

POLITECHNIKA WROCŁAWSKA

PROJEKT ZESPOŁOWY

Sprawozdanie

Piceon - Aplikacja wspierająca automatyczne segregowanie i oznaczanie zdjęć

Skład zespołu:

Michał KUZEMCZAK

nr 241491

Julia DOROBISZ

nr 241544

Kamil JARZĘBSKI

nr 241476

Maciej GOŁĘBIOWSKI

nr 241496

Prowadzący:
Dr inż. Andrzej RUSIECKI

4 CZERWCA 2020



Politechnika
Wrocławska

Spis treści

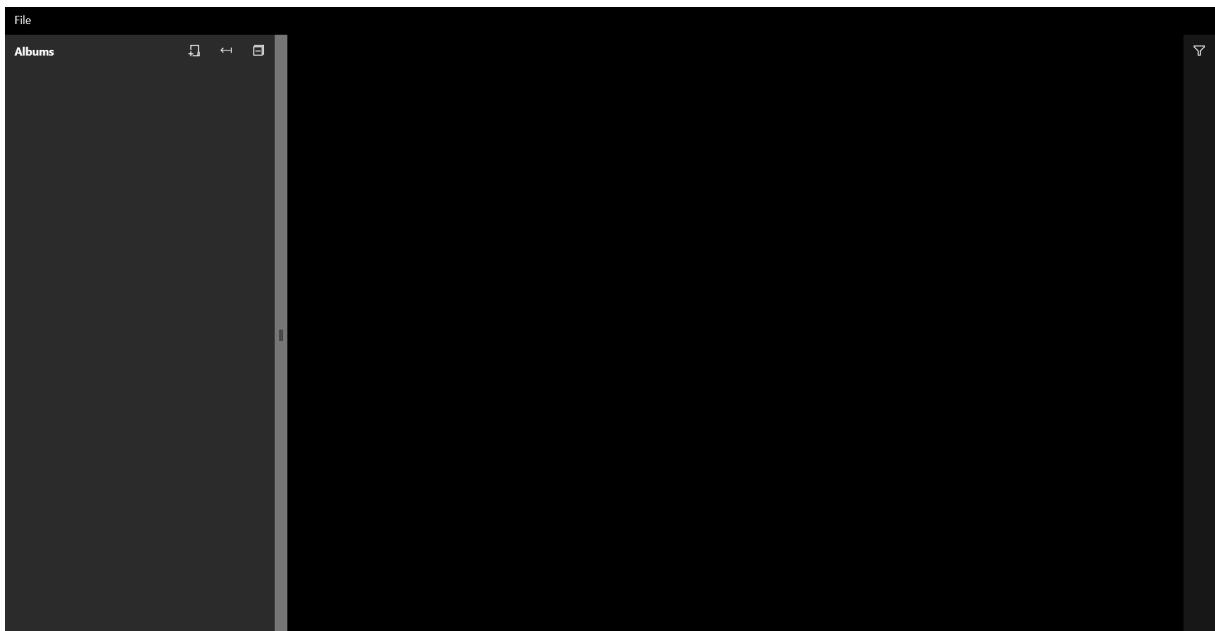
1	Interfejs aplikacji	2
1.1	Drzewko albumów	2
1.2	Galeria zdjęć	5
1.2.1	Wykrywanie podobieństwa zdjęć	7
1.3	Panel filtracji	8
1.3.1	Tagowanie zdjęć	10
2	Baza danych	11
3	Launcher	12
4	Komunikacja z pythonem i launcherem	
	RabbitMQ	12

Spis rysunków

1	Główny interfejs aplikacji	2
2	Dodawanie albumu	2
3	Importowanie zdjęć 1	3
4	Importowanie zdjęć 2	3
5	Skanowanie importowanych zdjęć	4
6	Grupowanie zdjęć według podobieństwa	5
7	Operacje na grupach	5
8	Dodawanie zdjęcia do grupy przez przeciągnięcie	6
9	Dodawanie zdjęcia do albumu przez przeciągnięcie	6
10	Podobieństwo obrazów i duplikatów	7
11	Ciekawe podobieństwo obrazów	7
12	Tagi wszystkich zdjęć w albumie	8
13	Tagi konkretnych zdjęć w albumie	8
14	Dodawanie tagów	9
15	Wyszukiwarka tagów	9
16	Wyszukiwanie po tagach	10
17	Przykładowa lokalizacja do tagowania zdjęcia	10
18	Diagram bazy danych	11
19	Komunikacja RabbitMQ	12

1 Interfejs aplikacji

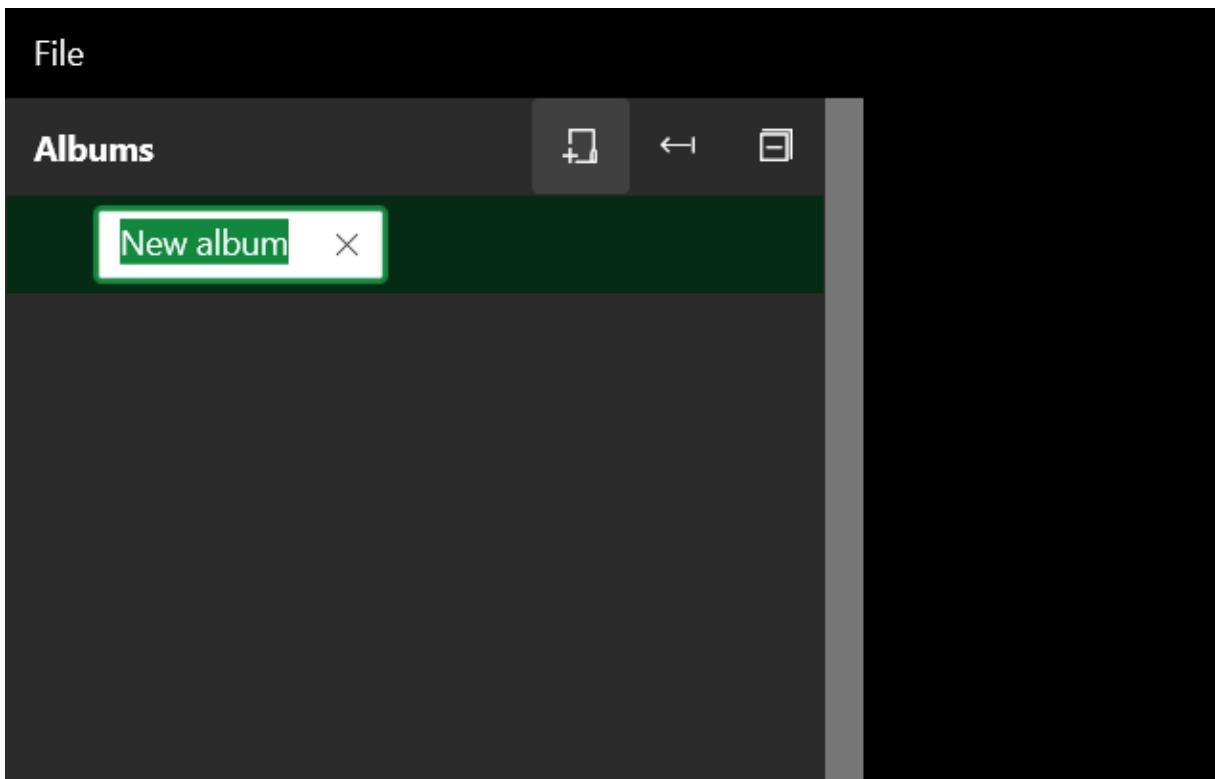
Główny widok aplikacji składa się z drzewka albumów po lewej, galerii zdjęć w centrum i rozwijanego panelu filtracji po tagach po prawej. U góry strony znajduje się pasek menu.



Rysunek 1: Główny interfejs aplikacji

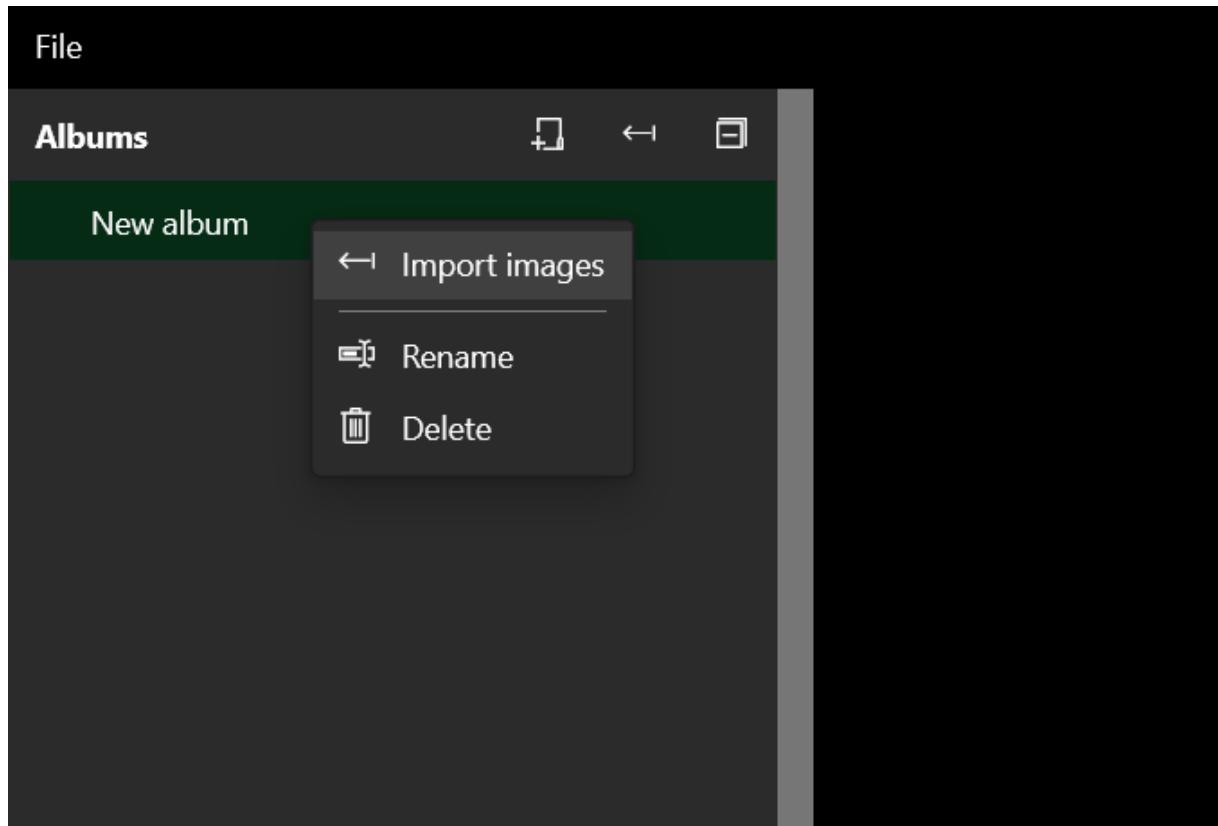
1.1 Drzewko albumów

U góry panelu drzewka albumów znajdują się przyciski pozwalające, od lewej, na dodanie nowego albumu lub pod-albumu, import zdjęć i zwinięcie całego drzewka.

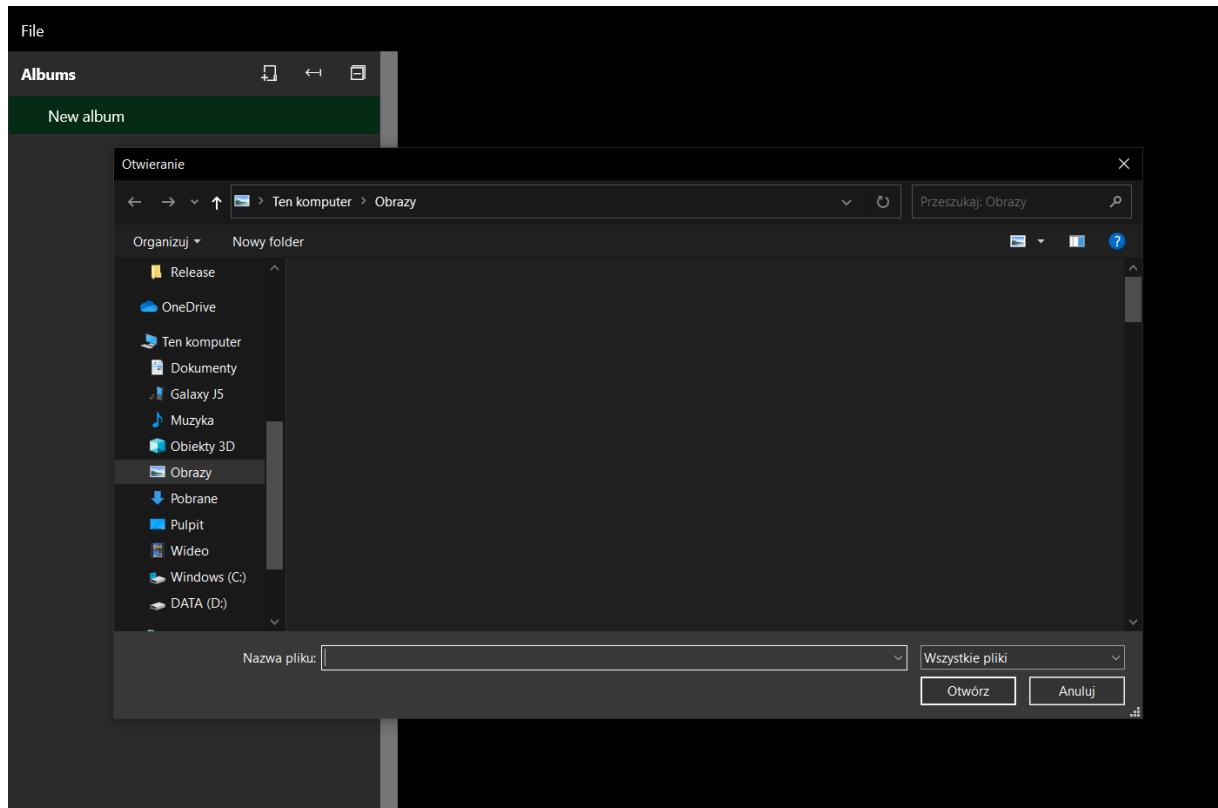


Rysunek 2: Dodawanie albumu

Zdjęcia możemy zainportować do istniejącego albumu. Po kliknięciu przycisku importu, otwiera się okno windowsowego pickera, w którym możemy wybrać pliki do importu.

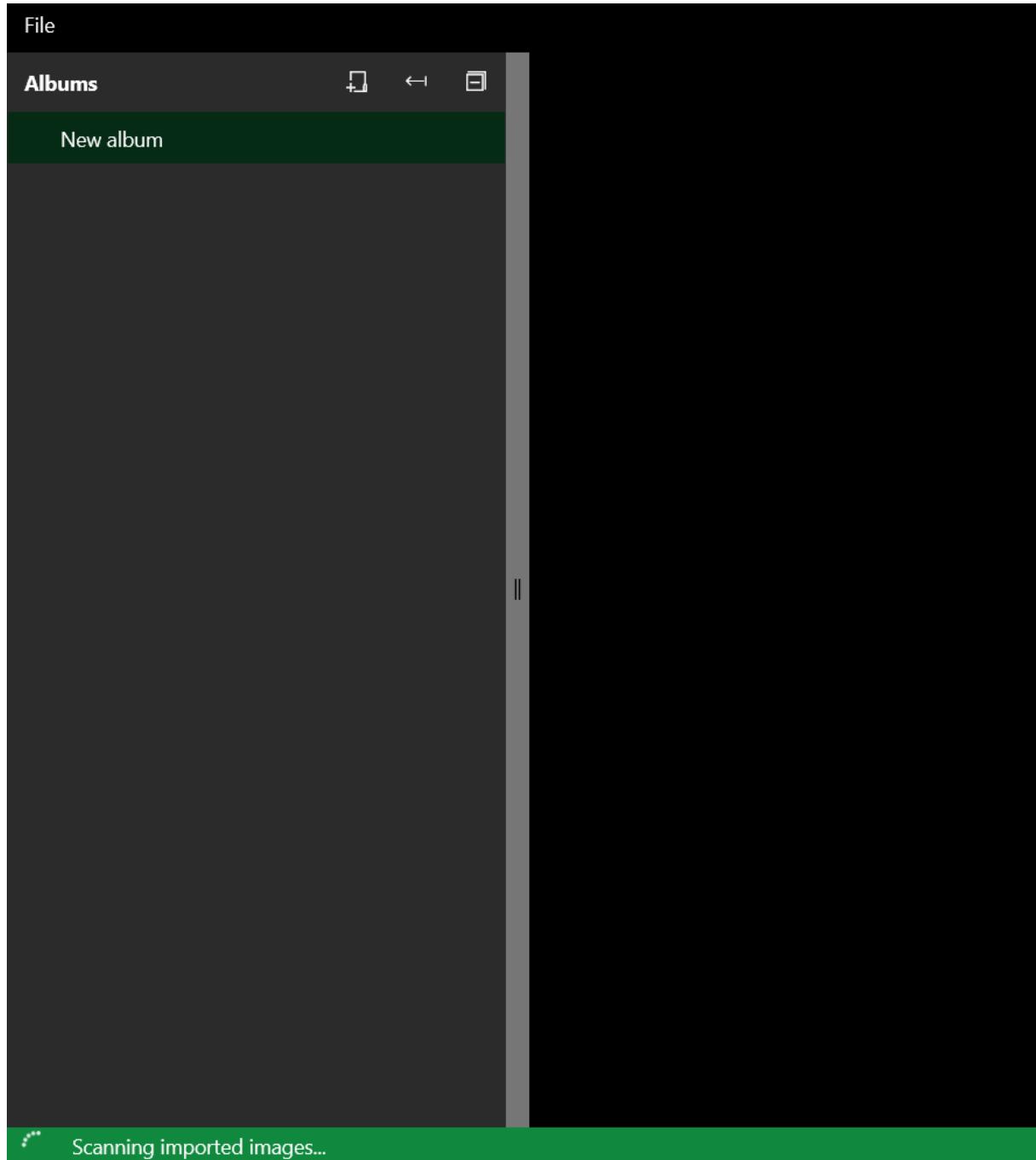


Rysunek 3: Importowanie zdjęć 1



Rysunek 4: Importowanie zdjęć 2

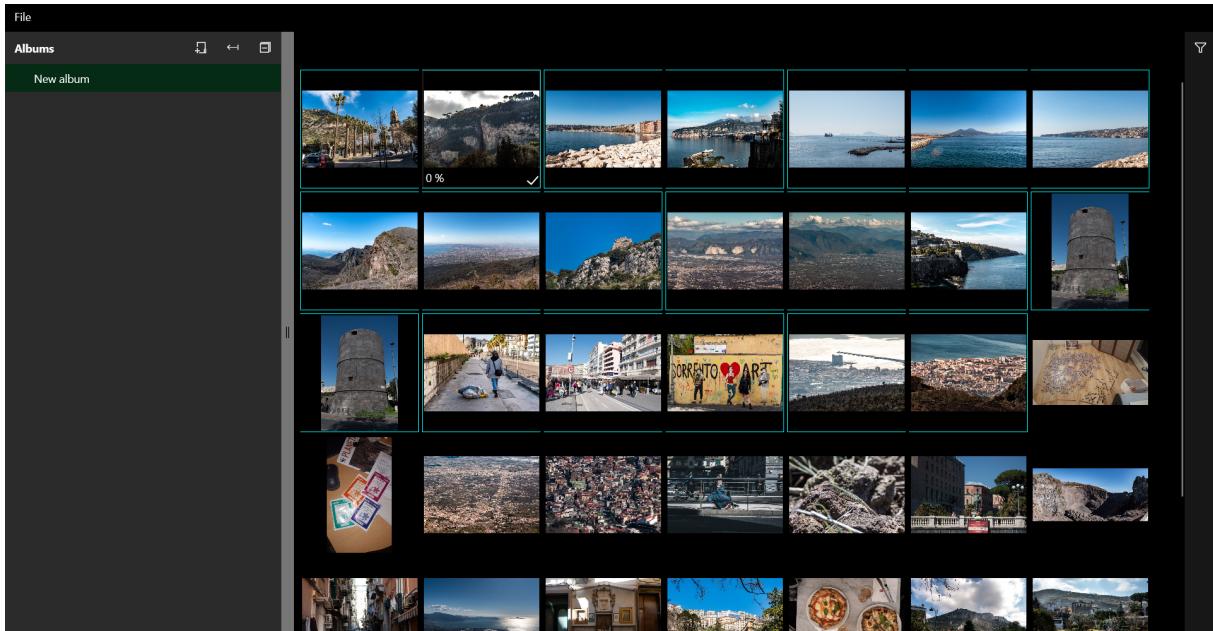
Wybrane zdjęcia zapisywane są w bazie danych i przekazywane do serwisów przetwarzających zdjęcia pod kątem metadanych (tagowanie lokalizacji) i pod kątem wizualnym (grupowanie zdjęć podobnych). W czasie przetwarzania zdjęć użytkownik może swobodnie je przeglądać. U dołu jest wtedy wyświetlane powiadomienie o działającym w tle procesie przetwarzania zdjęć.



Rysunek 5: Skanowanie importowanych zdjęć

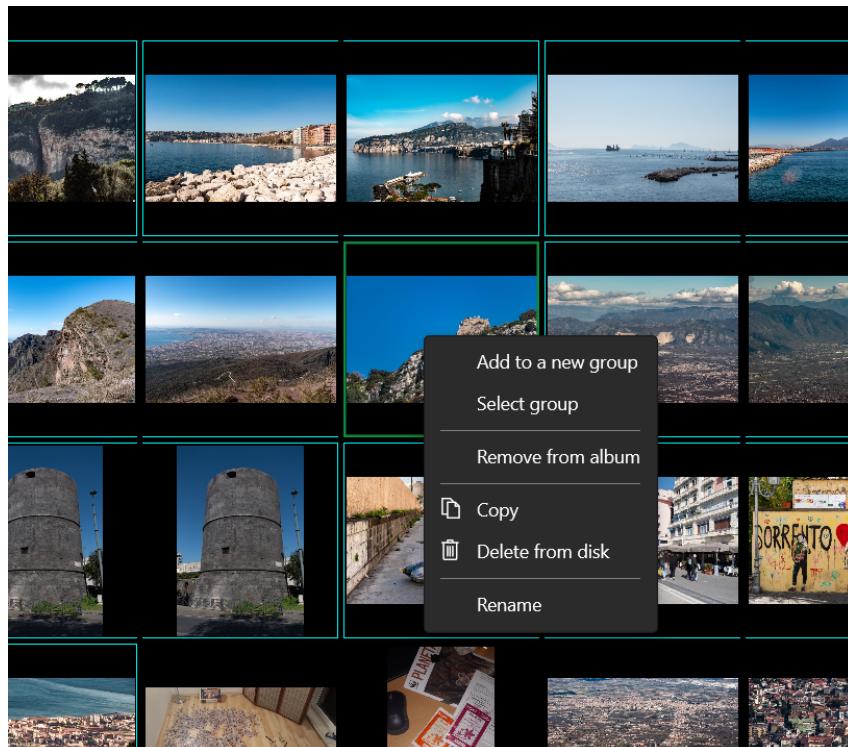
1.2 Galeria zdjęć

Po przetworzeniu, zdjęcia w galerii są pogrupowane jasnoniebieskimi ramkami (jeśli program znalazł podobieństwa). O tym, że zdjęcie zostało przetworzone, świadczy symbol ptaszka widoczny po najechaniu myszką.



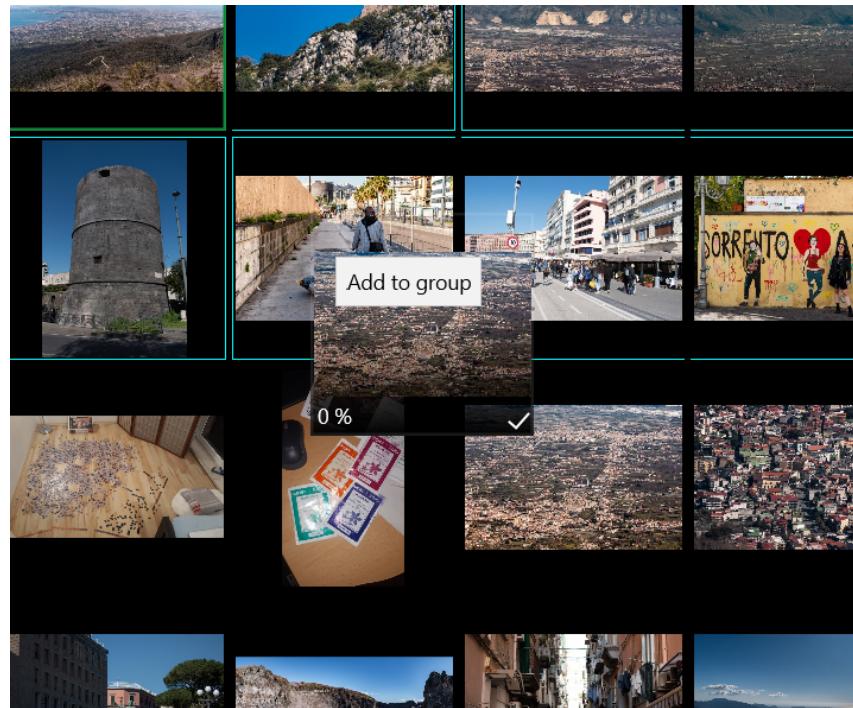
Rysunek 6: Grupowanie zdjęć według podobieństwa

Operacje na grupach takie jak łączenie grup, zaznaczanie całej grupy, dodawanie zdjęć do nowej grupy znajdziemy w menu prawego przycisku myszy po kliknięciu na miniaturkę zdjęcia.



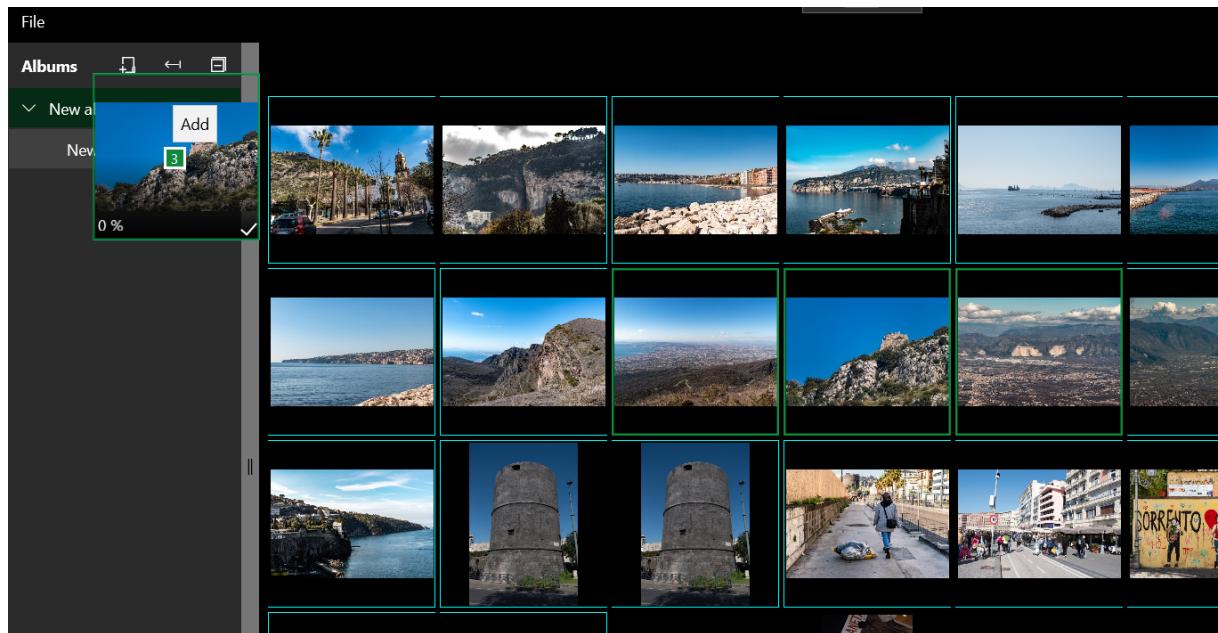
Rysunek 7: Operacje na grupach

Zdjęcia można także dodawać do grupy poprzez przeciągnięcie i upuszczenie.



Rysunek 8: Dodawanie zdjęcia do grupy przez przeciągnięcie

W ten sam sposób można także przenieść zdjęcie do innego albumu, wystarczy upuścić je na wybrany album w drzewku.



Rysunek 9: Dodawanie zdjęcia do albumu przez przeciągnięcie

1.2.1 Wykrywanie podobieństwa zdjęć

Do wykrywania podobieństwa zdjęć użyto języka Python, który posiada bardzo szeroki wachlarz bibliotek do działania na obrazach. Do porównania obrazów przez silną złożoność obliczeniową potrzeba było algorytmu który uchwyci abstrakcje rozumienia obrazu przez ludzki mózg oraz zapewni szybkie porównanie.

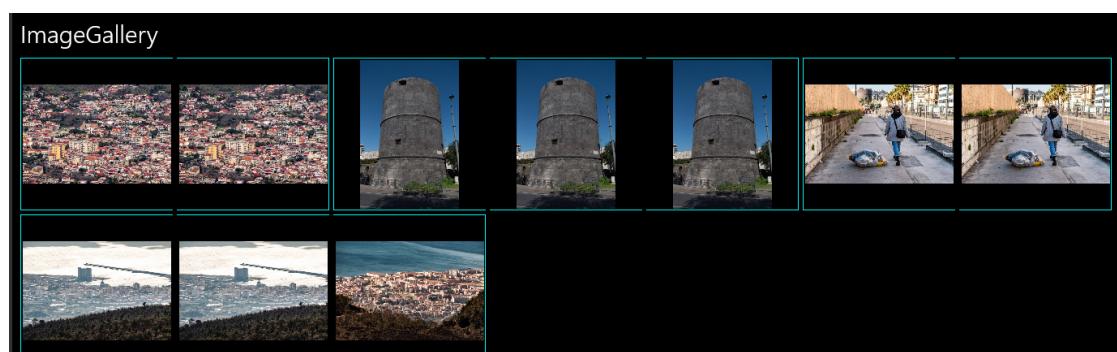
Przy pierwszym podejściu użyte zostały ekstraktory cech (binary descriptors) jak orb, sift, surf. Algorytmy te są odporne na rotacje, oświetlenie czy różne kąty obiektu względem sceny, jednak porównanie wektorów cech jest zbyt złożone więc mimo zadowalających rezultatów porównań szukano szybszych rozwiązań.

Druga próba obejmowała znalezienie i użycie gotowych modeli autokoderów do zakodowania obrazu oraz porównanie ich kodu różnymi miarami. Klastryfikacja nie dała zadowalających rezultatów, więc podjęto następną próbę znalezienia algorytmu.

Podejście trzecie: różne wariacje porównania histogramu od segmentów po histogram lokalnego wzorca (LBP). Ostatecznym rozwiązaniem jest euklidesowa miara podobieństwa pomiędzy histogramami lokalnego wzorca, z progiem dobranym doświadczalnie. Rezultaty mogą być uznane przez człowieka za ciekawe. Wykrywane duplikaty oraz pewne podobieństwa zaobserwować można na poniższym obrazie.



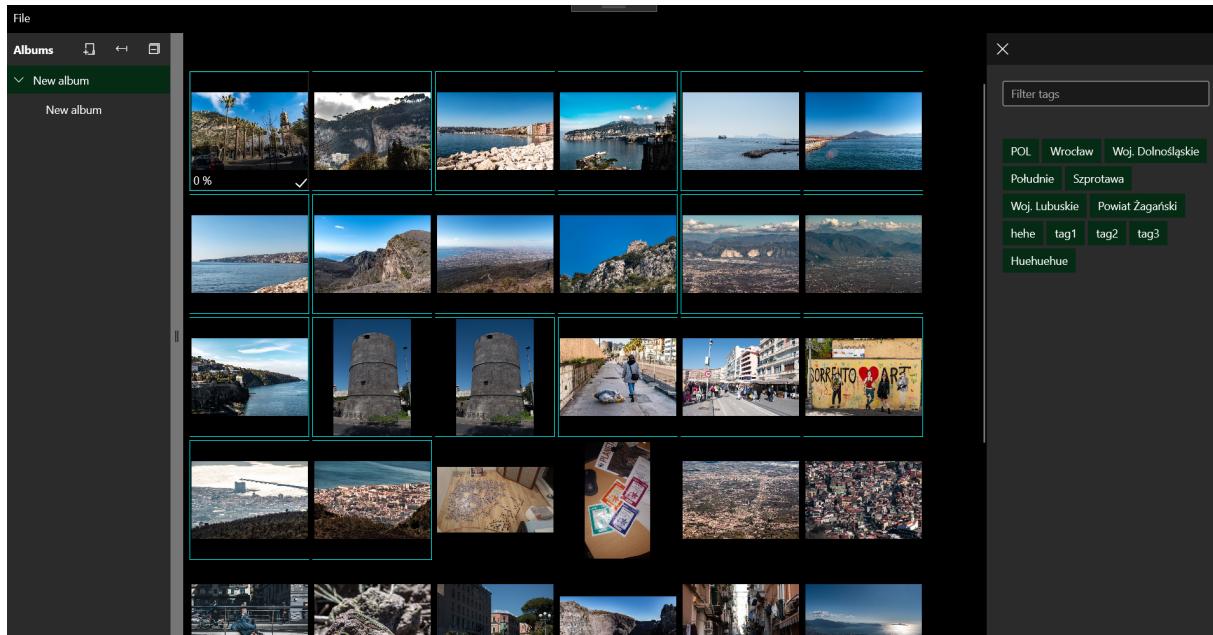
Rysunek 10: Podobieństwo obrazów i duplikatów



Rysunek 11: Ciekawe podobieństwo obrazów

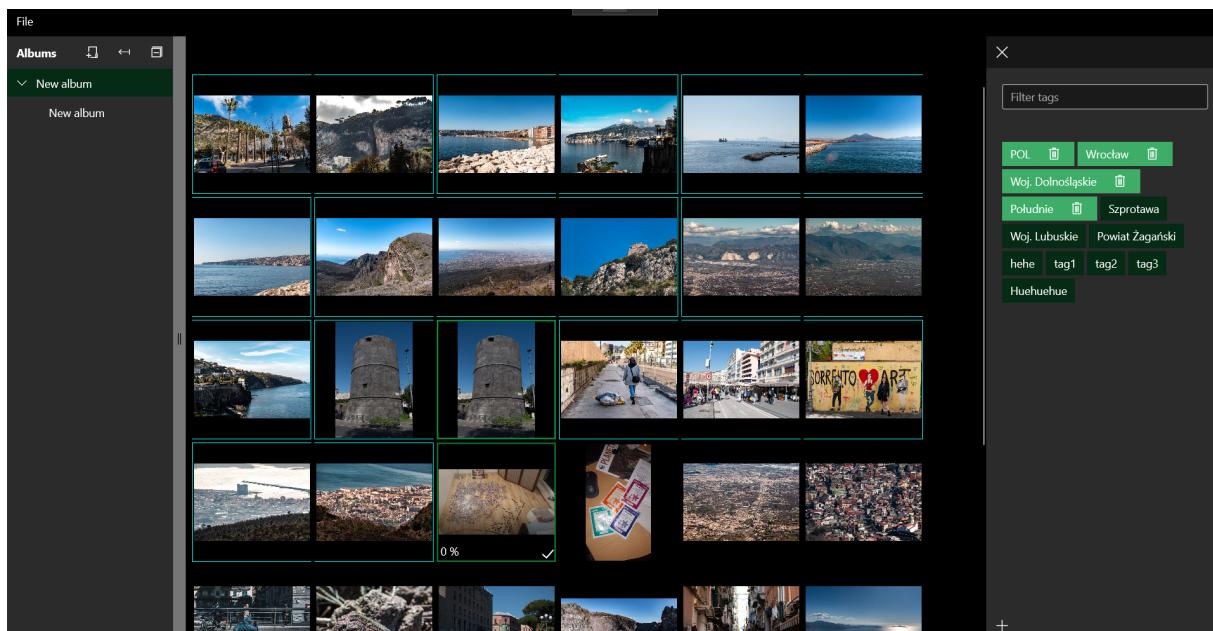
1.3 Panel filtracji

W panelu filtracji wyświetlane są wszystkie tagi zdjęć znajdujących się w aktualnie otwartym albumie.

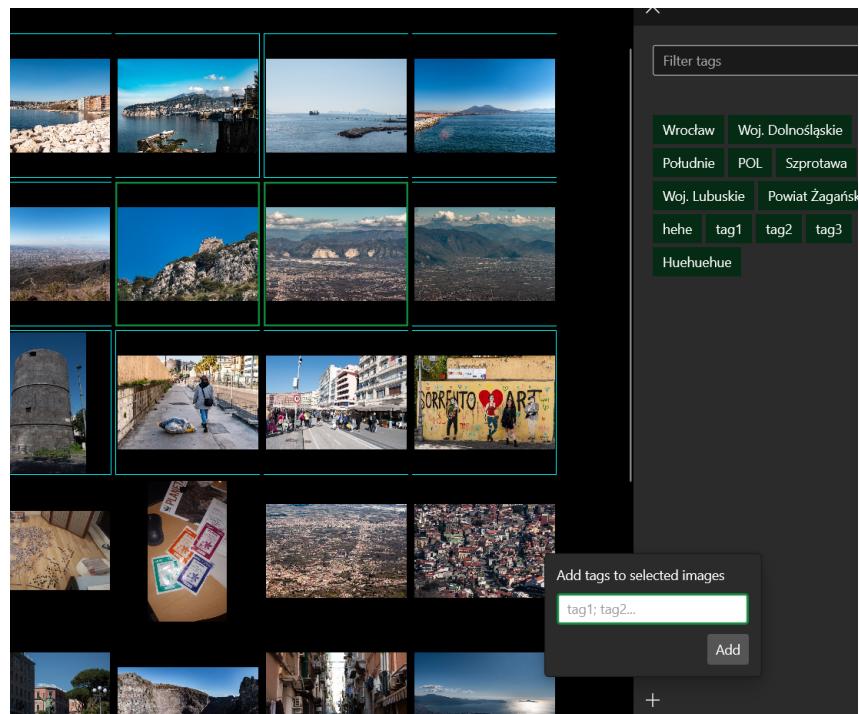


Rysunek 12: Tagi wszystkich zdjęć w albumie

Po zaznaczeniu zdjęć w galerii, ich tagi są podświetlane w panelu. Można je wtedy usuwać, można wtedy także dodać do zaznaczonych zdjęć nowe tagi klikając przycisk u dołu panelu.

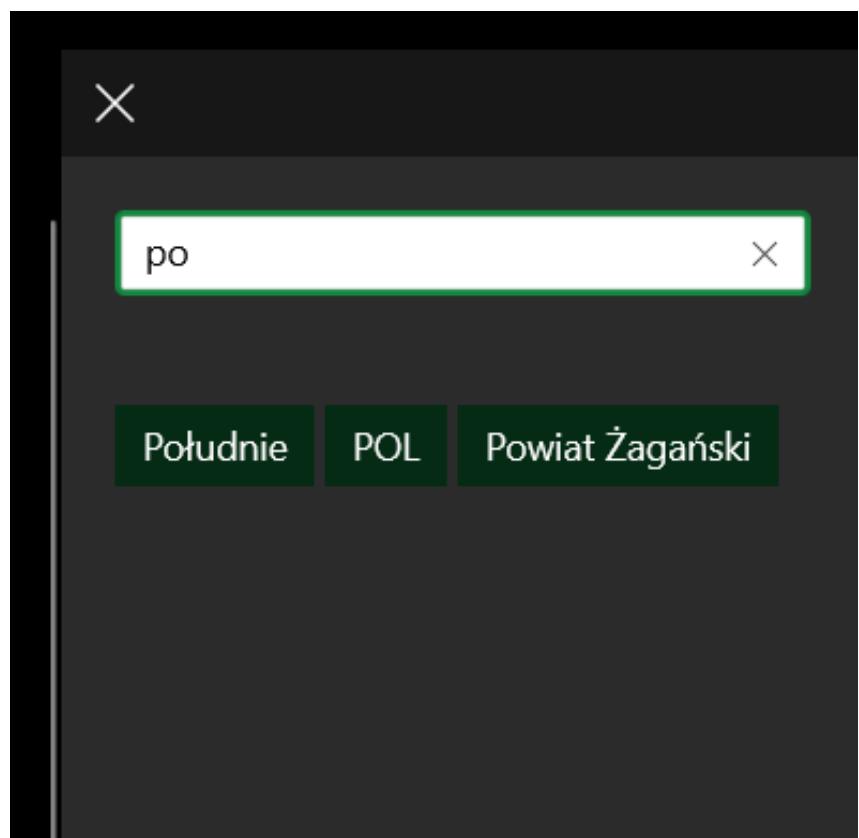


Rysunek 13: Tagi konkretnych zdjęć w albumie



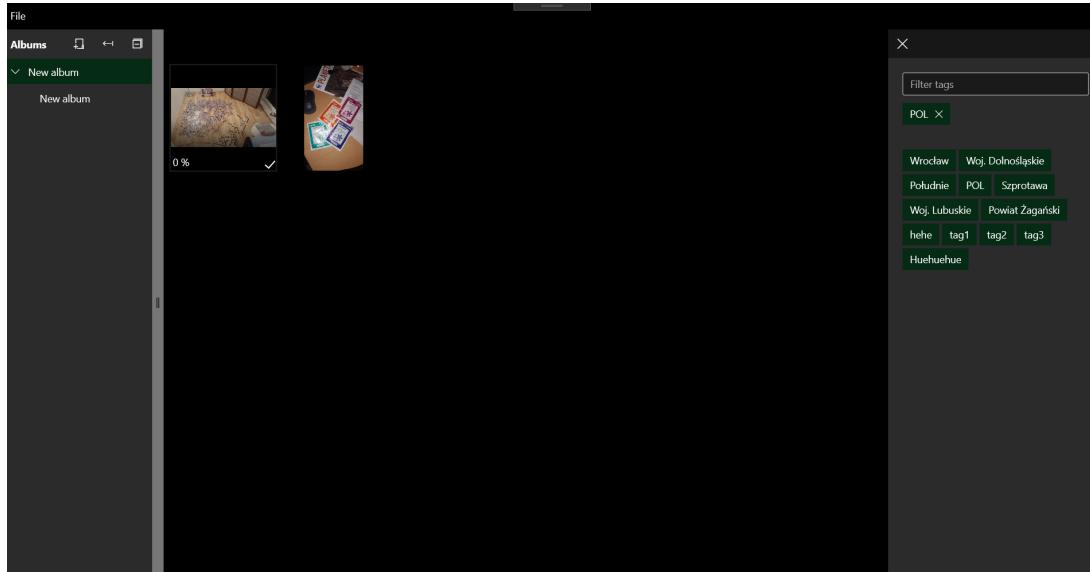
Rysunek 14: Dodawanie tagów

Wyszukiwarka u góry panelu pozwala na przeszukiwanie tagów.



Rysunek 15: Wyszukiwarka tagów

Po kliknięciu na jeden lub więcej tagów, ich duplikaty pojawiają się u góry listy tagów, a w galerii zdjęć zostają wyświetlane tylko te zdjęcia, które posiadają wszystkie wybrane tagi.



Rysunek 16: Wyszukiwanie po tagach

1.3.1 Tagowanie zdjęć

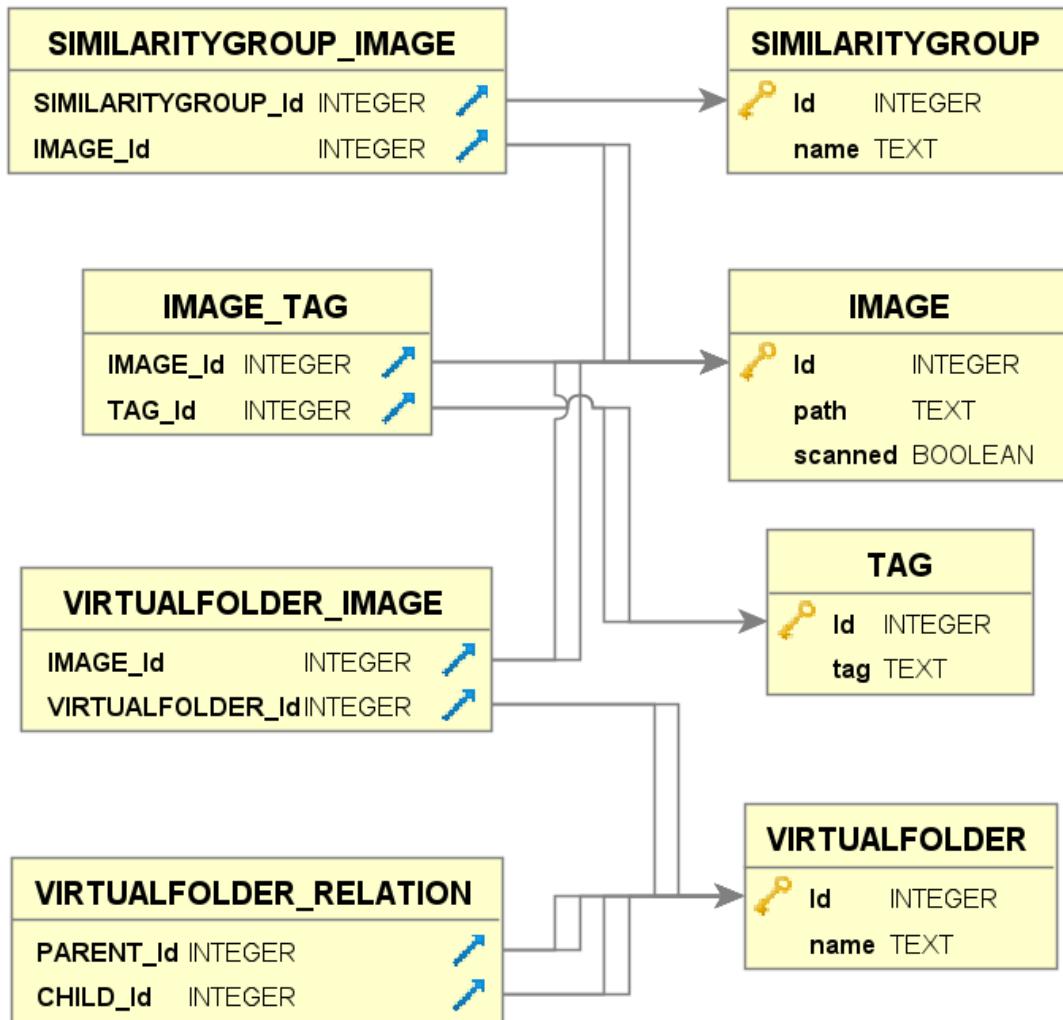
Tagi lokalizacyjne nadawane są zdjęciom przy importowaniu do aplikacji. Z metadanych zdjęcia wyciągana jest informacja o długości i szerokości geograficznej miejsca zrobienia zdjęcia i na jej podstawie wysyłane jest żądanie do serwisu geocode.arcgis.com. Serwis dokonuje operacji reverse-geocoding i zwraca adres w postaci JSON, przy pomocy którego powstają pojedyncze tagi. Tagi przechowywane są w specjalnie do tego przeznaczonej sekcji metadanych zdjęcia w takiej samej formie w jakiej można je dodać na przykład przy pomocy przeglądarki zdjęć systemu Windows, a zatem są trwałe i można je wyświetlić poza aplikacją Piceon.

```
"address": {  
    "Match_addr": "ulica Zygmunta Janiszewskiego 1/17, 50-372, Wrocław, Woj. Dolnośląskie",  
    "LongLabel": "ulica Zygmunta Janiszewskiego 1/17, 50-372, Wrocław, Woj. Dolnośląskie, POL",  
    "ShortLabel": "ulica Zygmunta Janiszewskiego 1/17",  
    "Addr_type": "PointAddress",  
    "Type": "",  
    "PlaceName": "",  
    "AddNum": "1/17",  
    "Address": "ulica Zygmunta Janiszewskiego 1/17",  
    "Block": "",  
    "Sector": "",  
    "Neighborhood": "Plac Grunwaldzki",  
    "District": "Wrocław",  
    "City": "Wrocław",  
    "MetroArea": "",  
    "Subregion": "Wrocław",  
    "Region": "Woj. Dolnośląskie",  
    "Territory": "",  
    "Postal": "50-372",  
    "PostalExt": "",  
    "CountryCode": "POL"  
},
```

Rysunek 17: Przykładowa lokalizacja do tagowania zdjęcia

2 Baza danych

Użytym systemem bazodanowym jest SQLite. W bazie danych przechowywane są informacje o zimportowanych zdjęciach, ich grupach, tagach i albumach. Poniższy diagram przedstawia zależności w bazie danych.



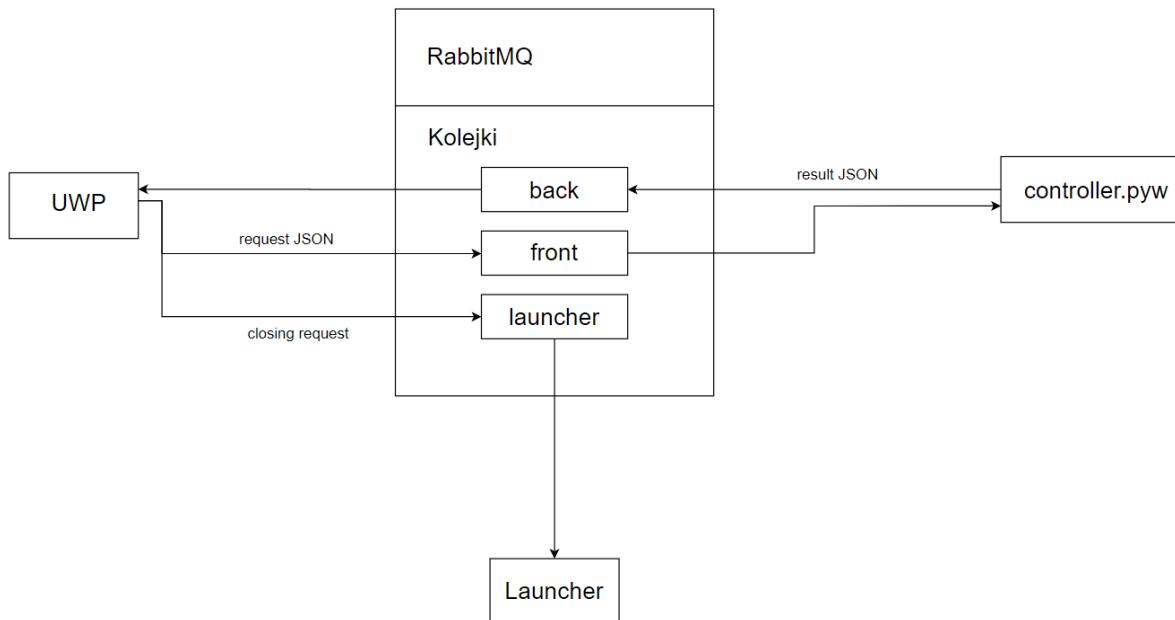
Rysunek 18: Diagram bazy danych

3 Launcher

Większa część aplikacji napisana jest na Universal Windows Platform, w C#/XAML, jednak część ciężkich obliczeniowo operacji przetwarzania obrazów napisana jest w pythonie. Kontrolą pythonowych skryptów zajmuje się jeden nadzędny skrypt - controller, który działa cały czas w tle. Ze względu na to, że środowisko UWP nie pozwala na proste odpalenie procesu z wiersza poleceń, napisano osobny program w C#, Launcher, odpowiedzialny za uruchomienie zarówno programu UWP, jak i pythonowego controllera. Dzięki temu uruchomienie programu odbywa się za pośrednictwem zwykłego pliku .exe. Launcher jest również odpowiedzialny za zakończenie procesu pythonowego kontrolera w reakcji na wiadomość otrzymaną przy zamknięciu aplikacji UWP.

4 Komunikacja z pythonem i launcherem RabbitMQ

Ze względu na to że aplikacja w UWP i pythonowy kontroler to niezależne procesy, komunikacja między nimi, a także między UWP a Launcherem odbywa się za pośrednictwem brokera wiadomości RabbitMQ. Wiadomość z UWP do pythona przesyłana jest jako string JSON, który zawiera identyfikator żądania, informacje o tym, jaka procedura jest wywoływana, a także jakie zdjęcia mają być przetworzone. W odpowiedzi python odsyła string JSON zawierający identyfikator żądania, rezultat operacji (DONE/ERROR) i dane wynikowe. Kiedy użytkownik zamknie aplikację UWP, wysyłana jest pojedyncza wiadomość na kolejkę, której nasłuchuje Launcher. Launcher wtedy zabija proces pythonowego kontrolera.



Rysunek 19: Komunikacja RabbitMQ