Informatyka Medyczna

Projekt

Dokumentacja

Wiktor Pawłowski

1. Temat projektu

A3. Generacja raportów konsultowanych danych obrazowych

Celem projektu jest stworzenie aplikacji generującej raporty odnośnie danych obrazowych w formacie DICOM. Na potrzeby realizacji projektu można założyć, że pliki pojawiają się w dedykowanym katalogu, zadaniem aplikacji jest więc okresowe śledzenie jego zawartości i wydobycie *metadanych*, takich jak modalność (CT, MR, RTG, ...), identyfikator badania i serii, identyfikator pacjenta i lekarza, dawka promieniowania, natężenie pola magnetycznego (stosownie do modalności). Pozyskane dane powinny być umieszczane w repozytorium pozwalającym na łatwe przetwarzanie. Dodatkowo powinna być dostarczona aplikacja z interfejsem graficznym pozwalająca tworzyć raporty na podstawie zadanych kryteriów.

2. Ogólny opis stworzonego projektu

Została stworzona dwuczęściowa aplikacja w języku Python. Wykorzystywana baza danych to baza MySQL.

Pierwszą częścią jest „Repository checker”. Okresowo przegląda on katalog, w którym pojawiają się pliki DICOM. W momencie, gdy pojawi się nowy plik, próbuje on wydobyć zdefiniowane wcześniej dane z pliku a następnie, zapisuje informacje w bazie danych. Przeglądane pliki z katalogu są sortowane malejąco ze względu na datę ich modyfikacji, a dodatkowo, zapisywana jest data najnowszego wcześniej przejrzanego pliku.

Drugą częścią jest aplikacja GUI służąca do przeglądania przetworzonych plików, zarządzania nimi, tworzenia raportów, dokonywania zmian w bazie danych oraz dodawania komentarzy do poszczególnych plików.

W ramach projektu powstała także maszyna wirtualna w celu łatwiejszego rozwoju projektu.

Repozytorium z projektem znajduje się na stronie https://github.com/wiktor145/im_projekt.

3. Instrukcja uruchomienia

Aby uruchomić „repository checker”, należy uruchomić plik repository_checker.py z katalogu repository_checker.

Aby uruchomić aplikację GUI, należy uruchomić plik dicom_repository.py z katalogu client_gui.

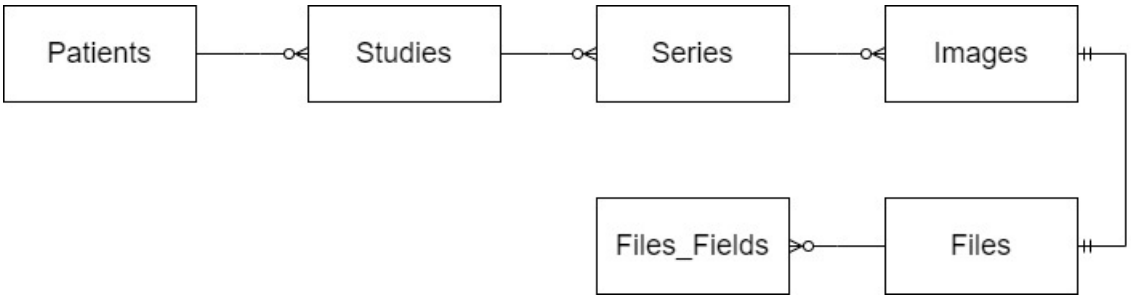
4. Wymagania techniczne

Preferowany system Linux
Baza MySQL
Python 3.8
Pakiety Pythona:
- Babel
- mysql-connector-python
- protobuf
- pydicom
- pytz
- setuptools
- six
- tkalendar

5. Krótki opis implementacji

Wszystkie pliki dotyczące aplikacji GUI (jej działania i wyglądu) znajdują się w katalogu client_gui. GUI zostało wygenerowane z pomocą narzędzia Page.
Domyślnie, katalogiem z plikami DICOM jest katalog data, a wyjściem dla raportów katalog reports.
W katalogu database znajduje się plik z klasą służącą do komunikacji z bazą danych (wstawianie nowych danych wyciąganie aktualnych itp.), klasy pythonowe odpowiadające strukturze bazy danych oraz plik db.sql pozwalającym stworzyć potrzebną bazę danych od zera.
Wszystkie dane konfiguracyjne oraz używane stałe znajdują się w pliku constants.py w katalogu other_classes. Znajdują się tam również zmienne z namiarami na bazę danych.

6. Schemat bazy danych



Patients – pacjenci
Studies – badania
Series – serie
Images – zdjęcia
Files – pliki
Files_fields – tagi DICOM plików

7. Opis aplikacji „repository checker” i jej możliwości

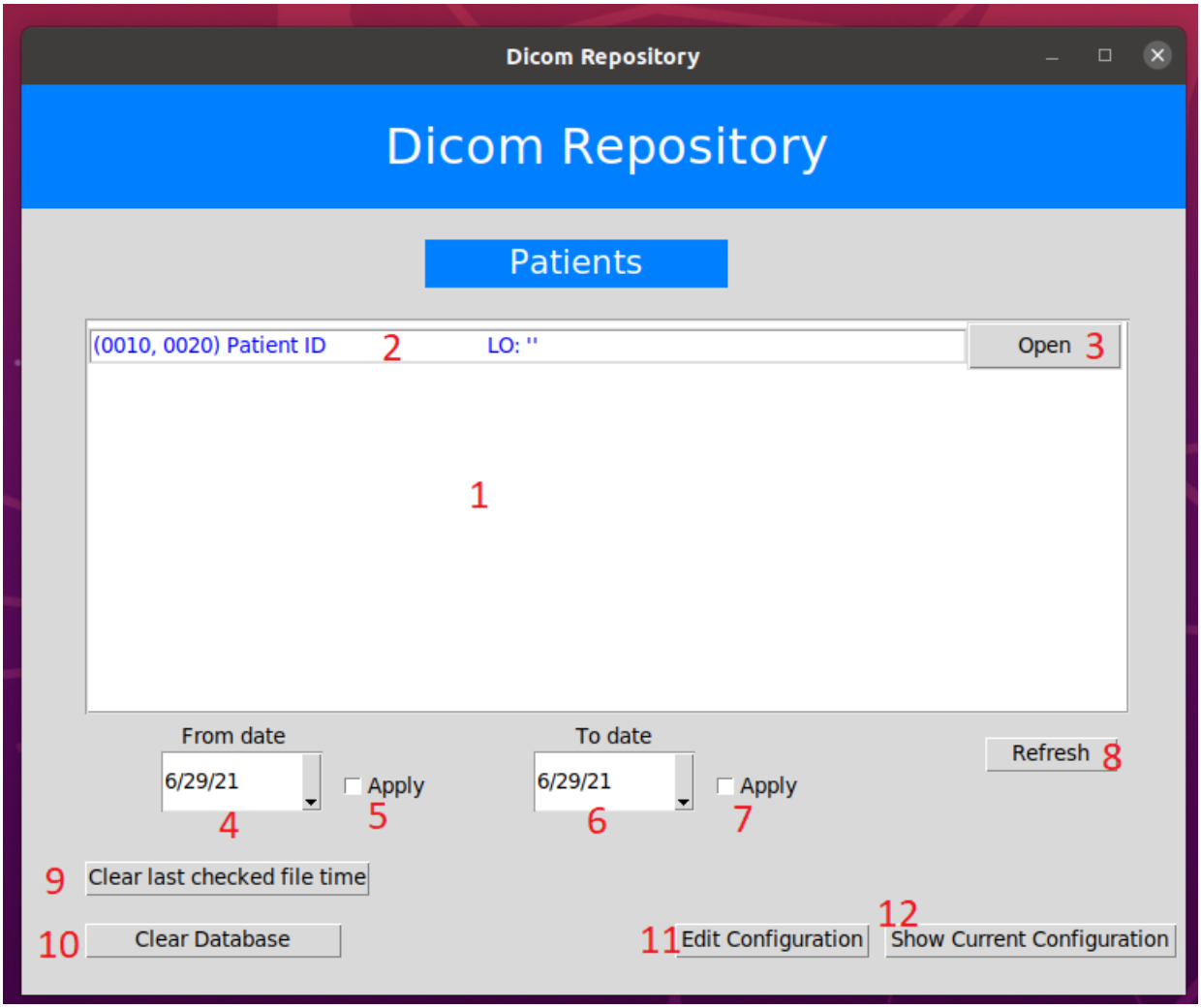
Jego działanie rozpoczyna się od połączenia z bazą danych oraz pobrania aktualnej konfiguracji dotyczącej tego, jakie dane powinny być wyciągane z plików DICOM (przy zmianie konfiguracji należy zrestartować repository checker).
Następnie w pętli, co 60 sekund (wartość konfigurowalna, dla zmiany potrzeby restart) odczytuje z pliku czas modyfikacji ostatnio przetworzonego pliku (jeśli istnieje).
Jeśli czasu nie ma – został usunięty przez odpowiednią flagę w pliku z czasem, wtedy przeglądane są jeszcze raz wszystkie pliki z katalogu.
Następnie, pobiera on listę plików z katalogu, wraz z ich systemowym czasem modyfikacji, sortuje malejąco względem czasu modyfikacji i dla każdego z plików z listy, sprawdza, czy został już wcześniej przetworzony (informacja w bazie danych) i jeśli nie, próbuje wyciągnąć z niego informacje DICOM-owe.
Jeśli się to nie powiedzie zupełnie (np. nieprawidłowy format, plik uszkodzony itp.), zapisuje on na bazie informację o tym, że jest on nieprawidłowy (nie jest on później przetwarzany).
Jeśli się powiedzie, na bazie zapisywane są wszystkie informacje zgodnie z wyciągniętymi informacjami.
Informacje są wyciągane dla pacjenta, serii, badania i samego pliku. Jeśli na bazie już istnieje np. podany pacjent, zostaje on „reużyty”.

W momencie, gdy dojdziemy do pliku, którego data modyfikacji jest mniejsza, niż ostatnio zapisany czas, przerywamy sprawdzanie i zapisujemy nowy czas, będący najnowszym przetworzonym plikiem.

Repository checker można zatrzymać „miękko” poprzez sygnał SIGINT lub SIGTERM.
Wtedy, po jednokrotnym przejściu, jego działanie się zakończy.

8. Opis aplikacji gui i jej możliwości

Ekran główny:

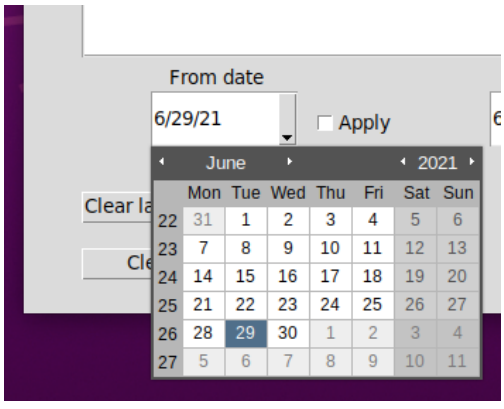


1. Pole, w którym pojawiają się pacjenci znajdujący się w bazie – pacjenci, którzy posiadają co najmniej jeden plik DICOM

2. Nazwa pacjenta – identyfikowana przez tag DICOM

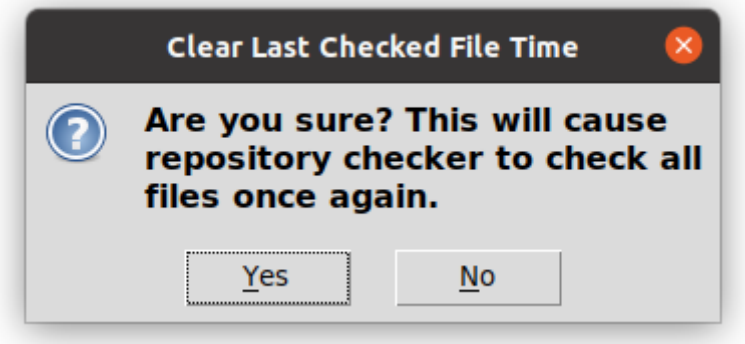
3. Przycisk służący to przejścia na ekran pacjenta – nowe okno

4. i 6. Filtry służące do ograniczenia widoczności pacjentów poprzez ograniczenia do tych, którzy posiadają jakiś plik z datą modyfikacji pomiędzy podanymi datami. Możliwe jest ograniczenie z dwóch stron, z dowolnej jednej, bądź z żadnej. Aby włączyć filtrowanie, należy wybrać odpowiednie daty/datę w 4 i/lub 6, następnie włączyć odpowiedni filtr za pomocą checkboxów 5 i/lub 7 oraz nacisnąć przycisk 8

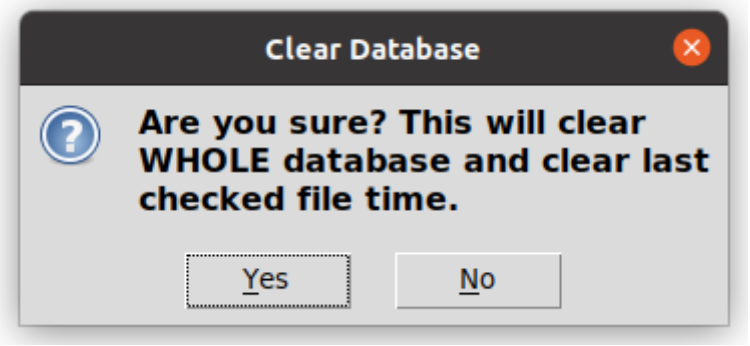


8. Przycisk służący do odświeżenia listy pacjentów – zarówno, gdy w bazie pojawią się nowe wpisy, jak i przy aplikowaniu filtrów

9. Przycisk służący do wyczyszczenia daty ostatnio sprawdzonego pliku. Po jego naciśnięciu (i potwierdzeniu), przy następnym przejściu repository checker przejrzy wszystkie pliki z odpowiedniego katalogu.

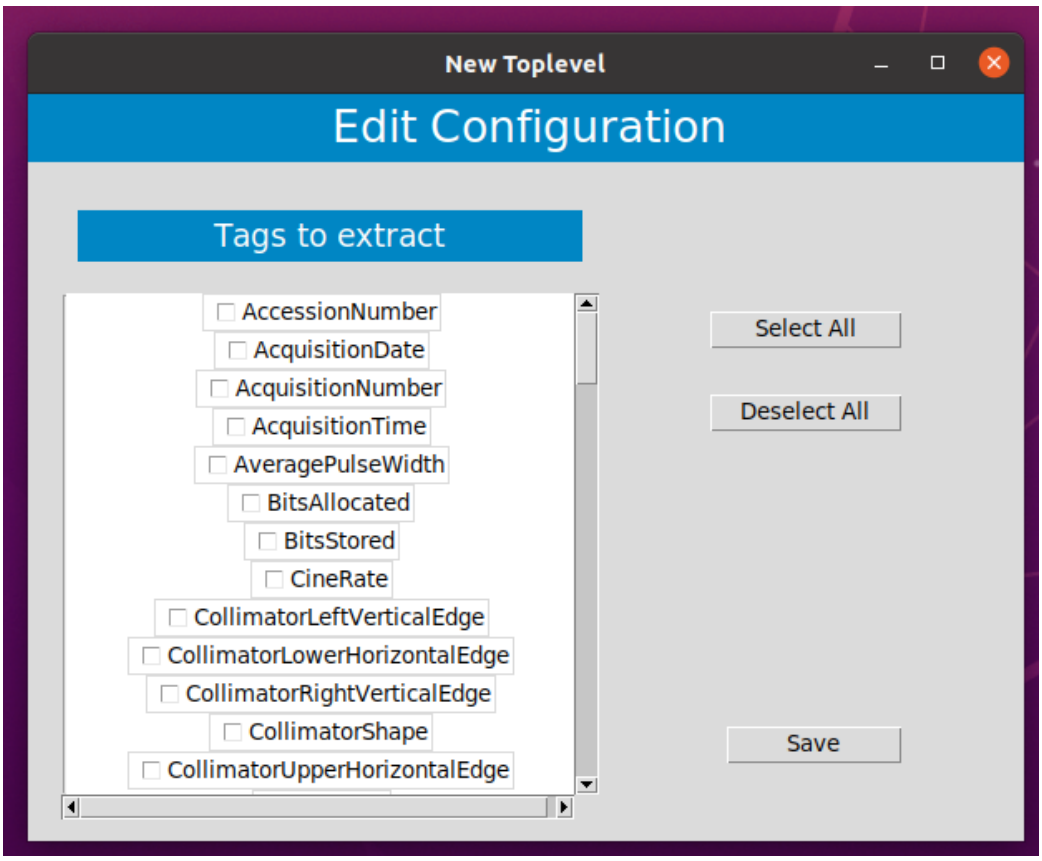


10. Przycisk służący do wyczyszczenia całej bazy danych oraz wyczyszczenia daty ostatnio sprawdzonego pliku. Spowoduje to powtórne przejście po wszystkich plikach z katalogu oraz wstawienie wyników do bazy danych.



11. Przycisk otwierający okno do edycji konfiguracji – po edycji należy zrestartować repository checker.

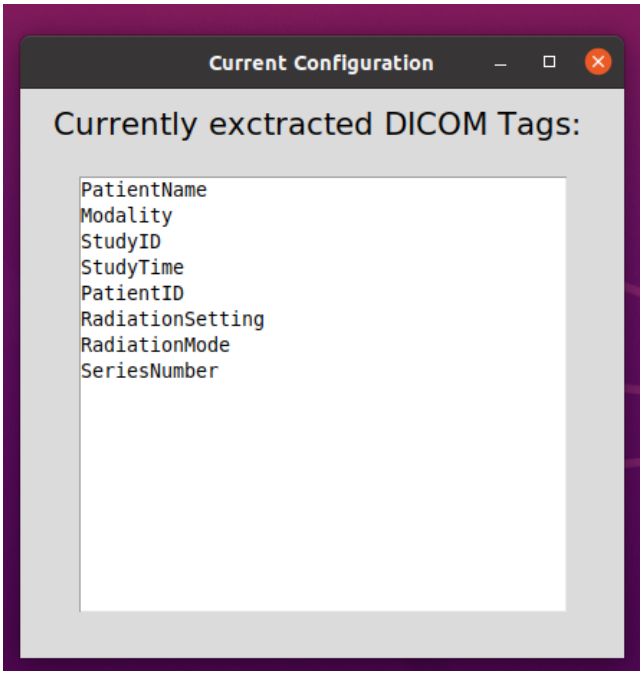
Okno edycji konfiguracji:



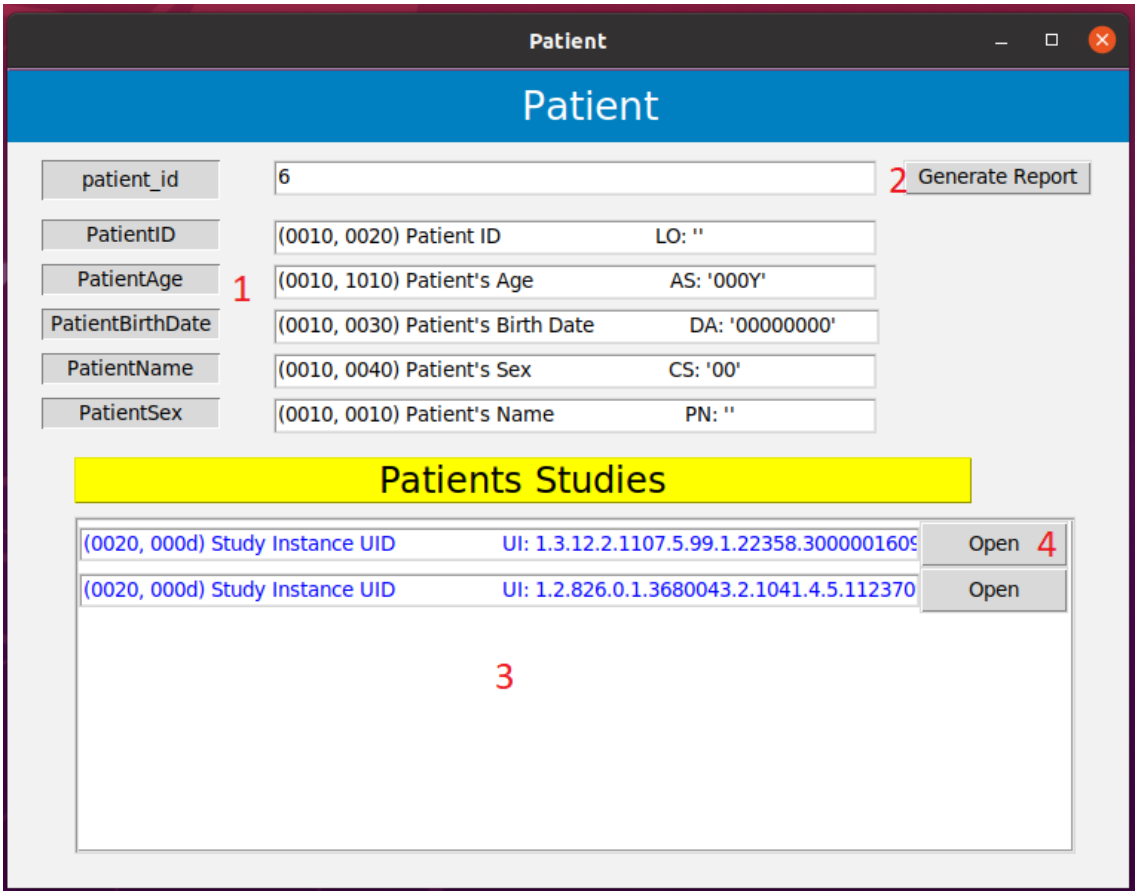
Działanie poszczególnych przycisków jest raczej oczywiste. Widoczne do wyboru tagi są zapisane w pliku configuration.py (katalog other_classes) w zmiennej small_list_of_fields (w tej samej klasie jest również zmienna all_keywords zawierająca wszystkie dostępne tagi DICOM).

12. Przycisk służący do wyświetlenia aktualnej konfiguracji – pól DICOM, które oprócz zdefiniowanych dla pacjenta, serii, badania i obrazu, są dodatkowo wyciągane z pliku i zapisywane na bazie.

Okno podglądu aktualnej konfiguracji:



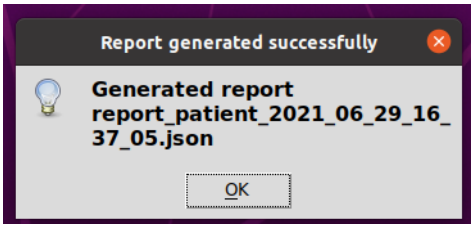
Ekran pacjenta



1. Lista podstawowych informacji o pacjencie

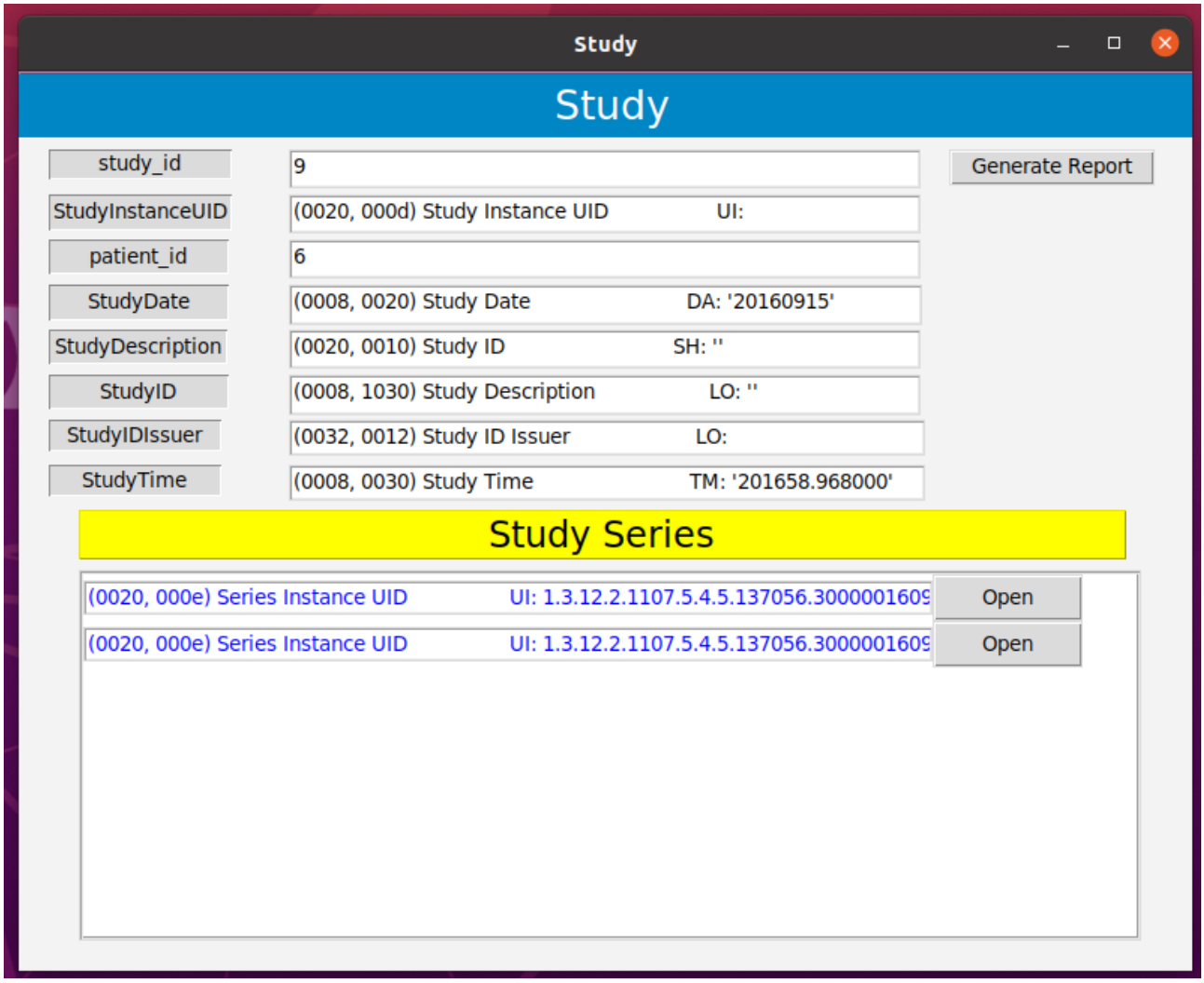
2. Przycisk służący do generowania raportu. Raport to plik json, sygnowany datą stworzenia, zawierający widoczne w 1. informacje o pacjencie, oraz ilość jego badań.


```
{
  "PatientID": "(0010, 0020) Patient ID                    LO: ''",
  "PatientAge": "(0010, 1010) Patient's Age                 AS: '000Y'",
  "PatientBirthDate": "(0010, 0030) Patient's Birth Date          DA: '00000000'",
  "PatientName": "(0010, 0010) Patient's Name                PN: ''",
  "PatientSex": "(0010, 0040) Patient's Sex                 CS: '00'",
  "patient_studies": 2,
  "db_patient_id": "6"
}
```

3. Lista badań pacjenta wraz z przyciskami do ich otwierania (4, nowe okno)

Ekran badania:

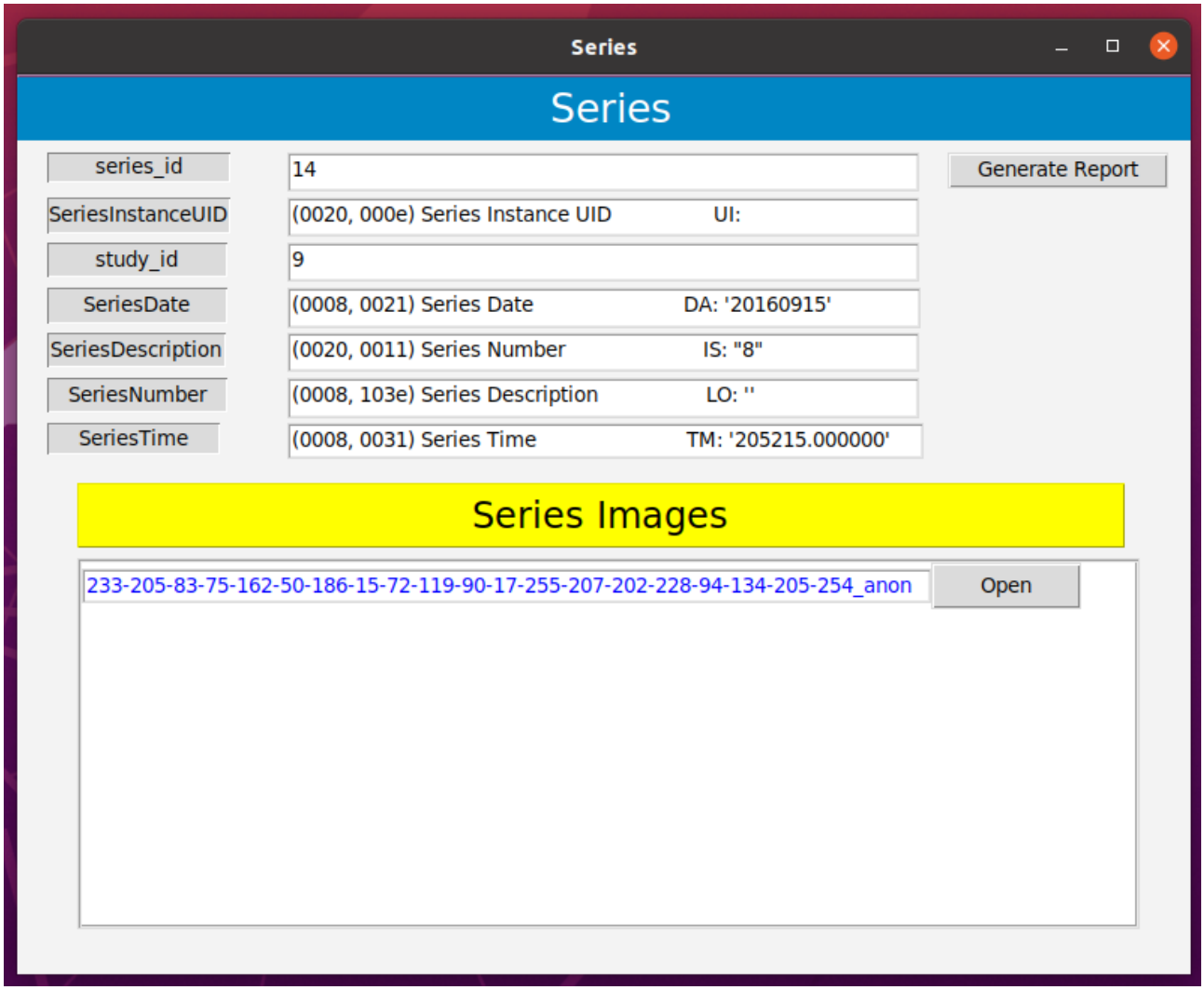


Jest on analogiczny do ekranu pacjenta.

Przykładowy generowany raport (dodatkowo mamy informacje na temat pacjenta):


```
{
  "StudyInstanceUID": "(0020, 000d) Study Instance UID                UI: 1.3.12.2.1107.5.99.1.22358.3000001609151402425000000016",
  "StudyDate": "(0008, 0020) Study Date                          DA: '20160915'",
  "StudyDescription": "(0008, 1030) Study Description                LO: ''",
  "StudyID": "(0020, 0010) Study ID                             SH: ''",
  "StudyIDIssuer": "(0032, 0012) Study ID Issuer                   LO: '16.09.15-20:16:59-DST-IVS'",
  "StudyTime": "(0008, 0030) Study Time                          TM: '201658.968000'",
  "study_series": 2,
  "db_study_id": "9",
  "db_patient_id": "6",
  "PatientID": "(0010, 0020) Patient ID                          LO: ''"
}
```

Ekran serii

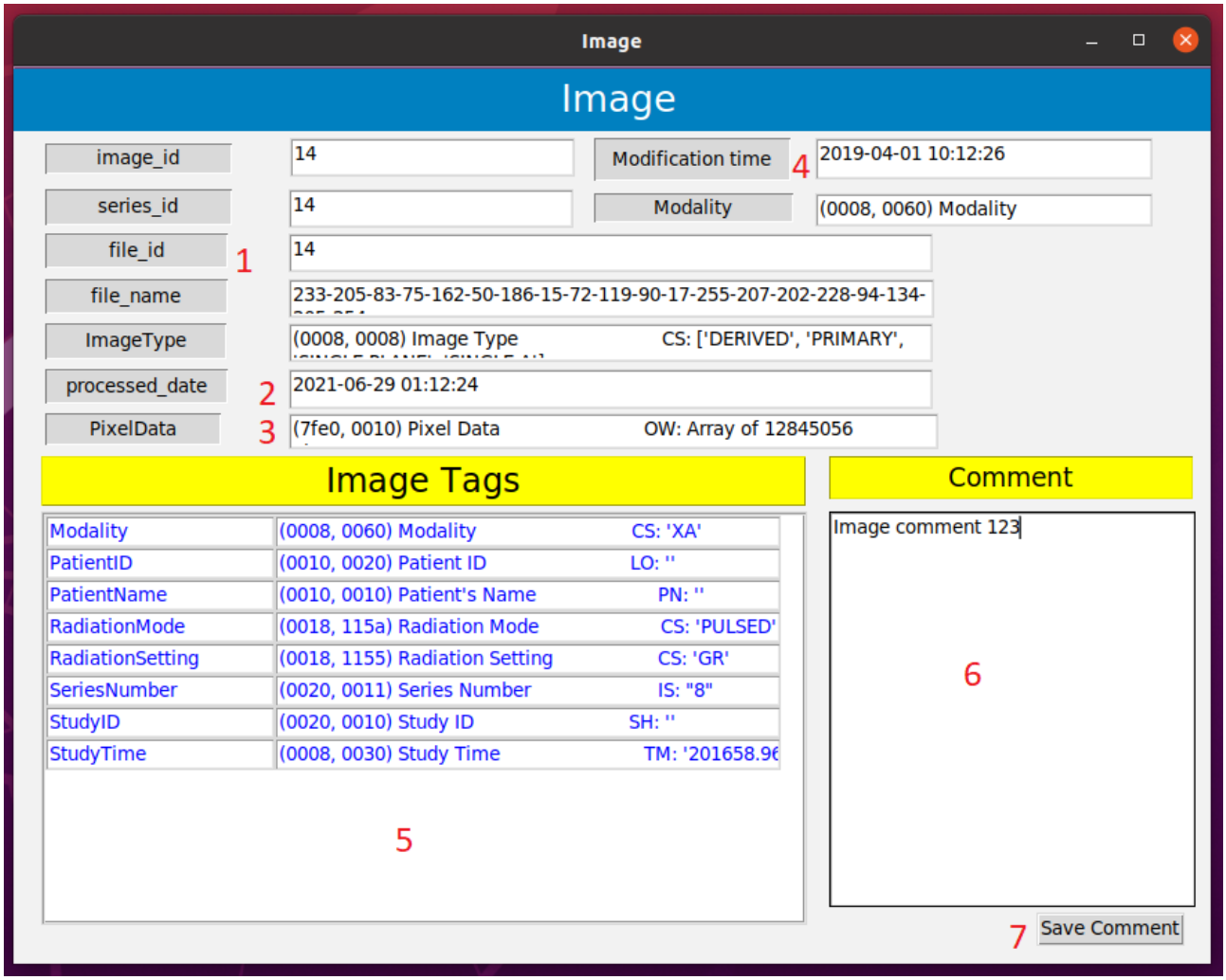


Jest on analogiczny do ekranu pacjenta.

Przy generowaniu raportu, oprócz ilości zdjęć dla serii, dodawana jest także informacja o ich modalności, jak również informacje o badaniu i pacjencie


```
{
  "SeriesInstanceUID": "(0020, 000e) Series Instance UID                UI: 1.3.12.2.1107.5.4.5.137056.30000016091514344882800000499",
  "SeriesDate": "(0008, 0021) Series Date                          DA: '20160915'",
  "SeriesDescription": "(0008, 103e) Series Description                LO: ''",
  "SeriesNumber": "(0020, 0011) Series Number                        IS: \"8\"",
  "SeriesTime": "(0008, 0031) Series Time                          TM: '205215.000000'",
  "series_images": 1,
  "db_series_id": "14",
  "db_study_id": "9",
  "Modality": "(0008, 0060) Modality                           CS: 'XA'",
  "StudyInstanceUID": "(0020, 000d) Study Instance UID                 UI: 1.3.12.2.1107.5.99.1.22358.30000016091514024250000000016",
  "patient_id": 6,
  "PatientID": "(0010, 0020) Patient ID                          LO: ''"
}
```

Ekran zdjęcia/pliku



1. Podstawowe informacje o zdjęciu/pliku.

2. Czas przeprocesowania pliku przez repository checker.

3. Informacje na temat PixelData obrazu (tylko rozmiar, brak dokładnych danych obrazowych).

4. Czas systemowej modyfikacji pliku.

5. Lista dodatkowo wyciągniętych z pliku tagów DICOM zdefiniowanych w konfiguracji.

6. Miejsce do wpisywania komentarza dotyczącego zdjęcia/pliku. Można go zapisać poprzez przycisk 7.