratios → chl:0.00  ndvi:0.00  ndwi:0.00

--- Procesando 2025-03-19 ---
[get_bands_for_date] Fecha 2025-03-19 | ventana: 2025-03-18 → 2025-03-20
 - Reductor temporal: MÁXIMO
[get_bands_for_date] raster shape: (10, 1751, 2759) (H=1751, W=2759)
[get_bands_for_date] CRS: PROJCS["WGS 84 / UTM zone 15N",GEOGCS["WGS 84",DATUM["Worl
d Geodetic System 1984",SPHEROID["WGS 84",6378137,298.257223563]],PRIMEM["Greenwic
h",0],UNIT["degree",0.0174532925199433,AUTHORITY["EPSG","9122"]]],PROJECTION["Transv
erse_Mercator"],PARAMETER["latitude_of_origin",0],PARAMETER["central_meridian",-93],
PARAMETER["scale_factor",0.9996],PARAMETER["false_easting",500000],PARAMETER["false_
northing",0],UNIT["metre",1,AUTHORITY["EPSG","9001"]],AXIS["Easting",EAST],AXIS["Nor
thing",NORTH]]
[2025-03-19] poly_px=4,777,500 | water_px=166,296 | valid_px=166,294 | cov_poly=3.5%
| cov_water=100.0%
      NaN ratios → chl:0.00  ndvi:0.00  ndwi:0.00

--- Procesando 2025-03-22 ---
```

```
[get_bands_for_date] Fecha 2025-03-22 | ventana: 2025-03-21 → 2025-03-23
 - Reductor temporal: MÁXIMO
[get_bands_for_date] raster shape: (10, 1751, 2759) (H=1751, W=2759)
[get_bands_for_date] CRS: PROJCS["WGS 84 / UTM zone 15N",GEOGCS["WGS 84",DATUM["Worl
d Geodetic System 1984",SPHEROID["WGS 84",6378137,298.257223563]],PRIMEM["Greenwic
h",0],UNIT["degree",0.0174532925199433,AUTHORITY["EPSG","9122"]]],PROJECTION["Transv
erse_Mercator"],PARAMETER["latitude_of_origin",0],PARAMETER["central_meridian",-93],
PARAMETER["scale_factor",0.9996],PARAMETER["false_easting",500000],PARAMETER["false_
northing",0],UNIT["metre",1,AUTHORITY["EPSG","9001"]],AXIS["Easting",EAST],AXIS["Nor
thing",NORTH]]
[2025-03-22] poly_px=4,777,500 | water_px=1,497 | valid_px=1,495 | cov_poly=0.0% | c
ov_water=99.9%
      NaN ratios → chl:0.00  ndvi:0.00  ndwi:0.00

--- Procesando 2025-03-24 ---
[get_bands_for_date] Fecha 2025-03-24 | ventana: 2025-03-23 → 2025-03-25
 - Reductor temporal: MÁXIMO
[get_bands_for_date] raster shape: (10, 1751, 2759) (H=1751, W=2759)
[get_bands_for_date] CRS: PROJCS["WGS 84 / UTM zone 15N",GEOGCS["WGS 84",DATUM["Worl
d Geodetic System 1984",SPHEROID["WGS 84",6378137,298.257223563]],PRIMEM["Greenwic
h",0],UNIT["degree",0.0174532925199433,AUTHORITY["EPSG","9122"]]],PROJECTION["Transv
erse_Mercator"],PARAMETER["latitude_of_origin",0],PARAMETER["central_meridian",-93],
PARAMETER["scale_factor",0.9996],PARAMETER["false_easting",500000],PARAMETER["false_
northing",0],UNIT["metre",1,AUTHORITY["EPSG","9001"]],AXIS["Easting",EAST],AXIS["Nor
thing",NORTH]]
[2025-03-24] poly_px=4,777,500 | water_px=1,161,477 | valid_px=1,161,477 | cov_poly=
24.3% | cov_water=100.0%
      NaN ratios → chl:0.00  ndvi:0.00  ndwi:0.00

--- Procesando 2025-03-26 ---
[get_bands_for_date] Fecha 2025-03-26 | ventana: 2025-03-25 → 2025-03-27
 - Reductor temporal: MÁXIMO
[get_bands_for_date] raster shape: (10, 1751, 2759) (H=1751, W=2759)
[get_bands_for_date] CRS: PROJCS["WGS 84 / UTM zone 15N",GEOGCS["WGS 84",DATUM["Worl
d Geodetic System 1984",SPHEROID["WGS 84",6378137,298.257223563]],PRIMEM["Greenwic
h",0],UNIT["degree",0.0174532925199433,AUTHORITY["EPSG","9122"]]],PROJECTION["Transv
erse_Mercator"],PARAMETER["latitude_of_origin",0],PARAMETER["central_meridian",-93],
PARAMETER["scale_factor",0.9996],PARAMETER["false_easting",500000],PARAMETER["false_
northing",0],UNIT["metre",1,AUTHORITY["EPSG","9001"]],AXIS["Easting",EAST],AXIS["Nor
thing",NORTH]]
[2025-03-26] poly_px=4,777,500 | water_px=3,815 | valid_px=3,815 | cov_poly=0.1% | c
ov_water=100.0%
      NaN ratios → chl:0.00  ndvi:0.00  ndwi:0.00

--- Procesando 2025-04-03 ---
[get_bands_for_date] Fecha 2025-04-03 | ventana: 2025-04-02 → 2025-04-04
 - Reductor temporal: MÁXIMO
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[10], line 8
      6 print(f"\n--- Procesando {d} ---")
      7 try:
----> 8     out = get_bands_for_date(d)
      9     B02,B03,B04,B05,B07,B8A,B08,B11,B12,SCL = [out[k] for k in BANDS]
     11     # NODATA -> NaN, escalar 0..1

Cell In[9], line 42, in get_bands_for_date(date_str)
     40 # Descargar a archivo temporal (Windows-safe)
     41 fd, tmp = tempfile.mkstemp(suffix=".tif"); os.close(fd)
---> 42 cube.download(tmp)
     44 with rasterio.open(tmp) as src:
     45     data = src.read()             # (nbands, H, W)

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\openeo\rest\datacube.py:2508, in DataCube.download(self, outputfile, format, options, validate, auto_add_save_result, additional, job_options, on_response_headers)
   2506 else:
   2507     res = self
-> 2508 return self._connection.download(
   2509     res.flat_graph(),
   2510     outputfile=outputfile,
   2511     validate=validate,
   2512     additional=additional,
   2513     job_options=job_options,
   2514     on_response_headers=on_response_headers,
   2515 )

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\openeo\rest\connection.py:1699, in Connection.download(self, graph, outputfile, timeout, validate, chunk_size, additional, job_options, on_response_headers)
   1697     ensure_dir(target.parent)
   1698     with target.open(mode="wb") as f:
-> 1699         for chunk in response.iter_content(chunk_size=chunk_size):
   1700             f.write(chunk)
   1701     # TODO: return target path instead of None? Or return a generic result wrapper?
   1702 else:

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\requests\models.py:820, in Response.iter_content.<locals>.generate()
    818 if hasattr(self.raw, "stream"):
    819     try:
--> 820         yield from self.raw.stream(chunk_size, decode_content=True)
    821     except ProtocolError as e:
    822         raise ChunkedEncodingError(e)

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\urllib3\response.py:1043, in HTTPResponse.stream(self, amt, decode_content)
   1041 else:
```

```
   1042     while not is_fp_closed(self._fp) or len(self._decoded_buffer) > 0:
-> 1043         data = self.read(amt=amt, decode_content=decode_content)
   1045         if data:
   1046             yield data

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\urllib3\response.py:935, in HTTPResponse.read(self, amt, decode_content, cache_content)
    932     if len(self._decoded_buffer) >= amt:
    933         return self._decoded_buffer.get(amt)
--> 935 data = self._raw_read(amt)
    937 flush_decoder = amt is None or (amt != 0 and not data)
    939 if not data and len(self._decoded_buffer) == 0:

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\urllib3\response.py:862, in HTTPResponse._raw_read(self, amt, read1)
    859 fp_closed = getattr(self._fp, "closed", False)
    861 with self._error_catcher():
--> 862     data = self._fp_read(amt, read1=read1) if not fp_closed else b""
    863     if amt is not None and amt != 0 and not data:
    864         # Platform-specific: Buggy versions of Python.
    865         # Close the connection when no data is returned
   (...)
    870         # not properly close the connection in all cases. There is
    871         # no harm in redundantly calling close.
    872         self._fp.close()

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\urllib3\response.py:845, in HTTPResponse._fp_read(self, amt, read1)
    842     return self._fp.read1(amt) if amt is not None else self._fp.read1()
    843 else:
    844     # StringIO doesn't like amt=None
--> 845     return self._fp.read(amt) if amt is not None else self._fp.read()

File C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.9_3.9.3568.0_x64__qbz5n2kfra8p0\lib\http\client.py:463, in HTTPResponse.read(self, amt)
    460 if amt is not None:
    461     # Amount is given, implement using readinto
    462     b = bytearray(amt)
--> 463     n = self.readinto(b)
    464     return memoryview(b)[:n].tobytes()
    465 else:
    466     # Amount is not given (unbounded read) so we must check self.length
    467     # and self.chunked

File C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.9_3.9.3568.0_x64__qbz5n2kfra8p0\lib\http\client.py:507, in HTTPResponse.readinto(self, b)
    502         b = memoryview(b)[0:self.length]
    504 # we do not use _safe_read() here because this may be a .will_close
    505 # connection, and the user is reading more bytes than will be provided
    506 # (for example, reading in 1k chunks)
--> 507 n = self.fp.readinto(b)
    508 if not n and b:
    509     # Ideally, we would raise IncompleteRead if the content-length
```

```
    510     # wasn't satisfied, but it might break compatibility.
    511     self._close_conn()

File C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.9_3.9.3568.0_x64
__qbz5n2kfra8p0\lib\socket.py:704, in SocketIO.readinto(self, b)
    702 while True:
    703     try:
--> 704         return self._sock.recv_into(b)
    705     except timeout:
    706         self._timeout_occurred = True

File C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.9_3.9.3568.0_x64
__qbz5n2kfra8p0\lib\ssl.py:1242, in SSLSocket.recv_into(self, buffer, nbytes, flags)
   1238     if flags != 0:
   1239         raise ValueError(
   1240           "non-zero flags not allowed in calls to recv_into() on %s" %
   1241           self.__class__)
-> 1242     return self.read(nbytes, buffer)
   1243 else:
   1244     return super().recv_into(buffer, nbytes, flags)

File C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.9_3.9.3568.0_x64
__qbz5n2kfra8p0\lib\ssl.py:1100, in SSLSocket.read(self, len, buffer)
   1098 try:
   1099     if buffer is not None:
-> 1100         return self._sslobj.read(len, buffer)
   1101     else:
   1102         return self._sslobj.read(len)

KeyboardInterrupt:
```

Se interrumpio la obtencion de datos ya que tomo demasiado tiempo en obtener los datos de 15 fechas, pero se consideraron más que suficientes para tener una idea general de la clorofila - cianobacteria. De ser necesario, se pueden obtener solo los datos de ciertas fechas basadas en las imagenes obtenidas en el documento de imagenes.ipynb

In [11]:
```
print("=== BLOQUE 6: Tabla + checks ===")
df = pd.DataFrame(rows).sort_values("date")
for col in ["coverage_pct_water","chl_mean","ndvi_mean","ndwi_mean","chl_median","c
    if col not in df.columns:
        df[col] = np.nan

print("Primeras filas:")
print(df.head(5).to_string(index=False))

if "error" in df.columns and df["error"].notna().any():
    print("\nErrores por fecha:")
    print(df.loc[df["error"].notna(), ["date","error"]].to_string(index=False))
else:
    print("\nSin errores reportados en fechas.")

print("\nResumen cobertura:")
print(df[["coverage_pct_water"]].describe())
```

```
df.to_csv("atitlan_indices_todas_las_fechas.csv", index=False)
```

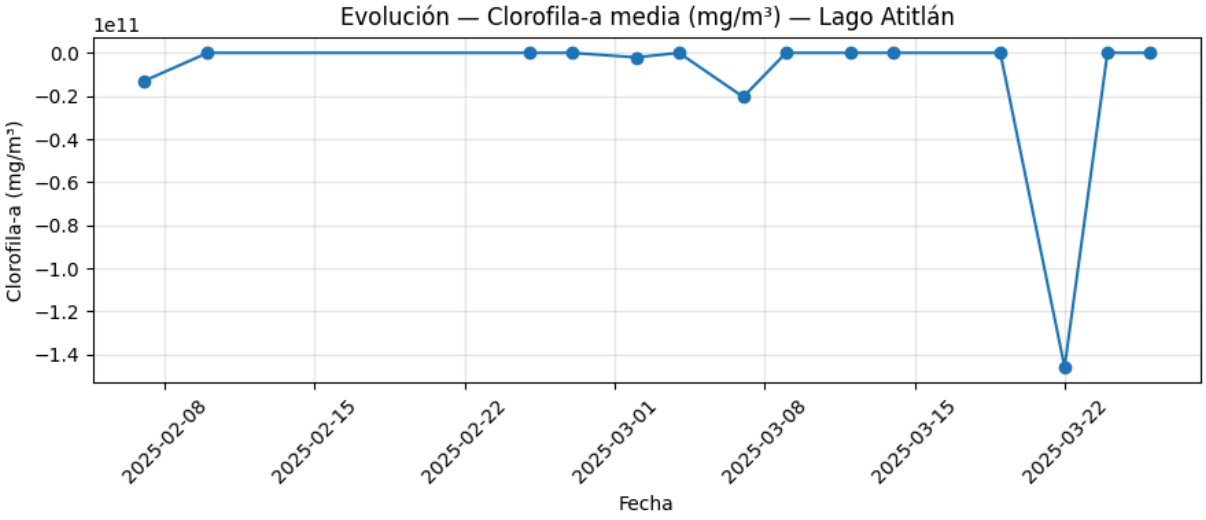
```
=== BLOQUE 6: Tabla + checks ===
Primeras filas:
      date  coverage_pct  coverage_pct_poly  coverage_pct_water      chl_mean  chl_m
edian     chl_p95  ndvi_mean  ndwi_mean  valid_px  water_px
2025-02-07      0.205518           0.205518            0.999811 -1.318510e+10    4.7
46356 1204.932007   0.904062   9.587302    981862    982048
2025-02-10      0.000263           0.000263            1.000000  4.040736e+00    4.2
60046    4.732763  -0.135770   0.436340      1258      1258
2025-02-25      0.049464           0.049464            1.000000  5.287579e-01    3.9
63018    4.691946   0.308929   0.595625    236316    236316
2025-02-27      0.268881           0.268881            0.999948  3.224710e+00    3.5
96182    4.517667  -0.143003   0.403314   1284579   1284646
2025-03-02      0.028497           0.028497            1.000000 -2.103549e+09    3.1
72209   10.806772   0.289753   0.760607    136144    136144

Sin errores reportados en fechas.

Resumen cobertura:
       coverage_pct_water
count           14.000000
mean             0.999886
std              0.000355
min              0.998664
25%              0.999988
50%              1.000000
75%              1.000000
max              1.000000
```

In [ ]:

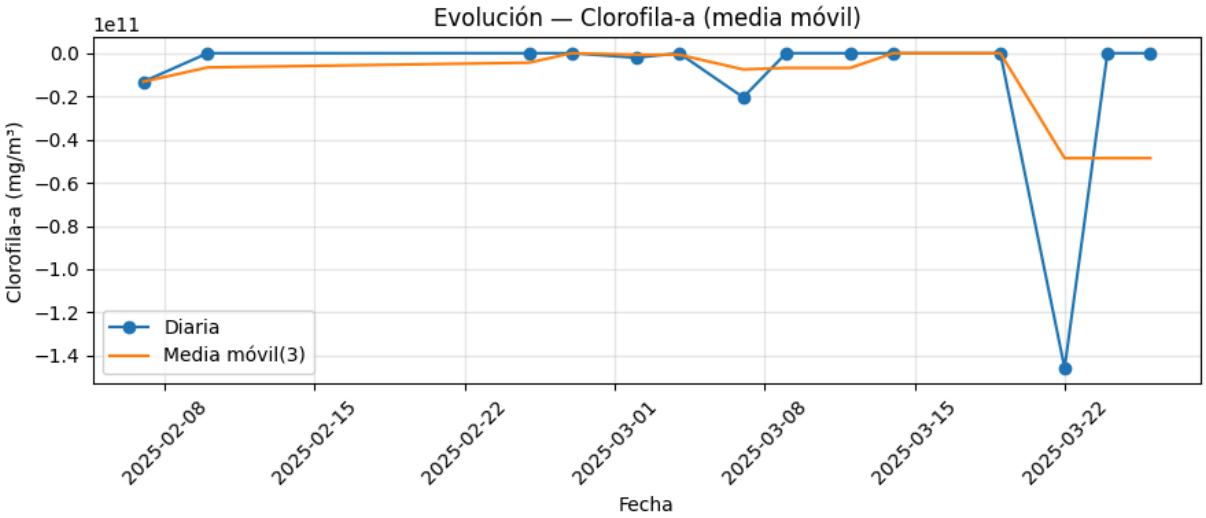
```
print("=== BLOQUE 7: Serie temporal + picos ===")
df_good = df[df["coverage_pct_water"].fillna(0) >= COVERAGE_THRESHOLD].copy()

if df_good.empty:
    print(f"⚠️ No hay fechas con cobertura >= {COVERAGE_THRESHOLD:.0%}. "
          "Prueba bajar temporalmente el umbral o revisa SCL/fechas.")
else:
    print(f"Fechas consideradas (cobertura >= {COVERAGE_THRESHOLD:.0%}): {len(df_go
    print(df_good[["date","coverage_pct_water","chl_mean","ndvi_mean","ndwi_mean"]]

    df_good.to_csv("atitlan_indices_temporales.csv", index=False)

    plt.figure(figsize=(9,4))
    plt.plot(df_good["date"], df_good["chl_mean"], marker="o")
    plt.title("Evolución — Clorofila-a media (mg/m³) — Lago Atitlán")
    plt.xlabel("Fecha"); plt.ylabel("Clorofila-a (mg/m³)")
    plt.grid(True, alpha=0.3); plt.xticks(rotation=45); plt.tight_layout()
    plt.show()
    plt.savefig("atitlan_serie_chl.png", dpi=200)

    df_good = df_good.sort_values("date")
    plt.figure(figsize=(9,4))
    plt.plot(df_good["date"], df_good["chl_mean"], marker="o", label="Diaria")
    plt.plot(df_good["date"], df_good["chl_mean"].rolling(3, min_periods=1).mean(),
    plt.title("Evolución — Clorofila-a (media móvil)")
```

```
    plt.xlabel("Fecha"); plt.ylabel("Clorofila-a (mg/m³)")
    plt.grid(True, alpha=0.3); plt.legend(); plt.xticks(rotation=45); plt.tight_lay
    plt.show()
    plt.savefig("atitlan_serie_chl_movil.png", dpi=200)

    plt.figure(figsize=(9,4))
    plt.plot(df_good["date"], df_good["chl_mean"], marker="o")
    plt.title("Evolución — Clorofila-a media (mg/m³) — Lago Atitlán")
    plt.xlabel("Fecha"); plt.ylabel("Clorofila-a (mg/m³)")
    plt.grid(True, alpha=0.3); plt.xticks(rotation=45); plt.tight_layout()
    plt.show()

    top_peaks = df_good.nlargest(3, "chl_mean")[["date","chl_mean","coverage_pct_wa
    print("\nTop-3 picos por clorofila media:")
    print(top_peaks.to_string(index=False))
```

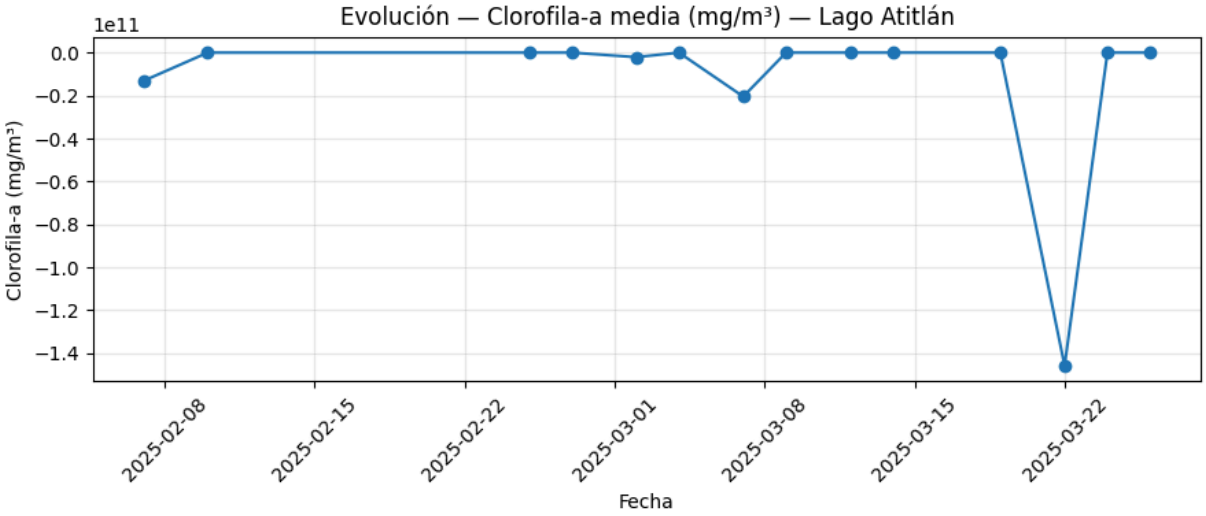
```
=== BLOQUE 7: Serie temporal + picos ===
Fechas consideradas (cobertura >= 40%): 14
         date  coverage_pct_water      chl_mean  ndvi_mean  ndwi_mean
0  2025-02-07            0.999811 -1.318510e+10   0.904062   9.587302
1  2025-02-10            1.000000  4.040736e+00  -0.135770   0.436340
2  2025-02-25            1.000000  5.287579e-01   0.308929   0.595625
3  2025-02-27            0.999948  3.224710e+00  -0.143003   0.403314
4  2025-03-02            1.000000 -2.103549e+09   0.289753   0.760607
5  2025-03-04            0.999989  1.684825e+00  -0.047707   0.383711
6  2025-03-07            1.000000 -2.045718e+10   0.660346   1.152470
7  2025-03-09            1.000000  3.153712e+00  -0.133639   0.428933
8  2025-03-12            1.000000  3.706319e+00  -0.004812   0.166499
9  2025-03-14            0.999999  2.433192e+00  -0.107496   0.462303
10 2025-03-19            0.999988  3.396026e+00  -0.111999   0.308703
11 2025-03-22            0.998664 -1.457839e+11  19.053434  21.067621
12 2025-03-24            1.000000  2.667520e+00  -0.131594   0.462572
13 2025-03-26            1.000000  3.049130e+00  -0.025212   0.475355
```




```
<Figure size 640x480 with 0 Axes>
```




```
<Figure size 640x480 with 0 Axes>
```

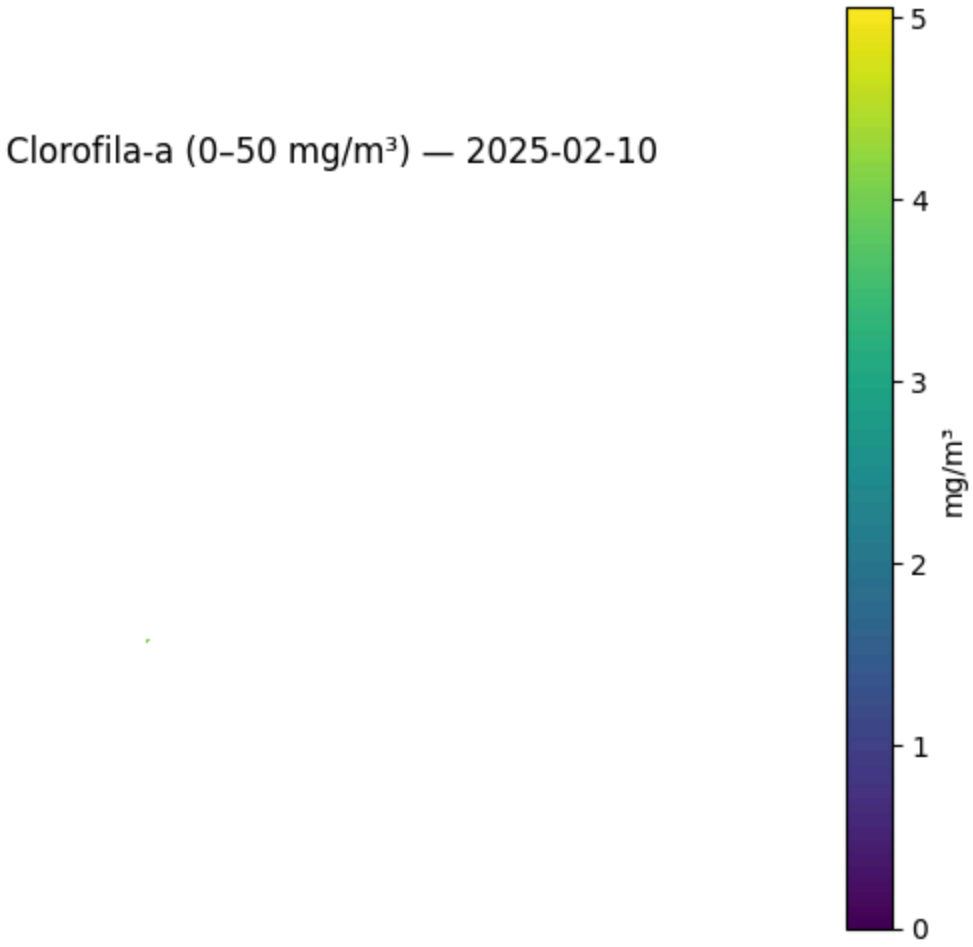



```
Top-3 picos por clorofila media:
      date  chl_mean  coverage_pct_water
2025-02-10  4.040736            1.000000
2025-03-12  3.706319            1.000000
2025-03-19  3.396026            0.999988
```

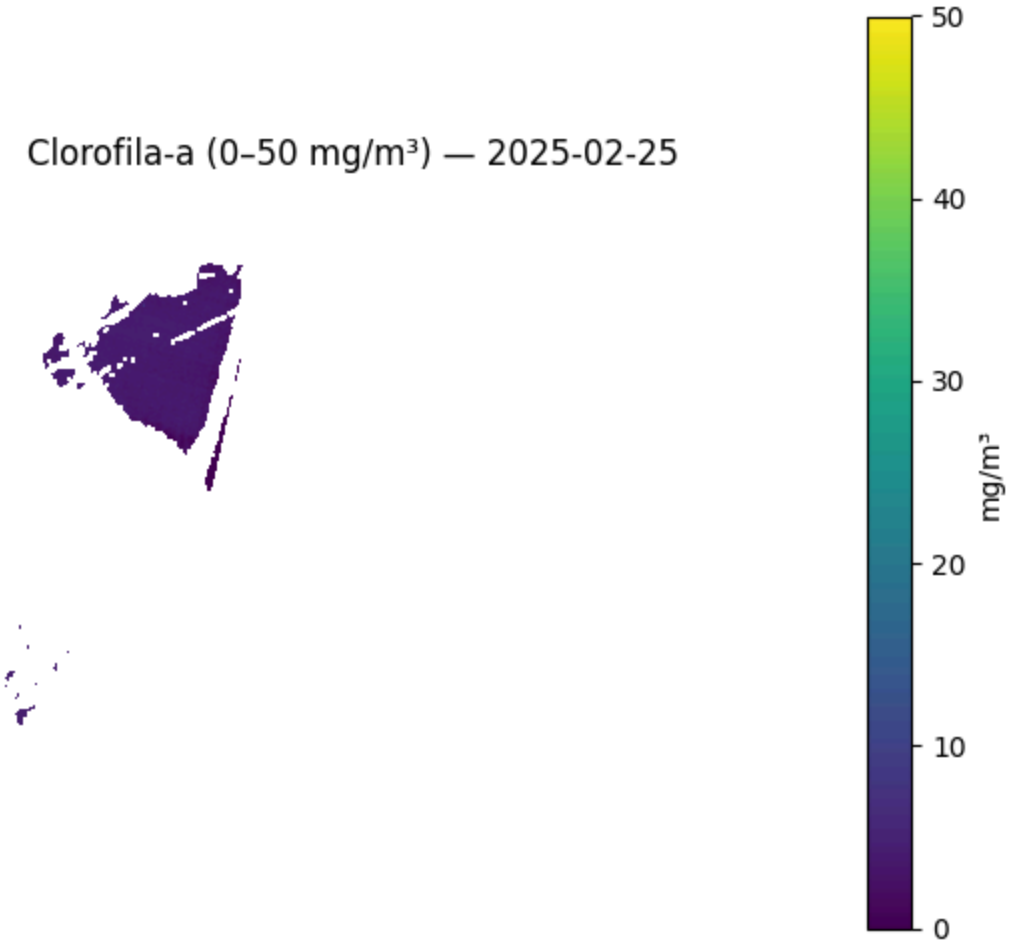
In [ ]:
```python
print("=== BLOQUE 8: Mapas y correlaciones ===")

# Mapas (dos mejores coberturas)
if not df_good.empty:
    best_two = df_good.nlargest(2, "coverage_pct_water")["date"].dt.strftime("%Y-%m
    if len(best_two) < 2:
        print(f"Solo {len(best_two)} fecha(s) con buena cobertura: {best_two}")
    else:
        print(f"Mejores coberturas: {best_two}")
    for d in best_two:
        arr = per_date_maps.get(d)
        if arr is None:
            print(f"[Mapa] No hay raster cache para {d}")
            continue
        plt.figure(figsize=(6,5))
        im = plt.imshow(arr, cmap="viridis")
        plt.title(f"Clorofila-a (0–50 mg/m³) — {d}")
```

```
        plt.axis("off"); plt.colorbar(im, label="mg/m³")
        plt.tight_layout(); plt.show()
        plt.savefig(f"atitlan_mapa_{d}.png", dpi=200)   # <-- guarda PNG
else:
    print("No hay fechas con buena cobertura para mapas comparativos.")

# Correlaciones (chl vs NDVI y NDWI)
from scipy.stats import pearsonr, spearmanr

def corr_report(x, y, labelx, labely, fname_stub=None):
    good = (~np.isnan(x)) & (~np.isnan(y))
    n = good.sum()
    if n >= 3:
        pr, pp = pearsonr(x[good], y[good])
        sr, sp = spearmanr(x[good], y[good])
        print(f"\nCorrelación {labelx} vs {labely} (n={n})")
        print(f"  Pearson r={pr:.3f} (p={pp:.3g}) | Spearman ρ={sr:.3f} (p={sp:.3g}
        plt.figure(figsize=(5,4))
        plt.scatter(x[good], y[good])
        m, b = np.polyfit(x[good], y[good], 1)
        xx = np.linspace(np.nanmin(x[good]), np.nanmax(x[good]), 100)
        plt.plot(xx, m*xx + b)
        plt.xlabel(labelx); plt.ylabel(labely)
        plt.title(f"{labelx} vs {labely}")
        plt.grid(True, alpha=0.3); plt.tight_layout(); plt.show()
        if fname_stub:
            plt.savefig(f"{fname_stub}.png", dpi=200)   # <-- guarda PNG
    else:
        print(f"Correlación {labelx} vs {labely}: datos insuficientes (n={n}).")

if not df_good.empty:
    print(f"Fechas válidas para correlación: {len(df_good)}")
    x_chl  = df_good["chl_mean"].values.astype(float)
    x_ndvi = df_good["ndvi_mean"].values.astype(float)
    x_ndwi = df_good["ndwi_mean"].values.astype(float)
    corr_report(x_chl, x_ndvi, "Clorofila media (mg/m³)", "NDVI medio", fname_stub=
    corr_report(x_chl, x_ndwi, "Clorofila media (mg/m³)", "NDWI medio", fname_stub=
else:
    print("Saltando correlaciones por falta de fechas válidas.")
```

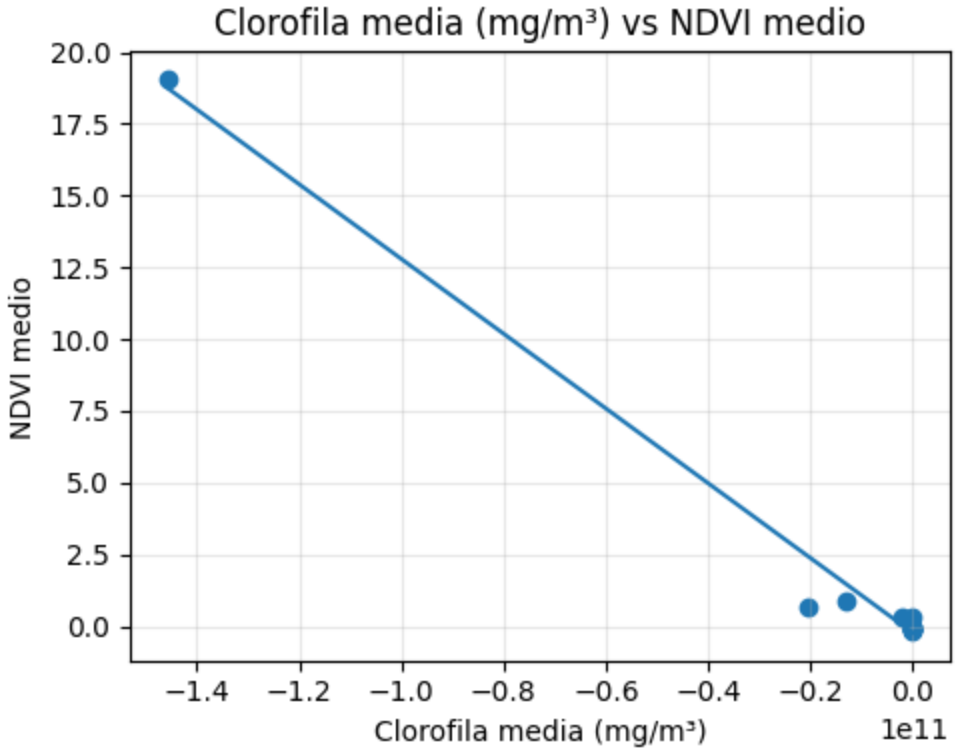
```
=== BLOQUE 8: Mapas y correlaciones ===
Mejores coberturas: ['2025-02-10', '2025-02-25']
```



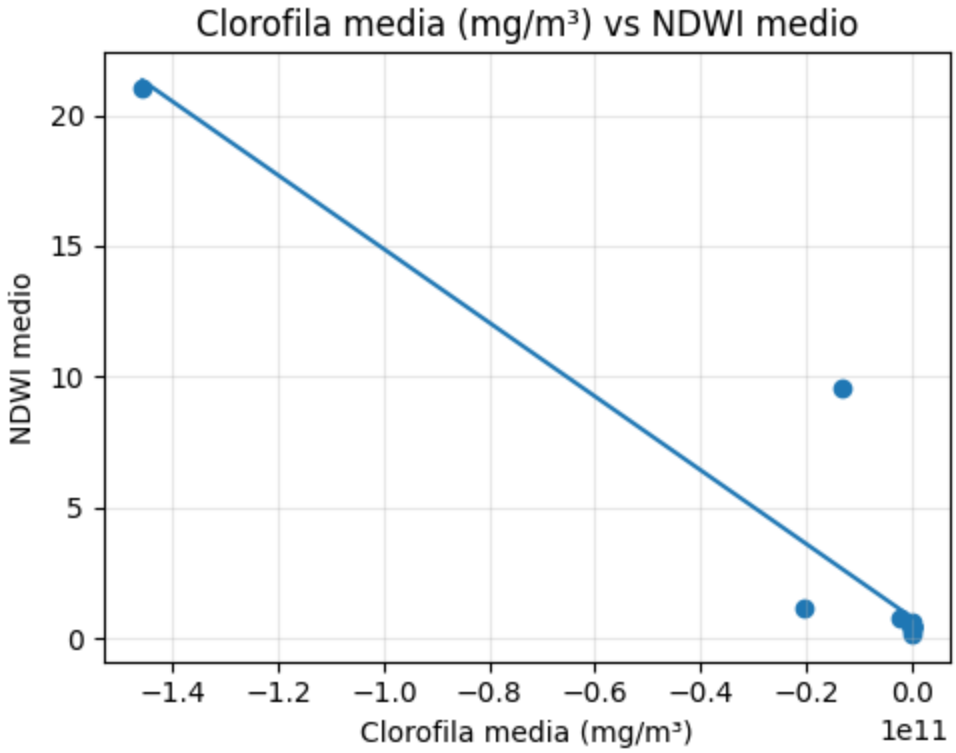
<Figure size 640x480 with 0 Axes>




```
Fechas válidas para correlación: 14

Correlación Clorofila media (mg/m³) vs NDVI medio (n=14)
  Pearson r=-0.994 (p=7.81e-13) | Spearman ρ=-0.798 (p=0.000628)
<Figure size 640x480 with 0 Axes>
```




```
Correlación Clorofila media (mg/m³) vs NDWI medio (n=14)
  Pearson r=-0.934 (p=9.94e-07) | Spearman ρ=-0.833 (p=0.000217)
<Figure size 640x480 with 0 Axes>
```




```
<Figure size 640x480 with 0 Axes>
```

En este caso no nos enfocamos en la busqueda de las imagenes, ya que mas que nada estamos buscando los indices de la clorofila dentro de estas imagenes. Dando un poco de contexto en la manera en la que funciona este codigo, va tomando las fechas que se nos dieron con menor nubosidad, verifica que tenga datos correctos (como por ejemplo que

dentro del bloque de la imagen haya pixeles de agua). Utilizando las bandas de estas fechas, tomamos la informacion relacionada con una mascara de agua que nos dice el procentaje de clorofila en el agua. Basandonos en estos datos podemos calcular la cantidad de cianobacteria en el lago de atitlan.

Por que buscamos la clorofila-a? Esto es ya que es un pigmento que esta presente en las algas, o en este caso en la cianobacteria. Si hay un alto valor de clorofila-a, puede indicar que hay un aumento o proliferacion de cianobacterias. Esto no siempre es cierto, pero solo basandonos en Sentinel-2, puede llegar a ser un buen indicador

En el caso de NDVI, que es un indice de vegetación, un nivel alto nos puede indicar cosas como vegetacion flotante o hasta contaminacion en ciertos casos. En este caso, tenemos un indice negativo, que nos puede decir que no hay mucha vegetacion flotante o que hay pigmentos que interfieren con la toma de datos.

Por ultimo, viendo el NDWI, que mide el contenido de agua en la superficie, al tener una correlacion negativa, nos dice que cuando la concentracion de clorofila-a aumenta, el agua disminuye. Esto pude darse por confusiones en los colores resaltados tomandolos como agua limpia.