Studio Projektowe

Wykonawcy: Anna Wójcik, Karolina Pieszczek

COLOR ♥ ME

Wstęp

ColorMe jest to prosta aplikacja przeglądarkowa, która wykorzystuje modele sztucznej inteligencji do automatycznego kolorowania czarno-białych zdjęć oraz lineartów. Obrazy na wejściu mogą być w skali szarości albo zawierać same kontury. Na wyjściu otrzymujemy gotowy kolorowy obraz. Aby bardziej dopasować końcowy wynik do naszych potrzeb możliwe jest nałożenie kolorowego filtru.

Opis rekomendowanego środowiska deweloperskiego oraz instrukcje

Projekt należy pobrać z repozytorium https://github.com/wojcikania12/ColorMe. W folderze *rest* znajduje się serwer aplikacji, natomiast w folderze *color_me* część odpowiedzialna za interfejs użytkownika.

Wymagania

Backend:
Python 3.7

Frontend:
Do stworzenia interfejsu użytkownika wykorzystano środowisko pracy stworzone przez Create React App. Aby móc uruchomić aplikację na swoim komputerze, będziesz potrzebować Node >= 8.10 oraz npm >= 5.6.

Biblioteki zewnętrzne

Backend:

- **numpy** - 1.18.5
- **Pillow** - wersja 8.0.1
- **flask** - wersja 1.1.2
- **flask_cors** - wersja 3.0.9
- **flask_restplus** - wersja 0.13.0
- **opencv-python** - wersja 4.4.0.46
- **tensorflow** - wersja 2.3.0

Frontend:

- Standardowe biblioteki ReactJS
- **ReactColor** - biblioteka wykorzystana do komponentu wyboru koloru filtra
- **SlickCarousel** - bibliotek użyta do wyświetlenia tzw. karuzeli, czyli pokazu slajdu kilku obrazków, widocznej na stronie głównej aplikacji.

Instrukcja uruchomienia serwera

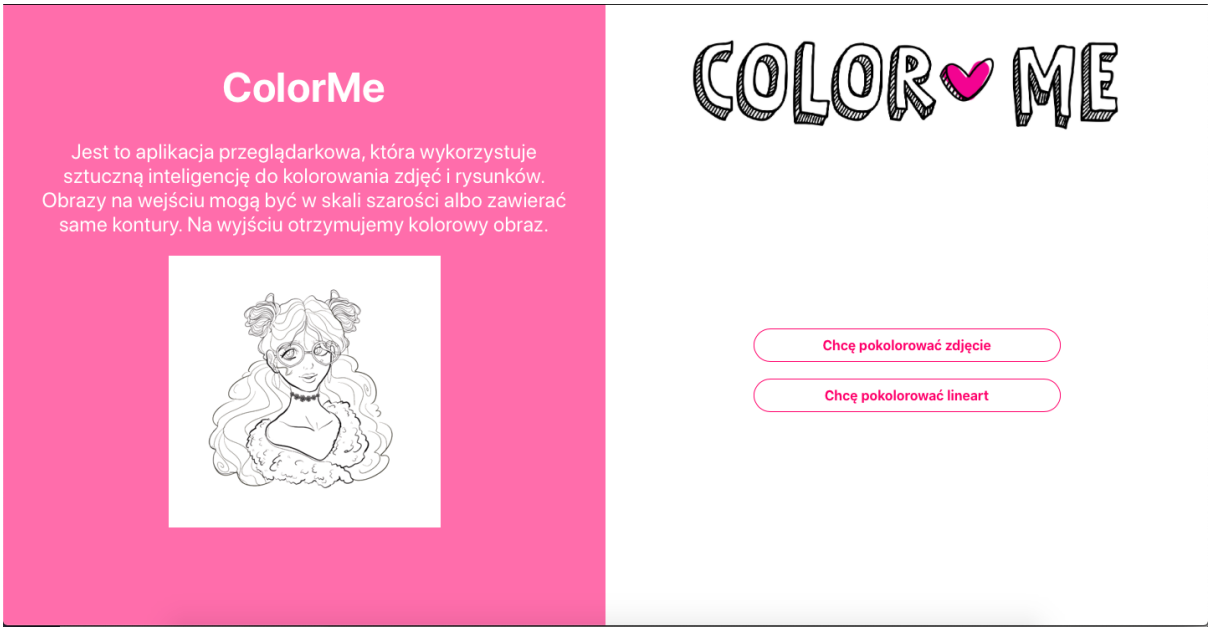
W celu uruchomienia serwera lokalnie(port 5000) należy wykonać plik *run_rest_interface.py*.

Instrukcja uruchomienia interfejsu użytkownika

W konsoli przejdź do pobranego folderu *./ColorMe/color_me/* i zawołaj komendę `npm install`.
Komenda ta pobierze potrzebne moduły, wykorzystane w aplikacji. Możesz uruchomić aplikację poleceniem `npm start`. Strona z aplikacją automatycznie się otworzy, jeśli jednak z jakiegoś powodu to się nie stało otwórz w przeglądarce stronę *http://localhost:3000/*. Pamiętaj, aby przed uruchomieniem interfejsu użytkownika uruchomić serwer.

Aplikacja jest gotowa do wdrożenia do środowiska produkcyjnego, zastosuj komendę `npm run build`. Dzięki temu uzyskasz zoptymalizowaną wersję swojej aplikacji. Znajdziesz ją w folderze *build*.

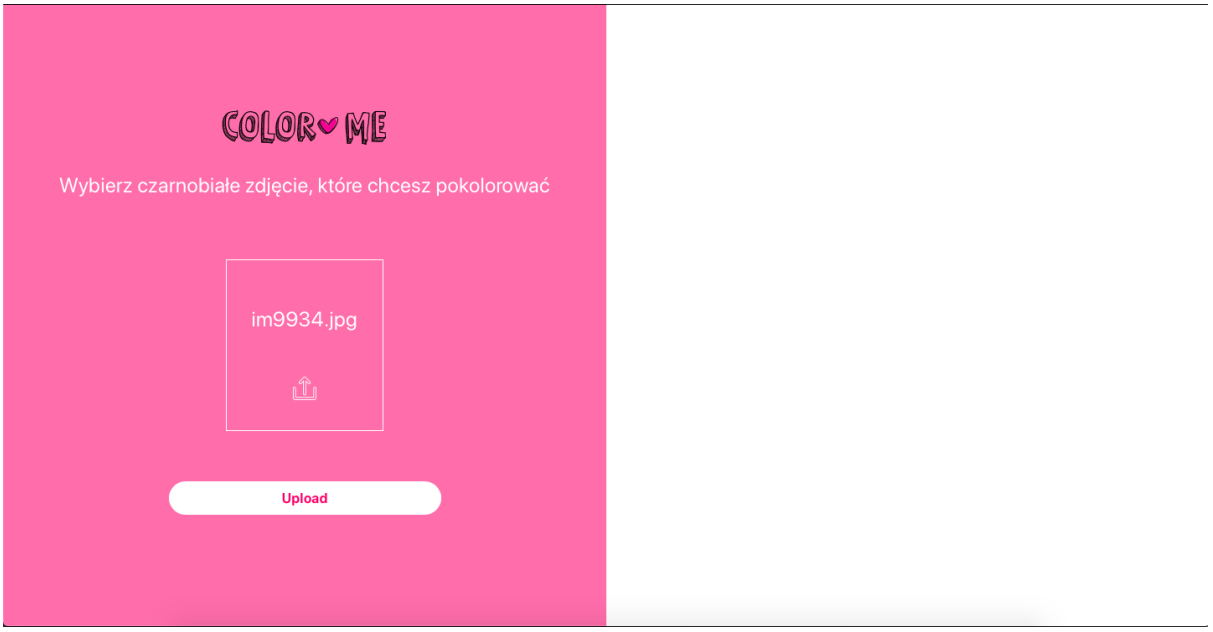
Instrukcja korzystania z aplikacji



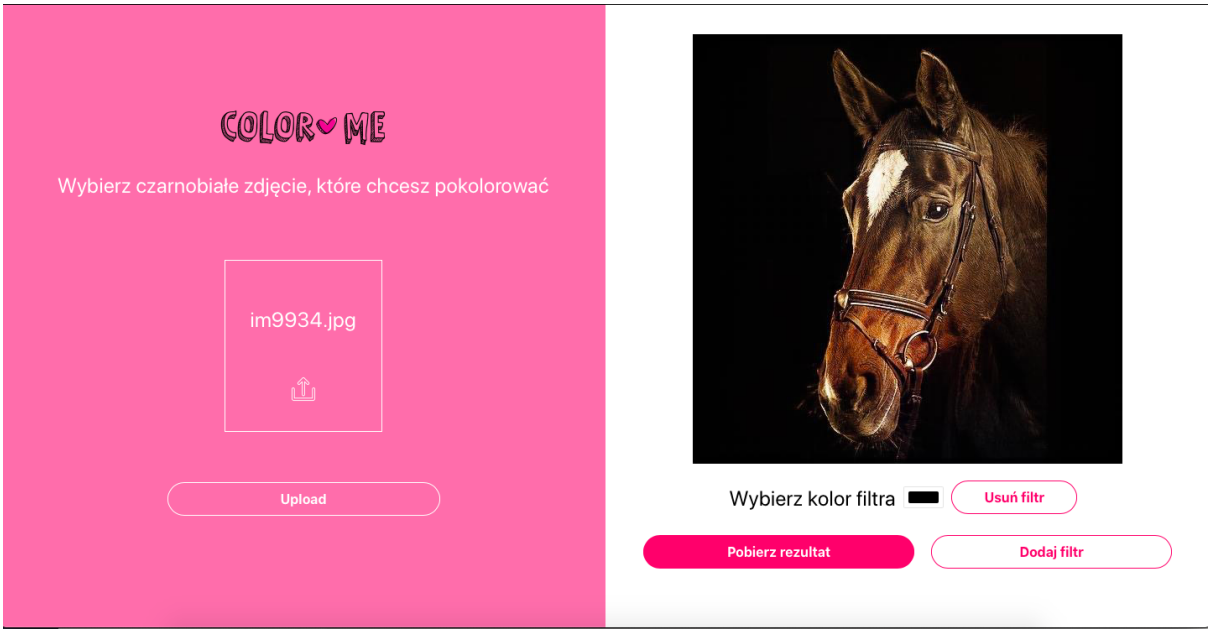
Po uruchomieniu aplikacji, na stronie głównej widzimy jej krótki opis wraz z przykładami, które zostały one uzyskane przy jej pomocy oraz dwa przyciski, które dają nam możliwość wyboru czy chcemy kolorować lineart czy zdjęcia. Kategorie te zostały rozróżnione ze względu na zastosowanie dwóch osobnych modeli sieci GAN.

Po wyborze kolorowania zdjęć, widzimy ekran podobny do głównej strony, z tym, że jedna część jest pusta.

Różowa część ekranu służy wyborowi zdjęcia, które chcemy kolorować. Gdy już się zdecydujemy klikamy przycisk “upload”, który przy pomocy odpowiedniego endpoint’u wysyła zdjęcie na serwer.

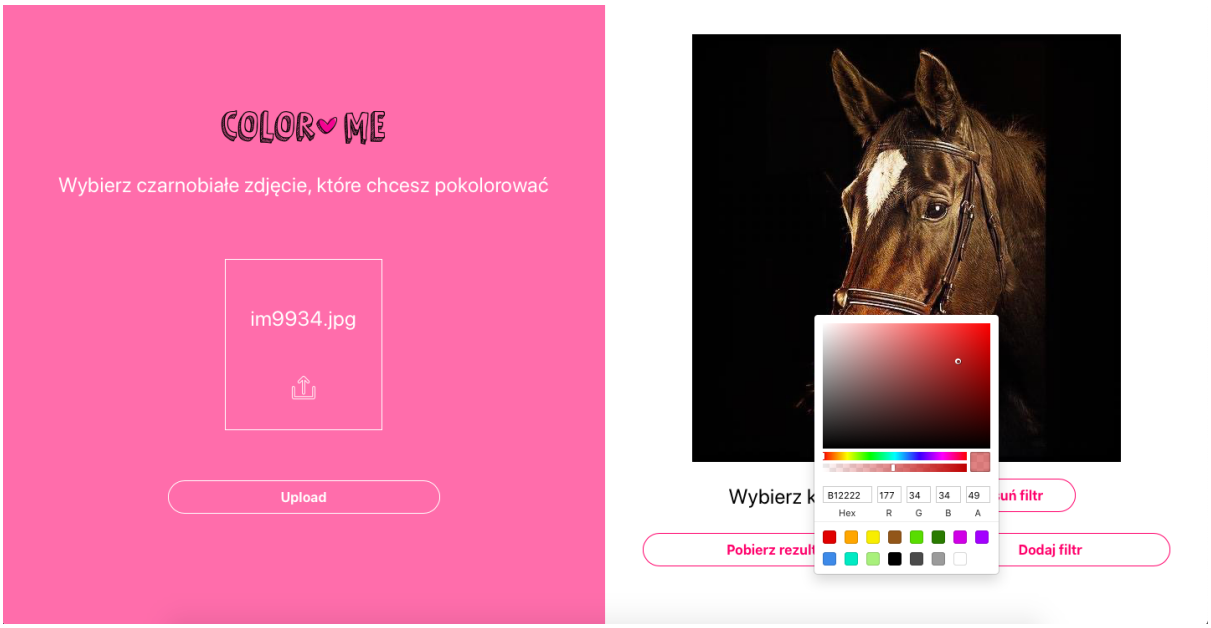


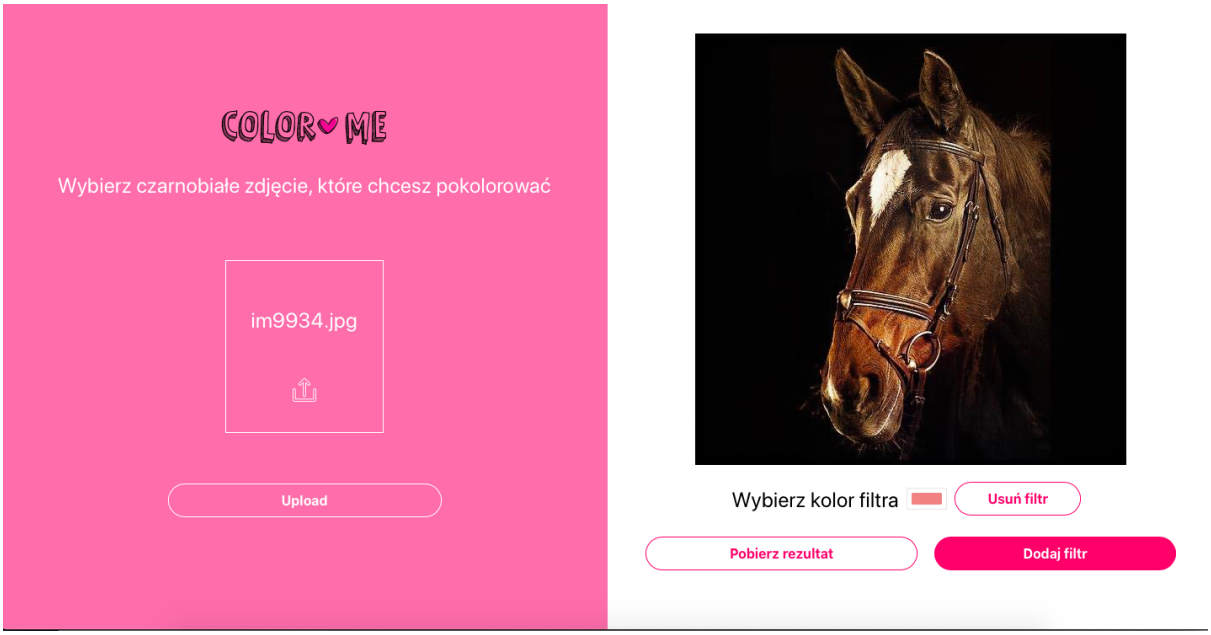
Odpowiedź z kolorowym zdjęciem pojawia się na, do tej pory, pustej części ekranu.



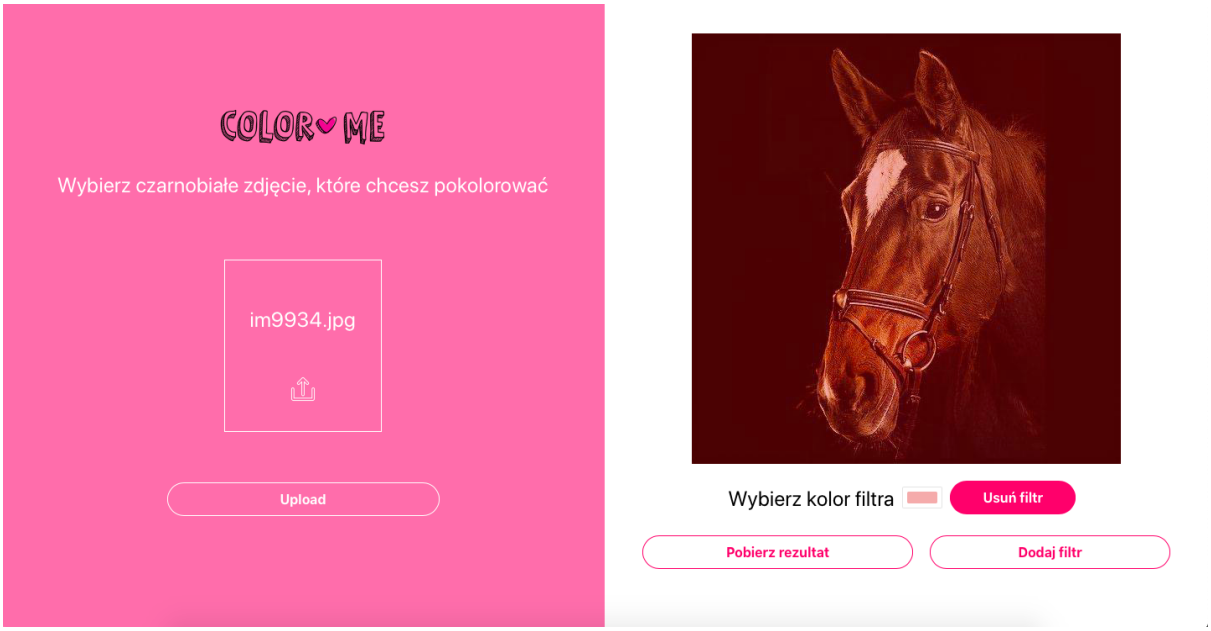
Otrzymany wynik możemy od razu pobrać lub lekko go zmodyfikować poprzez dodanie kolorowego filtra.

Po kliknięciu prostokąta z kolorem filtra otwiera nam się niewielki popup z możliwością wyboru koloru oraz jego transparentności.



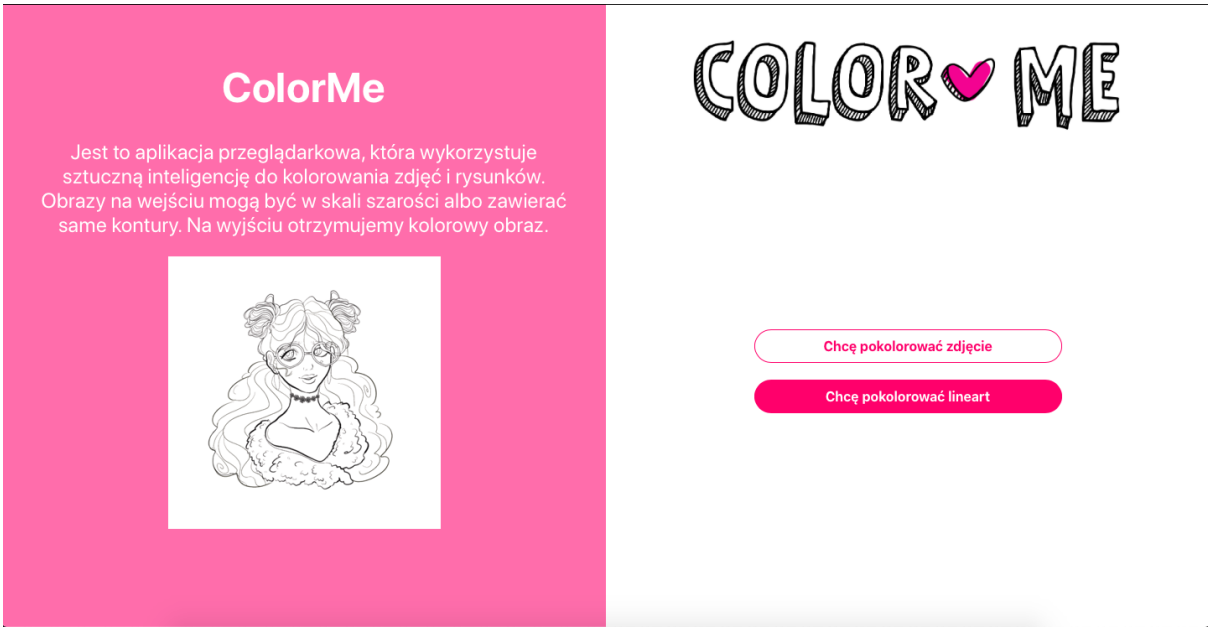


Aby zastosować wybrany filtr należy kliknąć przycisk “dodaj filtr”. Podobnie jak w przypadku przycisku “upload”, zostaje zawołany odpowiedni endpoint oraz wysłane zostają informacje o obrazku i filtrze.

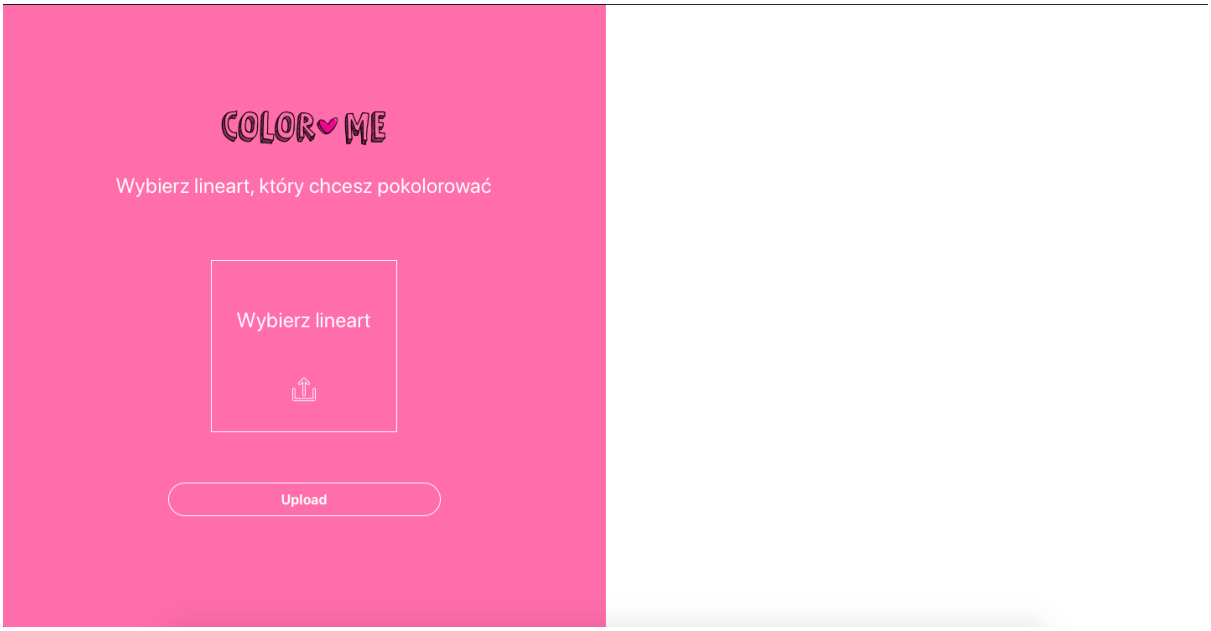


Po otrzymaniu odpowiedzi z serwera, rezultat pojawia nam się w tym samym miejscu, co poprzednio. Jeśli nie podoba nam się efekt i chcemy przywrócić wcześniejszą opcję klikamy “usuń filtr”. Jeśli chcemy dodać jeszcze jeden filtr ponawiamy całą procedurę. Gdy otrzymamy już efekt, który nas zadowala możemy pobrać zdjęcie.

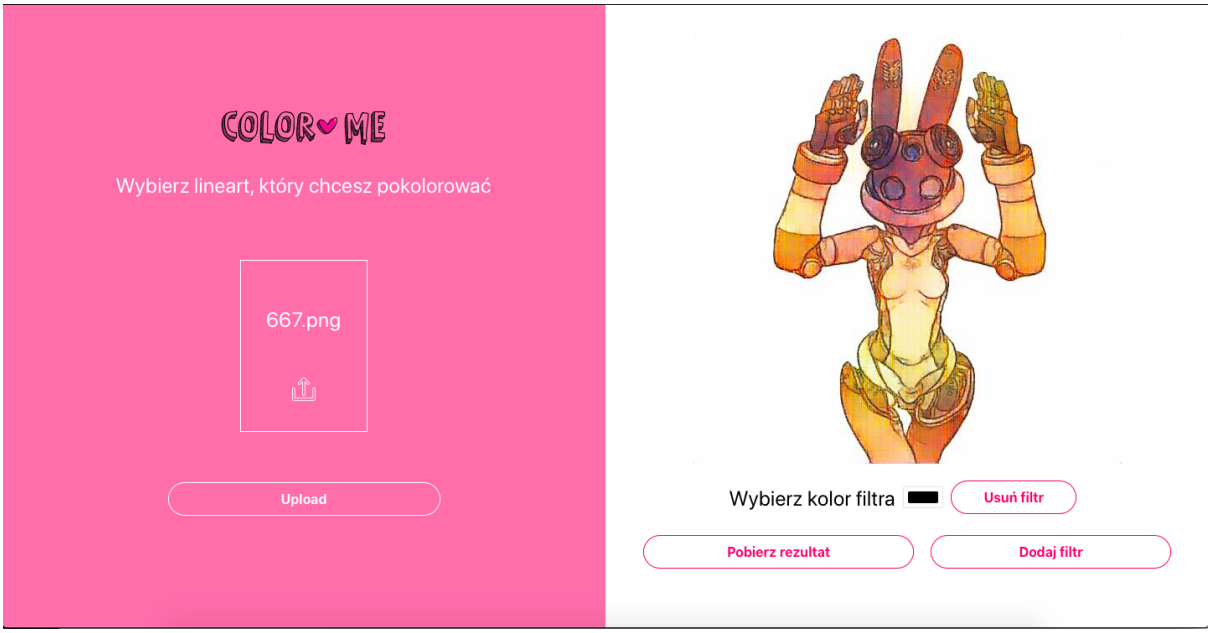
Aby powrócić do strony głównej należy kliknąć logo ColorMe.



W przypadku, gdy nie chcemy kolorować zdjęć lecz linearty, procedura wygląda analogicznie.



Zaczynamy od wyboru oraz wysłania rysunku na serwer, a następnie podobnie jak w przypadku zdjęć rezultat możemy lekko zmodyfikować poprzez nałożenie filtru i pobrać.

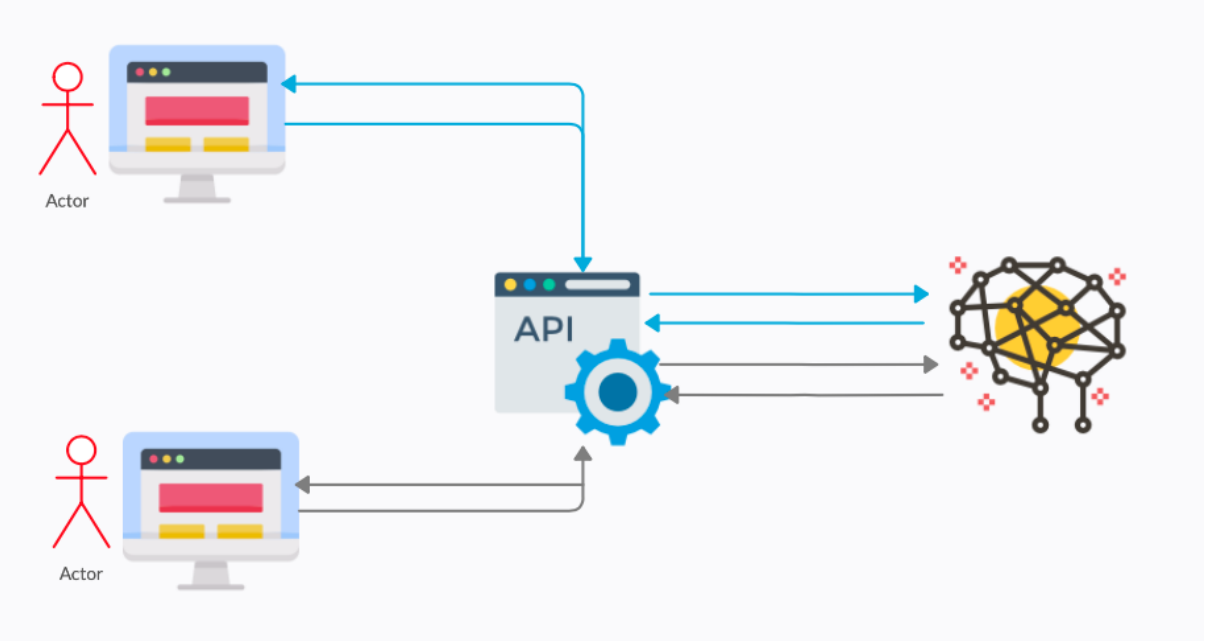


Architektura zastosowanego rozwiązania

Zastosowana została architektura restowa. Interfejs użytkownika komunikuje się z serwerem za pomocą wystawionych endpoint'ów. Serwer z kolei otrzymane dane przesyła do nauczonego modelu sieci GANs..

Technologie

- Frontend: React
- Backend: Flask
- AI: Python (TensorFlow oraz Keras)



Najważniejszą częścią aplikacji jest model AI odpowiedzialny za automatyczne kolorowanie zdjęć i rysunków.

Przypadki użycia

- **Nazwa:** Kolorowanie czarno-białych fotografii
  Inicjator: Użytkownik
  Cel: Otrzymać automatycznie pokolorowane zdjęcie
  Główny scenariusz:
  1. Użytkownik wgrywa obraz
  2. Użytkownik wybiera opcję kolorowania zdjęć
  3. System koloruje automatycznie zdjęcie z wykorzystaniem nauczonego modelu
  4. Użytkownik otrzymuje wynik
  5. Użytkownik opcjonalnie wybiera filtr nakładany na obraz

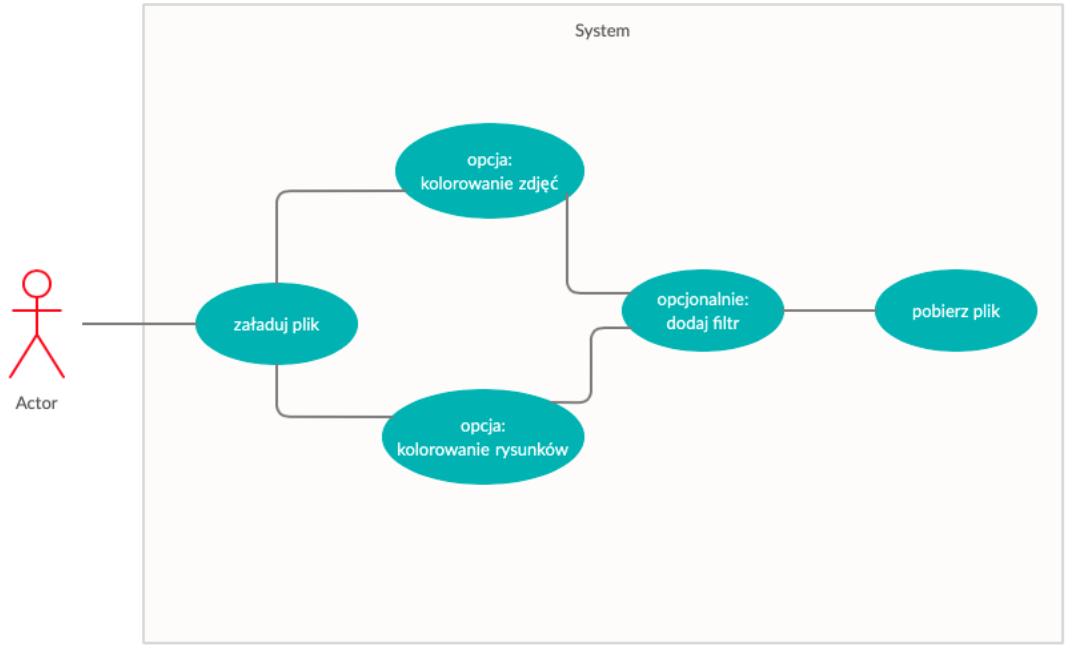
6. Użytkownik pobiera gotowy obraz

Rozszerzenia:

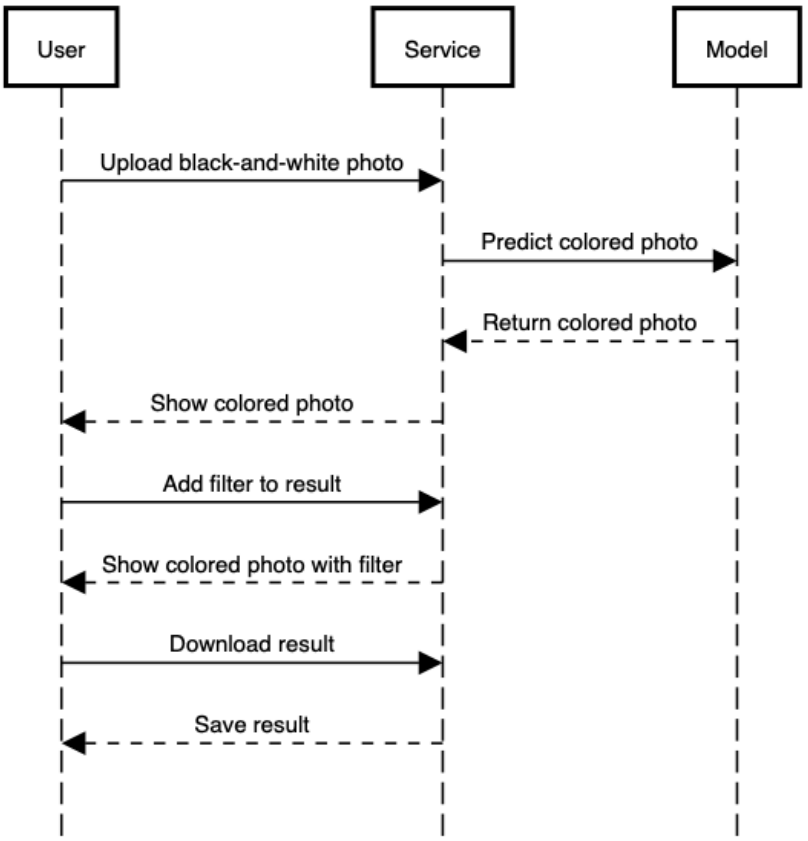
1a. Użytkownik przez pomyłkę wgrywa szkic

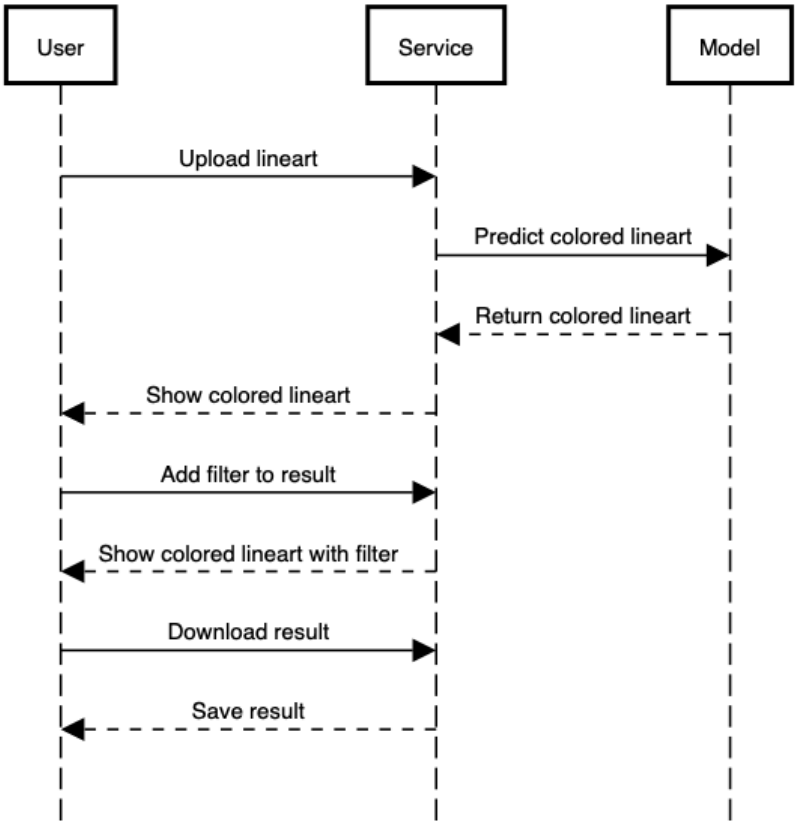
a. Niepowodzenie

- **Nazwa:** Kolorowanie lineartu
  Inicjator: Użytkownik
  Cel: Otrzymać automatycznie pokolorowany szkic
  Główny scenariusz:
  1. Użytkownik wgrywa obraz
  2. Użytkownik wybiera opcję kolorowania szkicu
  3. System koloruje automatycznie szkic z wykorzystaniem nauczonego modelu
  4. Użytkownik otrzymuje wynik
  5. Użytkownik opcjonalnie wybiera filtr nakładany na obraz
  6. Użytkownik pobiera gotowy obraz
  Rozszerzenia:
  1a. Użytkownik przez pomyłkę wgrywa zdjęcie
  a. Niepowodzenie

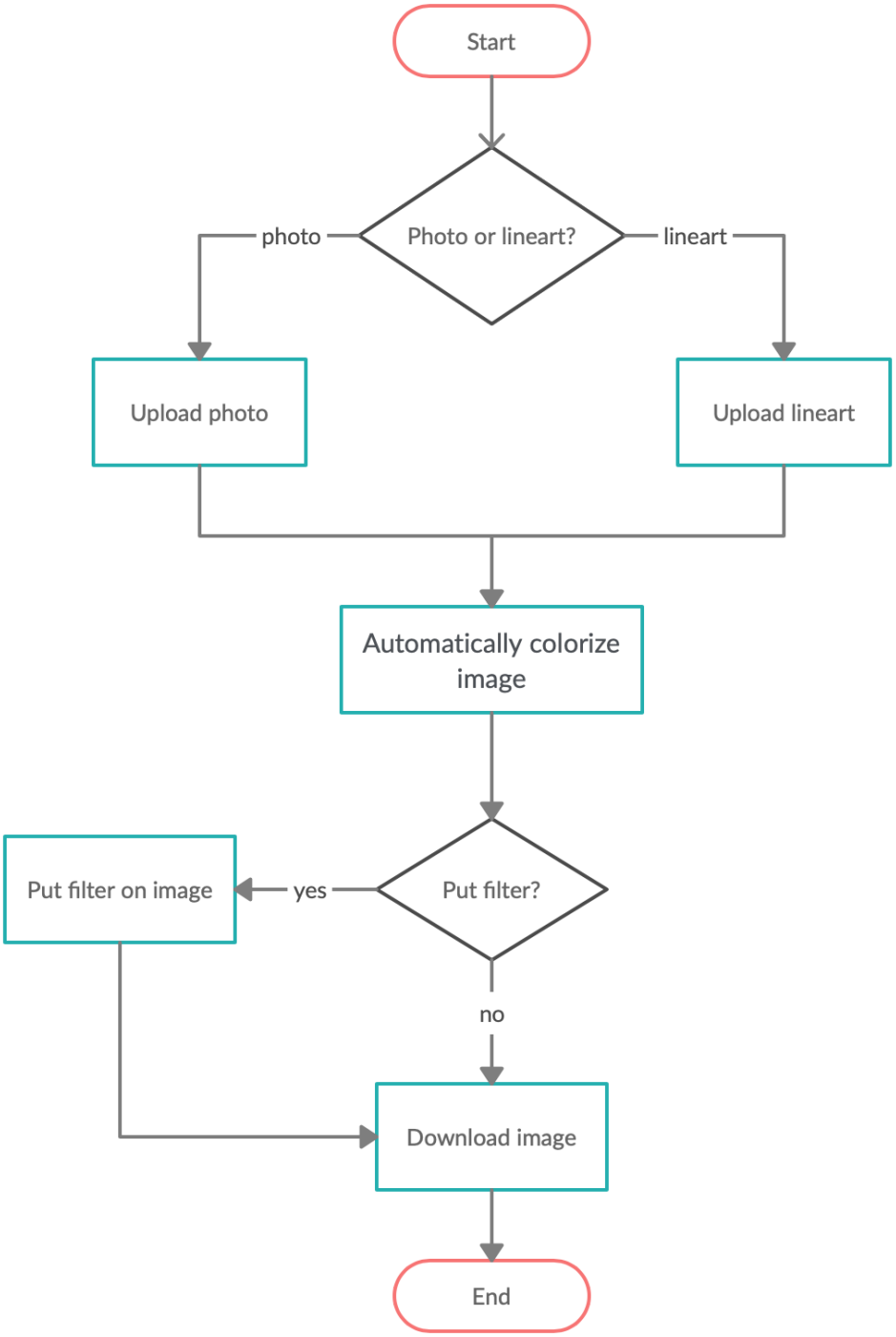


Diagramy sekwencji

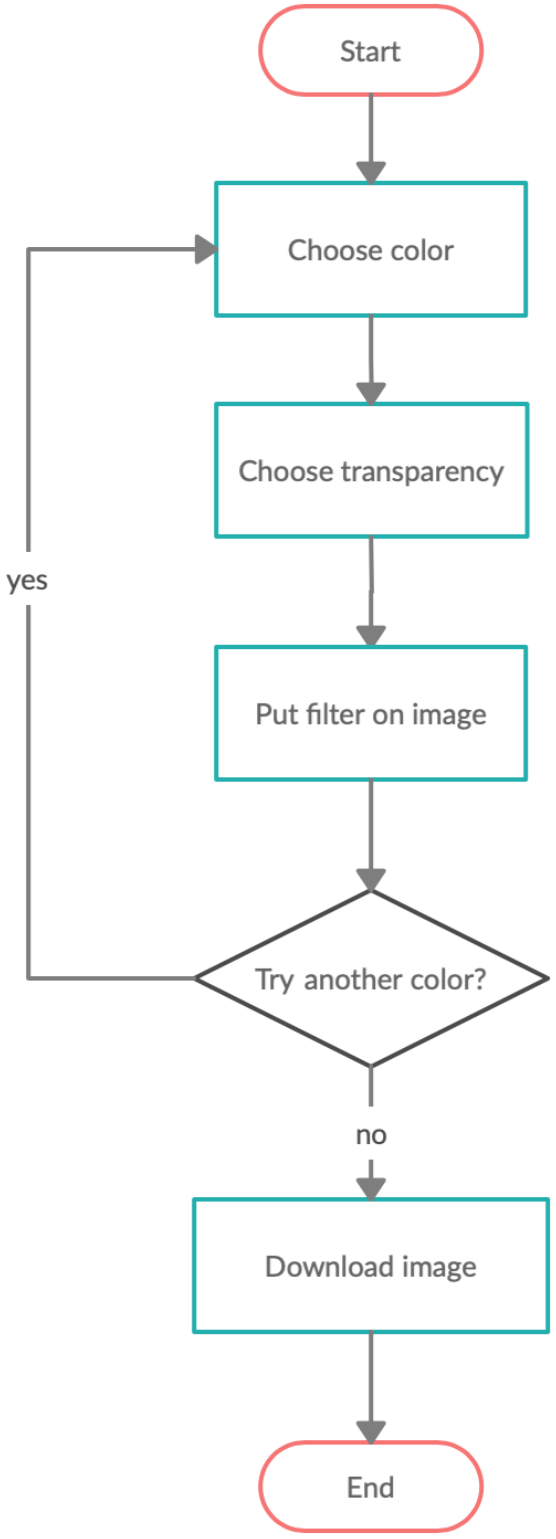




Kolorowanie



Wybieranie filtra

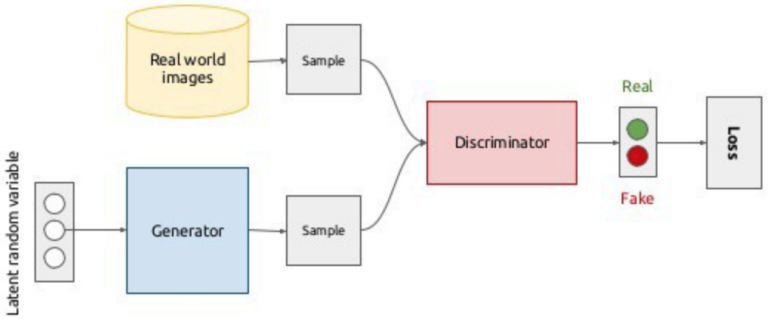


Kluczowe elementy architektury

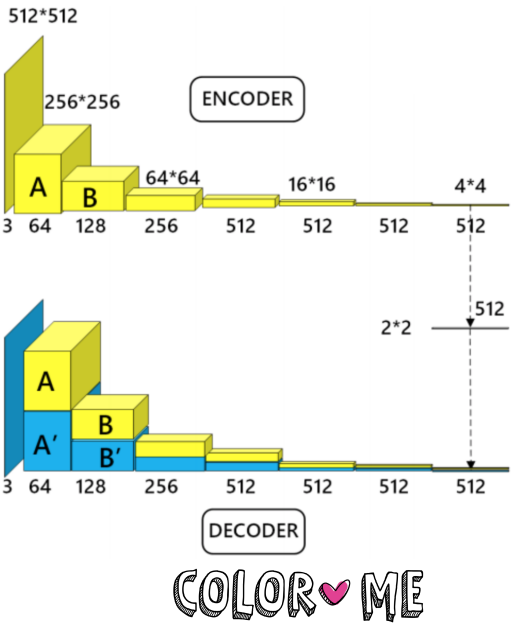
GAN (Generative Adversarial Network)

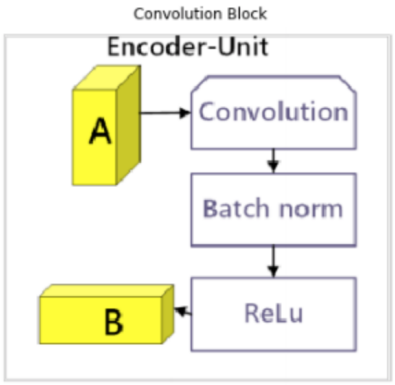
Sieci GAN składają się z 2 sieci neuronowych, generatora i dyskryminatora, które rywalizują ze sobą, aby zminimalizować lub zmaksymalizować określoną funkcję. Dyskryminator pobiera prawdziwe dane treningowe, a także wygenerowane (fałszywe) dane z generatora i musi wyprowadzać prawdopodobieństwo, że każdy obraz jest prawdziwy. Jego celem jest maksymalizacja liczby poprawnie klasyfikujących typ danych, które podaje, podczas gdy generator próbuje zminimalizować tę liczbę. Stara się, aby dyskryminator sklasyfikował wygenerowany wynik jako rzeczywisty, czyniąc dyskryminator mniej poprawnym.

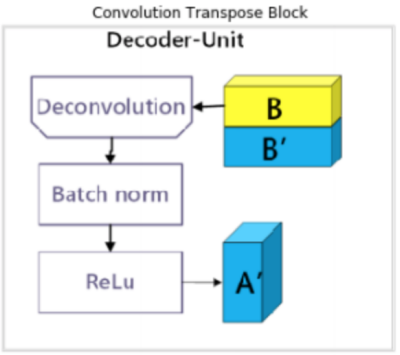
Struktura sieci



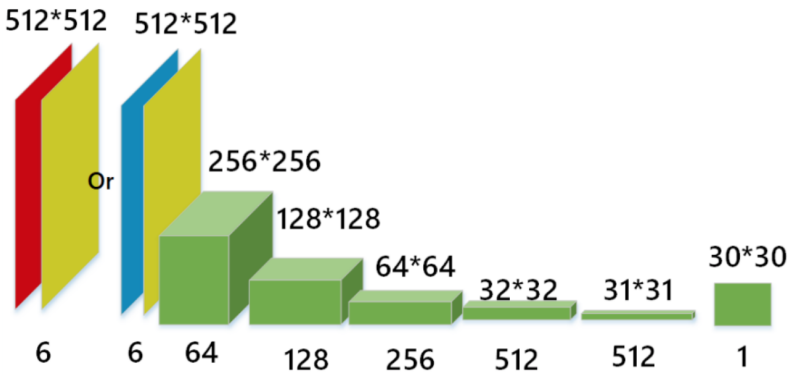
Generator



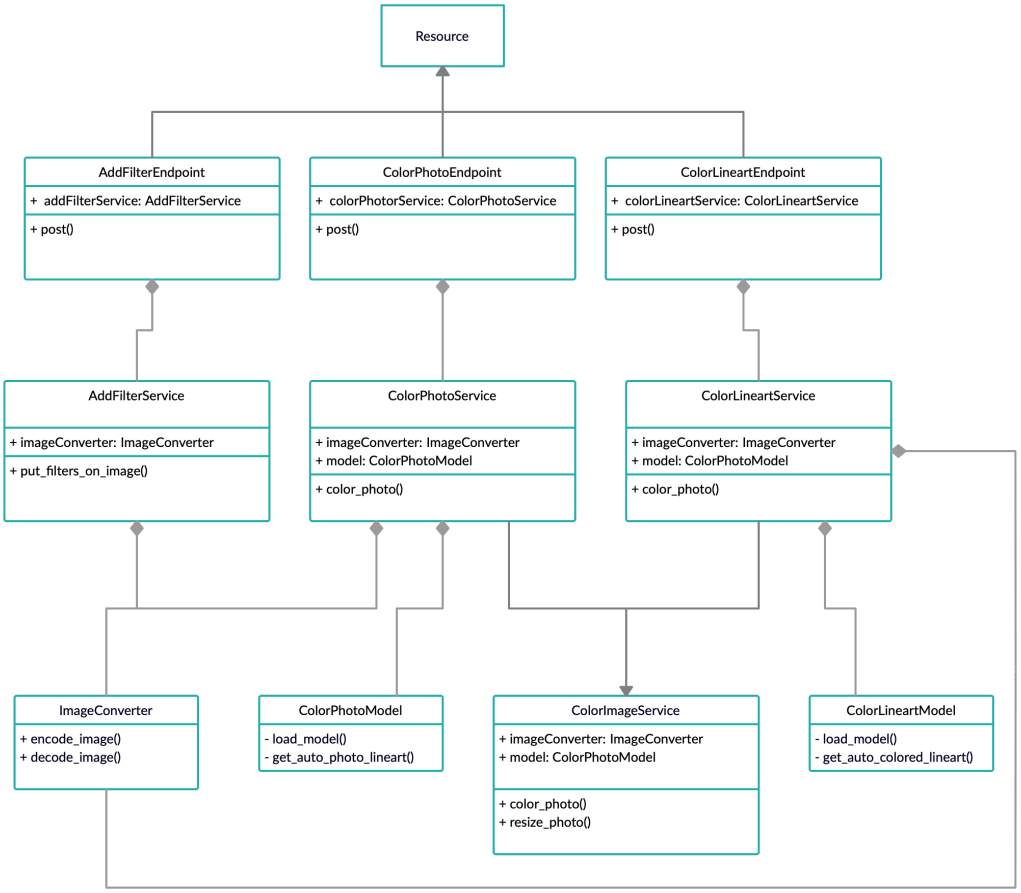




Dyskryminator

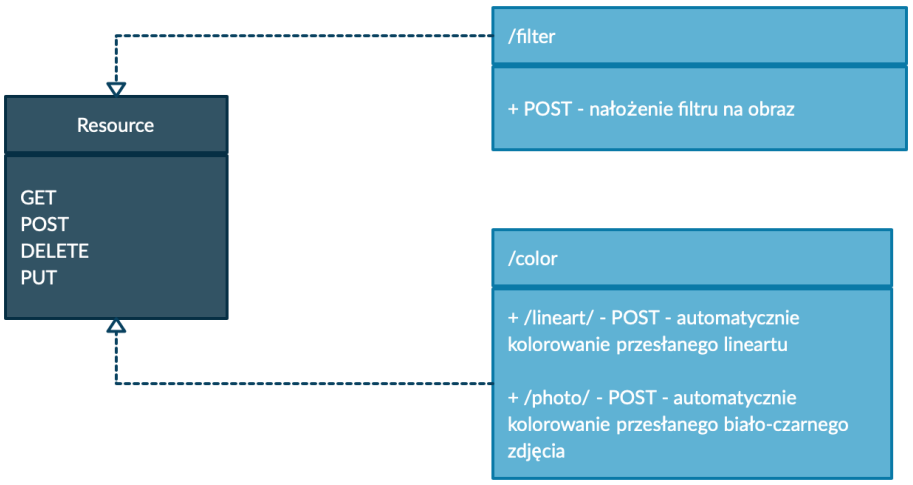


Serwis



Model REST

(Interfejs RESTowy serwisu backendowego)

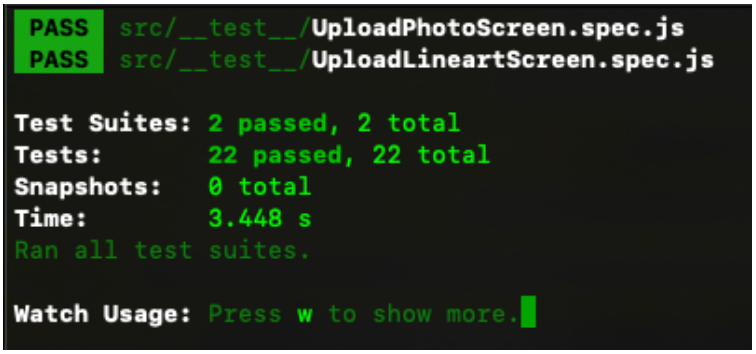


Testy

Frontend

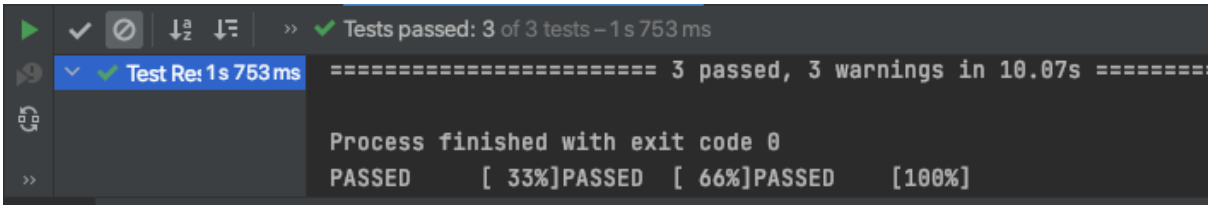
Do części frontendow'ej zostały zastosowane testy jednostkowe oraz manualne. Testy manualne, wykonane przez niezależnych testerów pokazują, że aplikacja jest przejrzysta oraz łatwo się odnaleźć w jej użytkowaniu.

Do testów jednostkowych zastosowana została biblioteka *react test renderer*. Są one podzielone na dwa pliki, osobno do części lineartowej i części przeznaczonej dla zdjęć. Rezultaty można zobaczyć na obrazku poniżej. Aby je wykonać należy w folderze *color_me*, po zainstalowaniu potrzebnych bibliotek, zawołać `npm test`.



Backend

W przypadku część backend'owej, zastosowane zostały testy integracyjne. Znajdują się one w folderze *test* i jest ich trzy, dla każdego z endpointów. Na obrazku poniżej widać rezultat. Do testów wykorzystana została biblioteka *pytest*, aby je uruchomić należy wykonać plik *enpoints_tests.py*.



Plan rozwoju aplikacji

1. Zakup domeny do hostowania serwera aplikacji oraz samej aplikacji, w celu łatwiejszego dostępu oraz korzystania
2. Rozszerzenie modelu odpowiedzialnego za kolorowanie lineartów na więcej kategorii niż postacie z mang.
3. Douczenie modelu koloryzującego czarno-białe fotografie, aby uzyskać bardziej zadowalający efekt.

Rozszerzenie aplikacji skupia się głównie na elementach warstwy logiki aplikacji (modele sieci). Do rozbudowy potrzebne będzie zebranie większej ilości danych oraz lepsze zasoby sprzętowe, potrzebne do uczenia sieci.