

Tworzenie prostej aplikacji wieloekranowej w Streamlit do wizualizacji aktualnych danych pogodowych

Aleksandra Szymańska 20.01.2025

Streamlit to biblioteka Pythonowa pozwalająca na szybkie i proste budowanie aplikacji webowych. Świeśnie nadaje się do wizualizacji danych statystycznych

Kod przykładu: https://github.com/RobinaHooda/streamlit_introduction

Instalacja

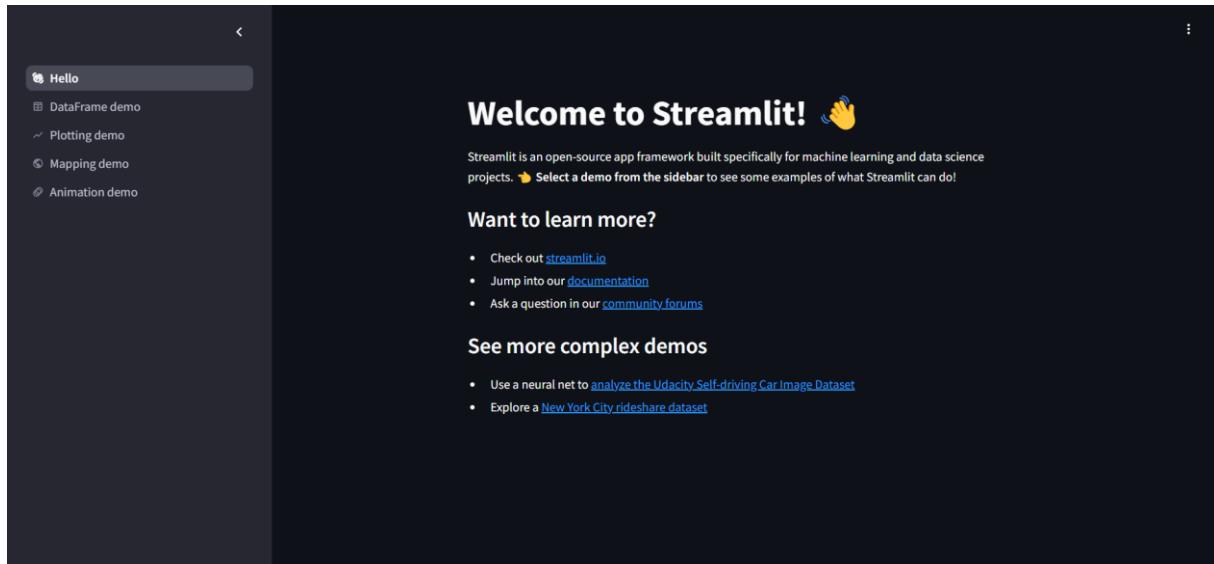
1. Skonfiguruj środowisko w Pythonie, np.: uruchamiając środowisko wirtualne
2. Zainstaluj poleceniem:

```
pip install streamlit
```

3. Zweryfikuj wpisując polecenie:

```
streamlit hello
```

Pod adresem <http://localhost:8501> wyświetli się strona powitalna z demo i przydatnymi linkami



Pierwsza aplikacja

1. By stworzyć najprostszą aplikację należy stworzyć nowy plik, np.: app.py

2. Zimportuj streamlit i wpisać „Witaj 🙌”

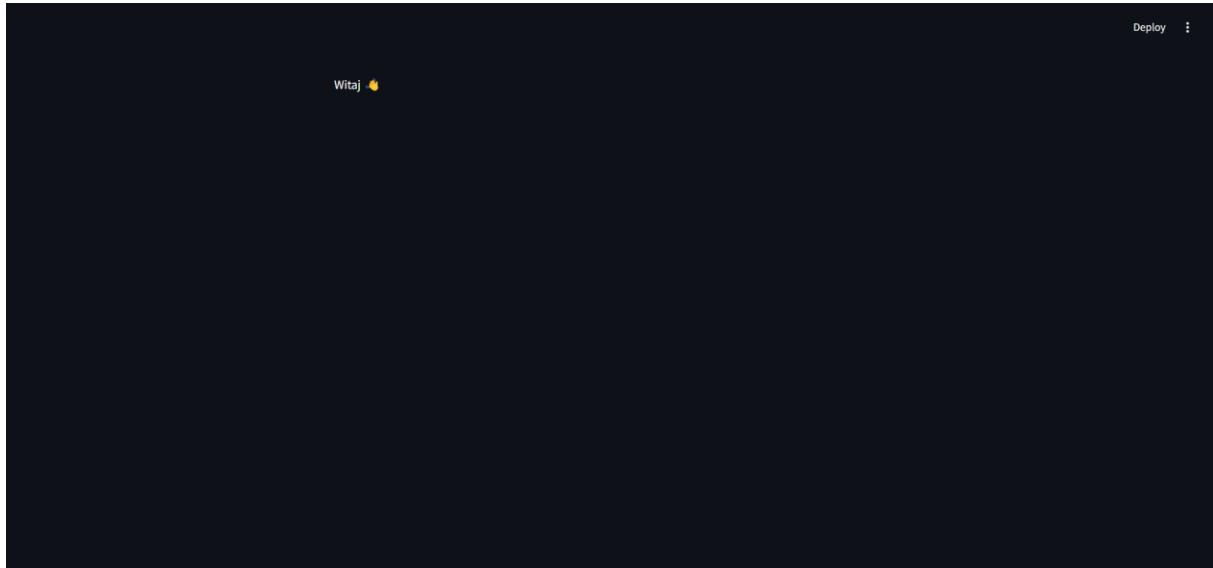
```
import streamlit  
"Witaj 🙌"
```

Streamlit wspiera „magiczne komendy” i automatycznie wypisuje umieszczone w ciele kodu zmienne lub dosłowne wartości w aplikacji. Odpowiada to użyciu funkcji streamlit.write()

3. Zapisz plik i wpisz w konsoli streamlit run [nazwa aplikacji]

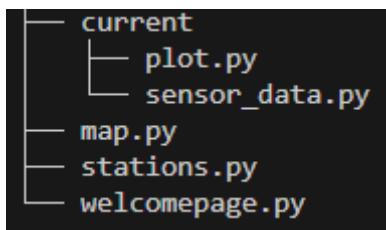
W tym przypadku streamlit run app

Pod adresem <http://localhost:8501> wyświetli się podstawowa strona z napisem „Witaj 🙌”



Właściwa aplikacja

1. Stwórz strukturę plików



2. Uzupełnij strukturę aplikacji

Zdefiniuj podstrony funkcją st.Page([nazwa_pliku], [tytuł wyświetlany w zakładce w przeglądarce, [ikona wyświetlana w zakładce w przeglądarce]])

```
import streamlit as st  
  
welcome_page = st.Page("welcomepage.py", title="Strona powitalna", icon="👋")  
stations_page = st.Page("stations.py", title="Stacje", icon="⚠")
```

```
map_page = st.Page("map.py", title="Mapa stacji", icon="🌐")
sensor_data_page = st.Page("current/sensor_data.py", title="Dane z sensorów",
icon="📡")
plot_page = st.Page("current/plot.py", title="Wykres najnowszych pomiarów",
icon="📈")
```

Dodaj element zarządzający nawigacją `st.navigation({[„grupa stron”]: [[[strona1], [strona2], [strona3]]]})`. Generuje on stronę wybraną przez użytkownika w pasku bocznym. Ma on strukturę słownika, gdzie kluczami są nazwy grup stron a wartościami tablice z ich stronami

Innym sposobem na dodanie nawigacji, jest dodanie folderu `pages` i plików stron o odpowiednich nazwach o strukturze `[numer w pasku bocznym]_[ikona aplikacji]_[tytuł aplikacji]`. Numer definiuje kolejność pojawiania się w panelu bocznym, ikona i tytuł to co będzie wyświetlane w zakładce w przeglądarce

```
pg = st.navigation({"Ogólne": [welcome_page, stations_page, map_page],
"Aktualne dane": [sensor_data_page, plot_page]})
```

Uruchom go funkcją `run()`. Uruchomi to pierwszą stronę dodaną do `st.navigation`

```
pg.run()
```

Po uruchomieniu wyświetli się pusta strona z panelu nawigacji po lewej stronie



3. Dodaj stronę powitalną

Zdefiniuj zmienne w stanie sesji poleceniem `st.session_state.[nazwa zmiennej] = None`

```
import streamlit as st
st.session_state.selected_station_id = None
st.session_state.selected_station_name = None
st.session_state.selected_sensor_id = None
st.session_state.selected_sensor_attribute = None
```

Wygeneruj tekst napisany w formacie markdown

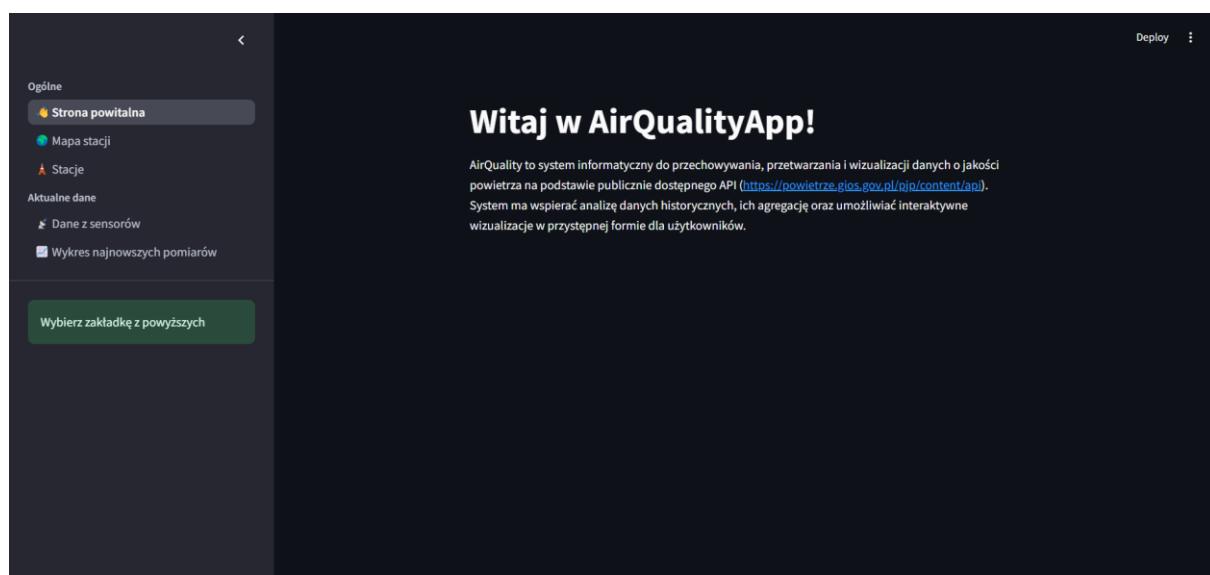
```
st.markdown(
```

```
"""
# Witaj w AirQualityApp!
AirQuality to system informatyczny do przechowywania, przetwarzania i
wizualizacji danych o jakości powietrza na podstawie publicznie dostępnego API
(https://powietrze.gios.gov.pl/pjp/content/api). System ma wspierać analizę
danych historycznych, ich agregację oraz umożliwiać interaktywne wizualizacje
w przystępnej formie dla użytkowników.
"""
```

Dodaj do panelu bocznego pole z poleceniem. Opcja success definiuje styl pola – kolor zielony.
Dostępne są też empty, info, warning i info

```
st.sidebar.success("Wybierz zakładkę z powyższych")
```

Po zapisaniu i odświeżeniu pojawi się zaktualizowany ekran.



4. Dodaj stronę z mapą

Zainstaluj i zimportuj dodatkowe biblioteki

```
import streamlit as st
import folium
from streamlit_folium import folium_static
import pandas as pd
import requests
```

Pobierz i sformatuj dane z api

```
url = "https://api.gios.gov.pl/pjp-api/rest/station/findAll"
response = requests.get(url)

if response.status_code == 200:
    stations = response.json()
else:
    stations = []
```

```

station_data = [
    {
        "id": station["id"],
        "stationName": station["stationName"],
        "latitude": float(station["gegrLat"]),
        "longitude": float(station["gegrLon"]),
        "city": station["city"]["name"],
        "addressStreet": station["addressStreet"]
    }
    for station in stations
]

stations_df = pd.DataFrame(station_data)

```

Zdefiniuj centrum mapy i mapę funkcją folium.Map([lokalizacja początkowa], [początkowe przybliżenie]). Wybierz współrzędne geograficzne, np.: średnią wszystkich zebranych i początkowe przybliżenie

```

map_center = [stations_df["latitude"].mean(), stations_df["longitude"].mean()]
m = folium.Map(location=map_center, zoom_start=6)
Dodaj stacje jako punkty na mapie funkcją folium.Marker([lokalizacja], [okienko wyświetlane po kliknięciu]).add_to([mapa]). Można też skonfigurować inne opcje wyświetlania i ich interaktywność

```

```

for _, row in stations_df.iterrows():
    folium.Marker(
        location=[row["latitude"], row["longitude"]],
        popup=f"{row['city']}<br>{row['addressStreet']}",
    ).add_to(m)

```

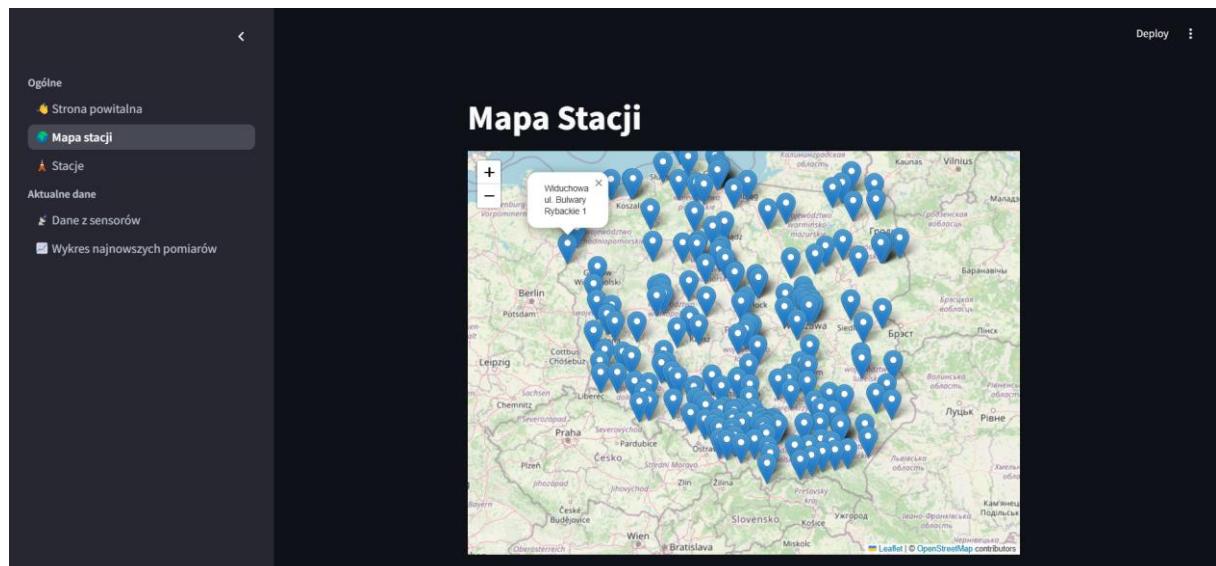
Wyświetl mapę

```

st.title("Mapa Stacji")
st_data = folium._static(m)

```

Tak wygląda ta podstrona



5. Dodaj stronę z listą stacji

Pobierz dane stacji z api

```
import streamlit as st
import pandas as pd
import requests
from st_aggrid import AgGrid, GridOptionsBuilder

url = "https://api.gios.gov.pl/pjp-api/rest/station/findAll"
response = requests.get(url)

if response.status_code == 200:
    stations = response.json()
else:
    stations = []

station_data = [
    {
        "id": station["id"],
        "stationName": station["stationName"],
        "latitude": float(station["gegrLat"]),
        "longitude": float(station["gegrLon"]),
        "city": station["city"]["name"],
        "addressStreet": station["addressStreet"]
    }
    for station in stations
]

stations_df = pd.DataFrame(station_data)
```

Dodaj opcje filtrowania tabeli. Wczytaj informacje wpisane przez użytkownika w polu
st.text_input([tekst wyświetlany w oknie wpisywania])

```
st.write("### Wszystkie stacje")
search_query = st.text_input("Wyszukaj w stacjach:")
filtered_df = stations_df.copy()
if search_query:
    filtered_df = filtered_df[
        filtered_df.apply(lambda row:
row.astype(str).str.contains(search_query, case=False, na=False).any(),
axis=1)
    ]
```

Skonfiguruj opcje wyświetlania ramek danych w GridOptionsBuilder

Dodaj ramkę danych w opcji GridOptionsBuilder.from_dataframe([ramka danych])

Dodaj podział na strony .configure_pagination([opcja, np.: paginationAutoSize=True])

Dodaj selekcje rzędów .configure_selection([opcja, np.: single])

Skonfiguruj wygląd kolumn .configure_column([kolumna], [opcja, np.: autoWidth=True])

Ukryj wybrane kolumny .configure_columns([kolumny], hide=True)

```
grid_options = gb.build()
grid_response = AgGrid(
    filtered_df,
    gridOptions=grid_options,
)
```

Zbuduj i wyświetl ramkę danych

```
grid_options = gb.build()
grid_response = AgGrid(
    filtered_df,
    gridOptions=grid_options,
)
```

Obsłuż kliknięcie w rząd przekierowując do strony z informacjami ze stacji. Zapisz informacje o id i nazwie stacji w stanie sesji st.session_state. Przekieruj do strony z aktualnymi danymi z sensorów st.switch_page([ścieżka do pliku])

```
selected_row = pd.DataFrame(grid_response.get("selected_rows", []))
if not selected_row.empty:
    selected_station = selected_row.iloc[0]
    st.session_state.selected_station_id = selected_station["ID"]
    st.session_state.selected_station_name = selected_station["Nazwa Stacji"]
    st.switch_page("current/sensor_data.py")
```

Oto efekt wykonania tego kodu

The screenshot shows a dark-themed web application interface. On the left, there is a sidebar with the following menu items:

- Ogólne
 - Strona powitalna
 - Mapa stacji
 - Stacje** (highlighted in red)
- Aktualne dane
 - Dane z sensorów
 - Wykres najnowszych pomiarów

The main content area is titled "Wszystkie stacje" and contains a search bar with the placeholder "Wyszukaj w stacjach:" and a text input field containing "Wrocław". Below the search bar is a table with the following data:

	id	stationName	latitude	longitude	city	add
114	Wrocław, ul. Bartnicza	51.115933	17.141125	Wrocław	ul. I	
117	Wrocław, wyb. Konrada-Korzeniowskiego	51.129378	17.02925	Wrocław	ul. I	
129	Wrocław, al. Wiśniowa	51.086225	17.012689	Wrocław	al. I	
143	Inowrocław, ul. Solankowa	52.793122	18.241044	Inowrocław	ul. I	

At the bottom of the grid, there is a pagination control with the text "1 to 4 of 4" and arrows for navigating between pages.

6. Dodaj stronę z danymi z sensorów

Przygotuj funkcje do pobierania danych sensorów i ich pomiarów. Dekorator @st.cache_data sprawia, że raz pobrane dane nie są pobierane ponownie, jeśli kod strony nie był aktualizowany.

```

import streamlit as st
import pandas as pd
import requests
from st_aggrid import AgGrid, GridOptionsBuilder

@st.cache_data
def fetch_station_sensors(station_id):
    url = f"https://api.gios.gov.pl/pjp-api/rest/station/sensors/{station_id}"
    response = requests.get(url)

    if response.status_code == 200:
        sensors = response.json()
    else:
        sensors = []

    return sensors

@st.cache_data
def fetch_sensor_measurements(sensor_id):
    url = f"https://api.gios.gov.pl/pjp-api/rest/data/getData/{sensor_id}"
    response = requests.get(url)

    if response.status_code == 200:
        measurements = response.json()
    else:
        measurements = []

    return measurements

```

Postępuj analogicznie, jak w przypadku stacji. Pobierz i zapisz w ramce danych potrzebne informacje. Jeśli nie wybrano stacji wyświetl ostrzeżenie, że należy to wcześniej zrobić

```

station_id = st.session_state.selected_station_id
station_name = st.session_state.selected_station_name

if not station_id:
    st.warning("Nie wybrano żadnej stacji.")
else:
    sensors = fetch_station_sensors(station_id)
    sensor_results = []
    for sensor in sensors:
        sensor_data = fetch_sensor_measurements(sensor["id"])
        print(sensor_data)
        if sensor_data:
            if sensor_data["values"][0]["value"]:
                latest_measurement = sensor_data["values"][0]
            else:
                latest_measurement = sensor_data["values"][1]
            sensor_results.append({
                "Sensor": sensor["param"]["paramName"],

```

```
        "ID Sensora": sensor["id"],
        "Data": latest_measurement["date"],
        "Wartość": latest_measurement["value"]
    })

results_df = pd.DataFrame(sensor_results)
```

Skonfiguruj opcje wyświetlania ramki danych i ją wyświetl

```
st.write(f"### Najnowsze dane z sensorów na stacji {station_id}.\n{station_name}")
gb_sensors = GridOptionsBuilder.from_dataframe(results_df)
gb_sensors.configure_selection("single")
gb_sensors.configure_auto_height(autoHeight=True)

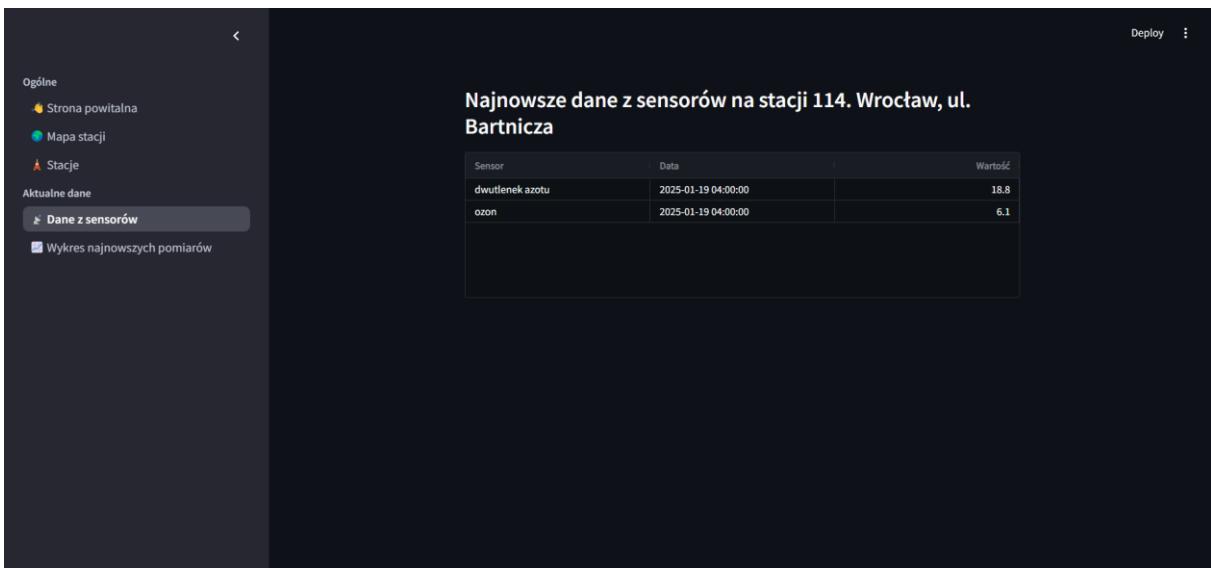
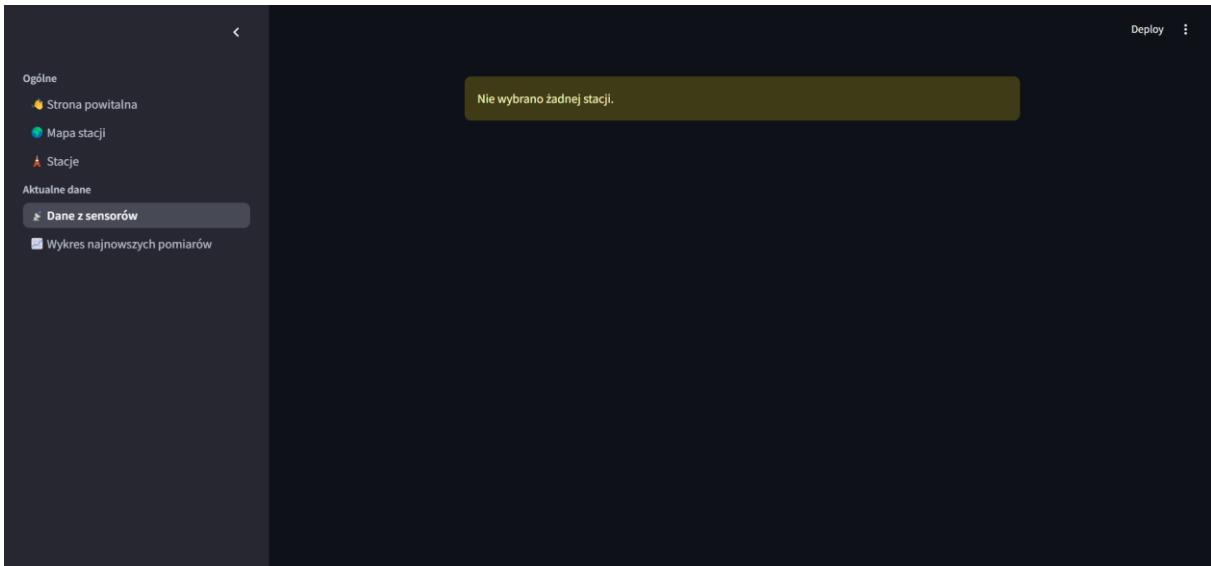
gb_sensors.configure_columns(["ID Sensora"], hide=True)

grid_options_sensors = gb_sensors.build()
grid_response_sensors = AgGrid(
    results_df,
    gridOptions=grid_options_sensors,
    fit_columns_on_grid_load=True,
)
```

Obsłuż kliknięcie w rzad. Zapisz odpowiednie informacje i przekieruj na stronę z wykresami z ostatnich 24 godzin

```
selected_sensor_row =
pd.DataFrame(grid_response_sensors.get("selected_rows", []))
if not selected_sensor_row.empty:
    selected_sensor = selected_sensor_row.iloc[0]
    st.session_state.selected_sensor_id = selected_sensor["ID Sensora"]
    st.session_state.selected_sensor_attribute = selected_sensor["Sensor"]
    st.switch_page("current/plot.py")
```

Ta działa ten ekran



7. Dodaj stronę z wykresem najnowszych pomiarów

Pobierz dane zebrane przez sensory. Jeśli nie wybrano sensora wyświetli ostrzeżenie, że należy to wcześniej zrobić

```
import streamlit as st
import pandas as pd
import requests
import plotly.express as px

@st.cache_data
def fetch_sensor_measurements(sensor_id):
    url = f"https://api.gios.gov.pl/pjp-api/rest/data/getData/{sensor_id}"
    response = requests.get(url)

    if response.status_code == 200:
        measurements = response.json()
```

```

    else:
        measurements = []

    return measurements

station_id = st.session_state.get("selected_station_id")
station_name = st.session_state.get("selected_station_name")
sensor_id = st.session_state.get("selected_sensor_id")
sensor_attribute = st.session_state.get("selected_sensor_attribute")
if sensor_id:
    sensor_measurements = fetch_sensor_measurements(sensor_id)
    valid_measurements = [
        {"Data": m["date"], "Wartość": m["value"]}
        for m in sensor_measurements["values"]
        if m["value"] is not None and m["value"] >= 0
    ]
    sensor_df = pd.DataFrame(valid_measurements)
    sensor_df = sensor_df.sort_values(by="Data")

```

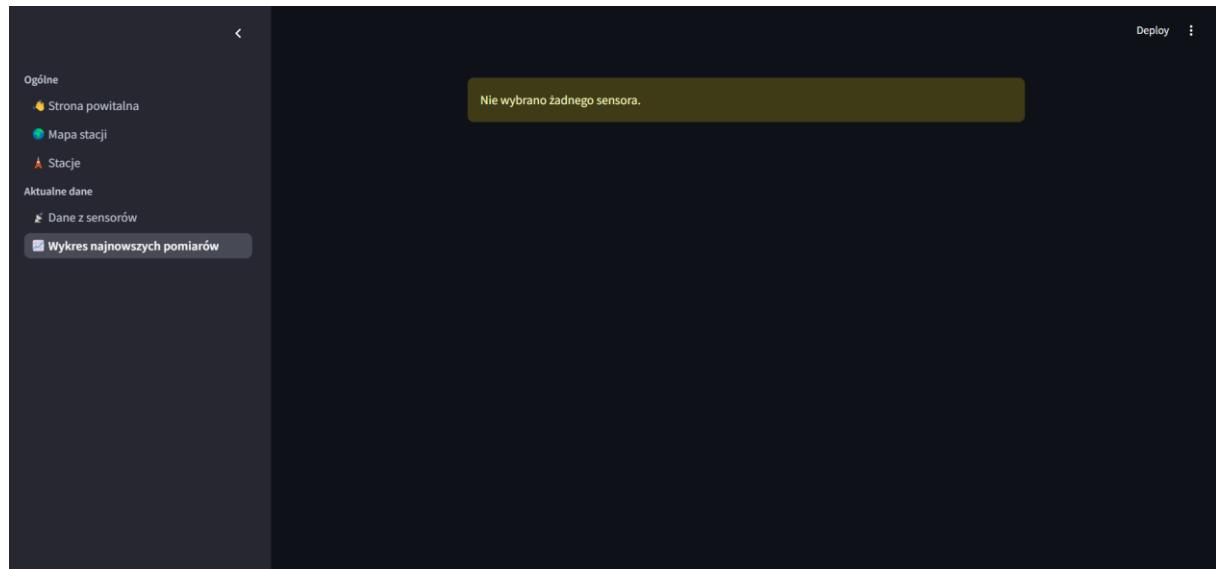
Wyświetl interaktywny wykres funkcją st.plot

```

    st.write(f"### Wykres danych dla stacji {station_id}. {station_name} dla
sensora {sensor_id}. {sensor_attribute}")
    fig = px.line(sensor_df, x="Data", y="Wartość", title="Wartości w czasie",
                  labels={"Data": "Data", "Wartość": "Wartość"})
    st.plotly_chart(fig, use_container_width=True)
else:
    st.warning("Nie wybrano żadnego sensora.")

```

To zobaczymy po wyświetleniu tej podstrony.





8. Dodaj nowe ekranы, np.: z ostrzeżeniami o przekroczonych normach, posługując się informacjami ze stron

<https://powietrze.gios.gov.pl/pjp/content/api>

<https://docs.streamlit.io/>

<https://streamlit.io/components>

<https://discuss.streamlit.io/>

9. Wdroż swoją aplikację za darmo z Streamlit Community Cloud, podążając za instrukcjami po wcisnięciu przycisku Deploy, wyświetlanego na każdej podstronie. Możesz też wyrenderować strony streamlit w widoku Django w twojej aplikacji.