

Baza iNaturalist — występowanie gatunków zagrożonych
na danym obszarze
projekt z Pracowni informatycznej

Natalia Okopna
nr albumu: 123454

prowadzący: dr hab. Wojciech Jakubowski

Grudzień 2021

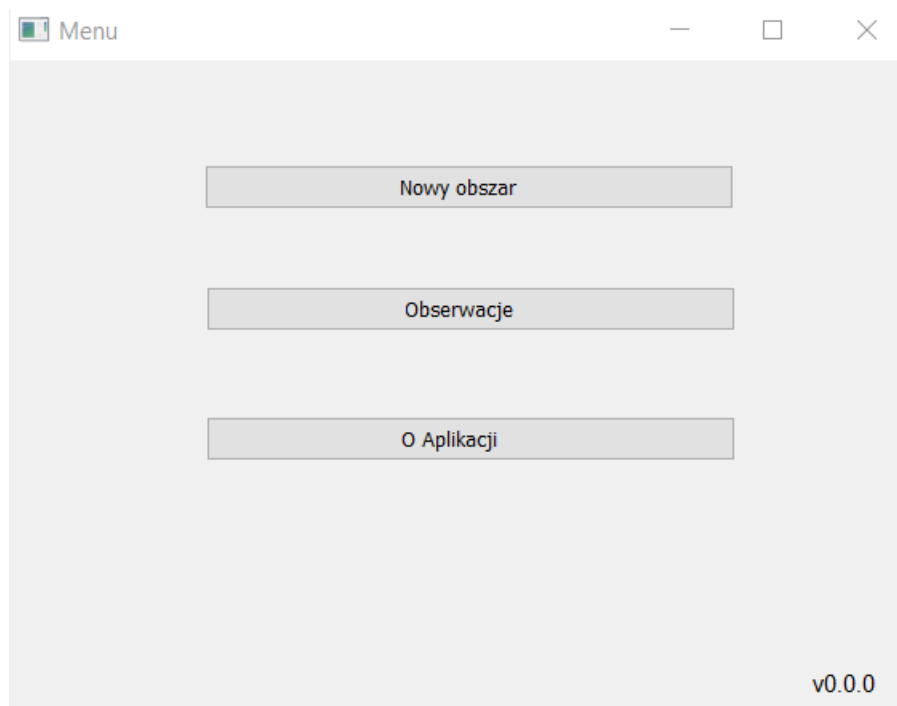
Spis treści

1	Menu główne	3
2	Wprowadzanie nowego obszaru	4
2.1	Okno wprowadzania obszaru	4
2.2	Walidacja danych	5
2.3	Tworzenie obszarów	6
2.4	Obszar spójny	7
3	Wczytywanie obserwacji	8
3.1	Wczytywanie obserwacji z dysku	8
3.2	Generowanie obserwacji z bazy	9
3.2.1	Zapytanie do bazy	9
3.2.2	Obserwacja w obszarze	9
4	Wizualizacja danych	10
4.1	Tworzenie mapy	10
4.2	Umieszczanie obszaru na mapie	12
4.3	Umieszczanie obserwacji na mapie	13
4.4	Znacznik	13
4.5	Wyświetlanie obserwacji	14
5	Zapisywanie obszarów i obserwacji	15
5.1	Postać pliku json	15

Projekt interface'u

1 Menu główne

Po otwarciu aplikacji użytkownik dostaje kilka możliwości do wyboru. Zapis nowego obszaru, wczytanie informacji oraz wyświetlenie informacji o aplikacji.



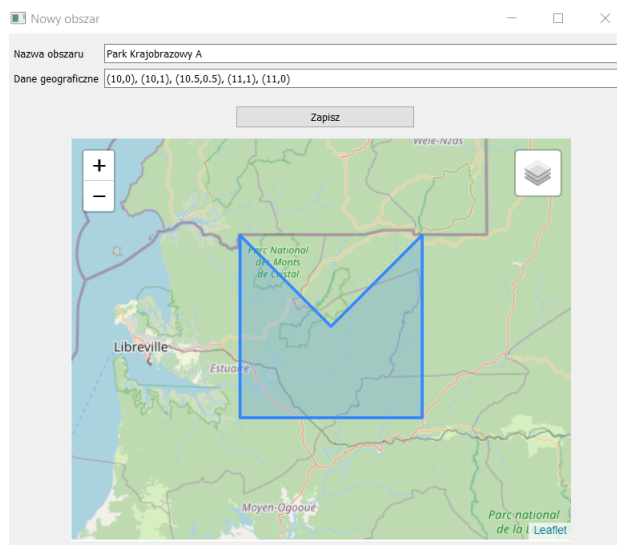
Rysunek 1: Menu

2 Wprowadzanie nowego obszaru

Aby wygenerować opcje na danym obszarze, musi być on wprowadzony do bazy.

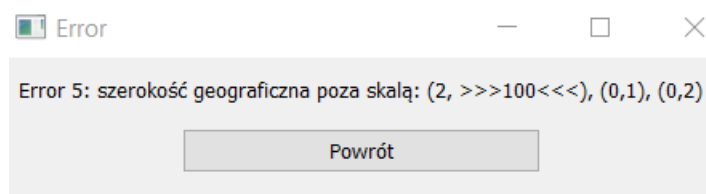
2.1 Okno wprowadzania obszaru

Po kliknięciu w menu głównym opcji *Nowy obszar* otwiera nam się następujące okno.



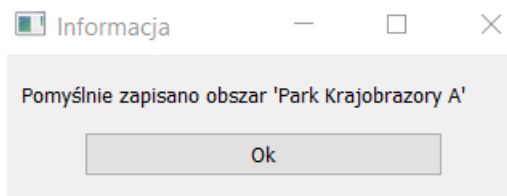
Rysunek 2: Okno 'Nowy obszar'

Docelowo po wprowadzeniu każdego punktu, generowany będzie interaktywnie obszar. Po zatwierdzeniu obszaru rozpoczyna się proces walidacji oraz sprawdzenie, czy obszar jest spójny, omówione w kolejnych podrozdziałach. W razie błędu wyświetli się następujące okno.



Rysunek 3: Okno 'Error'

W przeciwnym razie obszar zostaje zapisany do bazy. Pojawia się okno z informacją.



Rysunek 4: Okno informujące o zapisaniu obrazu

2.2 Walidacja danych

Aby korzystać z funkcji *shapely.Polygon* trzeba sprawdzić poprawność danych wprowadzonych przez użytkownika.

- Linia i punkt

W naszym programie obszar powinien zawierać minimum 3 punkty. Tworzymy funkcję *geo_less_than_3*.

```
1 def geo_less_than_3(lista):
2     if len(lista) < 3:
3         return 1
4     return 0
```

Listing 1: *geo_less_than_3*

Przykładowe wyjścia funkcji:

```
1 geo_less_than_3([('0', '0')])
2 #1
```

Listing 2: Za mało danych

```
1 geo_less_than_3([('0', '0'), ('0', '1')])
2 #1
```

Listing 3: Za mało danych

```
1 geo_less_than_3([('0', '0'), ('0', '1'), ('0', '2')])
2 #0
```

Listing 4: Minimalna liczba danych

- Inne znaki i zakres liczbowy. Sprawdzamy, czy dane wejściowe użytkownika to liczby i to liczby z zakresu szerokości i długości geograficznych świata. Tworzymy funkcję *geo_float*, która zwraca numer błędu lub przekonwertowaną listę danych geograficznych.

```
1 def geo_float(lista):
2     lista_float = []
3     try:
4         for long, lati in lista:
5             try:
6                 long = round(float(long), 5)
7                 if long < -180 or long > 180:
8                     return 3
9             except:
10                return 2
11            try:
12                lati = round(float(lati), 5)
13                if lati < -90 or lati > 90:
14                    return 5
15            except:
16                return 4
17            lista_float.append((long, lati))
18    except:
19        return 0
20    return lista_float
```

Listing 5: *geo_float*

Przykłady:

wpisujemy błędne dane:

```
1 geo_float([('0'), ('0', '1'), ('0', '2')])
2 #0
```

Listing 6: za krótka krotka

```

1 geo_float([(1, 0, 4), (0, 1), (0, 2)])
2 #0

```

Listing 7: za długa krotka

```

1 geo_float([(0, 'a'), (0, 1), (0, 2)])
2 #4

```

Listing 8: inne znaki

```

1 geo_float([('a', 0), (0, 1), (0, 2)])
2 #2

```

Listing 9: inne znaki

```

1 geo_float([(200, 0), (0, 1), (0, 2)])
2 #3

```

Listing 10: liczby poza skalą

```

1 geo_float([(2, 100), (0, 1), (0, 2)])
2 #5

```

Listing 11: liczby poza skalą

oraz poprawne dane.

```

1 geo_float([(1, 0), (0, 1), (0, 2)])
2 # [(1.0, 0.0), (0.0, 1.0), (0.0, 2.0)]

```

Listing 12: poprawne dane

2.3 Tworzenie obszarów

Do tworzenia obszarów będziemy korzystać z biblioteki *shapely.geometry*. Tworzymy przykładowy obszar używając funkcji *shapely.geometry.Polygon*. Jako argument dajemy listę punktów danych geograficznych.

```

1 from shapely.geometry import Polygon
2
3 test = [(10,0), (10,1), (10.5,0.5), (11,1), (11,0)]
4 test = Polygon(test)
5 test

```

Listing 13: Przykładowy obszar

Otrzymujemy obszar:



Rysunek 5: Przykładowy obszar 'test'

2.4 Obszar spójny

Sprawdzamy, czy wprowadzony przez użytkownika obszar jest poprawny za pomocą funkcji *shapely.geometry.Polygon.is_valid*. Funkcja zwraca wartość logiczną *True*, gdy obszar jest spójny. Może więc być to obszar wklęsły. Więcej można poczytać o tej funkcji w dokumentacji: <https://shapely.readthedocs.io/en/stable/manual.html#polygons>. Sprawdzamy poprawność stworzonego wcześniej obszaru Rysunek 5.

```
1 test.is_valid
2 #True
```

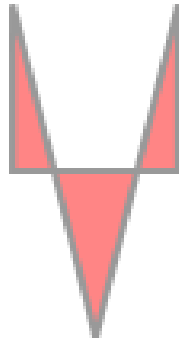
Listing 14: Obszar spójny

Tworzymy obszar, w którym linie między punktami nachodzą na siebie (obszar nie jest spójny).

```
1 test1 = [(0,0), (0,1), (0.5,-1), (1,1), (1,0)]
2 test1 = Polygon(test1)
3 test1
```

Listing 15: Obszar spójny

Otrzymujemy figurę: Sprawdzamy jego poprawność:



Rysunek 6: Obszar niespójny 'test1'

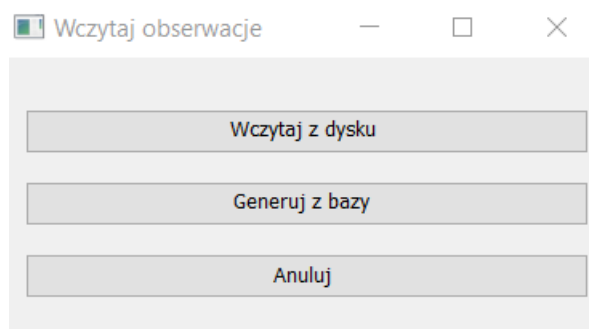
```
1 test1.is_valid
2 #False
```

Listing 16: Obszar niespójny

Funkcja *shapely.geometry.Polygon.is_valid* zwraca wartość *False*. Czyli obszar nie jest poprawny.

3 Wczytywanie obserwacji

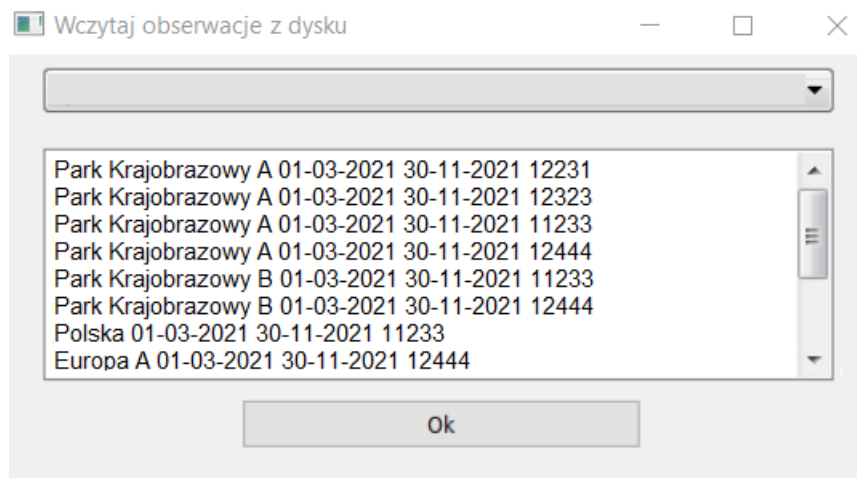
Po kliknięciu w menu głównym opcji *Obserwacje* wyświetla nam się następujące okno.



Rysunek 7: Wczytywanie obserwacji

3.1 Wczytywanie obserwacji z dysku

Po kliknięciu *Wczytaj z dysku* otwiera nam się okno *Wczytaj obserwacje z dysku*.

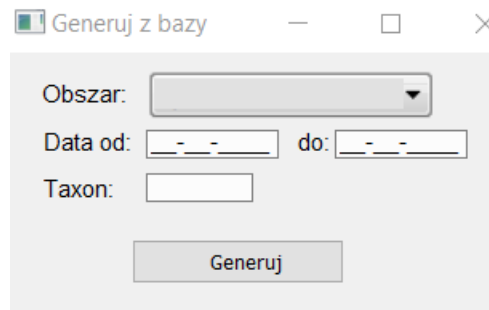


Rysunek 8: Wczytywanie obserwacji z dysku

Po kliknięciu *Ok* wyświetli się okno *Obserwacje* z wizualizacją danych omówioną w kolejnych punktach.

3.2 Generowanie obserwacji z bazy

Po kliknięciu *Wczytaj z dysku* otwiera nam się okno.



Rysunek 9: Generowanie z bazy

Możliwe błędy to: data spoza zakresu, zły takson, brak internetu. W razie błędów będą pojawiały się odpowiednie okna błędów. Po zatwierdzeniu danych i wciśnięciu przycisku *Generuj* zostaje wysyłane zapytanie do bazy i wygenerowanie obserwacji. Zostaje wyświetlone okno *Obserwacje* omówione w dalszej części.

3.2.1 Zapytanie do bazy

Zapytanie do bazy będzie kierowane poprzez funkcję `pyinaturalist.get_observation_species_counts`, która bierze jako argumenty takson, datę startową i końcową obserwacji oraz wartość 0-1 czy gatunek danej obserwacji ma być zagrożony w lokalizacji obserwacji oraz czy obserwacja jest zweryfikowana. Można też ustalić inne dodatkowe parametry o których więcej można poczytać w dokumentacji: https://pyinaturalist.readthedocs.io/en/stable/modules/pyinaturalist.v1.observations.html#pyinaturalist.v1.observations.get_observation_species_counts. Otrzymujemy zestaw obserwacji w postaci słownika JSON.

3.2.2 Obserwacja w obszarze

Następnie należy sprawdzić, które obserwacje należą do danego. Tworzymy punkt przy użyciu funkcji `shapely.geometry.Point`.

```
1 from shapely.geometry import Point
2 p1 = Point(10.5, 0.2)
```

Listing 17: Dodanie obszaru

Następnie korzystamy z funkcji `shapely.geometry.Polygon.contains()`.

```
1 test.contains(p1)
2 #True
```

Listing 18: Dodanie obszaru

```
1 p2 = Point(10.5, 1.2)
2 test.contains(p2)
3 #False
```

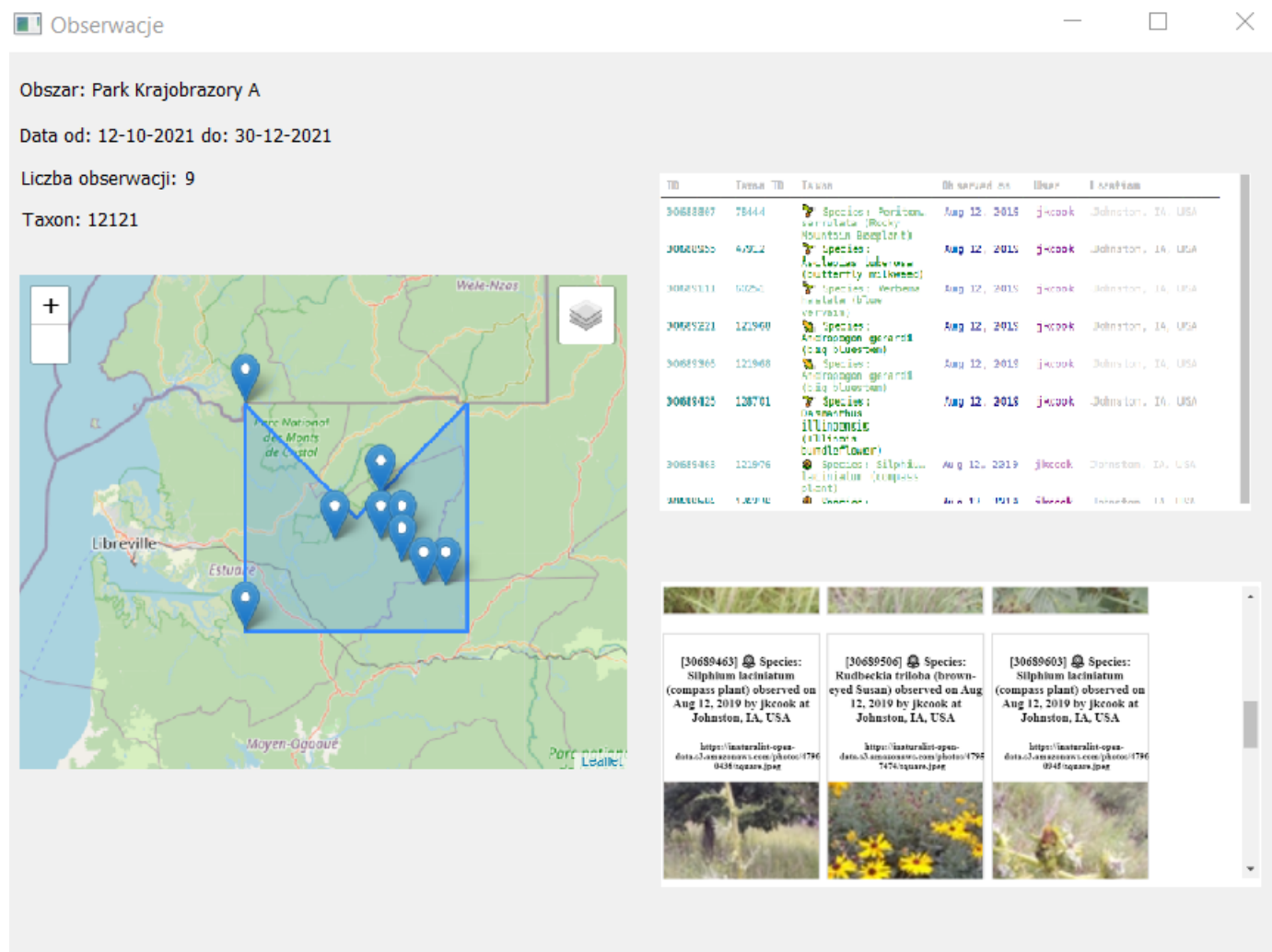
Listing 19: Dodanie obszaru

Punkt *p1* należy do obszaru *test*, punkt *p2* nie należy.

Gdy mamy przefiltrowane obserwacje, zostaje wyświetlić je na mapie oraz w odpowiedni wizualnie sposób.

4 Wizualizacja danych

Po wprowadzeniu obserwacji następuje wyświetlenie obserwacji. Poszczególne elementy okna są opisane w następnych podrozdziałach.



Rysunek 10: Obserwacje

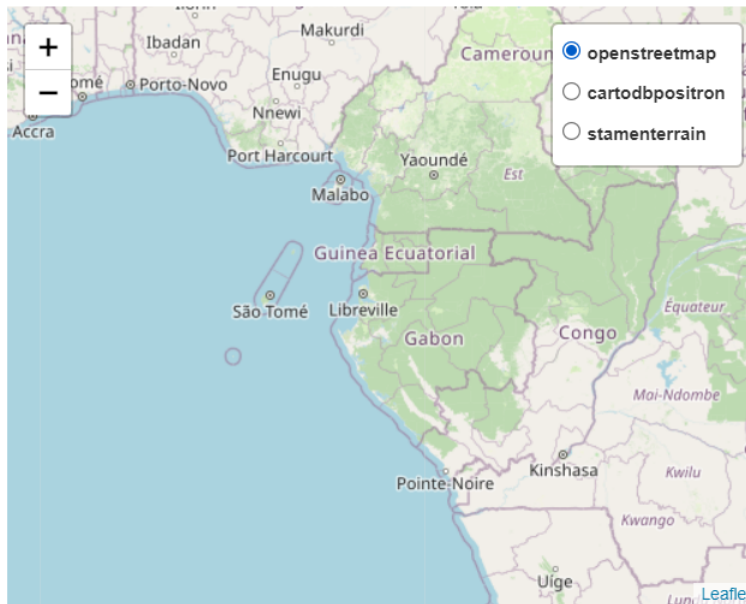
4.1 Tworzenie mapy

Aby umieścić obszary na mapie, skorzystamy z biblioteki *folium*: <https://python-visualization.github.io/folium/quickstart.html>.

Tworzymy obiekt *mapa* określając wymiary okna oraz widok. Domyślny tryb mapy to: *tiles = 'OpenStreetMap'*. W aplikacji będzie można zmienić tryb na *'cartodbpositron'* oraz *'Stamen Terrain'*.

```
1 import folium
2 mapa = folium.Map(width=500, height=400, location=[0,10], zoom_start=5)
3 folium.TileLayer('cartodbpositron').add_to(mapa)
4 folium.TileLayer('Stamen Terrain').add_to(mapa)
5 folium.LayerControl().add_to(mapa)
6 mapa
```

Listing 20: Mapa



Rysunek: Domyślny tryb



Rysunek: Tryb 'cartodbpositron'



Rysunek 11: Tryb 'Stamen Terrain'

4.2 Umieszczanie obszaru na mapie

Do mapy *mapa* dodajemy nasz obszar *test*.

```
1 import folium
2 mapa = folium.Map(width=500, height=400, location=[0,10], zoom_start=8)
3 folium.GeoJson(test).add_to(mapa) # dodanie obszaru
4 folium.TileLayer('cartodbpositron').add_to(mapa)
5 folium.TileLayer('Stamen Terrain').add_to(mapa)
6 folium.LayerControl().add_to(mapa)
7 mapa
```

Listing 21: Dodanie obszaru



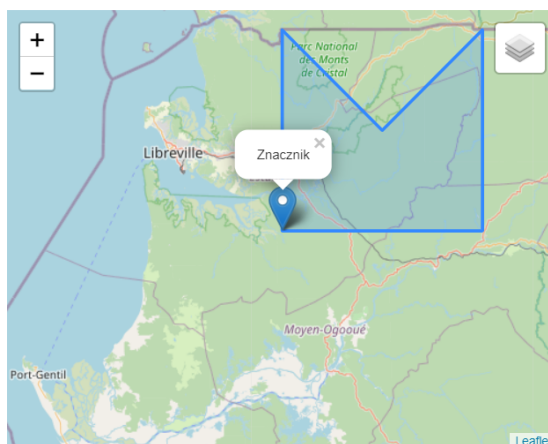
Rysunek 12: Mapa z obszarem

4.3 Umieszczanie obserwacji na mapie

Tworzymy obiekt punktu korzystając z funkcji `folium.Marker`.

```
1 folium.Marker([0, 10], popup="Znacznik").add_to(mapa)
2 mapa
```

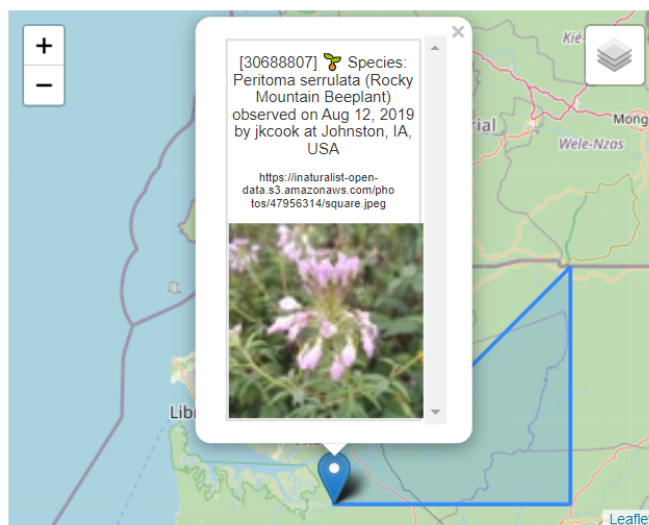
Listing 22: Dodanie obszaru



Rysunek 13: Mapa ze znacznikiem

4.4 Znacznik

Docelowo po kliknięciu na znacznik będziemy widzieli okno. Skorzystamy z biblioteki <https://github.com/karolzak/ipynplot> w zmienionej na nasze potrzeby formie. Ta część będzie uwzględniona dopiero w procesie tworzenia aplikacji w przyszłości.



Rysunek 14: Znacznik z informacjami

4.5 Wyświetlanie obserwacji

Oprócz mapy z obserwacjami (znacznikami) będzie obok widoczna metryczka z liczbą obserwacji i oknem ze wszystkimi obserwacjami w dwóch trybach:



Rysunek 15: Obserwacje z metryczką i zdjęciem

ID	Taxon ID	Taxon	Observed on	User	Location
30688807	78444	Species: <i>Peritoma serrulata</i> (Rocky Mountain Beeplant)	Aug 12, 2019	jkcook	Johnston, IA, USA
30688955	47912	Species: <i>Asclepias tuberosa</i> (butterfly milkweed)	Aug 12, 2019	jkcook	Johnston, IA, USA
30689111	60251	Species: <i>Verbena hastata</i> (blue vervain)	Aug 12, 2019	jkcook	Johnston, IA, USA
30689221	121968	Species: <i>Andropogon gerardi</i> (big bluestem)	Aug 12, 2019	jkcook	Johnston, IA, USA
30689306	121968	Species: <i>Andropogon gerardi</i> (big bluestem)	Aug 12, 2019	jkcook	Johnston, IA, USA
30689425	128701	Species: <i>Desmanthus illinoensis</i> (Illinois bundleflower)	Aug 12, 2019	jkcook	Johnston, IA, USA
30689463	121976	Species: <i>Silphium laciniatum</i> (compass plant)	Aug 12, 2019	jkcook	Johnston, IA, USA
30689506	136376	Species: <i>Rudbeckia triloba</i> (brown-eyed Susan)	Aug 12, 2019	jkcook	Johnston, IA, USA

Rysunek 16: Lista obserwacji z pełną informacją

5 Zapisywanie obszarów i obserwacji

Po wyjściu z okna *Obserwacje* pojawia się okno z zapytaniem, czy zapisać obserwację.

5.1 Postać pliku json

Obszary i obserwacje zapisują się w pliku typu json.

```
1 {
2   'areas': [
3     {
4       'name': 'Park Krajobrazowy A',
5       'loc': [(12.32121, 13.12113), (12.12121, 13.12121), (12.12521,
6         13.12151)]
7     },
8     {
9       'name': 'Park Krajobrazowy B',
10      'loc': [(12.32121, 15.12113), (12.12121, 15.12121), (12.12521,
11        15.12151)]
12    }
13  ],
14  'observations': [
15    {
16      'area_name': '',
17      'loc': [(11.32121, 15.12113), (11.12121, 15.12121), (11.12521,
18        15.12151)],
19      'events': [
20        Tutaj obserwacje jako slowniki json prosto z bazy pobierane
21        za pomoca funkcji pyinaturalist.get_observations() tak, aby byly konwertowalne za
22        pomoca funkcji pyinaturalist.Observation.from_json_list()
23      ]
24    },
25    {
26      'area_name': 'Park Narodowy C',
27      'loc': [(1.32121, 15.12113), (1.12121, 15.12121), (1.12521, 15.12151)],
28      'events': [
29        Jak wyzej
30      ]
31    }
32  ]
33 }
```

Listing 23: Schemat pliku JSON

Postać JSONów obserwacji: https://pyinaturalist.readthedocs.io/en/stable/user_guide.html#responses

And here is what that same observation looks like in JSON:

Observation response JSON

Previewing Responses

These responses can contain large amounts of response attributes, making it somewhat cumbersome if you just want to quickly preview results (for example, in a Jupyter notebook). For that purpose, the `print()` function is included to format response data as a condensed, color-highlighted table.

Examples:

```
>>> from pyinaturalist import get_observations, print
>>> observations = get_observations(user_id="micnose", per_page=5)
>>> print(observations)
```

ID	Taxon	ID	Taxon	Observed on	User
1
2
3
4
5

Rysunek 17: Format obserwacji