Kraków, 08.02.2026 r.

“Urban Heat Island Kraków”

Obliczenia w chmurze

Skład zespołu:

Emilia Brandys, nr. albumu: 409963

Zuzanna Sabat, nr. albumu: 410176

1. Cel projektu

Celem projektu jest wizualizacja miejsc o najwyższej temperaturze powierzchni na obszarze Krakowa na podstawie danych satelitarnych.

Aplikacja umożliwia:

- wybór daty (dostępnej w repozytorium/w chmurze),
- wyświetlenie punktów hotspotów,
- podgląd statystyk (min/avg/max) w jednostkach °C,

Projekt służy jako przykład użycia chmury: hosting statyczny + pipeline CI/CD + automatyzacja generowania danych.

2. Architektura

2.1 Komponenty

Frontend (statyczny)

- index.html (Leaflet) – mapa, legenda, selektor dat, warstwy punktów
- Granice Krakowa: pobierane z serwisu GIS (podkład satelitarny z ArcGIS REST jest przycinany do granic dostarczonych przez MSIP).
- Dane: data/hotspots_YYYY-MM-DD.geojson z hostingu.

Generator danych (Python)

- data/processing/make_hotspots_from_api.py
    - wyszukuje sceny Landsat C2 L2 w STAC,
    - wybiera scenę o najmniejszym zachmurzeniu,
    - pobiera raster temperatury (ST_B10; w Planetary Computer najczęściej asset lwir11),
    - przelicza wartości do °C na podstawie raster:bands.scale i offset,
    - wybiera TOP_K najcieplejszych punktów i zapisuje GeoJSON.
- data/processing/find_scenes.py – pomocniczo listuje sceny i ich czas akwizycji (UTC).

Usługa chmurowa Microsoft Azure

- Hosting: Azure Storage – Static Website (kontener $web).
- Upload/deploy: azcopy z SAS zdefiniowanym w sekretach repozytorium.

3. Przepływy działania

3.1 Przepływ: użytkownik ogląda mapę

1. Użytkownik otwiera URL strony (Static Website).
2. Przeglądarka pobiera index.html.
3. Frontend pobiera granice Krakowa (GeoJSON) z serwisu GIS.
4. Frontend pobiera plik data/hotspots_<data>.geojson.
5. Punkty są filtrowane do granic Krakowa (point-in-polygon).
6. Wyświetlane są:
    - warstwa punktów,
    - legenda dopasowana do przefiltrowanych punktów,
    - statystyki min/avg/max + liczba punktów.

3.2 Przepływ: generowanie hotspotów (pipeline)

1. Workflow generate-hotspots.yml
2. Setup Pythona i instalacja zależności.
3. Generator:
    - wykonuje STAC search (kolekcja landsat-c2-l2) dla DATE i BBOX,
    - wybiera scenę o najniższym eo:cloud_cover,
    - pobiera raster ST (Kelvin) i przelicza do °C,
    - zapisuje data/hotspots_DATE.geojson.
4. Załadowanie pliku do /$web/data/ przez azcopy (SAS).

3.3 Przepływ: deploy strony (CI/CD)

1. Push do master (zmiany w index.html lub data/**).
2. Workflow deploy.yml:
    - sprawdzenie repozytorium,
    - załadowanie index.html do $web/,
    - załadowanie data/ do $web/data/.

4. Sekrety i konfiguracja

4.1 Sekrety w repozytorium (GitHub Actions)

W repozytorium:

- AZURE_STORAGE_ACCOUNT – nazwa storage account.
- AZURE_STORAGE_SAS – SAS token z uprawnieniami do kontenera $web.

Minimalne uprawnienia SAS

- Scope: container $web
- Permissions: Read, Write, List
- Expiry: ograniczony czas: do 28.02.26r.

4.2 Ustawienia generatora (ENV)

- DATE – data sceny (YYYY-MM-DD)
- BBOX – zakres geograficzny (minLon,minLat,maxLon,maxLat)
- MAX_CLOUD – maks. zachmurzenie sceny
- N_POINTS – liczba próbkowanych pikseli
- TOP_K – liczba najcieplejszych punktów do zapisu

---

5. Koszty, limity, ryzyka

5.1 Koszty (szacunek jakościowy)

Azure Storage Static Website

- koszt przechowywania (małe pliki HTML + kilka GeoJSON) – zwykle bardzo niski,
- koszt operacji (GET/PUT) – niski przy małym ruchu,

Źródło danych

- Pobrania danych z Planetary Computer/Landsat są publiczne

5.2 Limity i ryzyka

- **SAS wygasa** → pipeline deploy/generowania przestaje działać (wymaga odnowienia SAS).
- Duża liczba dat i plików hotspots_*.geojson może zwiększać rozmiar repozytorium / transfer.
- Dane to temperatura powierzchni, nie powietrza – interpretacja wyników powinna to jasno komunikować.
- Rozdzielczość i czas przelotu: Landsat ma stałe okno dzienne (brak ujęć nocnych).

6. Monitoring i alerty

1. **Aplikacja - Błędy 404/5xx (braki plików danych)**
   Jeśli brakuje hotspots_*.geojson, użytkownik widzi pustą mapę.
2. **GitHub - Status workflow CI/CD**
   Umożliwia szybkie wykrycie, że wdrożenie się nie powiodło.
3. **Azure – metryki i alerty**

   Metryki:

   Transactions - pozwalają na obserwowanie realnej ilości wykonanych operacji, a także występujących pików, w zależności od typu magazynu i nazwy API.

   Availability – mierzy odsetek requestów zakończonych sukcesem. Pozwala na wykrycie problemów z brakiem połączenia z usługami.

   Used Capacity - określa ilość zajętego miejsca w magazynie. Umożliwia łatwe monitorowanie użycia miejsca w magazynie.

   Success Server Latency – czas obsługi żądania w Azure. Jeśli jest wysoki, może oznaczać przeciążenie magazynu.

   Success E2E Latency- mierzy pełny czas odpowiedzi między serwerem a klientem. Jeśli Server Latency jest niskie, a to wysokie, świadczy o problemach sieciowych. Jeśli oba mają niskie wyniki, oznacza to problem z magazynem.

   Bandwidth – egress oznacza bajty wysłane z magazynu, czyli koszty, a ingress mierzy bajty zapisane do magazynu i pokazuje synchronizacje.

   Alerty:

   - Dla Transactions
     Informuje o wzroście ilości transakcji – czyli kosztów. Ustawiony dynamicznie na podstawie łącznej ilości transakcji większej niż średnia w oknie godzinnym, sprawdzany co 30 min.
   - Dla Success E2E Latency
     Informuje o ewentualnych problemach sieciowych. Ustawiony statycznie na podstawie średniej wielkości opóźnienia w oknie godzinnym większej od 1s, sprawdzany co 30 min.

   - Dla Success Server Latency
     Informuje o ewentualnych przeciążeniach konta. Ustawiony statycznie na podstawie średniej wielkości opóźnienia w oknie godzinnym większej od 150 ms, sprawdzany co 30 min.

7. Procedura sprzątania zasobów

7.1 Usunięcie Resource Group

Usunięcie grupy zasobów kasuje magazyn i wszystkie dane:

az group delete -n heat_island --yes --no-wait

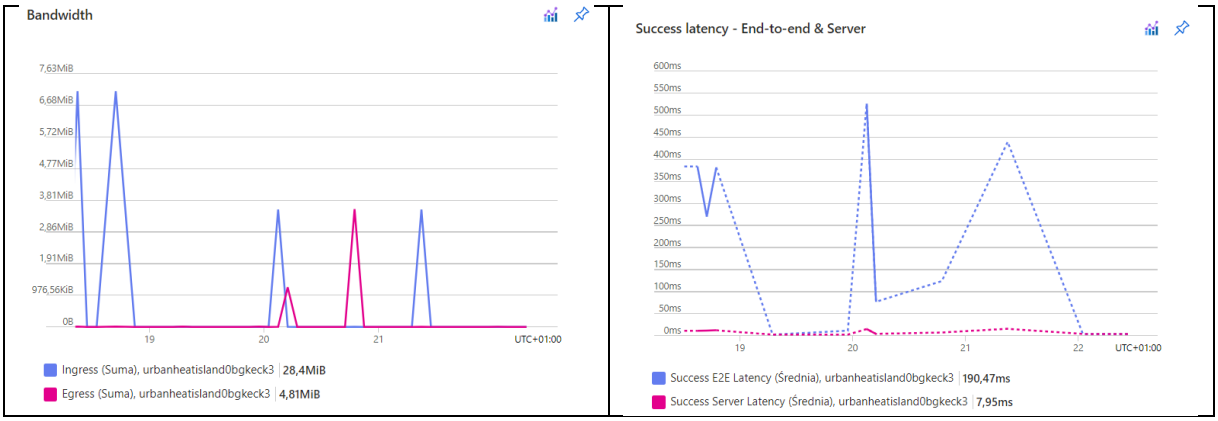
7.2 Wycofanie sekretów

- Usunięcie sekretów z repozytorium
- Unieważnienie SAS

8. Demo

Link do działającej aplikacji: https://urbanheatisland0bgkeck3.z1.web.core.windows.net/

Monitoring:



Availability

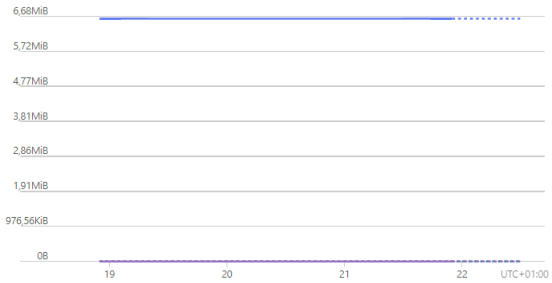
1/2

Blob availability (Średnia), urbanheatisland0bgkeck3 | 100%

File availability (Średnia), urbanheatisland0bgkeck3 | 100%

Queue availability (Średnia), urbanheatisland0bgkeck3 | --

Used capacity



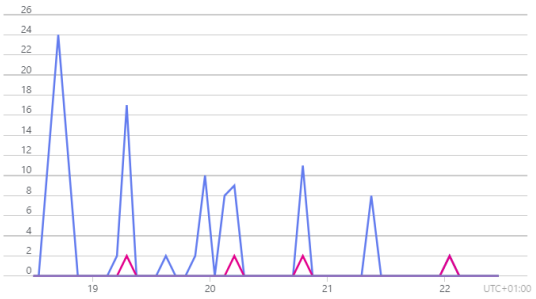
1/2

Blob capacity (Średnia), urbanheatisland0bgkeck3 | 6,61MiB

File capacity (Średnia), urbanheatisland0bgkeck3 | --

Queue capacity (Średnia), urbanheatisland0bgkeck3 | --

Transactions by storage type



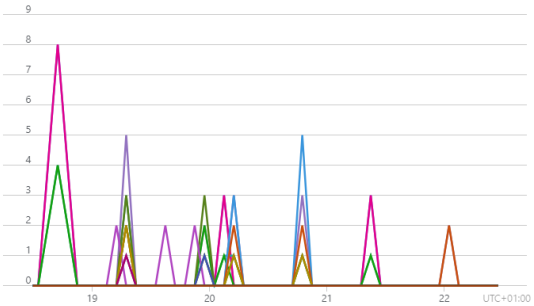
1/2

Blob transactions (Suma), urbanheatisland0bgkeck3 | 119

File transactions (Suma), urbanheatisland0bgkeck3 | 8

Queue transactions (Suma), urbanheatisland0bgkeck3 | --

Transactions by API name



1/2

GetBlobProperties | 22

PutBlob | 22

GetContainerProperties | 16

GetBlobServiceProperties | 15

ListBlobs | 14

GetWebContent | 8

GetFileServiceProperties | 8

BlobPreflightRequest | 8

Unknown | 6