Projekt zespołowy 2
Dokumentacja projektu

Dominika Młynarska (306779)
Bartosz Pomiankiewicz (311418)
Mikalai Stelmakh (316951)
Wojciech Grunwald (311566)

02.06.2023

Spis treści

1 Wprowadzenie

1.1 Cel projektu

Celem projektu jest stworzenie systemu zarządzającego kolekcjami zdjęć, umożliwiającego konwersję reprezentacji kolekcji pomiędzy wybranymi narzędziami do zarządzania zdjęciami.

1.2 Wstępna wizja projektu

Na początku określiliśmy kanoniczny format służący do przesyłania/pobierania metadanych pomiędzy różnymi API/desktopem. Zdecydowaliśmy się na kanoniczną formę metadanych w postaci sumy wszystkich formatów (zarówno IPTC, Exif jak i też innych niestandardowych informacji takich jak stopień polubienia, oznaczone osoby itd., definiując nowy znacznik) bez żadnej straty. Zapisujemy tak pobrane dane z API do bazy danych, która będzie przechowywać historię wszystkich przetworzonych zdjęć dokonanych przez jednego użytkownika (jego metadane). Nastepnie w celu przesłania zdjęcia do innego API przygotowujemy metadane zgodnie z formatem wspieranym przez dane API.
Na wstępie zdecydowaliśmy się na implementację tych funkcjonalności z wykorzystaniem dwóch popularnych API do zarządzania kolekcją zdjęć; *Google Photos* oraz *flickr*. Zakładamy, że nasza aplikacja poprzez podejście obiektowe w łatwy sposób umożliwiać będzie wdrożenie kolejnych API.
Wszystko to oprawione będzie w przejrzysty i prosty interfejs umożliwiający łatwy wybór odpowiednich parametrów wywołania poszczególnych funkcjonalności programu.

1.3 Organizacja pracy zespołowej

Główny podział:

- Dominika Młynarska - konwersja metadanych, API flickr
- Mikalai Stelmakh - zarządzanie bazą danych, testy
- Bartosz Pomiankiewicz - interfejs użytkownika
- Wojciech Grunwald - konwersja metadanych, API Google Photos

2 Analiza wymagań

2.1 Wymagania użytkownika i biznesowe

Nasz projekt ma za zadanie ułatwienie zarządzania personalną kolekcją zdjęć. Głównym problemem, na którym skupia się nasze rozwiązanie jest zapewnienie spójności metadanych obrazów podczas konwersji między różnymi programami do ich przechowywania.
Naszym celem jest odciążenie użytkownika systemu, uwalniając go od konieczności dbania o poprawny zapis metadanych podczas przenoszenia zdjęć z jednego narzędzia do drugiego. Użytkownik powinien mieć możliwość wyboru obrazów do przesłania, a także ich lokalizacji źródłowej i docelowej; nie powinien natomiast martwić się o nieoczekiwaną utratę informacji. Nasz program powinien zmniejszyć ilość czasu oraz pracy, którą użytkownik musi włożyć w transfer zdjęć. Proces ten, wykonywany poprzez ręczne przesyłanie do docelowego programu zdjęć wcześniej pobranych z innego narzędzia, może być żmudny i czasochłonny. Nasz program powinien zapewnić użytkownikowi możliwie prosty i czytelny interfejs, który udostępni mu w jednym miejscu wszystkie funkcje konieczne do wykonania transferu zdjęć.

2.2 Wymagania funkcjonalne i niefunkcjonalne

Program powinien:

(Wymagania funkcjonalne)

- umożliwić zalogowanie się do obsługiwanych serwisów,
- umożliwić wybór źródłowej i docelowej lokalizacji oraz zdjęć do przesłania,
- wyświetlać drzewo katalogów lokalizacji i umożliwić poruszanie się po nim w trakcie wyboru zdjęć przez użytkownika,
- konwertować metadane obrazów do postaci kanonicznej i zapisywać je w bazie danych,
- konwertować metadane z postaci kanonicznej do postaci, którą obsługuje program docelowy,
- zwracać informację o tym, czy transfer się powiódł,

(Wymagania niefunkcjonalne)

- być łatwo rozszerzalny – powinno być możliwe łatwe dodanie nowych obsługiwanych serwisów (obecnie jest to flickr oraz Google Photos),
- mieć możliwość rozszerzania i modyfikacji postaci kanonicznej, np. poprzez dodawanie i usuwanie używanych znaczników,
- zapewnić bezpieczeństwo logowania do poszczególnych serwisów,
- posiadać prosty i intuicyjny interfejs użytkownika, składający się z ok. 5 podstawowych ekranów,
- działać poprawnie podczas używania języków różnych narodowości, np. radząc sobie z obsługą różnych formatów daty i czasu.

2.3 Przypadki użycia

Z uwagi na charakter naszego systemu, którego głównym zadaniem z perspektywy użytkownika jest przeniesienie jego zdjęć między platformami, zdefiniowaliśmy jeden podstawowy przypadek użycia – transfer zdjęć. Przepływ podstawowy to główny scenariusz powodzenia, który kończy się poprawnym przesłaniem plików. Przepływy alternatywne opisują sytuacje, gdy w którymś z punktów wystąpi błąd lub użytkownik wycofa się z podjętych kroków. Poniżej opisany został przypadek konwersji z Google Photos do serwisu Flickr, jednak niezależnie od wybranych parametrów źródła/destynacji transferu (Google Photos, Flickr, w przyszłości inne platformy), poszczególne kroki użytkownika są analogicznie.

1. Nazwa przypadku użycia: transfer zdjęć (z Google Photos do Flickr).
   (a) Krótki opis: Przypadek użycia umożliwia Użytkownikowi przesłanie zdjęć ze swojego konta w serwisie Google Photos do serwisu Flickr. Użytkownik może wybierać zdjęcia z wyświetlanego drzewa katalogów, zdefiniować lokalizację docelową zdjęć, anulować chęć przesłania.
   (b) Aktor: Użytkownik systemu.
   (c) Wyzwalacz: Przypadek użycia rozpoczyna się, gdy Użytkownik uruchomi program.

2. Przepływ zdarzeń.

   (a) Przepływ podstawowy:

      (1) Użytkownik wybiera opcję „Google Photos” w polu „Choose Source app”.
      (2) Użytkownik wybiera opcję „Flickr” w polu „Choose Destination app”.
      (3) Użytkownik wybiera przycisk „Continue” i przechodzi do ekranu logowania Google Photos.
      (4) Użytkownik wpisuje dane i loguje się do serwisu.
      (5) Użytkownik zostaje przeniesiony do ekranu wyboru zdjęć i zaznacza pożądane obrazy w wyświetlanym drzewie katalogów.
      (6) Użytkownik akceptuje wybór przyciskiem „Continue” i przechodzi do ekranu logowania Flickr.
      (7) Użytkownik wpisuje dane i loguje się do serwisu.
      (8) Użytkownik zostaje przeniesiony do ekranu wyboru lokalizacji docelowej.
      (9) Użytkownik wybiera folder z wyświetlonego drzewa katalogów.
      (10) System pyta użytkownika, czy na pewno chce on przesłać zdjęcia do wybranego katalogu; Użytkownik akceptuje.
      (11) System wyświetla informację o powodzeniu konwersji.
      (12) Użytkownik wybiera opcję „Continue” i przechodzi do ekranu początkowego.

   (b) Przepływy alternatywne.

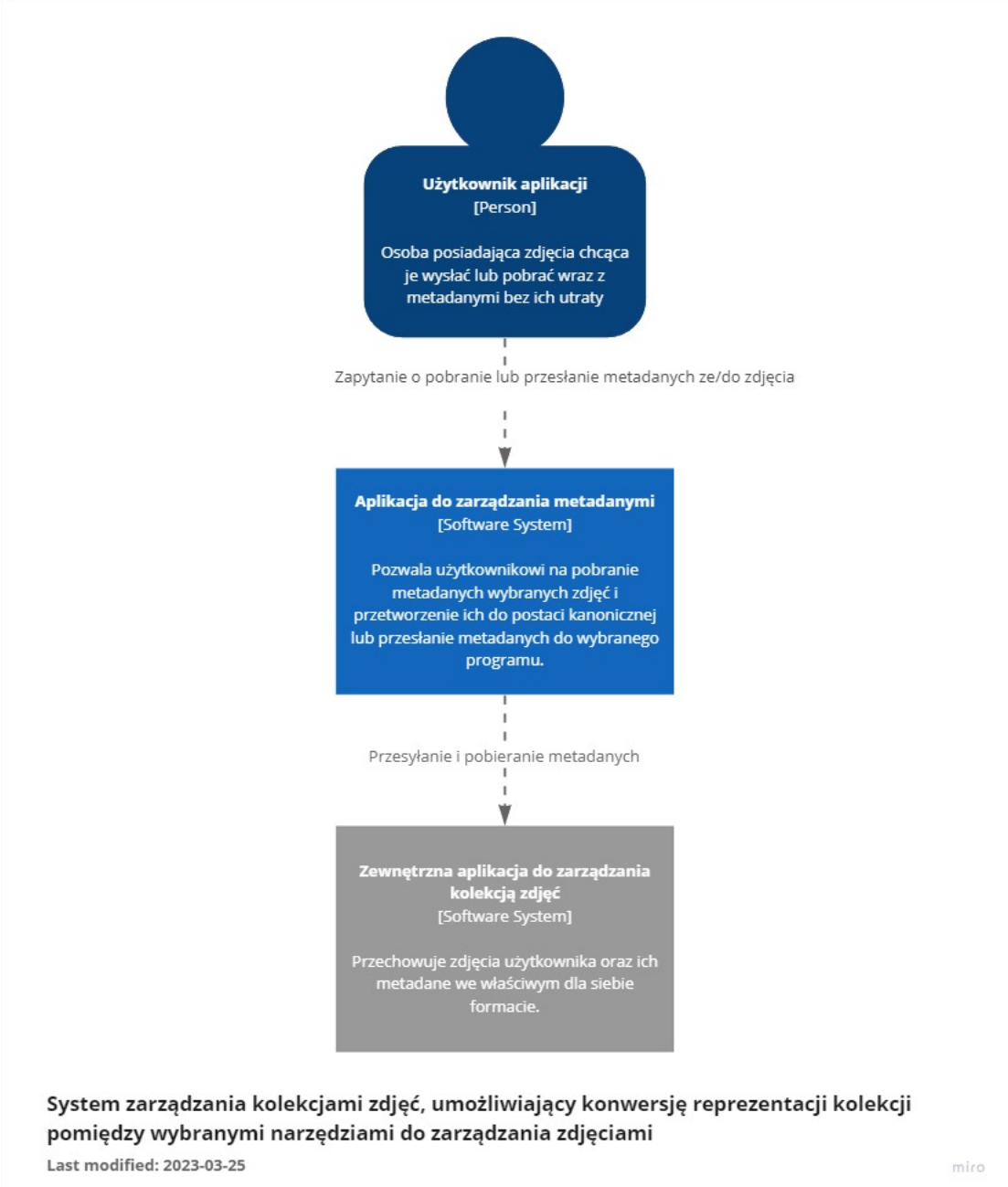
      1. Użytkownik podaje złe dane logowania do serwisu (punkty 4, 7)
         – Użytkownik jest proszony o ponowne podanie loginu i hasła.
      2. Użytkownik wybiera przycisk „Back” (punkty 3, 6)
         – System odrzuca opcje wybrane w tym punkcie i przenosi Użytkownika do poprzedniego ekranu.
      3. Użytkownik anuluje przesłanie zdjęć (punkt 10)
         – System nie dokonuje konwersji i przenosi Użytkownika do ekranu początkowego.
      4. Konwersja nie powodzi się (punkt 11)
         - System wyświetla informację dla Użytkownika o zaistniałych błędach,
         - Użytkownik wybiera opcję „Continue” i przechodzi do ekranu początkowego, gdzie może spróbować ponownie dokonać konwersji.

3 Architektura rozwiązania

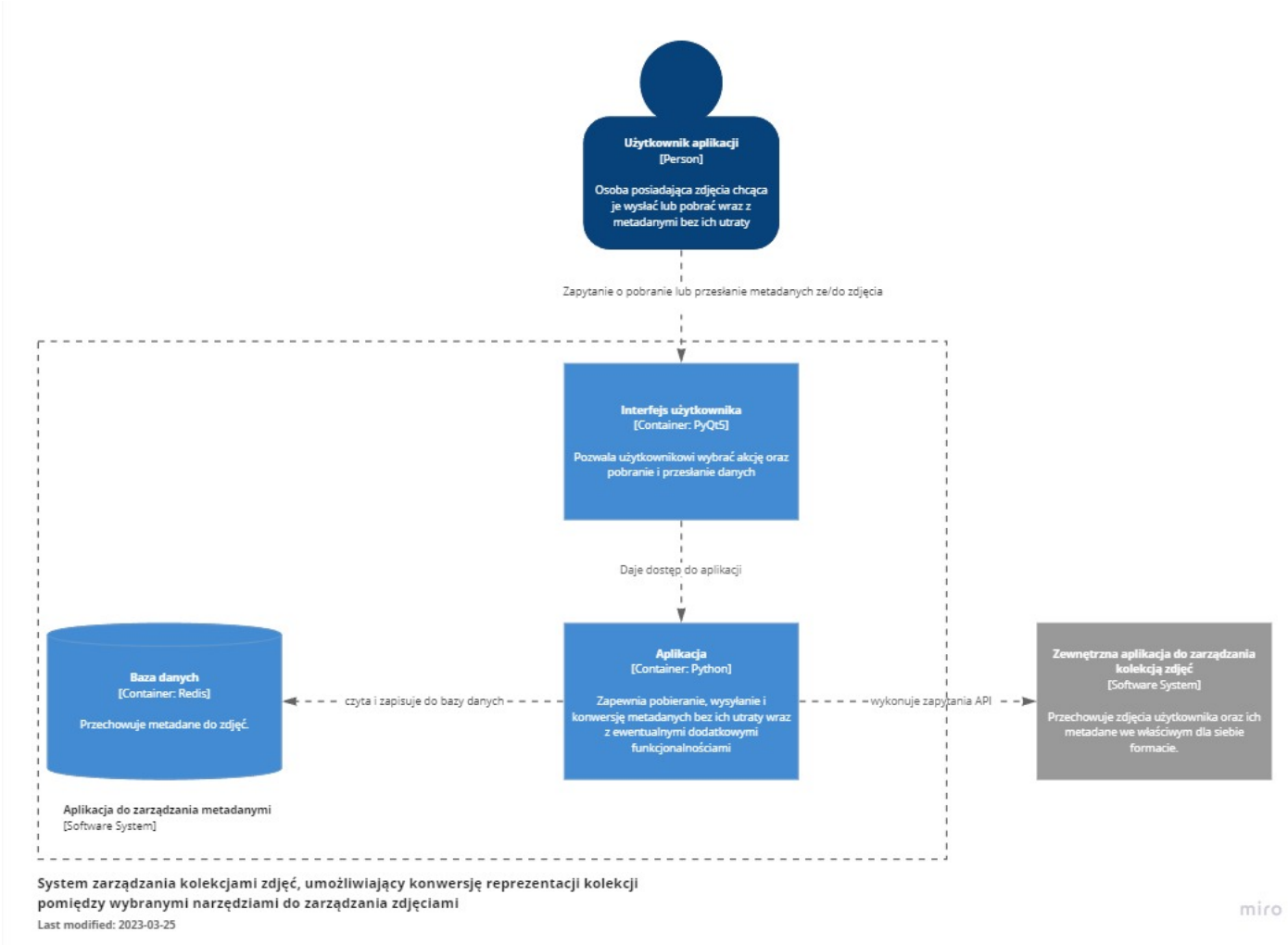
3.1 Zastosowane szablony architektoniczne

Użyliśmy do tego celu C4, czyli szczupłą technikę notacji graficznej służącą do modelowania architektury systemów oprogramowania.

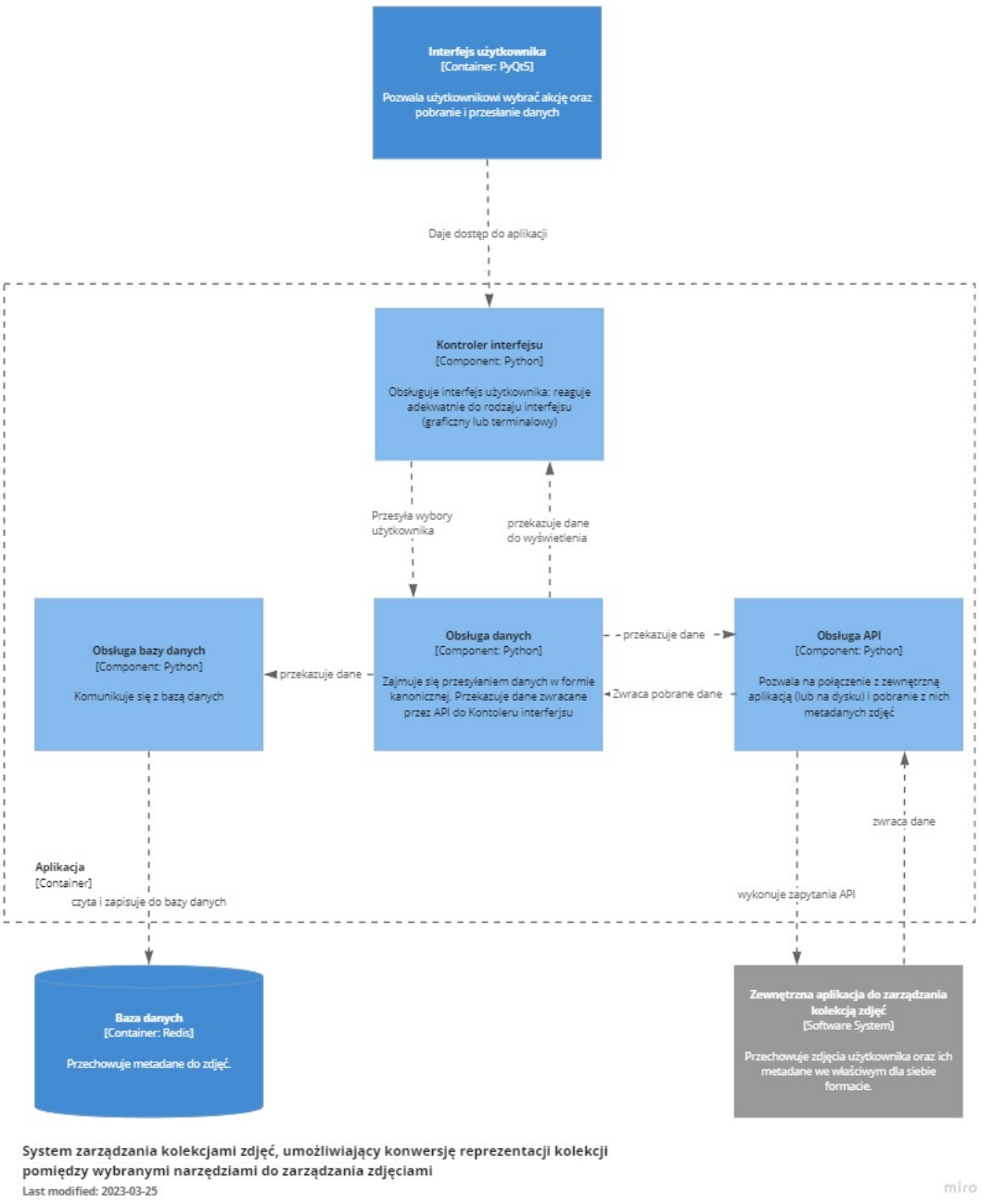
3.2 Diagram komponentów



Rysunek 1: System Context diagram



Rysunek 2: Container diagram



Rysunek 3: Component diagram

4 Dane trwałe - model logiczny danych

4.1 Sposób reprezentacji

Do przechowywania danych trwałych wybrano bazę nierelacyjną typu klucz-wartość Redis. Będzie używana ona lokalnie na każdej maszynie wykorzystującej aplikację.
Redis jest dobrym wyborem z kilku powodów:

- Liczba metadanych może się różnić dla różnych systemów zarządzania, ponieważ udostępniają one różne metadane,
- Przechowywane obiekty nie są ze sobą powiązane,
- Przechowywane dane nie podlegają aktualizacji w bazie danych.

4.2 Specyfikacja struktury danych

1. Kluczami w bazie są identyfikatory zdjęć w postaci stringów, np. "ANt9YmJTzA7PN1K7Zr2ilr", "Z3dvXr5f9O0RAOalgKzpN0", itd.

2. Wartością dla każdego klucza jest słownik zawierający wszystkie pola metadanych zdjęcia (EXIF + IPTC + XMP, czyli 400 pól) oraz dodatkowe informacje, takie jak osoby na zdjęciu oraz komentarze.

3. Pola metadanych są przechowywane w słowniku pod kluczem "basic" i są reprezentowane jako klucze-wartości, np. "Make": "Canon", "Model": "Canon EOS 5D Mark III", itd.

4. Dodatkowe pola przechowujące informacje pobrane za pomocą API, np. "people", "comments", "location" itd. Przy dodawaniu nowych API lista tych pól będzie rosła.

5. Przykładowy rekord w bazie:

```
{
    "basic": {
        "Make": "Canon",
        "Model": "Canon EOS 5D Mark III",
        "ExposureTime": 0.003125,
        "FNumber": 4.0,
        "ISO": 200,
        "DateTimeOriginal": "2022:04:27 15:10:30",
        "Software": "Adobe Photoshop Lightroom Classic 10.4",
        "...": "..."

    },
    "people": [
        {
            "name": "john_doe",
            "x": 100,
            "y": 200,
            "width": 50,
            "height": 100
        },
        {
            "name": "jane_smith",
            "x": 300,
            "y": 150,
            "width": 70,
            "height": 120
        }
    ],
    "comments": [
        {
            "owner": "joe_bloggs",
            "timestamp": 1234,
            "body": "Beautiful photo!"
        },
        {
            "owner": "boe_jogs",
            "timestamp": 3456,
            "body": "Cool!"
        }
    ]
}
```

5 Projekt i implementacja

5.1 Odnośnik do repozytorium kodu (dokładnie jedno repozytorium na projekt

`https://gitlab-stud.elka.pw.edu.pl/bpomiank/pzsp2_jdbim`

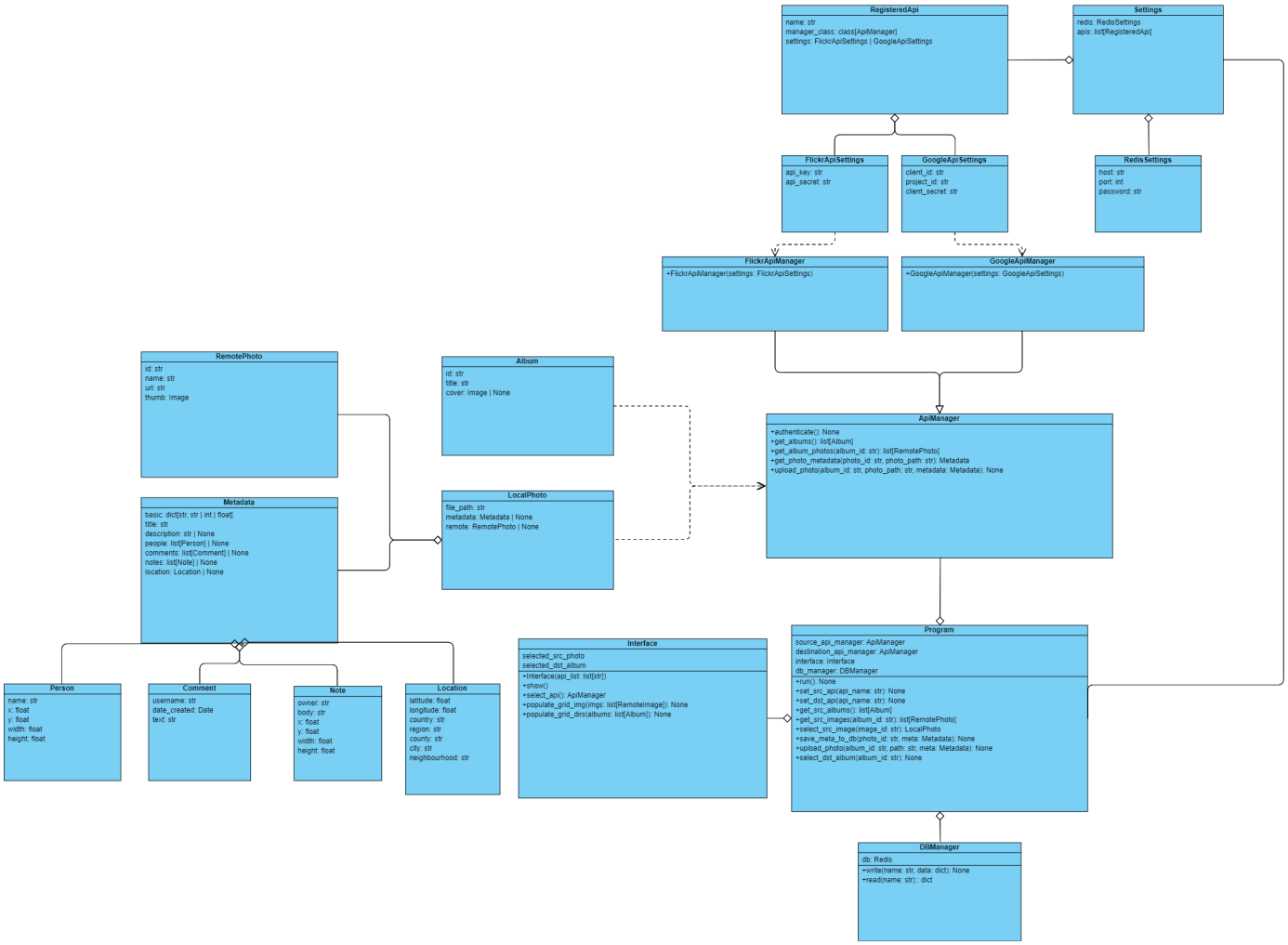
5.2 Lista języków programowania, bibliotek i środowisk

Całość implementacji w Pythonie, baza danych - Redis.
Użyte biblioteki w Pythonie:

- Związane z Google
  - google_auth_oauthlib.flow
  - google.auth.transport.requests
  - googleapiclient.discovery
  - itd...
- flickrapi
- Pillow do wyświetlania zdjęć
- Pydantic do zarządzania ustawieniami
- Requests do pobrania zdjęcia na desktop
- Os - kwestie związane ze ścieżkami
- Pickle - do zapisu tokenów
- Unittest - do testowania kodu

Do modelowania interfejsu: Qt Designer.

5.3 Diagram klas

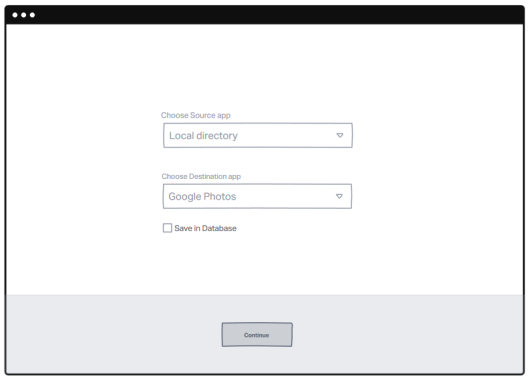


Rysunek 4: Diagram klas

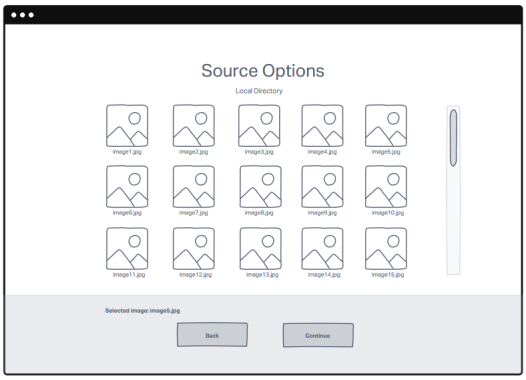
5.4 Statystyki

- Liczba plików - 40
- Linie kodu - 2671
- Liczba testów jednostkowych - 38
- Stopień pokrycia testów - 92% (wykluczając pliki z folderu gui/)

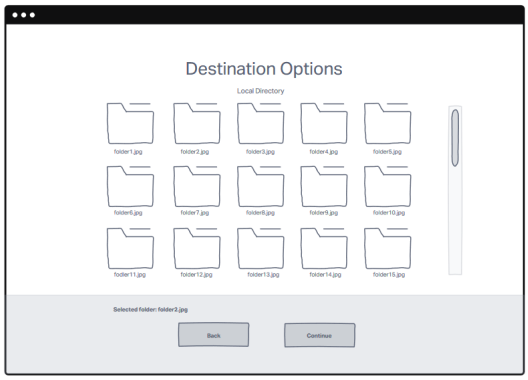
6 Warstwa prezentacji/interfejs użytkownika



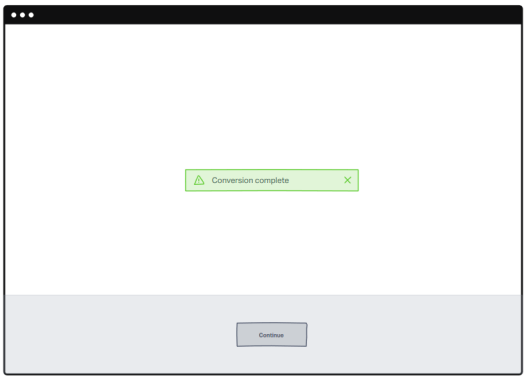
(a) Wybór programów (główny ekran)



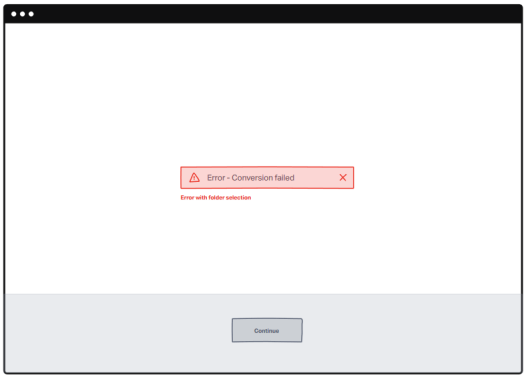
(b) Wybór zdjęcia do przesłania



(c) Wybór lokacji docelowej

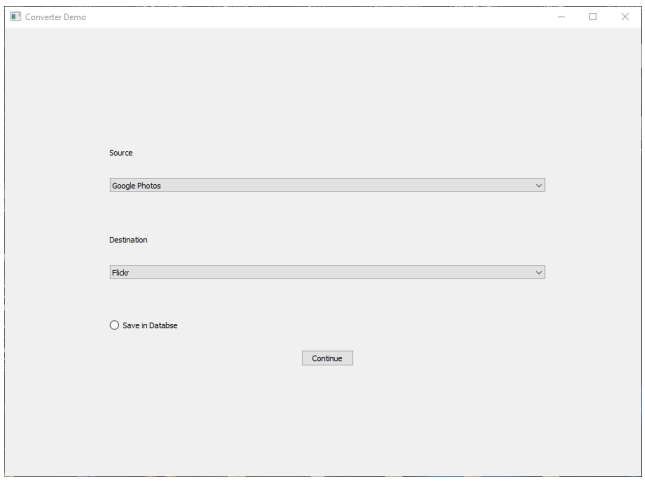


(d) Informacja o powodzeniu

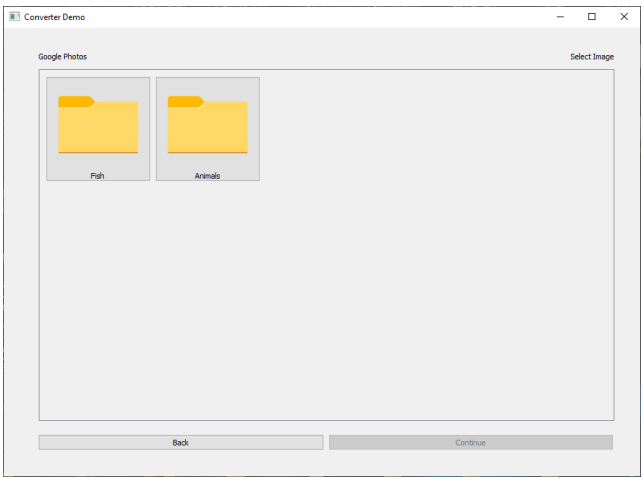


(e) Informacja o błędzie

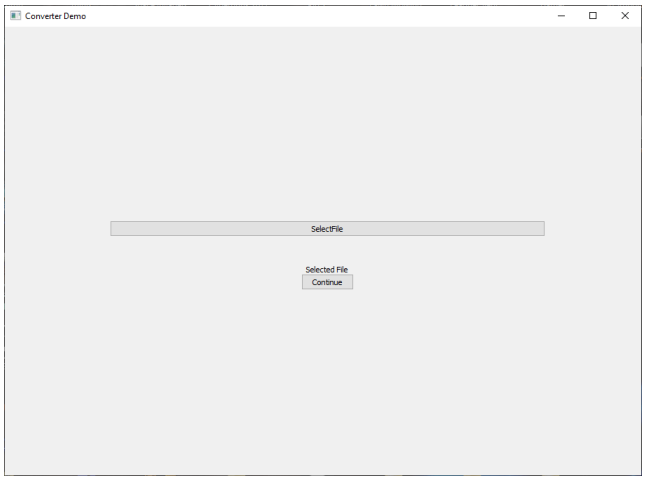
Rysunek 5: Prototypy ekranów aplikacji



(a) Wybór programów (główny ekran)



(b) Wybór zdjęcia do przesłania



(c) Wybór pliku lokalnego

Rysunek 6: Przykładowe ekrany z aplikacji

7 Wirtualizacja/konteneryzacja

Element nie występuje.

8 Bezpieczeństwo

Bezpieczeństwo korzystania z aplikacji silnie zależy od używanych API i sposobu, w jaki sposób autoryzowany jest użytkownik.

9 Podręcznik użytkownika

Do korzystania z programu potrzebny jest Python w wersji co najmniej 3.10 oraz baza danych Redis. W celu instalacji wszystkich niezbędnych bibliotek można wywołać w głównym katalogu projektu komendę:
pip install -r requirements.txt
Aby rozpocząć działanie programu, należy uruchomić w głównym katalogu skrypt 'main.py'.

Korzystanie z programu odbywa się poprzez interfejs graficzny. W trakcie transferu zdjęć konieczne będzie zalogowanie się do swoich kont w wybranych aplikacjach.
Po uruchomieniu, na ekranie startowym dostępne będą rozwijane menu, gdzie wybrać można dwa serwisy:

ten, z którego pobrane zostaną zdjęcia i ten, gdzie zostaną one przesłane. Można również zaznaczyć tutaj opcję przesyłania obrazów z dysku lub ściągnięcia ich na dysk.
W celu zapisania metadanych zdjęć w bazie danych, zaznaczyć trzeba opcję 'Save in database'.
Po kliknięciu przycisku 'Continue' program pokaże foldery z konta użytkownika w serwisie źródłowym (lub z dysku). Po wybraniu pożądanego katalogu, wyświetlone zostaną zawarte w nim zdjęcia. Każde zdjęcie można 'podejrzeć' i zobaczyć jego podstawowe metadane.
Gdy pożądane zdjęcie zostanie zaznaczone, a wybór potwierdzony przyciskiem 'Continue', nastąpi konwersja zdjęcia.
Po dokonaniu tych kroków, program zwróci informację o tym, czy transfer został zakończony powodzeniem. Następnie możliwe jest przejście z powrotem do ekranu startowego, gdzie można rozpocząć nową konwersję lub ponowić ostatnią próbę, jeśli nie przebiegła ona pomyślnie.

10 Podręcznik administratora

10.1 Podstawowe informacje

Program napisany jest w języku Python (wersja 3.10). Wymagana jest również baza danych Redis
Przed rozpoczęciem pracy z systemem należy zadbać o instalację wszystkich niezbędnych bibliotek. Można to zrobić poprzez wywołanie w głównym katalogu projektu komendy:
pip install -r requirements.txt
Znajdujący się w głównym katalogu plik 'main.py' jest głównym skryptem do uruchomienia przez użytkownika.

Poniżej znajdują się informacje dotyczące zalecanego sposobu implementacji rozszerzeń systemu, w szczególności dodawania obsługi nowych serwisów zarządzających kolekcjami zdjęć. W tej sekcji opisane zostały najważniejsze moduły i funkcjonalności systemu.

10.2 Dodawanie nowego serwisu – ApiManager

Zaprojektowany przez nas system umożliwia łatwe rozszerzenia o nowe obsługiwane programy. W tym celu w pakiecie 'api' wprowadzona została klasa abstrakcyjna ApiManager, która pełni rolę swoistego interfejsu i definiuje, jakie metody wykorzystywane są do przeprowadzania wszystkich niezbędnych operacji. W celu zaimplementowania w systemie nowego obsługiwanego serwisu, należy utworzyć klasę dziedziczącą po ApiManager i uzupełnić w niej metody:

- 'authenticate' – metoda uwierzytelnienia użytkownika w serwisie,
- 'get_albums' – uzyskanie listy wszystkich albumów użytkownika,
- 'get_album_photos' – uzyskanie listy zdjęć w danym albumie,
- 'get_photo_metadata' – pozyskanie metadanych zdjęcia,
- 'upload_photo' – przesłanie zdjęcia do serwisu.

10.3 Dodawanie nowego serwisu – konfiguracja

W pliku config.py znajduje się klasa Settings, która służy do przechowywania informacji o bazie danych i obsługiwanych programach. W celu dodania nowego serwisu potrzebne jest uzupełnienie listy zarejestowanych serwisów (settings.apis) o obiekt RegisteredApi, który będzie zawierał informacje o nowym serwisie: 'name' (nazwa), 'manager_class' (utworzona uprzednio klasa dziedzicząca po ApiManager) oraz 'settings'.

By uzupełnić atrybut ‘settings’, konieczne jest stworzenie klasy dziedziczącej po BaseSettings, zawierającej ustawienia potrzebne do uwierzytelniania aplikacji (przykładami takiej klasy są GoogleApiSettings, FlickrApiSettings).

10.4 Dodawanie nowego serwisu – postać kanoniczna metadanych

Postać kanoniczna metadanych zdefiniowana jest w pliku models.py. Znajdują się tam klasy odpowiadające poszczególnym informacjom o obrazach. Są to na przykład:

- Person – informacje o osobie znajdującej się na zdjęciu (imię, umiejscowienie na zdjęciu),
- Location – lokalizacja, w której zdjęcie zostało zrobione (m.in. kraj, region, miasto oraz współrzędne geograficzne),
- Title – tytuł,
- Description – opis,
- Notes – informacje o notatkach pozostawionych na zdjęciu przez użytkownika,
- Comments – komentarze pod zdjęciem.

Klasa Metadata agreguje wszystkie tak zdefiniowane rodzaje metadanych. Posiada również atrybut ‘basic’, w którym zapisywane są w formie słownika podstawowe informacje o zdjęciu, pobrane bezpośrednio z obrazu.
Podczas implementacji obsługi nowego serwisu należy zadbać, aby pozyskiwane przez nas metadane reprezentować poprzez tworzenie odpowiednich instancji powyższych obiektów. Jeśli dodawany serwis oferuje uzupełnianie kolekcji zdjęć o nowe informacje, niezawarte dotychczas w systemie, to możliwe jest dodawanie własnych typów metadanych. Można to zrobić przez zdefiniowanie nowych klas, które będą je reprezentować i dodanie ich jako atrybut do klasy Metadata.

10.5 Dodawanie nowego serwisu – zapis w bazie danych

Do przechowywania danych trwałych służy Redis - baza nierelacyjna typu klucz-wartość. Metody zapisu i odczytu metadanych zaimplementowane są w pliku db.py. Kluczami w bazie są identyfikatory zdjęć w postaci stringów, wartością dla każdego klucza jest słownik zawierający wszystkie pola metadanych zdjęcia. Przed zapisem słownik ten tworzony jest poprzez konwersję obiektu Metadata metodą ‘object_to_dict’ zaimplementowaną w pliku utils.py.

10.6 Dodawanie nowego serwisu – funkcje pomocnicze

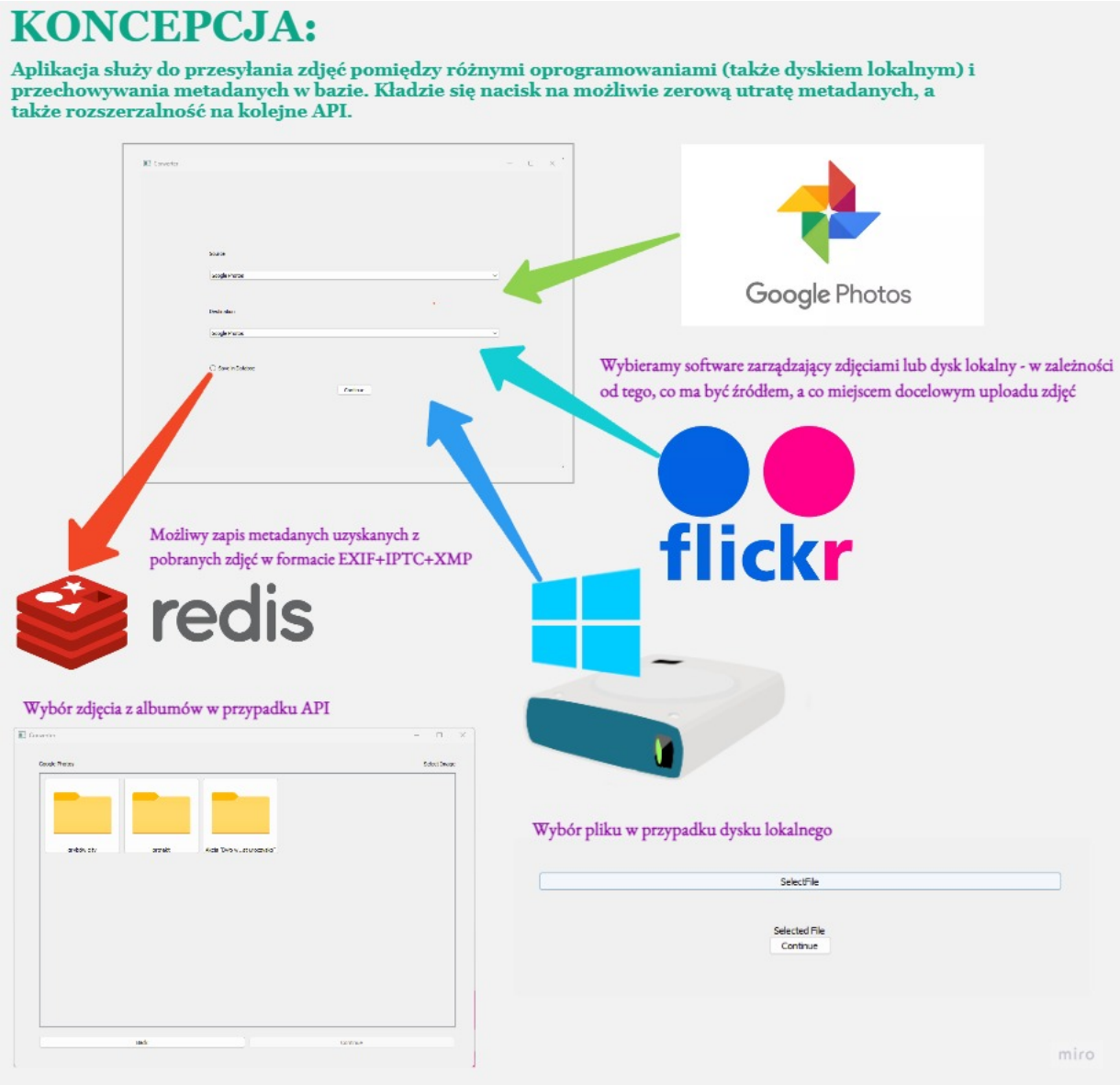
Oprócz ‘object_to_dict’ w utils.py znajdują się inne funkcje, które mogą okazać się przydatne w pracy z kodem. Są to:

- get_metadata – pobiera podstawowe metadane bezpośrednio z obrazu i zwraca je w postaci słownika,
- get_photo_from_url – zwraca obraz z podanego URL bez zapisywania go na dysku,
- download_photo – pobiera zdjęcie z podanego URL.

11 Podsumowanie

Realizacja projektu przebiegła raczej pomyślnie, choć napotkaliśmy pewne trudności. Głównie związane były one ze specyfiką poszczególnych API, np. różnicami w metodach uwierzytelniania użytkowników serwisów Flickr i Google, co powodowało pewne problemy implementacyjne. Dość trudne okazało się sformułowanie postaci kanonicznej metadanych, gdyż z naszych pierwszych analiz wynikało, że wszystkie istotne informacje na temat obrazów zawarte są już w standardowych formatach, takich jak EXIF czy IPTC. Ostatecznie jednak znaleźliśmy możliwe do pozyskania dane dotyczące zdjęć przechowywane przez różne serwisy, które nie są częścią podstawowych formatów metadanych i które warto dołączyć do proponowanej przez nas formy kanonicznej.
Niektóre elementy projektu zajęły więcej czasu niż przewidzieliśmy, czego efektem są pewne uproszczenia. Udało się jednak stworzyć działający system, który pozwala na transfer zdjęć przy zachowaniu istotnych informacji. Zastosowany przez nas system reprezentacji metadanych jest czytelny i łatwo rozszerzalny o nowe elementy. Sam program daje możliwość integracji z nowymi serwisami i jest ułatwieniem dla użytkownika, który może w przystępny sposób zarządzać swoją kolekcją zdjęć.

11.1 Rysunek



Rysunek 7: Wizualizacja projektu

11.2 Możliwe kierunki rozwoju

Aplikacja umożliwia łatwe rozszerzanie w kwestii dostępnych serwisów zarządzających kolekcją zdjęć. Dodatkowymi funkcjonalnościami, które warto rozwinąć w przyszłości, jest pobieranie większej liczby obrazów na raz a także bardziej efektywny sposób przesyłania zdjęć. Istnieje również wiele możliwości wprowadzenia ulepszeń w GUI, tak, by było coraz bardziej responsywne oraz lepiej spełniające oczekiwania klienta.

Literatura

[1] `https://neededapps.com/tutorials/what-are-image-metadata-exif-iptc-xmp/` - opis różnych formatów metadanych.

[2] `https://developers.google.com/photos/library/reference/rest` - dokumentacja API Google Photos.

[3] https://www.flickr.com/services/api/ - dokumentacja API Flickr.