

MEDCATEGORIZER

APLIKACJA DO GROMADZENIA, KLASYFIKACJI I TAGOWANIA ZDJĘĆ LARYNGOSKOPOWYCH PACJENTÓW

Spis treści

1. Wstęp	1
2. Architektura aplikacji	1
3. Komponenty, których funkcjonalności nie uległy znaczącej zmianie	4
4. Komponenty nowe lub zmienione	4
4.1. Rejestracja użytkowników	4
4.2. Panel administracyjny	5
4.3. Logowanie	6
4.4. Kartoteki pacjentów	7
4.4.1. Dodawanie nowego pacjenta do kartoteki	7
4.4.2. Usuwanie pacjenta z kartoteki	8
4.4.3. Dodawanie rekordu do kartoteki pacjenta	8
4.4.4. Żądanie sklasyfikowania obiektu	8
5. Klasyfikator	8
5.1. Włączenie do aplikacji, komunikacja podaplikacja - klasyfikator	8
5.2. Szczegóły techniczne klasyfikatora	9
5.2.1. Procedura uczenia klasyfikatora	9
5.3. Osiągnięte rezultaty	9
6. Obsługa administratorska	9
6.1. Zarządzanie profilami pacjentów	9
6.2. Zarządzanie rekordami pacjentów	10
6.3. Zarządzanie klasyfikatorami	10

1. Wstęp

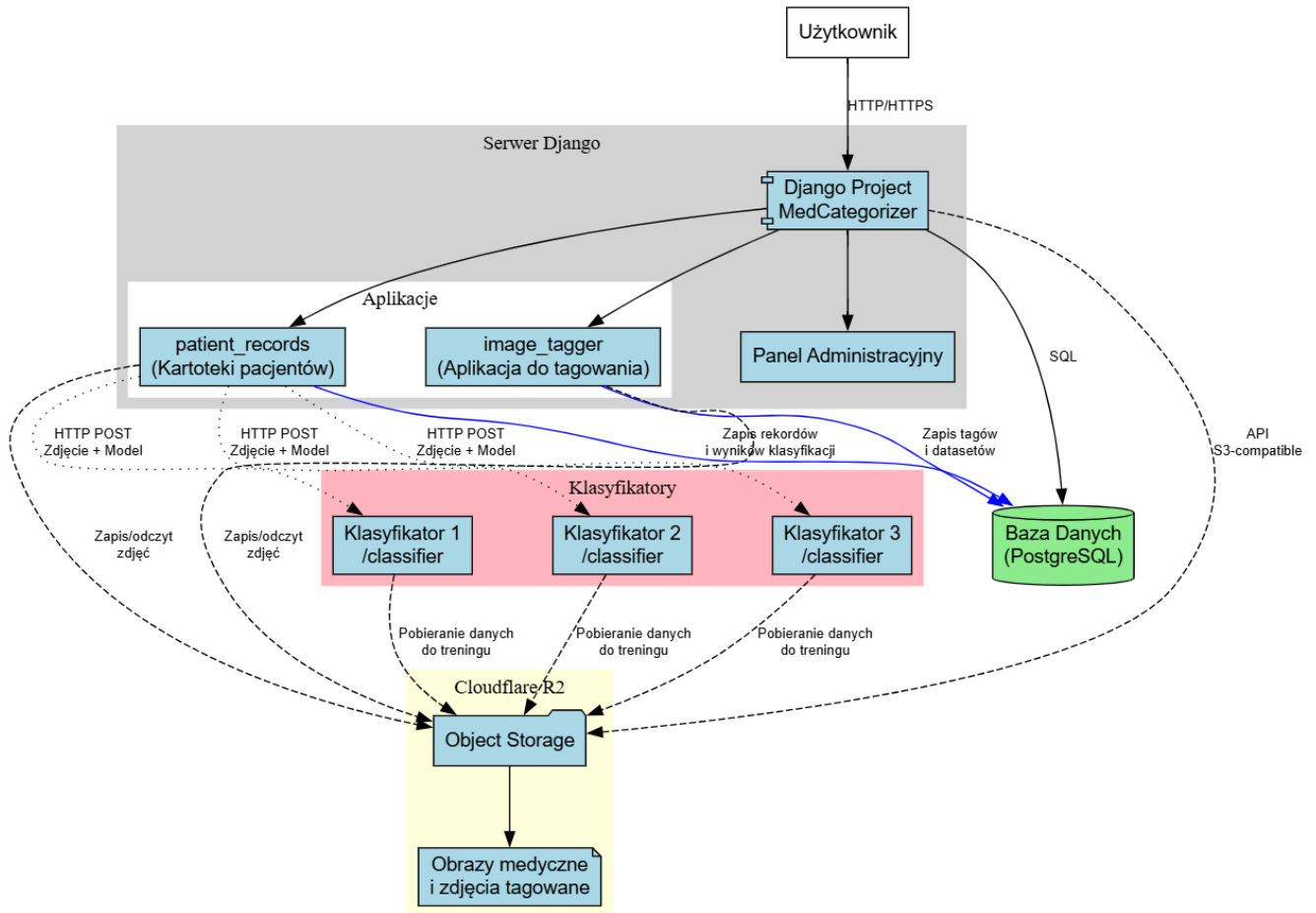
Aplikacja webowa MedCategorizer jest narzędziem bazującym na ramie uprzednio wydanej aplikacji MedTagger i stanowi jej rozszerzenie. MedCategorizer pozwala gromadzić zdjęcia laryngoskopowe pacjentów w ramach konta pacjenta a także je tagować (celem stworzenia bazy obrazów uczących klasyfikator) lub klasyfikować (na podstawie decyzji klasyfikatora). Szczegóły dotyczące komponentów MedTagger znajdują się w dokumencie „Aplikacja webowa do tagowania zdjęć” napisanym wcześniej - te komponenty nie uległy zmianie.

2. Architektura aplikacji

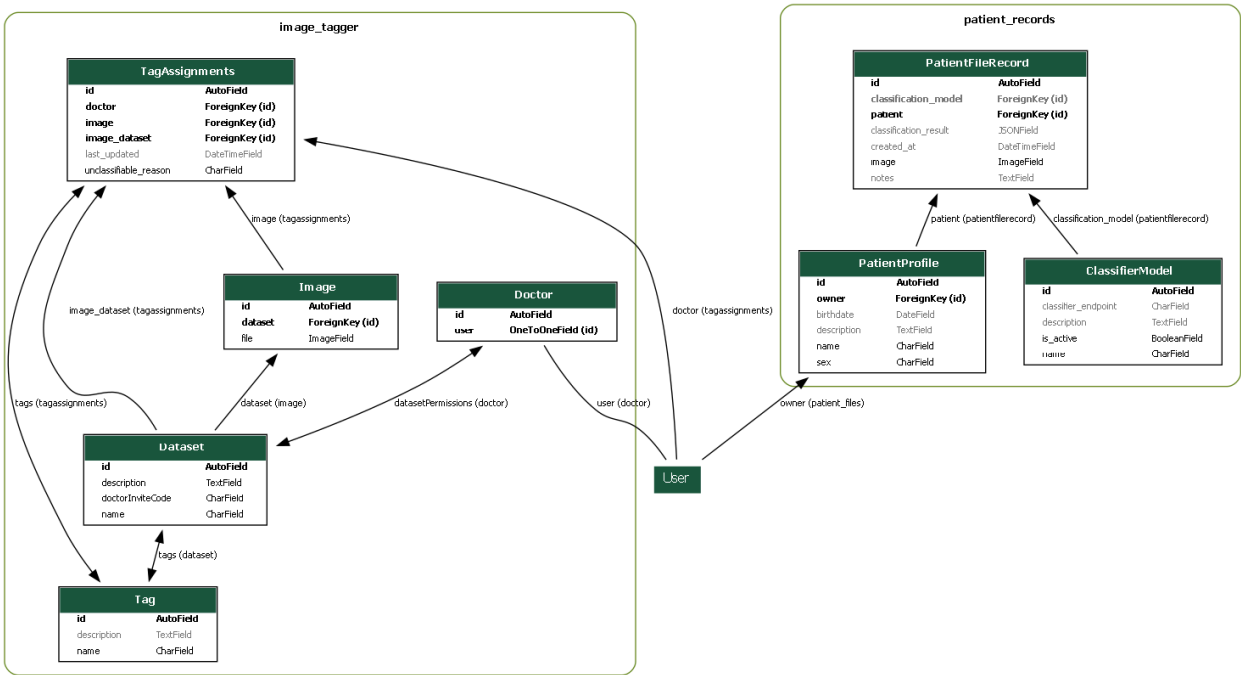
Podobnie jak w przypadku MedTagger, aplikacja webowa MedCategorizer została zbudowana przy użyciu frameworka Django, ze względu na jego prostotę oraz dużą ilość dostępnych bibliotek ułatwiających pracę. Aplikacja składa się z kilku części: panelu administracyjnego całej aplikacji, podaplikacji MedTagger oraz podaplikacji kartoteki pacjentów. Do przechowywania danych użytkowników, danych z kartoteki pacjentów, wyników klasyfikacji i oznaczeń nowych obrazów używana jest baza danych PostgreSQL. Do przechowywania samych zdjęć używana jest chmura Cloudflare R2. Kolejnym elementem aplikacji jest klasyfikator, który zrealizowano tak, aby mógł on być osobnym komponentem. Klasyfikator i kartoteka pacjentów z niego korzystająca są względem siebie niczym czarne skrzynki (black box). Pozwala to na łatwą wymianę, wybór innego typu klasyfikatora lub zwiększenie wydajności tego komponentu poprzez dołączenie nowych instancji. Klasyfikator ma dostęp do chmury Cloudflare R2, z której pobiera obrazy do analizy i sklasyfikowania. Komunikacja między podaplikacją kartoteki pacjentów a klasyfikatorem odbywa się za pomocą REST API. Klasyfikator przyjmuje adres w chmurze R2 obrazu do sklasyfikowania oraz nazwę modelu, który ma być użyty do klasyfikacji.

Klasyfikator zwraca wynik klasyfikacji w postaci łańcucha znaków, który wraz z danymi z kartoteki pacjenta zapisuje się do bazy danych PostgreSQL oraz wyświetla w otwartej karcie pacjenta.

Poniżej prezentujemy schematy: architektury, bazy danych oraz modele danych.



Rysunek 1: Schemat architektury aplikacji.



Rysunek 2: Schemat bazy danych.

Administracja Django

WITAJ, ADMIN. POKAZ STRONĘ / DOKUMENTACJA / ZMIEN HASŁO / WYLOGUJ SIĘ

Powrót • Dokumentacja • Modele • patient_records.PatientProfile

Zacznij pisać, aby filtrować...

IMAGE_TAGGER

Datasets

+ Dodaj

Images

+ Dodaj

Tag assignments

+ Dodaj

Tags

+ Dodaj

PATIENT_RECORDS

Classifier models

+ Dodaj

Patient file records

+ Dodaj

Patient profiles

+ Dodaj

UWIERZYTELNIENIE I AUTORYZACJA

Grupy

+ Dodaj

Użytkownicy

+ Dodaj

patient_records.PatientProfile

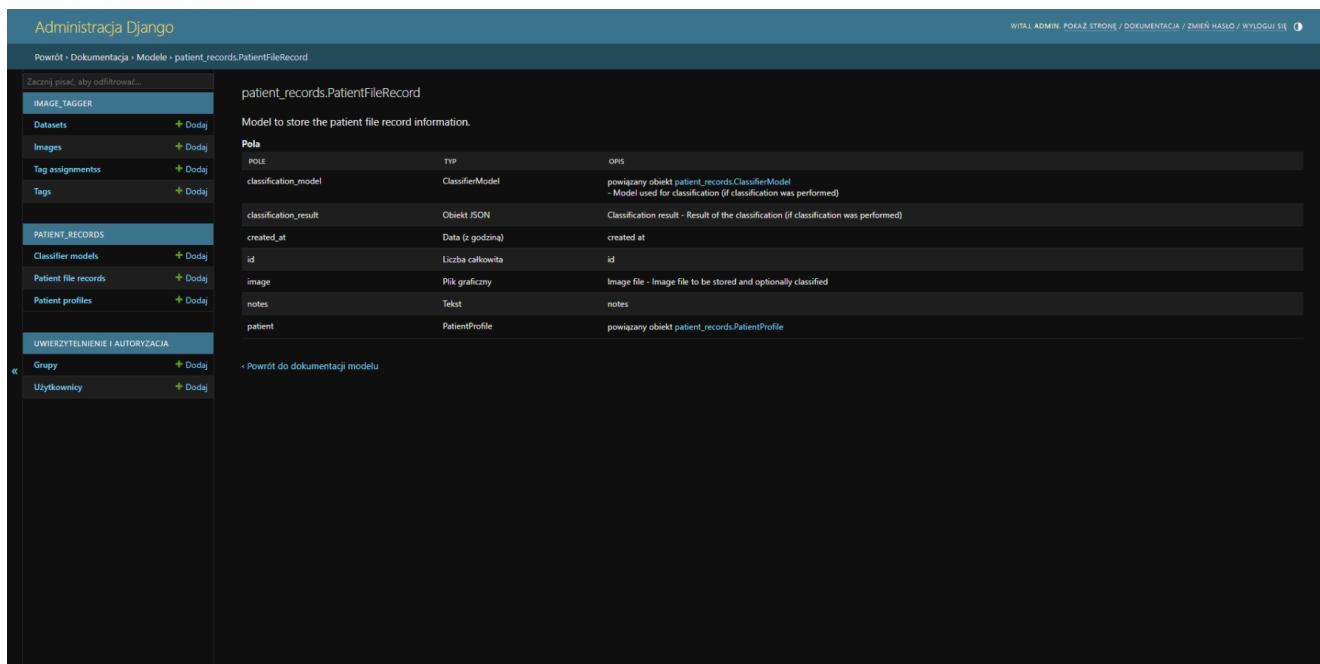
Model to store the patient file information.

Pola

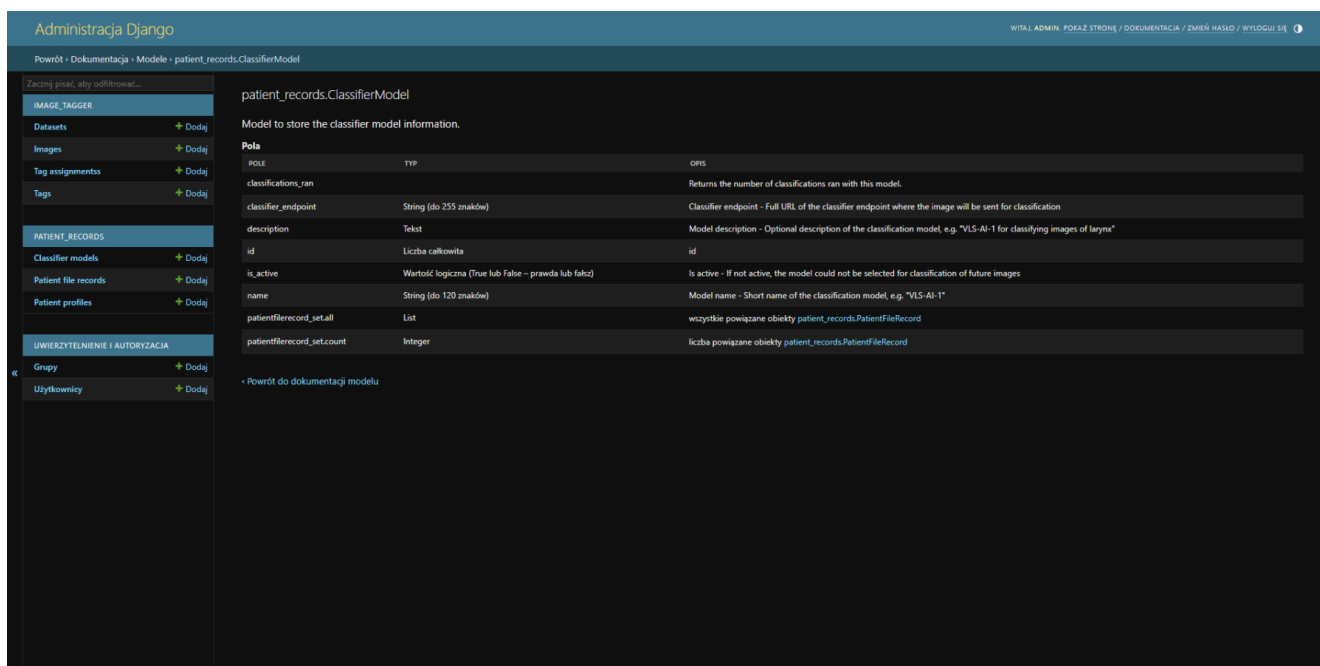
ROLE	TYPE	OPS
birthdate	Data (bez godziny)	Birthdate - Date of birth of the patient
description	Tekst	Optional notes about the patient or the file
id	Liczba całkowita	id
last_record_date		Returns the date of the last record associated with this patient profile.
name	String (do 120 znaków)	Patient name
owner	User	powiązany obiekt auth.User - User who owns the file
patientfilerecord_set.all	List	wszystkie powiązane obiekty patient_records.PatientFileRecord
patientfilerecord_set.count	Integer	liczba powiązane obiekty patient_records.PatientFileRecord
records_count		Returns the number of records associated with this patient profile.
sex	String (do 1 znaków)	Sex - Sex of the patient (M=Male, F=Female, O=Other, ?=Unknown)

- Powrót do dokumentacji modelu

Rysunek 3: Model danych - pacjent.



Rysunek 4: Model danych - rekord.



Rysunek 5: Model danych - klasyfikator.

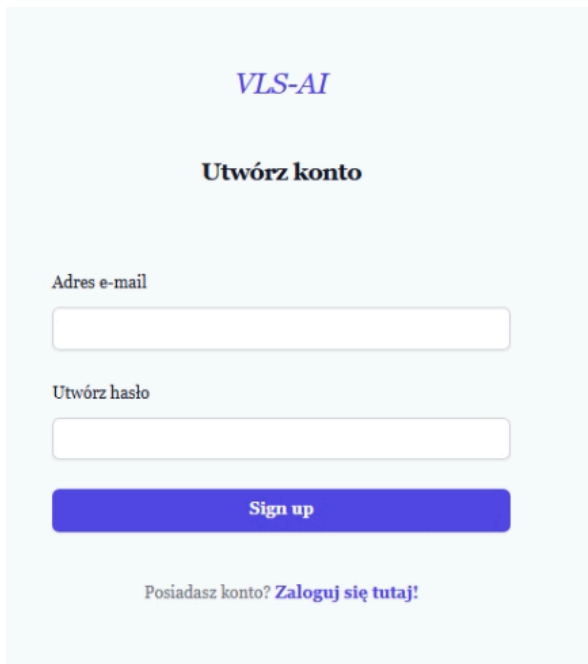
3. Komponenty, których funkcjonalności nie uległy znaczącej zmianie

1. Sposób przesyłania zbioru obrazów do oznaczenia.
2. Moduł oznaczania zdjęć (tu tylko wprowadzono możliwość eksportu tagów zdjęć).
3. Historia oznaczeń.

4. Komponenty nowe lub zmienione

4.1. Rejestracja użytkowników

Zmiana objęła sposób rejestracji użytkowników. Nie trzeba podawać kodu zapraszającego do dołączenia do datasetu zdjęć. Dołączanie do datasetu przeniesiono do podaplikacji ImageTagger.

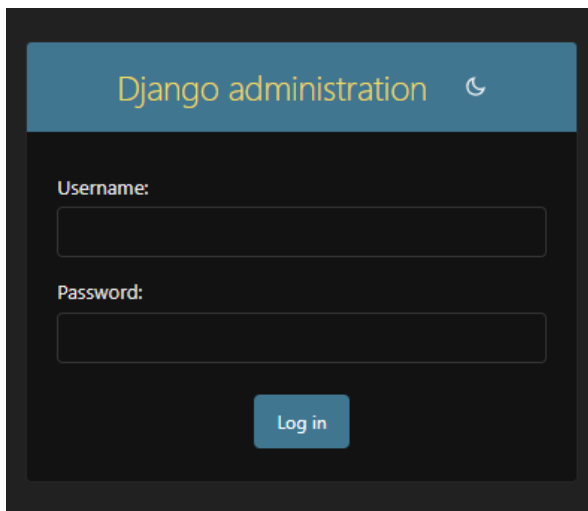


Formularz rejestracji do panelu administracyjnego VLS-AI. Tytuł formularza to "Utwórz konto". Formularz zawiera dwa pola tekstowe: "Adres e-mail" i "Utwórz hasło". Poniżej pól znajduje się przycisk "Sign up" w kolorze fioletowym. Na samym dole formularza znajduje się link "Zaloguj się tutaj!" w kolorze fioletowym, poprzedzony tekstem "Posiadasz konto?".

Rysunek 6: Nowy formularz rejestracji do panelu administracyjnego.

4.2. Panel administracyjny

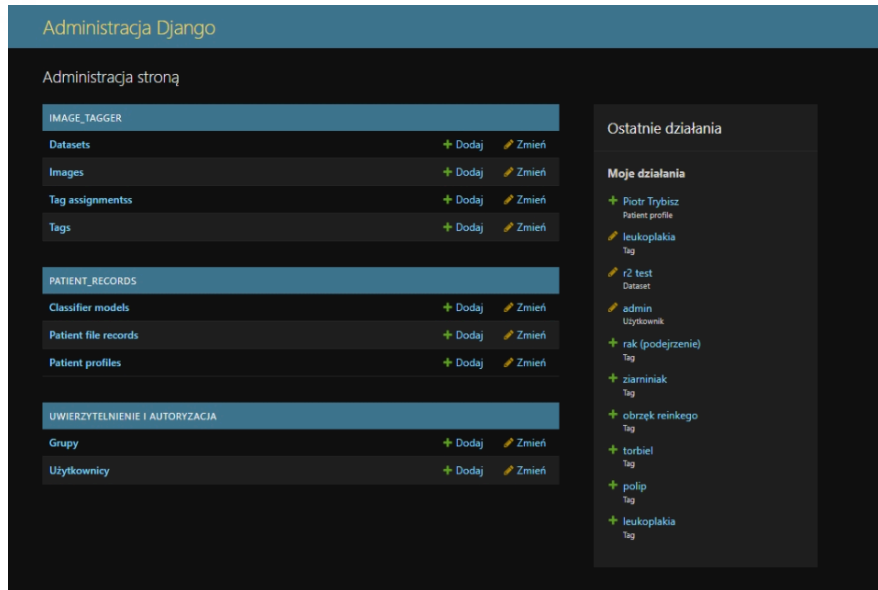
Aplikacja posiada panel administracyjny, który pozwala na zarządzanie użytkownikami, zdjęciami, oznaczeniami oraz kartoteką pacjentów. Jest to wspólny panel administracyjny dla obydwóch podaplikacji. Panel ten jest dostępny pod adresem /admin i wymaga zalogowania się za pomocą konta administratora.



Strona logowania do Django administration. Tytuł strony to "Django administration". Formularz zawiera dwa pola tekstowe: "Username:" i "Password:". Poniżej pól znajduje się przycisk "Log in" w kolorze niebieskim.

Rysunek 7: Logowanie do panelu administracyjnego.

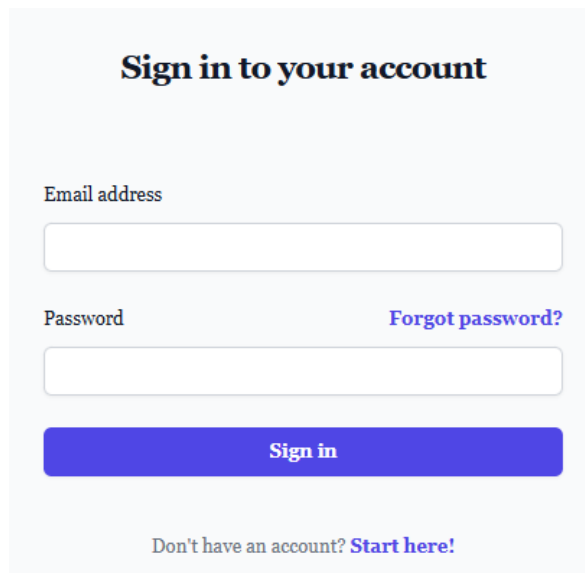
Po zalogowaniu widoczne są 3 główne sekcje: ImageTragger, Patient_records oraz Uwierzytelnienie i autoryzacja. Każda z głównych sekcji składa się z kilku sekcji opisanych poniżej.



Rysunek 8: Sekcje panelu administracyjnego.

1. **Datasets** - sekcja pozwalająca na zarządzanie zbiorami zdjęć.
2. **Images** - sekcja pozwalająca na zarządzanie pojedynczymi zdjęciami.
3. **Tag assignments** - sekcja pozwalająca na zarządzanie i wgląd w oznaczenia zdjęć.
4. **Tags** - sekcja pozwalająca na definiowanie dostępnych tagów.
5. **Classifier models** - sekcja zarządzania modelami klasyfikatora dostępnego w aplikacji.
6. **Patient file records** - sekcja pozwalająca na zarządzanie kartotekami pacjentów (wpisy z badań).
7. **Patient file profiles** - sekcja pozwalająca na zarządzanie profilami pacjentów.
8. **Grupy** - sekcja pozwalająca na zarządzanie grupami użytkowników.
9. **Użytkownicy** - sekcja pozwalająca na zarządzanie użytkownikami, ich hasłami oraz uprawnieniami.

4.3. Logowanie

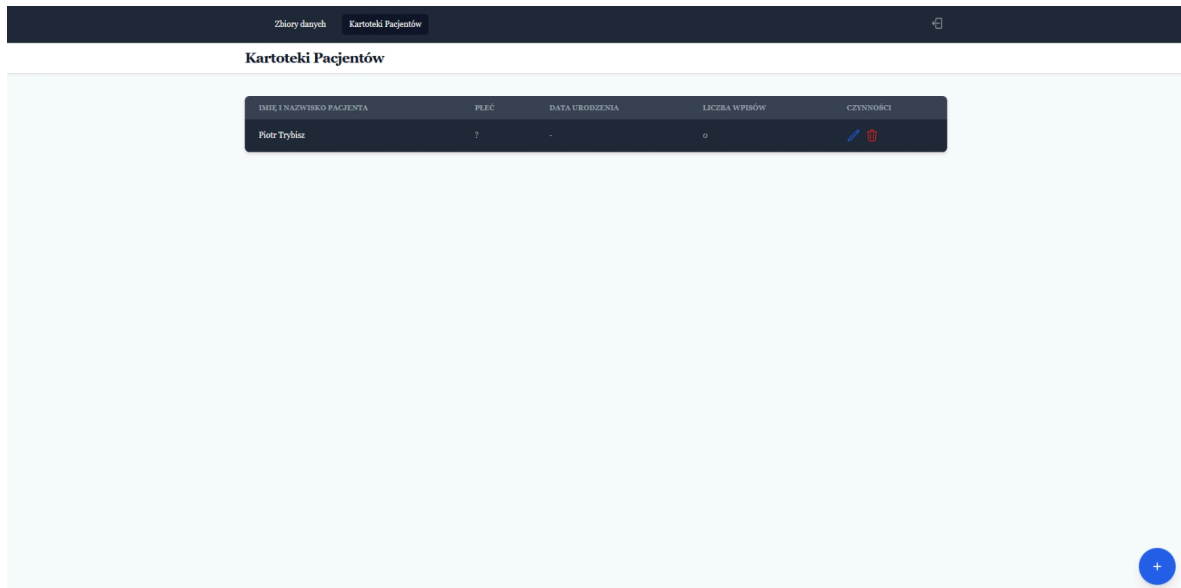


Rysunek 9: Formularz logowania

Moduł logowania się do aplikacji jest jeden, spójny dla wszystkich podaplikacji. Pomyślne zalogowanie przyznaje dostęp do każdej podaplikacji.

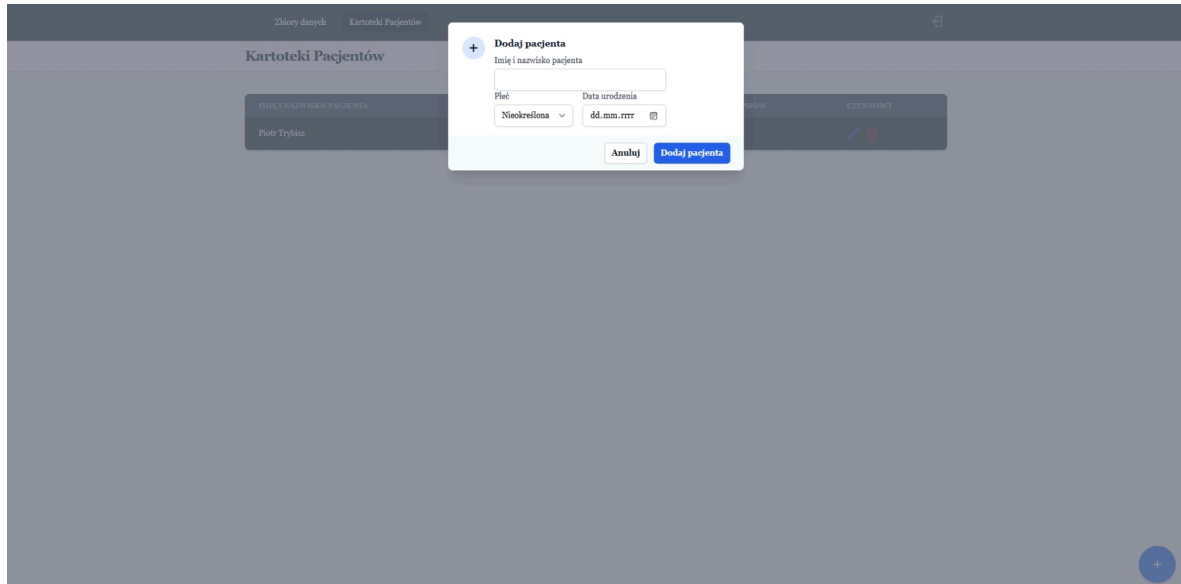
4.4. Kartoteki pacjentów

Komponent kartoteki pacjentów to miejsce, w którym zarządzamy pacjentem oraz wpisami z badań. W tym miejscu lekarz może wstawić zdjęcie krtani pacjenta i wysłać je do analizy przez klasyfikator.



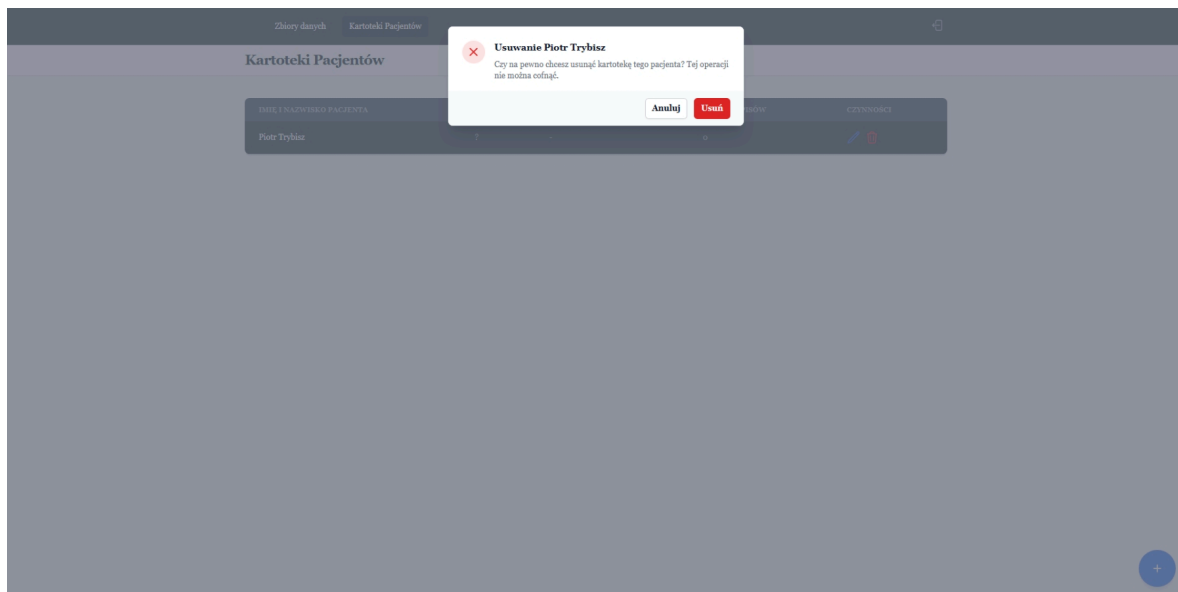
Rysunek 10: Widok kartoteki pacjentów

4.4.1. Dodawanie nowego pacjenta do kartoteki



Rysunek 11: Dodawanie pacjenta

4.4.2. Usuwanie pacjenta z kartoteki



Rysunek 12: Usuwanie pacjenta

4.4.3. Dodawanie rekordu do kartoteki pacjenta

[JB: oczekuję na zrealizowaną funkcjonalność]

☐
☐
☐

4.4.4. Żądanie sklasyfikowania obiektu

Lekarz dodając zdjęcie do kartoteki pacjenta (lekarz tworzy nowy rekord podczas wizyty) może zlecić jego sklasyfikowanie przy użyciu wybranego klasyfikatora. Podaplikacja kartoteki wysyła żądanie HTTP na stosowny endpoint klasyfikatora. Żądanie zawiera nazwę klasyfikatora oraz zdjęcie do sklasyfikowania (adres URL w chmurze R2). W odpowiedzi zwracany jest wynik klasyfikacji w postaci łańcucha znaków, który obrabia się celem estetycznej prezentacji w karcie oraz zapisuje się go do bazy danych.

[JB: tu obrazek]

☐
☐
☐

5. Klasyfikator

5.1. Włączenie do aplikacji, komunikacja podaplikacja - klasyfikator

Podjęto decyzję o zaimplementowaniu ramy klasyfikatora (w postaci usługi z endpointem), dzięki czemu podmiana modelu klasyfikatora jest możliwa a dalsze prace mogą być prowadzone bez przeszkód i przestojów z wykorzystaniem mock'owego klasyfikatora dopóki właściwy klasyfikator nie zostanie dostarczony. W ten sposób umożliwiliśmy pracę równoległą. Ustalono również wymogi związane z komunikacją między elementami systemu.

Komunikacja z klasyfikatorem odbywa się za pomocą REST API. Klasyfikator przyjmuje adres w chmurze R2 obrazu do sklasyfikowania oraz nazwę modelu, który ma być użyty do klasyfikacji. Klasyfikator zwraca wynik klasyfikacji w postaci łańcucha znaków i zapisuje go wraz z danymi z kartoteki pacjenta do bazy danych PostgreSQL. Na końcu, gdy właściwy klasyfikator - binarny w etapie 1. - został dostarczony, podmieniono mock'owy klasyfikator na właściwy.

5.2. Szczegóły techniczne klasyfikatora

Napisano skrypt w języku Python przeprowadzający klasyfikację binarną zdjęć. Wykorzystano biblioteki TensorFlow oraz Keras. Wykorzystano również MobileNetV2 jako bazowy model do klasyfikacji ze względu na fakt szybkiego uczenia.

5.2.1. Procedura uczenia klasyfikatora

1. Uruchomienie środowiska.
2. Wczytanie tagów z pliku CSV oraz zdjęć z chmury R2. Utworzenie krotek (tag, zdjęcie).
3. Odrzucenie krotek z informacją o nieczytelnym zdjęciu.
4. Podział krotek na 2 zbiory: krotki dla zdjęć zdrowych „zbiór zdrowy” i krotki dla zdjęć chorych „zbiór chory”.
5. Przygotowanie zbioru treningowego i walidacyjnego.
6. Obliczenie wag klas celem zbalansowania znacznej różnicy liczności zbiorów zdrowych i chorych.
7. Budowa modelu transfer learningowego MobileNetV2 wraz z podstawową data augmentation (tj. rotacja, flip, zoom, rescale).
8. Trening modelu.
9. Ewaluacja modelu na zbiorze walidacyjnym.
10. Zapis modelu do pliku.

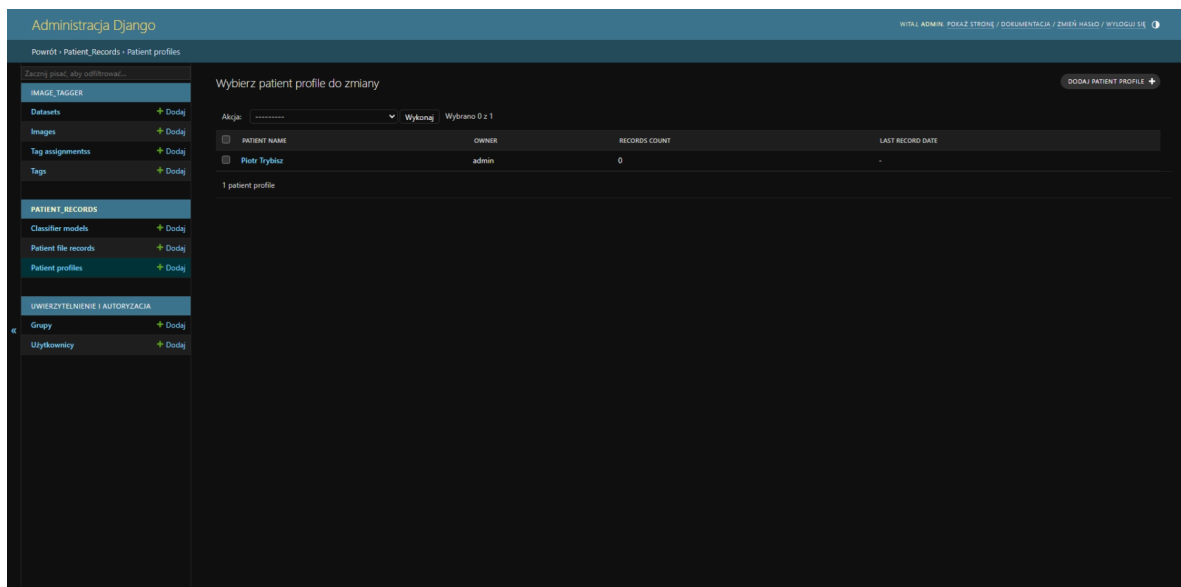
5.3. Osiągnięte rezultaty

Dane, którymi dysponujemy nie są liczne. Na moment pisania raportu zdjęć zdrowych jest ok. 20 a chorych ok. 80. Warto wspomnieć, że są to zdjęcia otagowane wielokrotnie, przez wielu lekarzy (otagowań jest 266). Wyniki nie są zadowalające, klasyfikator nie zwraca sensownych odpowiedzi.

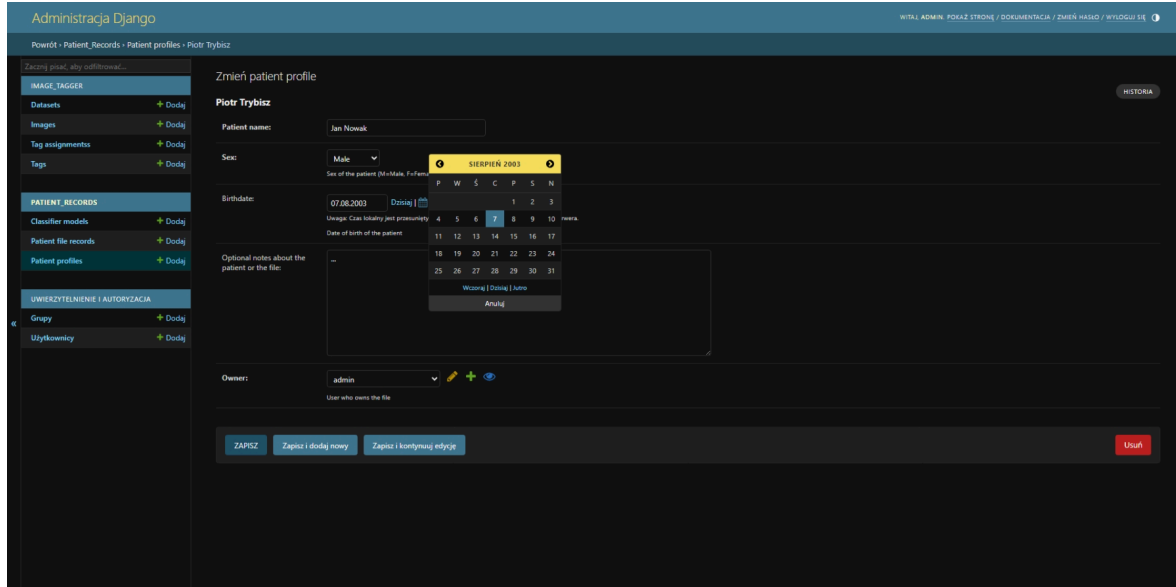
6. Obsługa administratorska

6.1. Zarządzanie profilami pacjentów

Administrator ma możliwość zarządzania profilami pacjentów. W tym celu może dodawać, edytować i usuwać profile pacjentów.



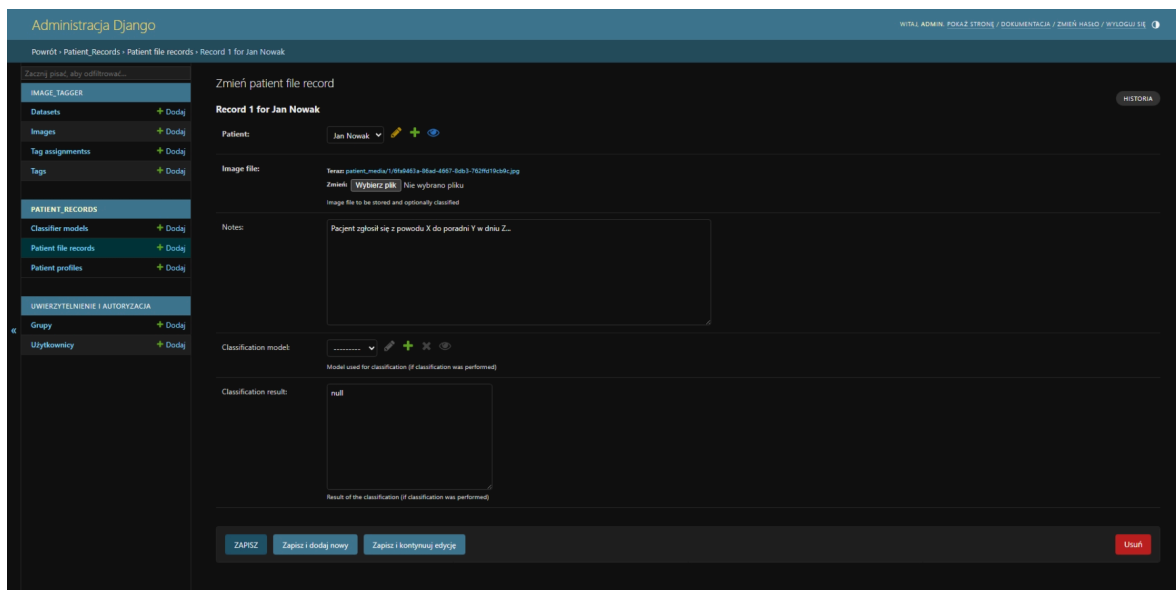
Rysunek 13: Zarządzanie kartoteką przez administratora



Rysunek 14: Edycja profilu pacjenta przez administratora

6.2. Zarządzanie rekordami pacjentów

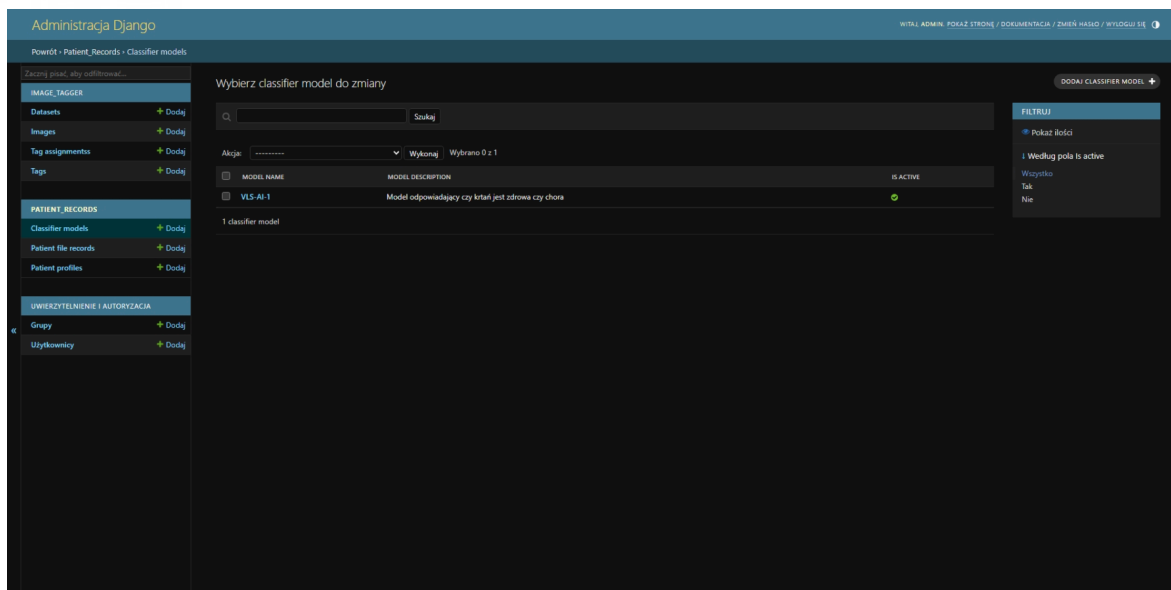
Administrator ma możliwość zarządzania rekordami pacjentów. W tym celu może dodawać, edytować i usuwać rekordy pacjentów.



Rysunek 15: Edycja rekordu przez administratora

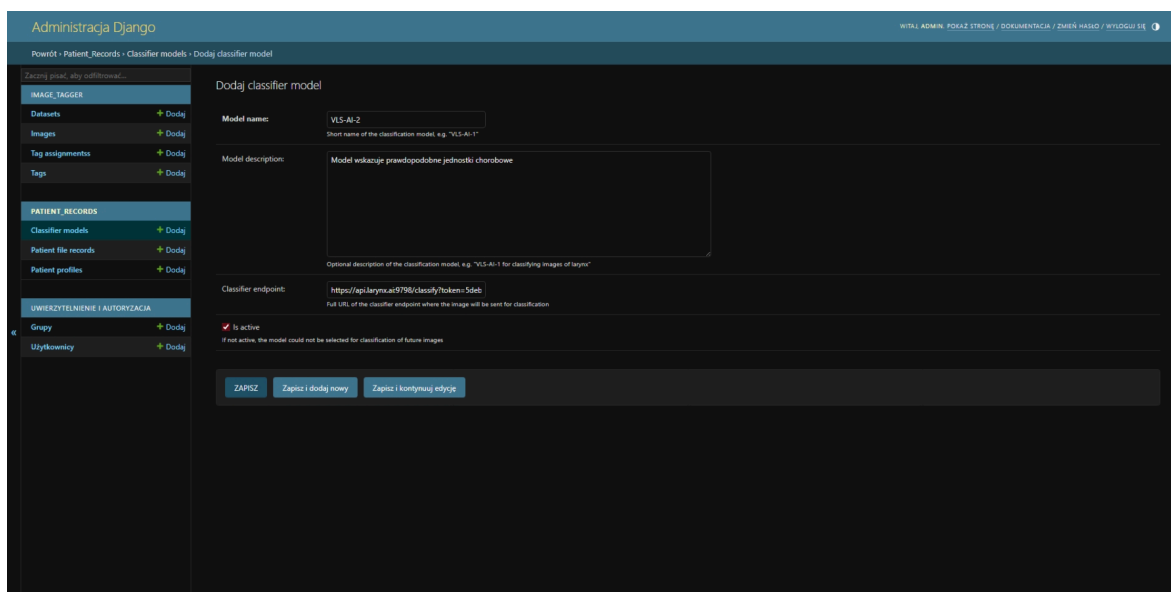
6.3. Zarządzanie klasyfikatorami

Zarząd nad dostępnymi klasyfikatorami pełni tylko administrator aplikacji. Może dodawać, usuwać lub wyłączyć korzystanie z klasyfikatora (usunąć go z listy rozwijanej) w widoku rekordu pacjenta. Administrator dba o to, aby podaplikacja kartoteki pacjentów współhistniała z klasyfikatorem na polu oferowanych funkcjonalności.



Rysunek 16: Lista wszystkich dodanych klasyfikatorów

Dodając klasyfikator, zapisuje się jego nazwę, opis oraz adres URL endpointu klasyfikatora, na który wysyła się obraz do klasyfikacji.



Rysunek 17: Dodawanie nowego klasyfikatora