Politechnika Warszawska

WYDZIAŁ MATEMATYKI
I NAUK INFORMACYJNYCH

Praca dyplomowa inżynierska

na kierunku Informatyka

Implementacja aplikacji wspomagającej diagnostykę nowotworu prostaty, wykorzystującej standard DICOM do integracji z realnym systemem szpitalnym

Łukasz Maciej Berwid

Numer albumu 268740

Rafał Buzun

Numer albumu 268744

Jakub Karolak

Numer albumu 268758

promotor

mgr inż. Piotr Sobecki

WARSZAWA 2018

....

podpis promotora

....

podpis autora

Streszczenie

Implementacja aplikacji wspomagającej diagnostykę nowotworu prostaty, wykorzystującej standard DICOM do integracji z realnym systemem szpitalnym

Celem pracy dyplomowej jest stworzenie narzędzia wspomagającego diagnostykę nowotworu prostaty oraz jego integracja ze szpitalną bazą danych. Proponowane narzędzie będzie umożliwiało obliczanie objętości prostaty bazując na wysegmentowanym fragmencie obrazu pochodzącego z obrazowania metodą multiparametrycznego Rezonansu Magnetycznego. Dodatkowo stworzona zostanie baza badań zawierająca informacje o referencyjnych wartościach objętości gruczołu krokowego pochodzących z estymacji bazującej na obrazowaniu Ultrasonografem. Narzędzie zostanie przygotowane do wdrożenia w szpitalnym systemie informatycznym.
Słowa kluczowe: przetwarzanie obrazu, analiza danych, bioinformatyka

Abstract

The aim of the diploma thesis is to create a tool that supports the diagnostics of prostate cancer and its integration with the hospital database. The proposed tool will be able to calculate prostate volume based on the segmented image from multiparameter Magnetic Resonance imaging. In addition, a test database containing information on reference prostate volume values from the estimation based on ultrasound imaging will be created. The tool will be prepared for implementation in the hospital IT system.
Keywords: image processing, data analysis, bioinformatics

Warszawa, dnia .....

Oświadczenie

Oświadczam, że moją część pracy inżynierskiej (zgodnie z podziałem zadań opisanym w pkt. ...) pod tytułem; „Implementacja aplikacji wspomagającej diagnostykę nowotworu prostaty, wykorzystującej standard DICOM do integracji z realnym systemem szpitalnym", której promotorem jest mgr. inż. Piotr Sobecki wykonałem samodzielnie, co poświadczam własnoręcznym podpisem.

.....

Spis treści

1. Wstęp

Rak prostaty to drugi – zaraz po raku płuc – najczęściej diagnozowany nowotwór u mężczyzn. W ciągu ostatnich lat w Polsce wzrasta zachorowalności na raka stercza, a także – choć w mniejszym stopniu – umieralność. KRN prognozuje, że w naszym kraju w 2015 r. zostanie wykrytych 14 tys. nowych przypadków raka prostaty, a 5 tys. chorych z wcześniej wykrytą chorobą umrze. W 2012 r., zaledwie trzy lata temu, z powodu tego nowotworu zachorowało 11 tys. mężczyzn, a 4,1 tys. zmarło. Odwrotną tendencję obserwuje się w zachodniej Europie i Stanach Zjednoczonych, gdzie odnotowuje się spadek zachorowalności i umieralności na ten nowotwór. Wszystko dzięki zastosowaniu najnowszych metod leczenia i wczesnej diagnostyce. [1]

Dlatego też, w niniejszej pracy podjęto się zbadania możliwości automatyzacji wykrywania prostaty na zdjęciach rezonansu magnetycznego., poprzez automatyczną segmentację prostaty i obliczanie objętości. Opisane poniżej algorytmy obliczania wielkości gruczołu mogą być wskazówka dla lekarza podczas stawiania diagnozy.

System powstały jako produkt poniższej pracy dzięki zastosowaniu modularnej architektury może być dostosowany do zaawansowanych systemów szpitalnych oraz rozszerzony o alternatywne algorytmy na przykład oparte o głębokie uczenie maszynowe.

Za implementację oraz wdrożenie algorytmu segmentacji prostaty bazującej na danych konkursowych przy użyciu konwolucyjnych sieci neuronowych oraz moduł prezentacji aplikacji odpowiada Rafał Buzun.

Za stworzenie, opisanie zbioru testowego, przeprowadzenie analizy biznesowej oraz statystycznej wyników badań rezonansu magnetycznego oraz rozpoznanie architektury szpitalnego systemu informatycznego i integrację z systemem PACS odpowiada Jakub Karolak.

Za zaprojektowanie architektury oraz implementacje serwera, w tym obsługę bazy danych i przetwarzanie plików DICOM oraz implementacja wybranych metod obliczania objętości prostaty odpowiada Łukasz Berwid

1.1. Diagnostyka raka prostaty

Objawemi świadczącymi o problemach z gruczołem prostaty jest ból lub pieczenie przy oddawaniu moczu lub nieregularne oddawanie moczu. Takie same objawy ma schorzenie nazywane ”łagodnym rozrostem gruczołu krokowego”. Mężczyźni nie zwykli bagatelizować takich problemów, więc rak prostaty w 75% jest wykrywany w bezobjawowym stadium ograniczonym do narządu, dzięki czemu łatwo jest go wyleczyć.

Pierwszym z badań, które jest wykonywane w przypadku podejrzenia występowania nowotworu jest badanie poziomu PSA we krwi. PSA jest antygenem sterczowym, który daje nam podstawowe informacje na temat problemów z gruczołem pacjenta.

Zazwyczaj następnym badaniem jest badanie palpacyjne per rectum. W przypadku tego badania, lekarz szuka nieprawidłowości w kształcie i twardości stercza. Badanie obarczone jest duża niedokładnością, ponieważ ok. 30% nowotworów znajduje się w rejonie prostaty nieosiągalnym przez badanie palpacyjne.

W przypadku podejrzeń nowotworu wykonuje się dwa kolejne badania. Jest to biopsja stercza oraz badanie USG. Biopsja o wyniku pozytywnym jednoznacznie definiuje posiadanie nowotworu. Badanie USG cechuje się mniejszą dokładnością, jest robione w celu określenia wytycznych do badania MR, czyli rezonansu magnetycznego.

Badanie MR cechuje się większą dokładnością, radiolog za pomocą oprogramowania jest w stanie znaleźć nawet najmniejsze zmiany w strukturze gruczołu. Często po wynikach rezonansu zlecana jest biopsja celowana, która ma sięgnąć do już zbadanych przez rezonans rejonów prostaty. Pełny zestaw badań, tj, badanie PSA, badanie palpacyjne, USG, MR oraz biopsja jest w stanie jednoznacznie postawić diagnozę.

W obecnej chwili, w szpitalu MSWiA w Warszawie, do opisu prawdopodobieństwa istnienia nowotworu w danym miejscu prostaty jest używana skala PI-RADS v2.

PIRADS

Ocena PI-RADS v2 opiera się na wykorzystaniu pięciopunktowej skali opracowanej w oparciu o obserwację, że istnieje zależność pomiędzy trzema badanymi obszarami: T2W, DWI oraz DCE a prawdopodobieństwem, że uwidocznione zmiany mają charakter nowotworowy.Zaszeregowanie do określonej kategorii według PI-RADS v2 powinno opierać się na wynikach badań mpMRI i nie powinno uwzględniać innych czynników, takich jak stężenie PSA, wynik badania per rectum, dane uzyskane z wywiadu lub przebytego leczenia. Przy wynikach rzędu 4-5 w skali PIRADS występuje wysokie prawdopodobieństwo posiadania istotnej klinicznie formy nowotworu. W takich przypadkach, jeśli jeszcze nie została wystawiona diagnoza, należy bezzwłocznie wykonać biopsję.

PSA

PSA czyli swoisty antygen sterczowy. Jest to związek wytwarzany w gruczole krokowym (prostacie). Obecność tego enzymu we krwi jest wykorzystywana w diagnostyce, monitorowaniu i leczeniu raka prostaty od 1986 roku. Fakt posiadania podwyższonego poziomu PSA we krwi nie jest zawsze jednoznaczny z chorobą. Badanie to daje około 25% fałszywie dodatnich wyników. Niestety fakt nieposiadania podwyższonego PSA może być równie mylący. W 20% przypadków to badanie daje fałszywie ujemny wynik. Standardy Europejskiego Towarzystwa Urologicznego zalecają coroczne oznaczanie PSA u mężczyzn po 50 roku życia. Kompletnie zdrowy mężczyzna powinien mieć PSA na poziomie <1,5 ng/ml. Przy tak niskiej wartości zaleca się kolejny pomiar w odstępie 5 lat.

PSAD

W celu poprawienia wykrywalności raka stercza opracowano współczynnik gęstości PSA, tzw. PSAD, który oblicza się dzieląc stężenia PSA w ng/ml przez objętość stercza. Ten współczynnik pomaga w ustaleniu, czy mamy do czynienia z rakiem stercza, czy też łagodnym rozrostem prostaty, podczas którego także wydziela się antygen PSA. Badania potwierdzają, że isnieje korelacja pomiędzy wartością współczyninka PSAD a występowaniem nowotworu.

Rezonans magnetyczny

Rezonans magnetyczny (MRI) to technika obrazowania medycznego stosowana w radiologii do tworzenia obrazów anatomii i procesów fizjologicznych organizmu. Skanery wykorzystują silne pola magnetyczne, gradienty pola magnetycznego i fale radiowe do generowania obrazów narządów w ciele. Rezonans nie obejmuje promieni rentgenowskich ani promieniowania jonizującego, co odróżnia go od tomografii komputerowej lub skanów PET.

Pełna nazwa badania brzmi *Obrazowanie metodą rezonansu magnetycznego*. Warto jest zaznaczyć fakt obrazowania, ponieważ istnieje jeszcze drugie zastosowanie tego aparatu, mianowicie spektroskopia, wykorzystywana w chemii i fizyce molekularnej.

Badanie objętości gruczołu

W szpitalach używane są trzy metody pomiaru objętości stercza. Pierwszym jest subiektywne badanie per rectum w ktrórym lekarz ręcznie sprawdza objętość prostaty. Dwie pozostałe opierają się na tych samych obliczeniach, różnią się sposobem pozyskiwania danych. Pierwszym z nich jest badanie MR, a drugim TRUS, czyli przezodbytnicze badanie USG. Lekarz ogląda gruczoł w różnych płaszczyznach, zaznaczając punkty skrajne. Po zaznaczeniu najbardziej skrajnych punktów wyliczane są trzy wymiary prostaty: szerokość, wysokość i długość. Aby uzyskać objętość gruczłu krokowego należy zastosować wzór

$$V = d * h * l * 0.54$$

gdzie d - szerokość h - wysokość l - długość Stała 0.52 jest próbą przybliżenia kształtu prostaty figurą geometryczną. Gruczoł przypomina elipsoidę, więc zastosowany został wzór na objętość elipsy

$$V = pi/6 * a * b * c$$

gdzie abc, to kolejne wymiary elipsy.

1.2. Architektura szpitala

Architektura szpitalna składa się z wielu komponentów w poniższej pracy zlokalizowaliśmy komponenty, które będą niezbędne do integracji z opisywanym system. Podstawowym elementem w sieci IT szpitala jest system informacji szpitalnej (HIS), który koncentruje się głównie na potrzebach administracyjnych szpitali. W wielu wdrożeniach system HIS jest kompleksowym, zintegrowanym systemem informatycznym zaprojektowanym do zarządzania wszystkimi aspektami funkcjonowania szpitala, takimi jak zarządzają danymi demograficznymi pacjentów, ubezpieczeniem informacje, planowaniem leczenia klinicznego, działaniem laboratorium oraz zarządzaniem historią pacjenta i badaniami.

1.2.1. System archiwizacji obrazu PACS

PACS czyli system archiwizacji i komunikacji obrazu) to technologia obrazowania medycznego stosowana głównie w szpitalach i jednostkach medycznych do bezpiecznego przechowywania i cyfrowej transmisji obrazów elektronicznych i raportów o znaczeniu klinicznym. Korzystanie

z PACS eliminuje potrzebę ręcznego przechowywania, wyszukiwania i wysyłania poufnych informacji, filmów i raportów. Daje to możliwość aby dokumentacja medyczna i obrazy były bezpiecznie przechowywane w zewnętrznych serwerach i dostępne z każdego miejsca na świecie przy użyciu oprogramowania PACS, stacji roboczych i urządzeń mobilnych. Ze względu na panujące prawo dotyczące ochrony danych osobowy i tego, że dane medyczne pacjentów są danymi wrażliwymi takie rozwiązania rzadko są stosowane w praktyce. Dane pacjentów są przechowywane na serwerach szpitalnych - tego wymaga prawo.

PACS ma cztery główne komponenty:

1. Sprzętowe urządzenia obrazujące.
2. Bezpieczna sieć do dystrybucji i wymiany obrazów pacjentów.
3. stacja robocza lub urządzenie mobilne do przeglądania, przetwarzania i interpretowania obrazów.
4. Elektroniczne archiwa do przechowywania i wyszukiwania obrazów oraz powiązanej dokumentacji i raportów.

Nie ma jednoznacznych standardów, którymi dostawcy się kierują wykorzystując standard DICOM, co utrudnia przenoszenie danych z jednego systemu do pracy w innym systemie.

1.2.2. Radiologiczny System Informatyczny RIS

System informacji radiologicznej (RIS) to sieciowy system oprogramowania do zarządzania obrazami medycznymi i powiązanymi danymi. RIS jest szczególnie przydatny do śledzenia zleceń obrazowania radiologicznego i informacji rozliczeniowych i jest często używany w połączeniu z PACS do zarządzania archiwami obrazów i rejestrowaniem.

RIS spełnia podstawowe funkcji:

1. Zarządzanie pacjentem - RIS może śledzić cały przebieg pracy pacjenta w dziale radiologii. Specjaliści mogą dodawać zdjęcia i raporty do EHRs, gdzie mogą być pozyskane i oglądane przez upoważniony personel radiologii.
2. Planowanie - RIS umożliwia personelowi umawianie się na wizyty zarówno w szpitalach, jak i ambulatoriach.
3. Monitorowanie pacjenta - Korzystając z systemu RIS, dostawcy mogą śledzić historię całej radiologii pacjenta od przyjęcia do wypisu i koordynować historię z przeszłymi, obecnymi i przyszłymi spotkaniami.
4. Raportowanie wyników - RIS może generować raporty statystyczne dla pojedynczego pacjenta, grupy pacjentów lub poszczególnych procedur.
5. Śledzenie obrazu - Tradycyjnie, specjaliści używają RIS do śledzenia poszczególnych filmów i związanych z nimi danych. Ponieważ EHR stały się standardem w branży medycznej, a ucyfrowienie obrazów i PACS zostały powszechnie przyjęte, działy radiologii i ich systemy RIS-PACS zostały zintegrowane w kliniczny obieg pracy całego przedsiębiorstwa medycznego.

1.2.3. Format DICOM

DICOM jest standardowym protokołem do zarządzania i przesyłania obrazów medycznych i powiązanych danych i jest stosowany w większości szpitali. DICOM to międzynarodowy standard komunikacji i zarządzania obrazami i danymi medycznymi. Celem jest zapewnienie przechowywania, udostępniania, współdziałania systemów używanych do produkcji, wyświetlania, wysyłania, odpytywania, przetwarzania, pobierania i drukowania obrazów medycznych, a także do zarządzania powiązanymi przepływami pracy.

DICOM jest używany na całym świecie do przechowywania, wymiany i przesyłania obrazów medycznych, umożliwiając integrację medycznych urządzeń obrazujących wielu producentów. Dane pacjenta i powiązane obrazy są wymieniane i przechowywane w znormalizowanym formacie. Standaryzacja ułatwia dzielenie i interpretację informacji między różnymi urządzeniami do przetwarzania obrazu.

1.3. Metody wspomagania pracy specjalisty

W dzisiejszym świecie praca radiologa jest ściśle powiązana z pracą przy użyciu dostarczanego oprogramowania zintegrowanego z infrastrukturą szpitalną. Radiolog odczytuje, opisuje i analizuje wyniki badania na komputerze. Jednakże wiele badań dostarczanych jest przez pacjenta w formie papierowej. O problemie znajdowania odpowiednich danych w szpitalnej dokumentacji opowiemy w dalszej części pracy.

//todo kuba Wejdźcie głębiej w temat: 1. Jakie narzędzia są teraz wykorzystywane? 2. Kto je dostarcza? 3. Jakie są metody wspomagania obecnie dostarczane? 4. Jakie są możliwości wspomagania? 5. Kto je dostarcza? To co napisaliście jest zbyt powierzchowne. NIe zawsze! Niektóre DICOM viewery mają zintegrowane narzędzia do wspomagania (np. automatyczne wykrywanie i oznaczanie zmian w płucach - one są drobne i jest ich dużo). Musicie do tego dojść Narzędzia wykorzystywane przez radiologów są dalekie od ideału. Są to, w większości, przeglądarki zdjęć, które radiolog musi analizować. W tych programach brakuje automatyzacji i sugestii dla lekarza. Za bardzo pomocny został uznany pomysł dokładnego liczenia objętości prostaty na podstawie zdjęć. Skraca to bardzo czas pracy lekarza oraz daje dokładniejsze wyniki. Obecnie nie istnieje na rynku narzędzie automatycznie segmentujące zdjęcia i dające lekarzowi sugestie na temat przeglądanego obrazu.

Celem poniższej pracy jest stworzenie i opis narzędzia, które zapewni automatyzację pracy lekarza oraz zapewni łatwą integrację z systemem szpitalnym

1.4. Słownik pojęć

Pojęcia których będziemy używać w poniższej pracy:

1. DICOM - (Digital Imaging and Communications in Medicine) Międzynarodowy standard służący do przesyłania, przechowywania, pobierania, przetwarzania oraz wyświetlania obrazów medycznych. Darmowy standard umożliwiający unifikację zdjęć medycznych pochodzących z maszyn wielu producentów.

2. Segmentacja – Proces podziału obrazu na części określane jako obszary (regiony), które są jednorodne pod względem pewnych wybranych własności.
3. Slice - Warstwa obrazu DICOM zawierająca jeden obraz
4. Maska - Obraz przedstawiający wysegmentowaną prostatę
5. Komponent - Usługa siwciowa lub element jednego z serwisów spełniający jedno określone zadanie
6. Endpoint - Adres pozwalający na dostęp do API aplikacji z zewnątrz
7. API - Interfejs pozwalający na komunikowanie się aplikacji między sobą
8. Backend - Część serwerowa aplikacji
9. Frontend - Warstwa prezentacji aplikacji, w przypadku naszej pracy to strona internetowa
10. HTTP - Internetowy protokół komunikacyjny
11. Web serwis - Usługa sieciowa, do której można się odwołać za pomocą wywołań funkcji określonych przez jej API za pomocą protokołu komunikacyjnego
12. JSON - Format wymiany danych komputerowych
13. Base64 - Rodzaj kodowania
14. RIS – (Radiological Information System) System zarządzania pacjentami
15. PACS - (Picture Archiving And Communication System) System archiwizacji zdjęć medycznych
16. Użytkownik – osoba mająca dostęp do aplikacji oraz bazy danych
17. Baza danych – baza z badaniami pacjentów, zawiera obraz w DICOM oraz dodatkowe informacje z badania
18. Anonimizacja – proces usuwający dane wrażliwe pacjenta

2. Dane szpitalne

//todo kuba Zaczęliśmy rozmów z doktor Sklindą jeszcze na naszej uczelni. Odbyliśmy 2 spotkania, na których doktor tłumaczyła nam problemy z jakimi się zmaga oraz pokazywała nam potencjalne problemy, które można rozwiązać łącząc je z pisaniem pracy inżynierskiej przez trzyosobowy zespół. Zgodziliśmy się na pewien zestaw wymagań, który w czasie się zmieniał. Po finalnym ustaleniu wymagań, jeden z nas udał się do szpitala MSWiA w Warszawie w celu poznania szpitalnej architektury oraz zgromadzeniu danych.

Niestety żadnego ze szpitalnych informatyków tego dnia nie zastaliśmy w pracy, więc pierwsza wizyta zakończyła się jedynie na gromadzeniu danych. Wkrótce potem zaczęliśmy wymieniać komunikację mailową z zespołem informatyków ze szpitala. Nasz początkowy pomysł, czyli integracja z bazą HIS upadł bardzo szybko, ponieważ okazało się, że oprogramowanie używane w szpitalu nie może być w żaden sposób modyfikowane przez ludzi nie będących pracownikami firmy dostarczającej to oprogramowanie.

Następnie, gdy znaleźliśmy przeglądarkę zdjęć DICOM, napisaliśmy maila do zespołu szpitalnych informatyków z pytaniem o możliwość instalacji tej przeglądarki na komputerach szpitalnych. Dostaliśmy przeczącą odpowiedź, argumentowaną niepewnością dotyczącą bezpieczeństwa oprogramowania, które chcieliśmy zainstalować. Po wspólnej naradzie uznaliśmy, że najlepszym rozwiązaniem będzie kopiowanie danych na dysk twardy i przenoszenie ich na komputer, który nie jest podłączony do sieci szpitalnej. Doktor Sklinda zgodziła się pracować w ten sposób, uznając, że i tak jest to dla niej pomoc, pomimo niewygody związanej z fizycznym transferem danych.

2.1. Proces pozyskania danych

Kompleksowe dane pacjentów w szpitalu MSWiA są przechowywane w formie papierowej. Każdy pacjent ma oddzielną koszulkę w segregatorze, w której znajdują się wyniki wszystkich badań wykonanych poza szpitalem MSWiA, wywiady z wizyt u lekarzy specjalistów oraz wyniki badań krwi zrobionych w szpitalu. W szpitalnej bazie danych przechowywane są wywiady środowiskowe, wyniki oraz opisy badań MR.

Nie dostaliśmy pozwolenia na instalację zewnętrznego oprogramowania na komputerach podłączonych do sieci szpitalnej. Według nas jest to zrozumiały ruch ze strony szpitalnych informatyków. Oprogramowanie otwartego przez nas wykorzystywane mogłoby być łatwym punktem ataku hakerskiego. Gdy wszystkie dotychczasowe plany poniosły porażkę, zdecydowaliśmy na jedyne rozwiązanie, które zadowoliłoby obie strony. Używamy dysku twardego, ponieważ kompleksowy plik zawierający badanie jednego pacjenta zajmuje średnio ponad Gigabajt danych.

Na komputerze niepodłączonym do sieci szpitalnej doktor może dokonać analizy i predykcji za pomocą naszego oprogramowania. Następnie musi sprawdzić identyfikator pacjenta w arkuszu kalkulacyjnym i odnieść się z wynikami wygenerowanymi przez nas do prawdziwego przypadku.

Uzyskane rozwiązanie jest dalekie od optymalnego. Nie jest ono ani wygodne ani szybkie. Jest natomiast bezpieczne, ponieważ eliminuje większość potencjalnych punktów ataków hakerskich. Przy pracy z bardzo delikatnymi danymi kryterium bezpieczeństwa musi być priorytetem.

Procedura pozyskiwania danych polegała na otworzeniu karty pacjenta leczonego przez współpracującego z nami lekarza (dr n. med. Katarzyna Sklinda), znalezieniu jego wywiadu środowiskowego oraz opisu badania w bazie danych RIS. Następnie należało ręcznie wypełnić wszystkie pola w arkuszu kalkulacyjnym, które dotyczyły badanego pacjenta. Dane z wywiadu środowiskowego zostały zakodowane, w celu przyspieszenia wypełniania arkusza oraz zachowania ogólnej estetyki. Legendę opisu tych danych przedstawimy w dalszej części pracy.

Karty pacjentów były diametralnie różne. Wahały się od jednej strony do całego pliku kartek. Naszym zadaniem było wypełnienie jak największej ilości kolumn na podstawie otrzymanych danych.

Największym kłopot podczas wykonywania tej pracy sprawił nam fakt, iż każdy szpital używa innego formatowania swoich dokumentów, więc nierzadko mieliśmy trudności z odczytaniem istotnych dla nas informacji.

Po uzupełnieniu arkusza kalkulacyjnego należało jeszcze wykonać niezbędne obliczenia objętości oraz PSAD.

Jedną z dróg, którą zamierzaliśmy obrać na początku tego wyzwania było użycie systemu OCR na zeskanowanych kartach pacjentów. System OCR (z ang. Optical Character Recognition) to oprogramowanie pozwalające na znajdowanie i rozpoznawanie fragmentów tekstu w plikach graficznych. Niestety musieliśmy porzucić tę drogę z powodu niejasności kart pacjentów i niejednoznacznych wyników oraz korekt niejednokrotnie nanoszonych długopisem przez lekarza.

Taka procedura pozyskiwania danych jest wyjątkowo nieoptymalna oraz podatna na błędy. Przy obecnej infrastrukturze szpitalnej ręczne przepisywanie danych jest jedynym sposobem na digitalizację zasobów pacjentów. Komfort pracy lekarzy znacząco by się podniósł gdyby zostały wyeliminowane wszystkie papierowe karty pacjentów. Na wypełnienie kompletnych danych jednego pacjenta potrzebowaliśmy ok. 15 minut. Lekarz radiolog dziennie wykonuje około 4 badań MR (w szpitalu MSWiA), więc można założyć, że marnuje około godziny dziennie na niepotrzebne przeglądanie papierowej dokumentacji. Wraz ze współpracującym z nami doktorem postanowiliśmy stworzyć program, który pozwoli na oszczędzanie tego czasu. Wszyscy lekarze, z którymi się konsultowaliśmy, przyznali, że w codziennych obowiązkach pomógłby im program wyliczający objętość prostaty na podstawie plików DICOM. Jest to jedna z płaszczyzn, na których informatyka jest w stanie pomóc oszczędzać czas lekarzy, co przekłada się na komfort oraz zdrowie pacjentów. Uznaliśmy ten problem za bardzo poważny, dlatego właśnie zdecydowaliśmy się na realizację obranego tematu.

3. Specyfikacja

Celem pracy dyplomowej jest stworzenie narzędzia wspomagającego diagnostykę nowotworu prostaty. Powinno ono wspierać pracę lekarza w diagnozie oraz zapewnić automatyczne znajdowania prostaty na zdjęciach rezonansu magnetycznego. Priorytetem aplikacji jest to aby posiadała modularną strukturę, która zapewni łatwą integracja ze szpitalną bazą danych oraz szybki rozwój lub zamianę istniejących komponentów. Aplikacja aby zapewnić łatwy dostęp do szpitalnych baz danych musi posiadać komponent odpowiadający za integrację z systemem PACS. Na potrzeby poniższej pracy utworzyliśmy naszą instancję bazy danych w oparciu o otwarte rozwiązanie ORTHANC [8] w celach deweloperskich.

Narzędzie będzie w stanie przyjąć nowe dane pacjenta w formacie DICOM, przetworzyć zdjęcia zawarte w pliku, wykonać segmentację gruczołu prostaty oraz wyświetlić użytkownikowi przewidywaną maskę. Ponadto użytkownik jest w stanie samodzielnie modyfikować predykcję maski używając do tego opcji edycji. W takim wypadku jako segmentacja zostaje zapisana najnowsza wersja maski stworzonej przez lekarza. Dzięki temu w ramach prowadzonej diagnostyki przeprowadzanej przez specjalistów powstają wysokiej jakości zbiory danych potrzebne do wprowadzenia metod wspomagania opartych o narzędzia sztucznej inteligencji. Kolejny moduł to moduł liczenia wysegmentowanego gruczołu. Aplikacja powinna zapewniać kilka alternatywnych algorytmów obliczania, dzięki czemu będziemy mogli porównywać otrzymane wyniki.

Ponieważ chcemy dostosować aplikację pod jak największą ilość środowisk, serwer backendowy powinien być możliwy do instalacji na każdym systemie operacyjnym.

3.1. Opis biznesowy

Aktualnie jest dostępnych wiele przeglądarek formatu DICOM, lecz żadna z nich nie zawiera modułu pozwalającego na segmentację gruczołu prostaty. Jedną z ważniejszych informacji jakie wykorzystuje się przy diagnostyce raka gruczołu prostaty jest objętość. Na tej podstawie szacuje się parametr PSAD, który silniej niż w przypadku PSA koreluje z obecnością nowtworu w prostacie. Ponadto, żadne z narzędzi używanych obecnie w szpitalu nie posiada modułu pozwalającego na liczenie objętości gruczołu.

Nasze rozwiązanie ułatwia pracę lekarza przedstawiając aproksymowaną objętość gruczołu prostaty. Dodatkowo użytkownik jest w stanie samodzielnie poprawić predykcję używając opcji edycji maski. Dzięki takiej informacji wraz z rozwojem systemu mamy dostęp do coraz dokładniejszych danych dotyczących segmentacji pacjenta. Aplikacja ułatwi pracę lekarza oraz zgromadzi dane zawierające dokładne segmentacje utworzone przy użyciu algorytmu, które następnie podlegają kontroli lekarza. Narzędzie dostarcza również informacje na temat objętości

gruczołu krokowego biorąc pod uwagę precyzję urządzenia na którym zostały wykonane zdjęcia.

Objętość jest kluczową informacją dla lekarza przy wystawianiu diagnozy. Wizualizacja wysegmentowanego gruczołu prostaty oraz obliczona wartość objętości pomoże lekarzowi w diagnostyce, a dzięki łatwemu rozszerzaniu aplikacji rozwój i modyfikacje modułu obliczania objętości lub segmentacji można łatwo zmieniać algorytmy, za pomocą, których je wyznaczamy.

3.2. Wymagania funkcjonalne

Podstawowym wymaganiem aplikacji jest możliwość liczenia segmentacji, objętości oraz integracja ze szpitalnym systemem PACS. Metody realizacji każdego z nich zostały opisane w dalszej części pracy.

Podstawą interakcji użytkownika z systemem jest strona WWW - użytkownik powinien być w stanie otworzyć ją na dowolnym komputerze z dostępem do internetu, wyposażonym w przeglądarkę:

1. Google Chrome w wersji 49 lub wyższej
2. Mozilla Firefox w wersji 52 lub wyższej
3. Safari w wersji 10.1 lub wyższej

Posiadanie przeglądarki innej niż wymienione lub w starszej wersji nie oznacza że strona nie będzie działać, jednak nie da się zagwarantować że będzie to działanie w pełni poprawne. Każdy z użytkowników ma dostęp do tych samych danych oraz takie same możliwości Funkcjonalność aplikacji WWW rozbita jest na trzy podstrony:

1. Wyświetlenie listy dostępnych pacjentów z aplikacyjnej bazy danych lub ze szpitalnego systemu PACS po wpisaniu danych logowania lekarza
2. Dodanie nowego pacjenta z pliku lub ze szpitalnej bazy danych. Dane szpitalne przed wysłaniem na stronę serwera powinny być zanonimizowane tak aby serwer nie przechowywał żadnych danych wrażliwych.
3. Wyświetlenie danych pacjenta w tym obrazu z rezonansu magnetycznego, maski uzyskanej poprzez segmentacje obrazu, danych z pliku DICOM oraz wartości objętości gruczoły na podstawie aktualnej maski.

3.3. Diagram przypadków użycia

Po wejściu na stronę internetową wyświetlana jest lista pacjentów znajdujących się obecnie w bazie danych. Dodatkowo po wcześniejszym połączeniu się z systemem szpitalnym PACS ma możliwość wyświetlenia pacjentów ze szpitalnej bazy danych. W ramach środowiska deweloperskiego dane, którymi inicjalnie zasilona jest aplikacyjna baza danych pochodzą z konkursu. [2] Lekarz ma możliwość dodania nowego pacjenta z pliku lub ze szpitalnej bazy danych. Aby dodać pacjenta z pliku wystarczy w przeglądarce wybrać pacjenta, obraz DICOM zostanie pobrany z bazy PACS, zanonimizowany po stronie przeglądarki, a następnie wysłany

do serwera. Dodatkowo użytkownik ma możliwość dodania pacjenta z pliku DICOM poprzez przeciągnięcie zbioru zdjęć pacjenta na okno przeglądarki. W przypadku gdy wybrany pacjent nie znajduje się jeszcze w bazie danych aplikacja tworzy nowa encję opisującą tego pacjenta, w przeciwnym przypadku, jeżeli obraz nie znajduje się jeszcze w bazie danych dla pacjenta o takim identyfikatorze, aplikacja dodaje kolejne zdjęcia dla już istniejącego pacjenta. Następnie lekarz ma możliwość oglądania zdjęć pacjenta, oraz danych jakie o tym pacjencie aktualnie znajdują się w bazie danych. Strona wyświetla kolejne obrazy, które zostały dodane dotyczące wybranego przypadku. Dodatkowo wyświetlane są informacje z pliku DICOM dotyczące informacji o pacjencie i wyników badań jeżeli taki znajdowały się w pliku, który został wysłany do bazy danych. Ponadto aplikacja wyświetla maskę przedstawiającą wynik procesu segmentacji prostaty. Użytkownik ma możliwość ukrycia maski oraz jej edycji za pomocą dostarczonych narzędzi. Aplikacja dodatkowo liczy objętość wysegmentowanej prostaty na podstawie masek znajdujących się w bazie danych. W aplikacji dostępne jest kilka alternatywnych algorytmów liczenia objętości. Wartość nie jest przeliczana automatycznie w przypadku edycji. Lekarz po edycji wszystkich masek, naciska przycisk na stronie, który uruchamia proces liczenia objętości

Na poniższym rysunku przedstawiono w postaci diagramu UML zbiór przypadków użycia platformy dla lekarza i dla administratora systemu (programisty).

Rysunek 3.1: Diagram przypadków użycia

W tabeli 3.1 zebrano szczegółowe opisy przypadków użycia.

3.4. Wymagania niefunkcjonalne

Tabla 3.2 przedstawia wymagania niefunkcjonalne w naszym systemie.

Tablica 3.1: Przypadki użycia

Nazwa	Opis	Odpowiedź systemu
Połącz z systemem PACS	Użytkownik po podaniu informacji dostępu do szpitalnego systemu PACS zostaje z nim połączony	Lista pacjentów znajdujących się w wybranym systemie
Pobierz pacjenta z systemu PACS	Pobranie obrazu lub obrazów w formacie DICOM i wysłanie ich po uprzedniej anonimizacji do serwera aplikacyjnego	Informacja o sukcesie lub błędzie przy dodawaniu użytkownika
Dodaj pacjenta z pliku	Wysłanie pliku lub plików DICOM do serwera aplikacyjnego	Informacja o sukcesie lub błędzie przy dodawaniu użytkownika
Wyświetl listę pacjentów dostępnych w systemie	Pobiera listę pacjentów dostepnych w systemowej bazie danych, inicjalnie baza danych jest zasilona danymi pochodzącymi z archiwum danych pacjentów ze zdiagnozowanym rakiem prostaty	Lista identyfikatorów dostępnych pacjentów
Zobacz pacjenta	Wyświetla dane dotyczące danego pacjenta aktualnie znajdujące się w bazie danych, w szczególności listę obrazów rezonansu magnetycznego oraz maskę z wysegmentowaną prostatą	Dane pacjenta
Segmentuj	Oblicza segmentację dla wybranego obrazu	Maska opisująca prostatę na wybranym obrazie
Policz objętość	Liczy objętość na podstawie listy masek na podstawie wybranego algorytmu	Wartość objętości prostaty

3.5. Środowisko sprzętowe i programowe

Opiszemy w dużym skrócie technologie wykorzystane w poniższej pracy, decyzje jakie stały za poszczególnymi wyborami, oraz alternatywne technologie i przyczyny dlaczego się na nie nie zdecydowaliśmy.

3.5.1. Języki i technologie

Docker

Podstawą całej architektury backendowej aplikacji jest wykożystanie kontenerów czyli środowisku w którym każdy serwis uruchamiany jest w odizolowanym środowisku zawierającym wszystkie zależności wymagane do jego działania. Ponieważ każdy z nas miał inne środowisko deweloperskie, docker zapewnia nam, że serwisy będą działały w dokładnie ten sam sposób na każdej maszynie. Ma to również zalety dla środowisk produkcyjnych, ponieważ niezależnie czy udostępnimy naszą aplikację jako program on-premise czy zdecydujemy się na umieszczenie go w chmurze, konfiguracja będzie tak samo nieskomplikowana. Zapewnia to niezależność poszczególnych komponentów oraz odizolowanie aplikacji od systemu operacyjnego. Zdecydowaną przewagą Docker'a nad

Tablica 3.2: Wymagania niefunkcjonalne

Obszar wymagań	Nr wymagania	Opis
Użyteczność	1	Aplikacja będzie aplikacją internetową dostępną na systemach Linux lub Windows. Aplikacja będzie zawierać przyjazne użytkownikowi karty z opcją segmentacji oraz obliczania objętości prostaty. Dodatkowo lekarz będzie miał możliwość dodawania kolejnych pacjentów
Niezawodność	2	Aplikacja będzie w stanie wykonywać funkcjonalności opisane wyżej. Ponadto Aplikacja będzie zapisywać logi z błędami oraz przyczyną awarii
Wydajność	3	Aplikacja WWW powinna działać płynnie na każdym komputerze z dowolnym systemem operacyjnym wyposażonym w odpowiednią przeglądarkę. Obliczenia zależnie od rozmiarów danych wejściowych aplikacja powinna wykonywać takim czasie aby praca użytkownika nie była przez nią spowolniona, przy tym nie blokując interfejsu użytkownika
Utrzymanie	4	Aplikacja będzie miała architekturę modułową, która będzie łatwo rozszerzalna. Cały kod aplikacji zostanie udokumentowany. W ramach środowiska developerskiego powstanie obraz Docker.

wirtualizacją jest możliwość uruchomienia aplikacji w wydzielonym kontenerze, ale bez konieczności emulowania całej warstwy sprzętowej i systemu operacyjnego. Do zarządzania kontenerami użyliśmy udostępnianego przez Docker narzędzia docker-compose. W ten sposób w przypadku potrzeby modyfikacji jednego z serwisów, aplikacja dalej pozostaje aktywna a zmianie ulega tylko jeden moduł.

Środowisko .NET Core

Serwis backendowy, który odpowiada za integrację z bazą danych oraz komunikację z warstwą prezentacji został napisany przy użyciu frameworku .NET Core. Główną zaleta tego wyboru jest wieloplatformowość rozwiązania. Dzięki nowemu środowisku uruchomieniowemu aplikacja może działać na dowolnym systemie operacyjnym. Kożystamy z najnowszej na chwilę pisania pracy wersji (2.1), która wspiera większość pakietów dostępnych dla .NET Framework. Dodatkowo głównym założeniem przy powstawaniu systemu było aby był oparty o architekturę mikroserwisów przy użyciu kontenerów, które również są wspierane przez framework.

Język programowania C#

System został wykonany z wykorzystaniem języka programowania C# oraz platformy programistycznej .NET Framework. Wybór tej technologii był podyktowany doświadczeniem zespołu w jej wykorzystaniu. .NET Framework został wykorzystany do napisania modułu backendowego odpowiedzialnego za obsługę akcji użytkownika oraz komunikację z bazą danych. Ponieważ serwer powinien działać na wszystkich systemach operacyjnych użyliśmy .Net core oraz ASP.NET MVC Core.

Entity Framework Core

Entity Framework Core jest dedykowanym frameworkiem dla aplikacji działających w środowisku .NET Core zapewniający dostęp i zarządzanie bazą danych. Jest to narzędziem typu ORM (ang. Object RelationalMapping). Dzięki użyciu tego narzędzia jako warstwy dostępu do bazy danych mogliśmy w łatwy sposób stosować podejście code-first przy projektowaniu aplikacji. Dodatkowo w przeciwieństwie do wersji z .NET Framework Entity Framework zapewnia wydajność porównywalną do natywnych wywołań zapytań bazodanowych za pomocą dedykowanego sterownika dla .NET czyli ADO.NET

SQL Server 2016

Microsoft SQL Server jest systemem służącym do zarządzania bazą danych stworzonym przez firmę Microsoft. Platforma spełnia wszystkie podstawowe funkcjonalności jakich oczekujemy od bazy danych ale dodatkowo ponieważ w przypadku operowania na dużych wolumenach danych, co jest bardzo prawdopodobne w przypadku danych szpitalnych, oferuje wiele możliwości optymalizacji oraz zaawansowane narzędzia do utrzymania oraz analityki. Jest to platforma preferowana do projektów działających w .NET Framework. Pomino stosowania w projekcie podejścia *Code first* nie wykluczamy, że w przypadku rozwoju platformy będzie trzeba je zmienić.

Język programowania Python

W aplikacji wykożystaliśmy język Python do modułu odpowiadającego za segmentację prostaty na obrazie. Wybór ten podyktowany jest dużą ilością dostępnych pakietów do uczenia maszynowego oraz głębokiego uczenia. Python jest wspierany na wszystkich większych systemach operacyjnych oraz łatwo tworzy się kontenery wykorzystujące Pythona. Ponadto brak procesu kompilacji wspomagał proces prototypowania algortytmu, co w przypadku uczenia maszynowego ma duże znaczenie.

Framework Flask

Aby zapewnić komunikację za pomocą protokołu http między serwisem segmentacji, a pozostałymi elementami systemu potrzebowaliśmy biblioteki, która udostępnia taką funkcjonalność.Framework zapewnia elementarną implementacją serwera http, który pozwala na komunikację z pozostałymi komponentami. Ponieważ segmentacja wspiera tylko i wyłącznie jeden punk dostępowy uznaliśmy, że wybór rozwiązania bardziej rozbudowanego jest nieuzasadniony. Django czy Pyramid, to dwa bardzo popularne rozwiązania dla serwerów http zaimplementowane w języku Python jednak o wiele bardziej rozbudowanej konfiguracji nie przynosząc zarazem odczuwalnych korzyści.

Biblioteka Keras

Do implementacji algorytmu segmenacji gruczołu krokowego na zdjęciach potrzebowaliśmy biblioteki pozwalającej na tworzenie sieci neuronowych. Podczas poszukiwań najbardziej odpowiedniego rozwiązania kierowaliśmy się kilkoma kryteriami. Przede wszystkim chcieliśmy aby rozwiązanie było napisane w języku Python oraz wspierało użycie biblioteki TensorFlow. Dodatkowo chcieliśmy aby biblioteka umożliwiała szybkie prototypowanie oraz była przenośna między systemami operacyjnymi. Keras jest obecnie jednym z najpopularniejszych bibliotek do takiego zastosowania. Zapewnia wysoko poziomowy interfejs ułatwiający tworzenie rozwiązań opartych o uczenie maszynowe. Decyzja o

użyciu frameworku zapadła na podstawie spełeniania wszystkich powyższych założeń oraz doświadczeniem zespołu w jego wykorzystaniu.

Tensorflow

TensorFlow jest biblioteką typu open source do obliczeń numerycznych i uczenia maszynowego. Biblioteka łączy ze sobą szereg modeli i algorytmów głębokiego uczenia. Korzysta z Pythona, aby zapewnić wygodne API do budowania aplikacji, a obliczenia wykonywane są w C ++. Biblioteki nie wykożystaliśby bezpośrdenio tylko przez naladkę Keras, która udostępnia wysokopozomowe API.

OpenCV, OpenCvSharpCore

OpenCV [4] to biblioteka oprogramowania do nauki i uczenia maszynowego typu open source. Jest to chyba najpopularniejsza biblioteka do przetwarzania obrazów. Do integracji z .NETem użyliśmy wrappera działającego w środowisku .NET Core OpenCVSharpCore. Zapewnia dostęp do metod biblioteki poprzez kod C#. W naszej pracy została wykorzystana w module liczącym objętość do znajdowania otoczki maski oraz liczenia pola powierzchni poligonów.

React

React okazał się idealnym rozwiązaniem, ponieważ jest jedynie biblioteką dostarczającą komponenty, która może być rozszerzalna przez kolejne biblioteki. Projekt ten dojrzałym framework'iem stworzonym i rozwijanym przez Facebook. Przygotowując aplikację SPA postanowiliśmy znaleźć rozwiązanie, które najbardziej dopasuje sie do naszych wymagań.

Ponadto wszystkie wymienione technologie zostały wybrane do realizacji pracy, ponieważ zespół, który nad nią pracował, miał doświadczenie w pracy z nimi wyniesione z poprzednich projektów realizowanych w ramach studiów i pracy zawodowej.

3.5.2. Rozważane alternatywne technologie

W momencie pisania pracy ustaliliśmy, że najpopularniejszymi aktualnie rozwiązaniami są następujące projekty:

Java

Jeden z najpopularniejszych obecnie języków programowania, teoretycznie spełniający wszystkie nasze wymagania. Zapewnia modularność, jest wspierany przez wszystkie języki programowania, ma wiele dostępnych bibliotek gotowych do użycia w aplikacji. Odrzuciliśmy ten wybór z powodu dużej ilości konfiguracji potrzebnej do utworzenia systemu jaki tutaj opisujemy. Dodatkowo ponieważ przetwarzamy pliku zajmujące często nawet powyżej 100MB obawialiśmy się problemów z wydajnością zwalniania pamięci w Javie. W przypadku przetwarzania dużych wolumenów danych Java wymaga dużej ingerencji w mechanizm zwalniania pamięci.

Angular

Projekt Angular jest jednym z najpopularniejszych obecnie rozwiązań do tworzenia warstwy prezentacji aplikacji. Początkowo wydawał się być doskonałym wyborem dla naszego zastosowania (dojrzały projekt, wspierany przez Google), aczkolwiek wymuszał on używanie konkretnych rozwiązań stworzonych na potrzeby tego konkretnego frameworka

do wykonywania zapytań. Z tego powodu zdecydowaliśmy się zrezygnować z tego projektu, ponieważ naruszał nasze kryterium dotyczące modularności.

Vue

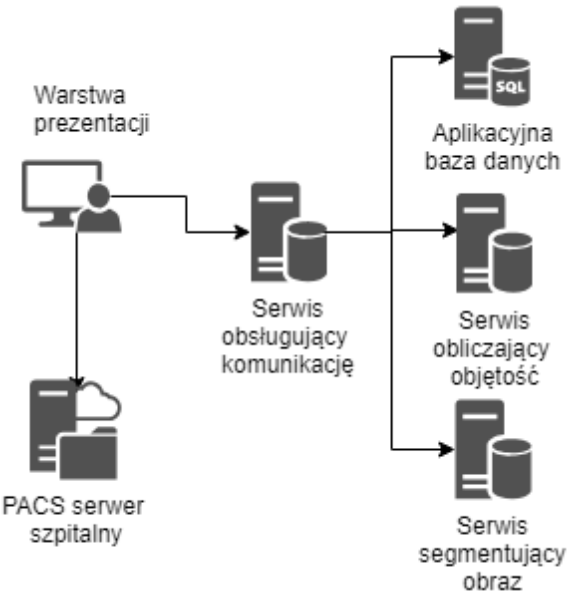
Vue jest coraz bardziej popularnych frameworkiem, który wyróżnia się modularnością, przenośnością oraz niskim progiem wejścia. Nie dopasował się do naszych wymagań ze względu na niedojrzałość projektu, przez co dostępne komponenty nie spełniały naszych oczekiwań. Wciąż jest to rozwiązanie nie wspierane przez dużych klientów biznesowych.

4. Opis aplikacji

W tym rozdziale przedstawione zostaną efekty pracy. Na wstępie przedstawimy wysoko poziomową architekturę całego rozwiązania. Następnie opiszemy poszczególne moduły zawarte w aplikacji oraz zasadę ich działania, przedstawiając szczegóły techniczne implementacji. Dzięki podziałowi aplikacji na osobne serwisy działające niezależnie, zawarte w dalszej części opisu modułów również opiszemy niezależnie. Kontenery oddzielają aplikacje od siebie oraz infrastruktury, na której działają zapewniając dodatkową warstwę ochronną dla aplikacji.

4.1. Architektura systemu

System składa się z modułów realizujących ściśle określone zadania, niezależne zadania, dlatego też łatwo wydzielić je jako osobne serwisy działające niezależnie od siebie. Każdy z serwisów działa jako osobny kontener. Konteneryzacja polega na tym, że umożliwia się uruchomienie wskazanych procesów aplikacji w wydzielonych obiektach, które z punktu widzenia aplikacji są odrębnymi instancjami środowiska uruchomieniowego. Każdy kontener posiada wydzielony obszar pamięci, odrębny interface sieciowy z własnym prywatnym adresem IP oraz wydzielony obszar na dysku.



Rysunek 4.1: Schemat architektury aplikacji

Komunikacja serwera backendowego z serwerem PACS odbywa się za pośrednictwem aplikacji internetowej, z której korzysta klient. Jest to zrobione specjalnie i daje nam możliwość łatwej integracji z systemem szpitalny. W zasadzie jest to jedyna opcja jaką znaleźliśmy

w trakcie naszego badania architektury szpitalnej nie wymagająca ustawiania specjalnego połączenia VPN między naszym serwisem a serwerami szpitalnymi. Dzięki temu zyskujemy niezależność. Aplikacja może działać na dowolnym środowisku przy tym zapewniając klientowi dostęp i wysyłanie danych pacjenta na nasze serwisy do przetwarzania. Rozwiązanie w którym nasz serwer komunikuje się bezpośrednio z bazą danych szpitalnych ma kilka dużych wad. Przede wszystkim przesyłamy poufne dane pacjentów przez sieć. Aby to robić bezpiecznie potrzebowalibyśmy dedykowanego bezpiecznego połączenia, którego nie udało nam się uzyskać przez cały okres pisania pracy.

Reszta architektury jest standardowa w przypadku aplikacji o architekturze mikroserwisowej. Mamy główny serwer odpowiadający za komunikację z użytkownikiem i obsługę bazy danych. Serwisy do liczenia objętości i segmentacji są niezależne i nie mają bezpośredniego dostępu do bazy danych. Jedyna modyfikacja jaką można by wykonać w przypadku intensywnego użytkowania aplikacji to skonfigurować osobne bazy dla każdego serwisu aby uniknąć niepotrzebnego przesyłania danych przez sieć. Jednak w naszym przypadku proponowane rozwiązanie jest bezpieczniejsze, łatwiejsze w utrzymaniu i w zupełności wystarczające.

4.2. Komunikacja między modułami

Komunikacja między komponentami odbywa się za pomocą protokołu *http*. Każdy z serwisów udostępnia swoje punkty dostępu. W przypadku serwisu kalkulującego objętość oraz wykonującego segmentację dostęp jest jedynie z poziomu kontenera.

Serwisy są lokalizowane dzięki mechanizmom Dockera, który tworzy podsieć dla wszystkich działających w obrębie jednego Docker-compose. Dostęp do serwisów odbywa się przez adres postaci http://*nazwa-serwisu*:*port* (jeżeli nie zmieniony to serwis działa na porcie 80). Dostęp z zewnątrz zapewnia jedynie serwis obsługujący komunikacje, a pozostałe serwisy są niedostępne.

Każdy z naszych serwisów udostępnia REST API. Najwięcej metod ma moduł do komunikacji z klientem, którego pełna dokumentacja jest opisana w Załączniku 1. Może być przydatne w przypadku chęci użycia naszego rozwiązania z inną warstwą prezentacji np. aplikacją okienkową.

5. Opis modułów

W poniższym rozdziale opiszemy poszczególne moduły. Wytłumaczymy drogę dojścia do rozwiązania oraz ewentualne modyfikacje jakie można zastosować przy dalszym rozwoju. Skupiliśmy się na unikaniu zależności pomiędzy modułami. Wszystkie moduły zaprojektowaliśmy w sposób modularny, tak aby dowolnie móc je wymieniać.

5.1. Moduł segmentacji

Moduł odpowiada za znajdowanie na obrazie wejściowym prostaty. Algorytm działa na zdjęciach w formacie PNG i zwraca również obraz w tym samym formacie. Do obliczeń użyliśmy konwolucyjnych sieci neuronowych, na podstawie danych z konkursu [2]. Problem segmentacji gruczołu prostaty jest niezwykle istotny z perspektywy lekarzy, którzy na podstawie wysegmentowanej maski gruczołu są w stanie łatwiej postawić diagnozę dotyczącą raka prostaty.

5.1.1. Segmentacja obrazu

Segmentacja obrazu jest procesem dzielenia obrazu cyfrowego na wiele segmentów). Celem segmentacji jest uproszczenie i / lub zmiana reprezentacji obrazu na obszary, które są bardziej znaczące i łatwiejsze do analizy. Segmentacja obrazu jest używana do lokalizowania obiektów i granic (linii, krzywych itp.) w obrazach. Jest to proces przypisywania etykiety do każdego piksela na obrazie, tak że piksele z tą samą etykietą mają pewne cechy

Na początku wyjaśnijmy pojęcie segmentacji. Proces polega na podziale obrazu na obszary, które są jednorodne pod względem pewnych wybranych cech. Przez obszar najczęściej rozumiemy zbiór pikseli obrazu. Zadanie segmentacji nie tylko wymaga zrozumienia co się znajduje na obrazie, lecz również ustalenia gdzie się ten obszar znajduje. Celem segmentacji jest wyeksponowanie pewnych własności obrazu, które są istotniejsze od pozostałych. Istnieje wiele algorytmów segmentacji, najpopularniejszymi są: progowanie obrazu, wykrywanie krawędzi, rozszerzanie obszaru. W przypadku każdej z wyżej wymienionych metod wynik nie zmienia się dla kolejnego uruchomienia algorytmu z tymi samymi danymi wejściowymi. W ostatnich latach głębokie sieci neuronowe oraz konwolucyjne sieci neuronowe wykazały, że radzą sobie z problemem klasyfikacji lepiej niż statyczne rozwiązania.[5]. W naszej pracy postanowiliśmy zastosować sieci neuronowe jako bardziej nowoczesne podejście. Mając nadzieję, że da ono lepsze rezultaty.

5.1.2. Sieci neuronowe

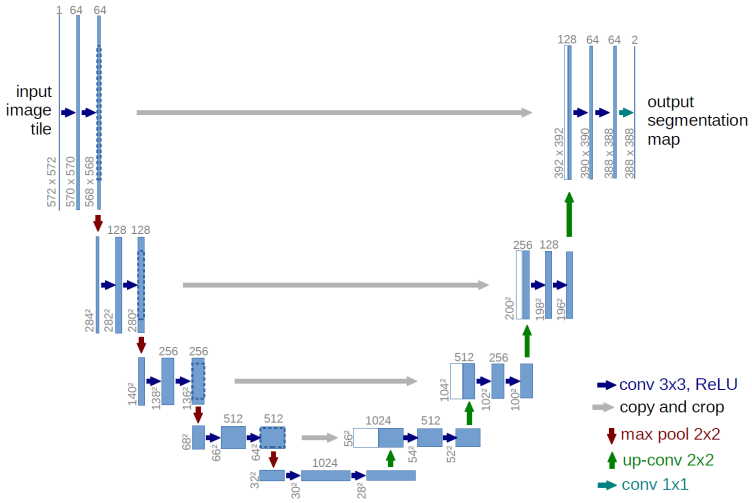
Głęboka sieć neuronowa (DNN) to sztuczna sieć neuronowa z wieloma warstwami między warstwami wejściową i wyjściową. DNN znajduje poprawną matematyczną manipulację, aby

przekształcić dane wejściowe na dane wyjściowe, niezależnie od tego, czy jest to relacja liniowa czy nieliniowa. Sieć przechodzi przez warstwy obliczające prawdopodobieństwo każdego wyjścia. Każda manipulacja matematyczna jako taka jest uważana za warstwę, a złożone DNN mają wiele warstw, stąd nazwa "głębokieśieci. Celem jest, aby ostatecznie sieć została nauczona w zakresie rozkładania obrazu na funkcje, identyfikowania trendów istniejących we wszystkich próbkach i klasyfikowania nowych obrazów według ich podobieństw bez konieczności wprowadzania danych przez człowieka.

Konwolucyjna sieć neuronowa (CNN) to rodzaj sztucznej sieci neuronowej wykorzystywanej w rozpoznawaniu i przetwarzaniu obrazów, która jest specjalnie zaprojektowana do przetwarzania obrazów. W przeciwieństwie do sieci użytej do głębokiego uczenia maszynowego, każda ukryta warstwa aktywacyjna jest obliczany jako iloczyn kartezjański wyników kolejnych wartstw. Jednak w sieciach konwolucyjnych każda ukryta aktywacja jest obliczana przez pomnożenie małego lokalnego sygnału wejściowego względem ciężarów. Ciężary są następnie dzielone na całej przestrzeni wejściowej. Po obliczeniu ukrytych jednostek, maksimum warstwa pomaga usunąć zmienność ukrytych jednostek. W szczególności każda jednostka gromadząca maksimum otrzymuje aktywacje ze znalezionej ilości splotowych pasm, i wyprowadza maksimum aktywacje z tych splotów. Większość CNN działa w rozpoznawaniu obrazu ma niższe warstwy sieciowe, które są splotowe, a wyższe warstwy sieciowe są w pełni połączone. W tej sekcji omówimy ile potrzeba warstw splotowo-w pełni połączonych, co to optymalna liczba ukrytych jednostek na warstwę, co jest najlepsze strategia łączenia i najlepszy typ wejścia dla CNN. tradycyjnych sieciach neuronowych.

U-Net

Sieć U-Net jest zbudowana na podstawie architektury pełnych sieci konwolencyjnych(ang. Fully Convolutional Network) zmodyfikowaną w taki sposób, aby uzyskiwać lepsze rezulataty segmentacji na obrazach medycznych. Lepsze rezultaty pochodzą z kształtu architektury U, który skutkuje większą ilością cech w scieżce wstępującej co przekłada się na lepszy przepływ informacji o kontekście i lokalizacji na obrazach o niskiej rozdzielczości.



Rysunek 5.1: Architektura sieci UNet

W procesie uczenia sieci neuronowych najbardziej czasochłonnym elementem jest proces trenowania. Uczenie sieci neuronowych odbywa się na kartach graficznych ze względu na ich

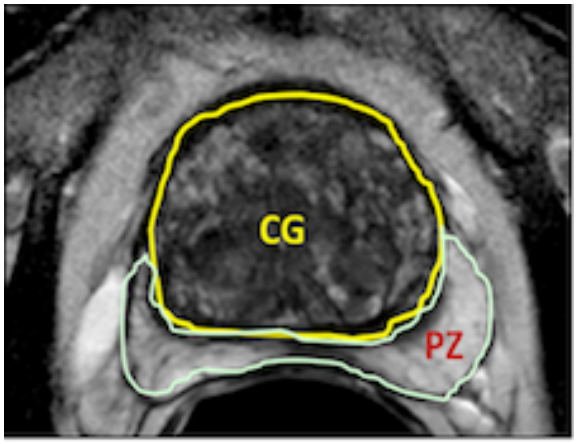
lepsze przystosowanie do przetwarznia wielkiej ilości operacji w sposób zrównolelony.CPU jest przeznaczony do przełączania zadań. Nie jest przeznaczony do wysokiej przepustowości. Musi przełączać się między wieloma zadaniami i przechowywać konteksty podczas przełączania i dbać o pamięć współdzieloną. Jest przeznaczony do zadań o niskiej przepustowości i wysokiej latencji. Z drugiej strony procesor graficzny został zaprojektowany z myślą o dużym obciążeniu pracą i przepustowości. Jest przeznaczony do zadań o niskim opóźnieniu i wysokiej przepustowości.

5.1.3. Opis danych

Najlepszym źródłem pozyskania daych dla naszych celów badawczych są strony konkursowe ponieważ zawierają obrazy z już odnalezioną prostatą na zdjęciu. Organizowanych jest wiele konkursów [2] w celu opracowania stabilnego algorytmu wyznacznia gruczołu prostaty, jednak my po analizie zdecydowaliśmy się na dane z konkursu - *Automated Segmentation of Prostate Structures Challange* [6] który dostarczył danych użytych do trenowania naszej sieci.

Dane

Dostarczone dane składały się z plików DICOM zawierających dane pacjentów oraz z maski zawierających obszary gdzie znajduje się prostata. Dane były przygotowane przez lekarzy, oraz podległy wcześniejszemu procesowi anonimazacji, stąd w plikach nie ma żadnych danych wrażliwych na temat pacjentów. Osoby poddano badaniom za równo pod maszynami o mocy 1.5T oraz 3T z użyciem cewki endorektalnej oraz cewki powierzchniowej. Maski zostały przygotowane przez zespól lekarzy z uniwersytetów Boston University School of Medicine oraz Case Western University. Zdjęcia zostały przygotowane w formacie nrrd, gdzie zaznaczone są dwa regiony: część obwodową gruczołu krokowego (PZ) oraz centralny gruczoł(CG).



Rysunek 5.2: Zaznaczone regiony prostaty na zdjęciu z badania MR

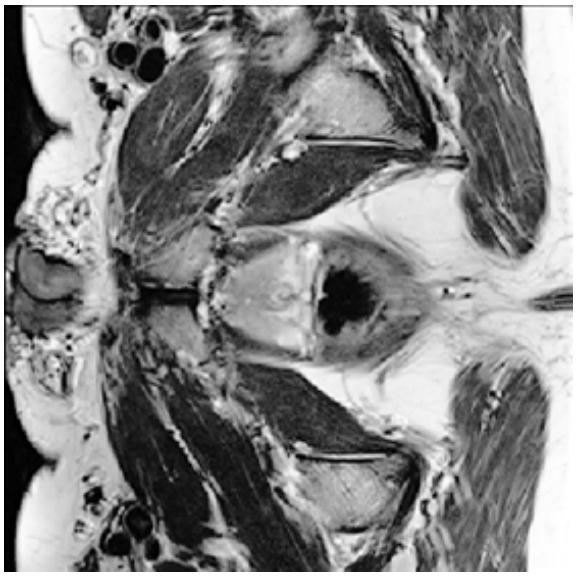
Zbiór treningowy 60 pacjentów, łącznie 1073 obrazów

Zbiór testowy 10 pacjentów, 147 obrazów

Zbiór walidacyjny 10 pacjentów, 146 obrazów

Obrazy wejściowe

Dane zostały przetransformowane z formatu DICOM w taki sposób, aby wyeksportować tag pixelData i przetransformować go do postaci tablicy zawierającą intensywność pixeli.



Rysunek 5.3: Przykładowy obraz z badania MRI

Oznaczone maski

Przygotowane maski podzielone zostały na dwa regiony gruczołu PZ oraz CG. Obszar CG został oznaczony wartościami 2, obszar PZ wartościami 1 natomiast pozostały obszar wartością 0. Pnieważ nasz problemy był bardziej ogólny nie było potrzeby wprowadzenia rozróżnienia między strefami prostaty. Dlatego dane poddaliśmy procesowi normalizacji maski do wartości z przedziału [0;1].

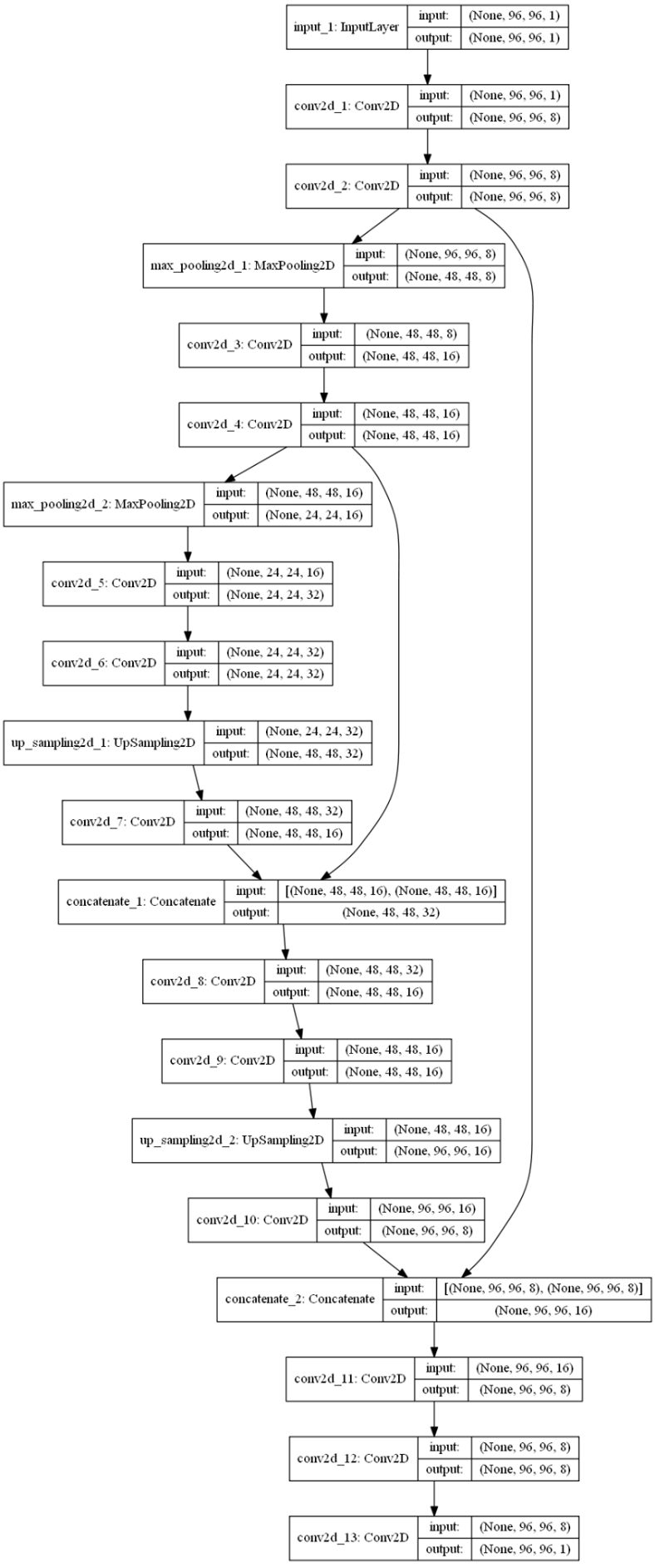
Rysunek 5.4: Przykładowy obraz z badania MRI

5.1.4. Opis rozwiązania

W przypadku analizy obrazów medycznych problemem jest mały zbiór danych testowych dobrej jakości. Postanowiliśmy do tego celu wykożystać dane pochodzące z konkursu opisane powyżej. Eksplorację danych rozpoczeliśmy od przeglądania obrazów DICOM. Na wstępie zauważyliśmy, że granice pomiędzy tkankami miękkimi nie są dobrze podkreślone. Co więcej dane jakie dostaliśmy są niskiej rozdzielczości. Segmentacja gruczołu krokowego nie jest

zadaniem trywialnym z powodu podobnych intensywności obrazu między gruczołem prostaty a sąsiadującą tkanką oraz z powodu niejednorodności intensywności obrazu w okolicy prostaty. Co więcej, gruczoł krokowy ma dużą różnorodność wyglądu i kształtów u różnych pacjentów. Z uwagi na bardzo ograniczony zbiór testowy zastosowaliśmy mechanizmy *Data augmentation*, dzięki czemu w efektywny spoób zwiększyliśmy zbiór testowy. Rowiązanie jest standarowym procesem stosowanym przy przetwarzaniu obrazów medycznych, polega na zwielokrotnienu zbioru testowego poprzez zastosowanie na obrazach inicjalnych podstawowych operacji przetwarzających obraz takich jak np. rotowanie obrazu, sotsowanie filtru Gaussowskiego lub progowania.

Pracę nad naszym algorytmem rozpoczęliśmy od analizy aktualnie wykorzystywanych algorytmów do segmentacji obrazów. Jednym z założeń było znalezienie rozwiązania, które pomimo niewielkiego zbioru danych treningowych poradzi sobie z obrazami o niskiej rozdzielczości. Najbardziej dopasowanym rozwiązaniem okazała się uproszczona sieć U-net, która powstała jako zgłoszenie do konkursu Automated Segmentation of Prostate Structures [**?**]. Wybraliśmy te rozwiązanie ponieważ sieć jest symetryczna, dzięki czemu zyskujemy większą dokładność znajdowania regionów obrazu. Rozwiązanie nauczyliśmy ponownie na przygotowanych przez nas danych. Architekturę sieci przedstawia poniższy diagram.



Rysunek 5.5: Architektura sieci

Pierwszy blok składa się z następujących warstw

conv2d_1 3x3 Warstwa konwolucyjna + funckja aktywacji ReLu

conv2d_2 3x3 Warstwa konwolucyjna + funckja aktywacji ReLu

max_pooling2d_1 2x2 Max Pooling

Kolejne dwa bloki zawierają identyczne warstwy. Te 3 bloki są nazywane ścieżką zstępującą. Jej zadanie to ustalenie znaczenia danych na obrazie. Następnie ten kontekst będzie przetransformowany do ścieżki wschodzącej. Kolejnym krokiem jest znalezioenie bloku będącego wąskim gardłem.

conv2d_5 2x2 Max Pooling

conv2d_6 2x2 Max Pooling

Następny blok składa się z następujących warstw

up_sampling2d_1 up sampling 2x2

con2d_7 3x3 Warstwa konwolucyjna + funckja aktywacji ReLu

max_pooling2d_1 Połączenie z kontekstem danych

conv2d_7 3x3 Warstwa konwolucyjna + funckja aktywacji ReLu

Tę część sieci nazywamy ścieżką wstępująca odpowiedzalną za ustalenie dokładnej lokalizacji segmentacji.

Szybko dotarło do nas, że sieć zaczyna uzyskiwać dobre wyniki, lecz była zbyt wrażliwa na dane. Dlatego dołączyliśmy dodatkowe wariancje obrazów wewnętrznych.

Rozwiązanie oparte na prostej architekturze U-Net rozszerzyliśmy o odwrócenie wartości pixeli. Pomysł ten pojawił się obserwując jak lekarz diagnozuje obrazy z badania MRI oraz na co zwraca uwagę. W celu poradzenia sobie ze słabym kontrastem Pani doktor odwracała kolory pixeli.

Po augmentacji obrazów zaczeliśmy dobieranie parametrów sieci. Rozpoczeliśmy sprawdziań krzyżowy parametrów sieci. Na tej podstawie ustaliliśmy, że odpowiednia liczba epok wynosi 30 natomiast batch_size=32.

5.1.5. Alternatywne podejścia

Każde z wyżej opisanych podejść jest warte rozważenia, jednak z uwagi na stopień skomplikowania nie podjęliśmy się badaniu ich w pracy inżynierskiej.

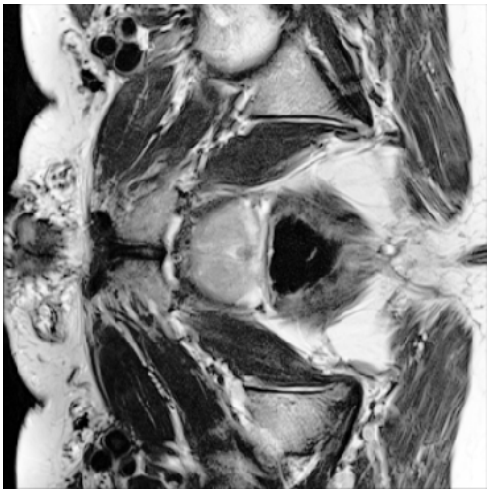
5.1.6. Wyniki

Do ustalenia skuteczności użyliśmy miary wykorzystywanej w zadaniach segmentacji tj. współczynnika podobieństwa Sørensena. Miara ta mówi jak podobne są dwa obiekty. Jej wartość to wspólny obszar regionu segmentacji podzielony przez całkowity rozmiar dwóch obrazów.

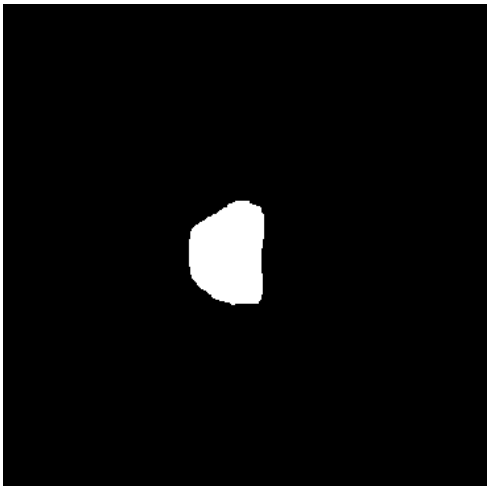
Po nauczeniu sieci osiągneliśmy współczynnik podobieństwa równy 0.68 na danych testowych. Jest to wartość lekko niższa niż wartość uzyskana w rozwiązaniu zgłoszonym do konkursu.

Różnica może być spowodowana różnymi danymi (zostały one lekko zmienione od czasu konkursu).

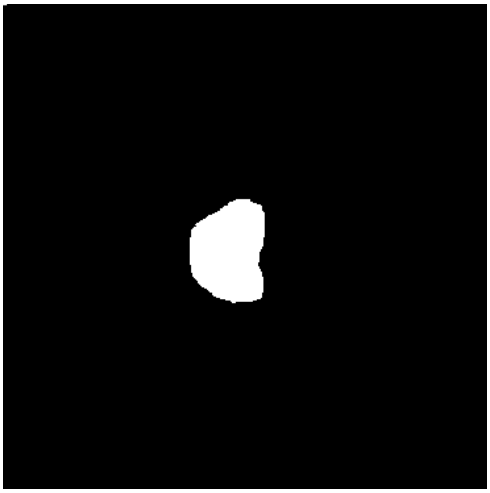
Analizując wyniki segmentacji zauważyliśmy, że nasza sieć radzi sobie bardzo dobrze w momentach, gdzie granica pomiędzy organami jest mocno zarysowna.



Rysunek 5.6: Zdjęcie z badania MRI



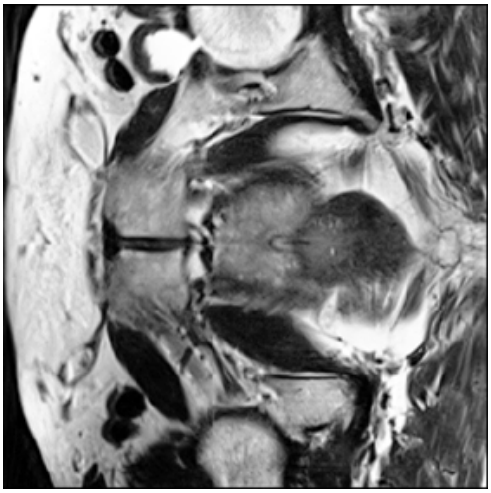
Rysunek 5.7: Maska wyznaczona przez lekarzy



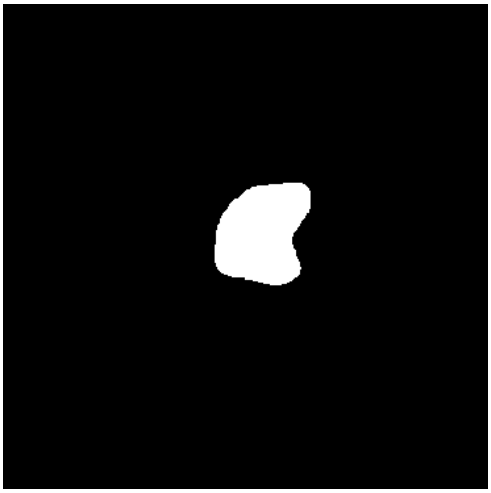
Rysunek 5.8: Poprawna predykcja

Problemy sieci pojawiają się natomiast w momencie, gdy obraz jest rozmyty i nie można

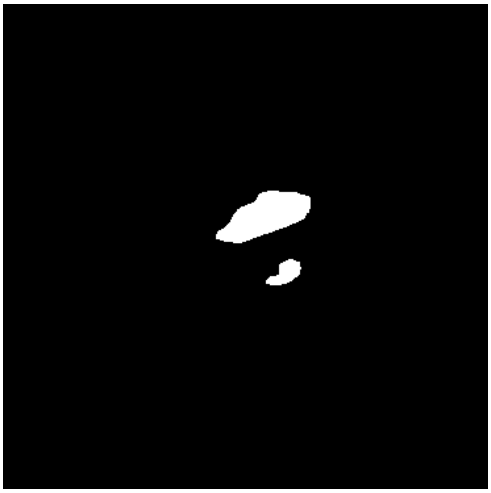
jasno wyznaczyć granic pomiędzy ścianami narządów.



Rysunek 5.9: Zdjęcie z badania MRI



Rysunek 5.10: Maska wyznaczona przez lekarzy



Rysunek 5.11: Poprawna predykcja

Skuteczność sieci może zostać poprawiona poprzez dołożenie obrazów rozmytych. Tworząc obrazy treningowe z szumem sieć nauczy się odnajdować szukany region pomimo rozmytych krawędzi.

5.2. Moduł obliczania objętości

Jedynym z celów pracy było zaproponowanie alternatywnej metody do liczenia objętości. Obecnie w szpitalu objętość gruczoły obliczana jest za pomocą aproksymacji przedstawionej wzorem:

$$V = (h_{max} - h_{min}) * S_{max} * 0.54$$

gdzie
h_{max} - wysokość górnego końca gruczoły prostaty
h_{min} - wysokość dolnego końca gruczoły prostaty
S_{max} - pole powierzchni warstwy o największym, każdym liczonym jako:

$$S_d = (i_{d_{max}} - i_{d_{min}}) * (j_{d_{max}} - j_{d_{min}})$$

gdzie
$i_{d_{max}}$ - pozycja najdalej wysuniętego pixela prostaty poziomo
$i_{d_{min}}$ - pozycja najmniej wysuniętego pixela prostaty poziomo
$j_{d_{max}}$ - pozycja najdalej wysuniętego pixela prostaty pionowo
$j_{d_{min}}$ - pozycja najmniej wysuniętego pixela prostaty pionowo
d - indeks warstwy

Metoda polega na znalezieniu prostopadłościanu w który możemy wpisać gruczoł i obliczenie jego objętości. Następnie wartość jest przemnażana przez stałą aby przybliżyć kształt elipsą. Metoda wydaje się być niedokładna, co potwierdzają lekarze.

Postanowiliśmy zaproponować alternatywne rozwiązanie. Objętość w dalszych obliczeniach będziemy obliczać jako sumę voxeli znajdujących się w każdej z warstw obrazu.

$$V_{total} = \sum_{i=0}^{n} \sum_{d_i,j=0}^{m} V_{d_{ij}} \; gdzie \; d_{ij} \; pixelem \; maski$$

gdzie
n - liczba wartstw pilku DICOM
m - liczba pixeli na warstwie

Objętość voxela liczona jest wzorem:

$$V_{d_{ij}} = PixelSpacing_0 * PixelSpacing_1 * SliceThickness$$

gdzie
$PixelSpacing_0$ - pierwsza wartość atrybutu PixelSpacing (0028, 0030) z pliku DICOM
$PixelSpacing_1$ - druga wartość atrybutu PixelSpacing (0028, 0030) z pliku DICOM
$SliceThickness$ - atrybut SliceThickness (0018, 0050) z pliku DICOM

Ze wzoru wynika, że wartości atrybutów nie wpływają na poprawność wyniku, stąd w dalszej części opisu dla uproszczenia przyjmiemy parametry pochodzące z pliku DICOM jako stałe

równe 1. Przy takich założeniach wartość objętości jest równa polu powierzchni maski dla danej warstwy.

5.2.1. Liczenie objętości

Po pierwsze proponujemy aby wszystkie obrazy nie były traktowane jako całość, a objętość całej prostaty liczyć jako objętość kolejnych warstw nachodzących po sobie, a końcowa objętość jest suma objętości.

$$V = \sum_{i=0}^{n-1}((S_i + S_{i+1}) * (h_{d_{i+1}} - h_{d_i}))/2$$

gdzie
n - liczba warstw
d_i - indeksy kolejnych warstw obrazu
S_i - objętość kolejnych warstw obrazu
h_{d_i} - lokalizacja warstwy o indeksie d_i

Liczenie pola powierzchni maski została opisana w kolejnej sekcji.
W przypadku naszego systemu algorytm poprawić o niedokładność segmentacji prostaty. Zdarzają się sytuacje gdzie kolejna warstwa nie jest warstwą ostatnią na której znajduje się prostata a algorytm nie znalazł na zdjęciu prostaty. W tym przypadku bierzemy kolejną następującą po niej niepustą warstwę.

5.2.2. Liczenie pola powierzchni

Z uwagi na niedokładności masek zwracanych przez algorytm segmentacji proponujemy 3 alternatywne funkcje poprawiające obraz. Proponowe sposoby liczenia pola powierzchni:

1. Pole powierzchni prostokąta na którym można opisać maskę. Najprostszy algorytm, wzorujący się bezpośrednio na obecnym podejściu stosowanym w szpitalu. Jednak w tym przypadku każdą warstwę opisujemy innym prostokątem, przez co wyniki powinny być dokładniejsze

2. Znalezienie otoczki wypukłej największego spójnego fragmentu maski. Alternatywnym podejściem do rozwiązania problemu jest algorytm znajdujący otoczkę wypukła dla największego spójnego elementu. Jest rozwiązaniem gdzie wyszukujemy kształt o największym polu powierzchni, a następnie liczymy tylko jego pole pomijając miejsca gdzie również została znaleziona prostata. W tym przypadku pole powierzchni jest policzone dużo dokładniej, niż w pierwszym przypadku jednak gdy maska nie jest spójna mogą być mocno zaniżone.

3. Znalezienie otoczki wypukłej dla wszystkich elementów maski. Jest to modyfikacja poprzedniego algorytmu , w której szukamy otoczki wypukłej dla wszystkich pixeli maski.

5.2.3. Analiza wyników

//todo przerobic

5.2.4. Alternatywne podejścia

Algorytmy opisane powyżej są jednymi z wielu możliwości poprawienia efektywności badania objętości prostaty w sposób analityczny. Jak widać w dużej mierze objętość opiera się o dokładność segmentacji. Wartymi do rozważenia modyfikacjami jest również

Pole powierzchni prostokąta na którym można opisać największy spójny element uzyskany w procesie segmentacji. Z uwagi na małe zróżnicowanie względem opisanych przez nas metod nie uwzględniliśmy jej w algorytmach udostępnianych przez naszą aplikację

Wybór elementów wokół których szukamy otoczki wypukłej na podstawie pewnej funkcji. Jest to zagadnienie skomplikowane, ponieważ możliwych funkcji jest bardzo dużo. Na przykład moglibyśmy szukać otoczki wokół wszystkich elementów maski znajdujących się w określonej odległości od środka największego elementu maski.

Przy poprawianiu maski uwzględniamy wszystkie warstwy. Czyli w przypadku gdy aktualna maska jest niespójna z poprzedzającą oraz następującą po sobie, wówczas zamiast niej bierzemy maskę będącą średnią dwóch wymienionych.

5.3. Moduł backedowy

Moduł zaimplementowany w .NET Framework odpowiada za komunikację z warstwą prezentacji, komunikację między resztą serwisów oraz dostępem do bazy danych. Podstawowe funkcje

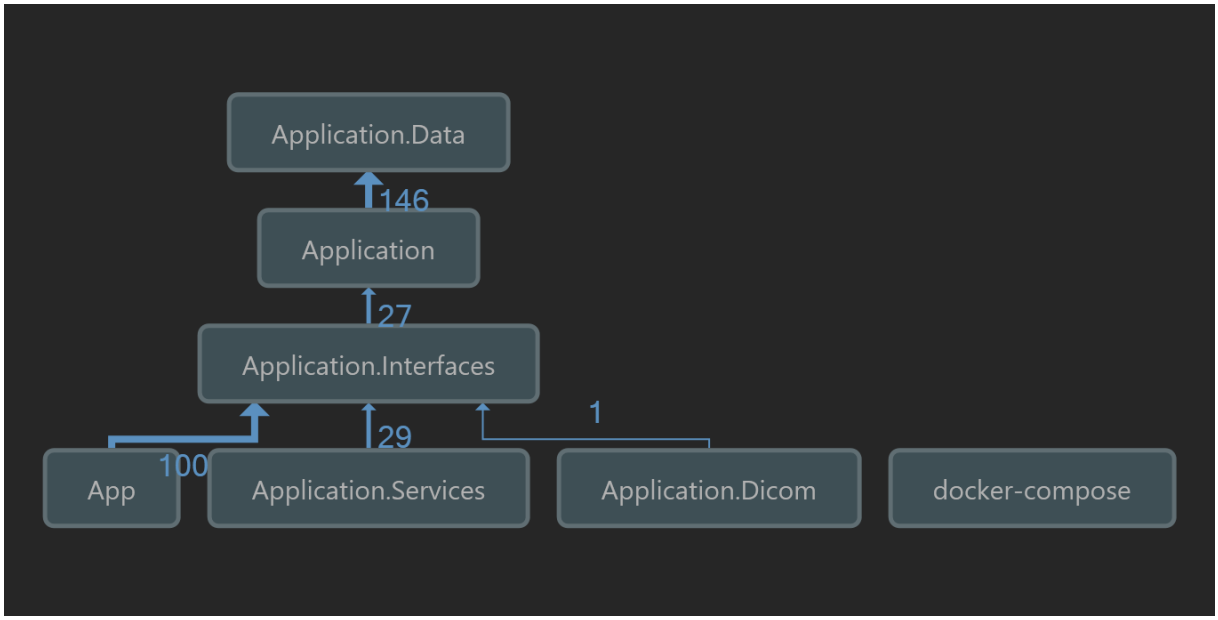
1. Udostępnia API do komunikacji z naszym modułem frontendowym oraz umożliwiający integrację z dowolnym innym systemem, który mógłby wykorzystywać nasze API. Interfejs zapewnia pełen dostęp do tabel czyli wszystkie operacje *CRUD* jakie mogą być potrzebne przy rozwoju systemu. Dodatkowo daje zapewnia komunikację użytkownika ze wszystkimi kontenerami. W tym przypadku zakres operacji jest ograniczony do minimum.

2. Obsługę bazy danych. W tym obsługę dodawania nowych pacjentów z plików oraz systemu PACS, migrację bazy danych oraz obsługę kwerend i procedur.

3. Komunikację z pozostałymi komponentami backendowymi.

Staraliśmy się w pracy zastosować wszystkie najlepsze praktyki wykożystywane przy projektowaniu oprogramowania takie jak zasady *SOLID* oraz wzorce projektowe jak *Dependency Injection*. Za spełnienie tego ostatniego w aplikacji odpowiada kontener Autofac.

5.3.1. Diagram zależności



Rysunek 5.12: Diagram zależności

Jak widać podstawowe projekty w systemie to

1. Application.Data - który jest warstwą dostępu do danych w aplikacji, definiuje encje oraz zarządza połączeniem z bazą danych.
2. Application - definiuje podstawowe obiektu współdzielone między modułami takie jak modele, wyjątki, logowanie, mapowanie obiektów.
3. Applicaition.Interfaces - projekt definiuje wszystkie kontrakty za pomocą których klasy komunikują się ze sobą
4. Applicaition.Services - zawiera implementację interfejsów zdefiniowanych w projekcie *Interfaces*.
5. Application.Dicom - odpowiada za obsługę plików DICOM. Moglibyśmy zaimplementować go w projekcie *Services* jednak z uwagi na złożoność zdecydowaliśmy umieści go w osobnym module.

Projektując architekturę rozwiązania zależało nam aby komponenty były ze sobą luźno powiązane, przez co unikniemy niepotrzebnych nad którymi trudno zapanować. Moduły podstawowe są zależne tylko do projektu *Interfaces* co zapewnia oczekiwane luźne łączenie i komunikację pomiędzy nimi. Warto zauważyć, że projekt który udostępnia nasze REST API jest zależny tylko od projektu *Interfaces*.

5.3.2. Baza danych

Plik wejściowy jest plikiem binarnym zawierającym wiele informacji o pacjencie, w przypadku aplikacji, którą tworzymy przy każdym odwołaniu do kolejnych warstw przetwarzanie pliku jest bardzo nieefektywne. Dlatego postanowiliśmy plik przetwarzać tylko raz przy dodawaniu, a przy odwołaniach zwracać dane z bazy danych.

Ponieważ według standardu DICOM [3] zawiera ponad 4 tysiące tagów. Po przeanalizowaniu plików wejściowych zauważyliśmy, że w naszym przypadku uzupełniona jest tylko część danych.

W bazie danych przechowujemy wybrane dane istotne z punktu widzenia działania aplikacji oraz potrzeb lekarza. Wybraliśmy strukturę gdzie dla jednego pacjenta informację na temat obrazów oraz danych są wspólne i nie zmieniają się dla kolejnych warstw obrazu. Dlatego jedyne informacje dotyczące obrazu są przechowywane dla każdego pliku osobno. Poniżej zamieszczono diagram encji jakie znajdują się w bazie danych.

Rysunek 5.13: Schemat bazy danych

5.4. Moduł prezentacji

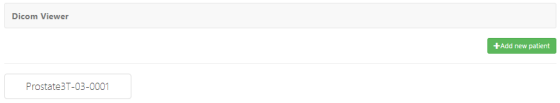
Aplikacja jest stworzona w architekturze SPA (Single Page Application). Wykorzystanie tego podejścia pozwala na wyświetlanie wielu widoków bez konieczności pobierania ponownie tych samych informacji, dzięki temu zachowując płynne działanie strony internetowej. Stusując te podejście zyskujemy na płynności działania oraz lepsze odczucia użytkownika przy korzystaniu z serwisu.

Naszą aplikację stworzyliśmy przy pomocy biblioteki React. Ułatwia ona myślenie o logice biznesowej aplikacji dzieląc stronę na komponenty. Każdy komponent składa się z co najmniej jednego HTML tagu i jest odpowiedzialny za inną część widoczną na stronie. Granulacja komponentów sprzyja pisaniu złożonych aplikacji poprzez separację stanów każdego z komponentów. Naszą aplikację stworzyliśmy przy pomocy biblioteki React. Ułatwia ona myślenie o logice biznesowej aplikacji dzieląc stronę na komponenty. Każdy komponent składa się z co najmniej jednego HTML tagu i jest odpowiedzialny za inną część widoczną na stronie. Granulacja komponentów sprzyja pisaniu złożonych aplikacji poprzez separację stanów każdego z komponentów.

5.4.1. Widok startowy

Pierwszy ekran jaki jest wyświetlany w naszej spełnia trzy podstawowe funkcjonalności.

1. Wyświetlenie listy pacjentów dostępnych w aplikacji.
2. Dodanie nowego użytkownika do bazy danych.
3. Integracja z systemem PACS.

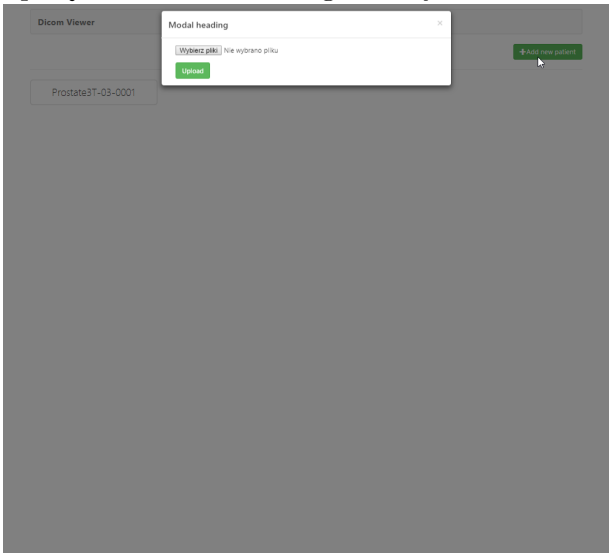


Rysunek 5.14: Ekran startowy

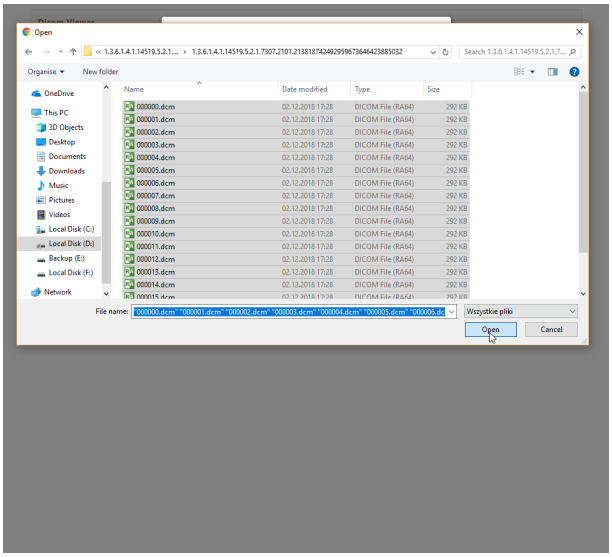
Aby dodać nowego pacjenta, należy nacisnąć przycisk *Add New Patient*.Pojawi się nowe okno w którym użytkownik powinien wybrać wszystkie pliki z danymi pacjenta, następnie nacisnąc *Upload*. Po udanym procesie należy zamknąć okienko.

Typowym scenariuszem jest dodawanie pacjenta wraz ze wszystkimi obrazami jakie mamy na jego temat. Pliki średnio ważą około 10 MB, w zależności od ilości wartstw

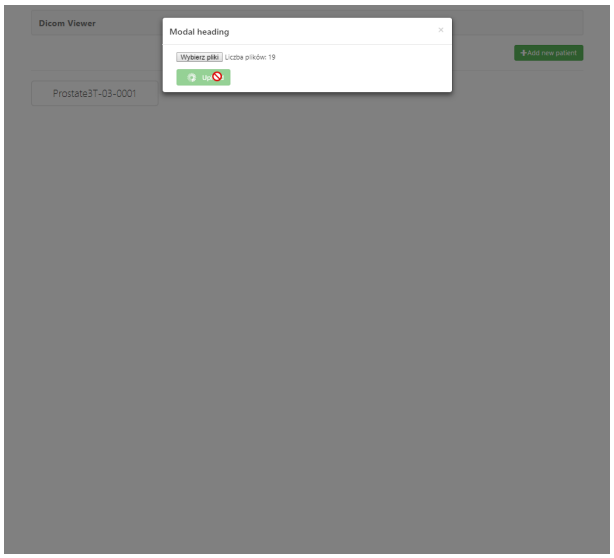
obrazu oraz szybkości połączenia internetowego czasy dodawania mogą być wydłużone.



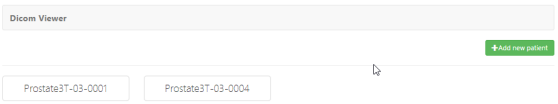
Rysunek 5.15: Dodawanie pacjenta



Rysunek 5.16: Wybór plików



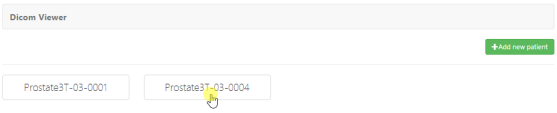
Rysunek 5.17: Proces zakończony



Rysunek 5.18: Proces zakończony

5.4.2. Widok pacjenta

Lista pacjentów wyświetla ich identyfikatory w systemie. Aby przejść do widoku wyświetlającego wszystkie informacje o pacjencie należy nacisnąć na wybranego pacjenta.



Rysunek 5.19: Przejście do widoku pacjenta

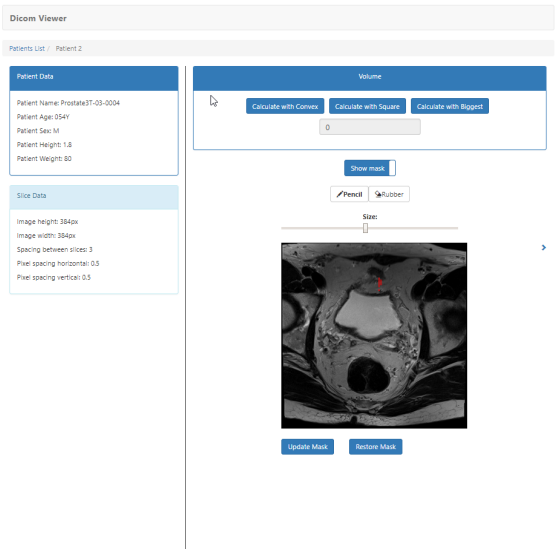
Widok pacjenta przedstawia nam wszystkie informacje jakie posiadamy na temat danego przypadku badanego. W szczególności:

Dane pacjenta z pliku DICOM, takie jak: nazwa, wiek, płeć, wzrost, waga oraz inne. Wyświetlamy tylko informacje, które w danym pliku się znajdowały. Jeżeli jakieś dane miały wartość pustą pomijamy je. Jeżeli dane podlegały wcześniejszej anonimizacji dane wrażliwe jakie zobaczymy na interfejsie użytkownika nie będą danymi odpowiadającymi tym z systemu szpitalnego.

Informacje na temat obrazu, takie jak: wymiary obrazu (szerokość i wysokość), odległość między kolejnymi obrazami w badaniu, odległości między konkretnymi pixelami na obrazie.

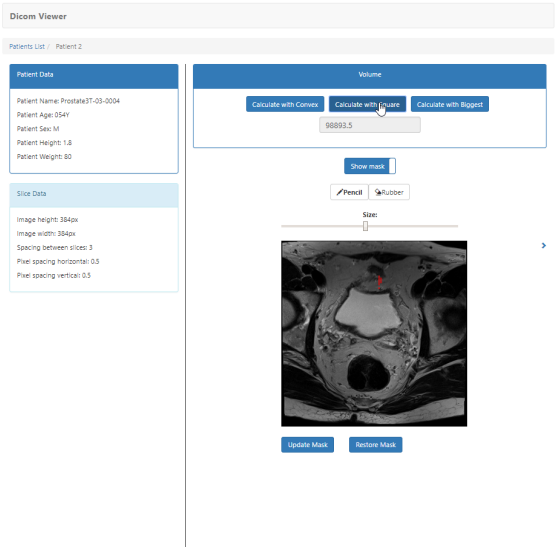
Informacje na temat objętości oraz przyciski do kalkulacji objętości według wybranego algorytmu.

Obraz z rezonansu magnetycznego oraz maskę znalezioną w procesie segmentacji wraz z możliwością edycji.



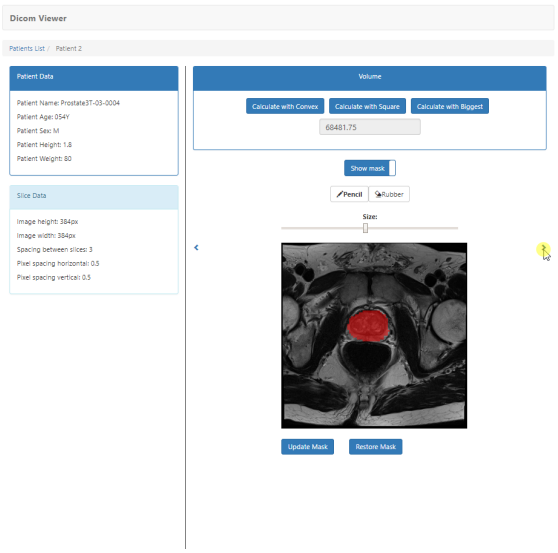
Rysunek 5.20: Widok pacjenta

Aby obliczyć objętość na podstawie masek znajdujących się obecnie w bazie danych należy nacisnąć odpowiedni przycisk znajdujący się na górze strony. Udostępniamy trzy alternatywne metody opisane w poprzednim rozdziale. Po naciśnięciu przycisku objętość jest obliczana oraz automatycznie aktualizowana na stronie tak jak i w bazie danych.



Rysunek 5.21: Kalkulacja objęctości

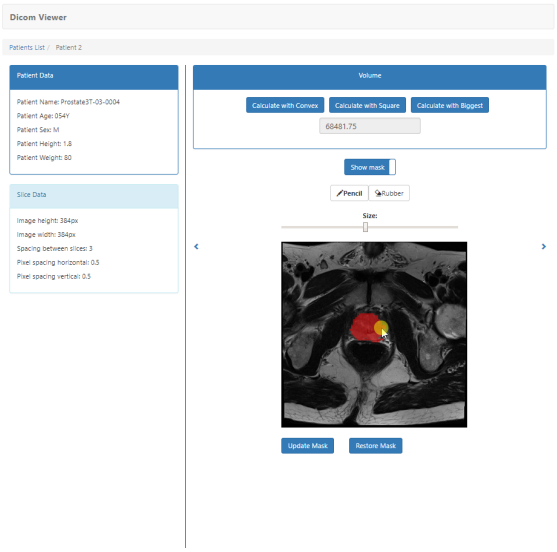
Aby przełączać się pomiędzy poszczególnymi warstwami należy nacisnąć przycisk po prawej lub lewej stronie obrazu.



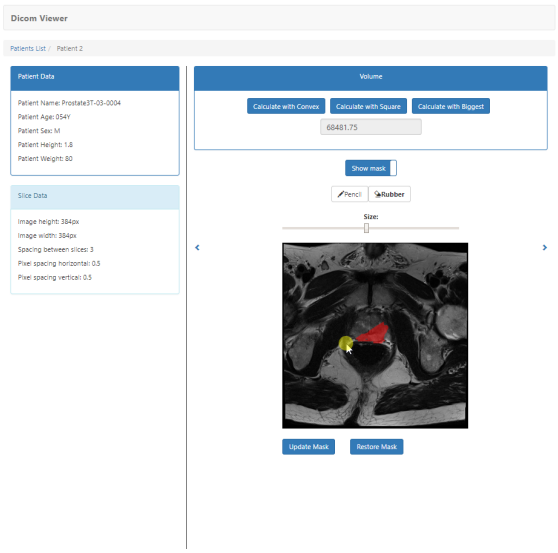
Rysunek 5.22: Przechodzenie pomiędzy kolejnymi obrazami

5.4.3. Edycja obrazu

Zapewniamy możliwość edycji masek obliczonych w procesie segmentacji. Udostępniamy opcje rysowani *Pencil* oraz usuwania *Rubber*. Po wybraniu narzędzia użytkownik może rysować po obrazie.

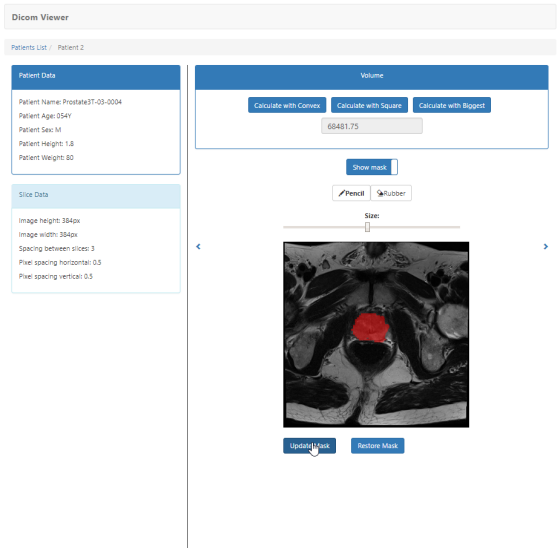


Rysunek 5.23: Rysowanie maski



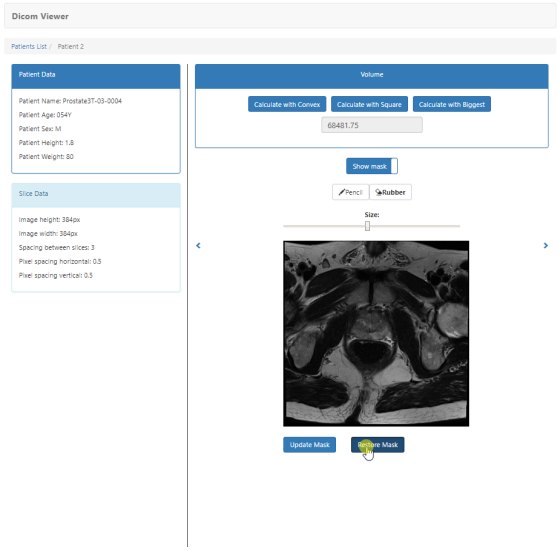
Rysunek 5.24: Usuwanie maski

Po zakończeniu edycji należy nacisnąć przycisk *Update mask*.



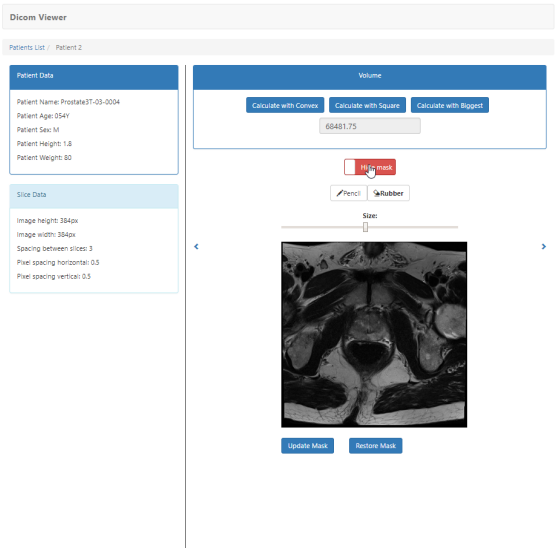
Rysunek 5.25: Aktualizacja mask

W przypadku gdy chcemy przywrócić pierwotną maskę znalezioną przez nasz algorytm segmentacji należy wybrać opcję *Restore mask*



Rysunek 5.26: Resetowanie maski

Dodatkowo zapewniamy możliwość ukrycia maski w przypadku aby lekarz mógł przeglądać oryginał zdjęcia bez przysłaniania go maską.



Rysunek 5.27: Ukryj maskę

//todo pacs

6. Podsumowanie rezultatów

Ten rozdział jest poświęcony przedstawieniu rezultatów pracy nad omawianym systemem. Omówione zostaną testy, które zostały przeprowadzone w celu zbadania, czy stworzone rozwiązanie odpowiada założeniom, przedstawionym w rozdziale *Specyfikacja*. Rozdział obejmuje także omówienie wniosków wyciągniętych podczas pracy nad tworzonym systemem.

6.1. Integracja z systemem informatycznym

Podstawowym wymaganiem dotyczącym naszej pracy jest możliwość integracji i używania go razem z obecnymi w szpitalu systemami. Podczas naszej współpracy ze szpitalem niestety doszliśmy do wniosku, że nie jest możliwa integracja z systemem używanym na co dzień w szpitalu. System ten jest napisany przez firmę Philips i nie ma w nim możliwości instalacji wtyczek, a cały kod źródłowy jest utajniony. Przez co w naszym systemie proponujemy obejście tego problemu poprzez łączenie się z siecią szpitalna przez przeglądarkę i w ten sposób pobieranie danych. Rozwiązanie umożliwiające łatwą integrację z infrastrukturą szpitalną bez rozszerzania funkcjonalnosci przegladarek DICOM. Alternatywnie można podłaczyc serwer do szpitalnego systemu PACS przy użyciu sieci VPN. Rozwiązanie przy użyciu klienta po stronie przeglądarki ma kilka zalet, takich jak na przykład redukcje kosztów po stronie obsługi IT i umożliwia wprowadzanie zmian po stronie klienta bez konieczności manualnej instalacji oprogramowania. Co ma istotne znaczenie w używanych systemach szpitalnych.

6.2. Trudności

W trakcie realizacji projektu napotkaliśmy kilka trudności. Podstawowymi było:

1. Problemy z integracją z siecią szpitalną. Wniosek jaki powstał po etapie analizy biznesowej, to, że uzyskanie odpowiednich uprawnień oraz dostępów do celu realizacji projektu jest niemożliwe. Stąd w naszej pracy integrację wykonaliśmy z lokalną instancją bazy PACS. Może to wprowadzać problemy przy realnej próbie integracji z systemem szpitalnym, gdyby okazało się, że systemy różnią się między sobą metodami dostępu w szczególności metodami udostępnianymi przez REST API.

2. Dane testowe jaki użyliśmy do uczenia sieci pochodzą z konkursu i zbiór testowy jest bardzo ograniczony. Może to powodować problemy z działaniem na realnych danych szpitalnych.

3. Nie byliśmyw stanie sprawdzić dokładności algorytmu segmentacji z danymi realnych pacjentów szpitala, z uwagi na brak dostępu do danych. W analizie wyników brakowało

nam najdokładniejszej informacji jaką jest objętość prostaty po zabiegu prostatektomii. Stąd nie jesteśmy w stanie zweryfikować wyników względem faktycznych wartości.

Trudności spowodowały rozwiązanie problemów w alternatywne sposoby niż początkowo zakładaliśmy. Jednak aby dopuścić rowiązanie do użytku produkcyjnego powinny być rozwiązane.

6.3. Wnioski

Przedstawiony system może poprawić komfort pracy lekarza radiologa. Jednak w zarówno w organizacji pracy radiologów jak i w naszym rozwiązaniu pozostaje wiele miejsca do wprowadzania usprawnień. Aplikacja powstała przy wykorzystaniu najnowszych dostępnych technologii stąd w środowisku produkcyjnym może to sprawiać problemy. Jednak wszystkie biblioteki jakich użyliśmy w pracy są stale rozwijane i zapewniają dobre wsparcie dla deweloperów.

Bibliografia

[1] *Strona internetowa Rynek Zdrowia*, http://www.rynekzdrowia.pl/Serwis-Onkologia/Eksperci-wzrasta-za

[2] *Dane konkursowe*, https://wiki.cancerimagingarchive.net/plugins/servlet/contentId=23691656/#conte

[3] *Standard DCIOM*, https://www.dicomstandard.org/

[4] *Open CV* https://opencv.org/

[5] *Przetwarzanie danych w medycynie* http://www.image-net.org/

[6] *Dane do uczenia sieci* https://wiki.cancerimagingarchive.net/display/Public/NCI-ISBI+2013+Challeng

[7] *Zgłoszenie do konkursu*, https://github.com/mirzaevinom/prostate_segmentation

[8] *Serwer PACS* https://www.orthanc-server.com/

Wykaz symboli i skrótów

HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
np.	Na przykład
tzw.	Tak zwany

Spis rysunków

Spis tabel

Spis załączników

1. Załącznik 1 - Opis API

6.4. Załącznik 1 - Opis API

API

Prostate segmentation rest API

api

Folder for api

GET http:///api/Dicom

http:///api/Dicom

HEADERS

Accept

text/plain, application/json, text/json

Example Request

http:///api/Dicom

```
curl --request GET \
  --url http:/api/Dicom \
  --header 'Accept: text/plain, application/json, text/json'
```

POST http:///api/Dicom

http:///api/Dicom

HEADERS

Content-Type

application/json-patch+json

Example Request

http:///api/Dicom

```
curl --request POST \
  --url http:/api/Dicom \
  --header 'Content-Type: application/json-patch+json'
```

GET http:///api/Dicom/:id

http:///api/Dicom/:id

HEADERS

Accept

text/plain, application/json, text/json

PATH VARIABLES

id

{{id}}

Example Request

http:///api/Dicom/:id

```
curl --request GET \
  --url http:/api/Dicom/:id \
  --header 'Accept: text/plain, application/json, text/json'
```

PUT http:///api/Dicom/:id

http:///api/Dicom/:id

HEADERS

Content-Type

application/json-patch+json

PATH VARIABLES

id

{{id}}

Example Request

http:///api/Dicom/:id

```
curl --request PUT \
  --url http:/api/Dicom/:id \
  --header 'Content-Type: application/json-patch+json'
```

DEL http:///api/Dicom/:id

http:///api/Dicom/:id

PATH VARIABLES

id

{{id}}

Example Request

http:///api/Dicom/:id

```
curl --request DELETE \
  --url http:/api/Dicom/:id
```

GET http:///api/Dicom/Indexes/:id

http:///api/Dicom/Indexes/:id

HEADERS

Accept

text/plain, application/json, text/json

PATH VARIABLES

id

{{id}}

Example Request

http:///api/Dicom/Indexes/:id

```
curl --request GET \
  --url http:/api/Dicom/Indexes/:id \
  --header 'Accept: text/plain, application/json, text/json'
```

GET http:///api/Dicom/File/:id

http:///api/Dicom/File/:id

HEADERS

Accept

text/plain, application/json, text/json

PATH VARIABLES

id

{{id}}

Example Request

http:///api/Dicom/File/:id

```
curl --request GET \
  --url http:/api/Dicom/File/:id \
  --header 'Accept: text/plain, application/json, text/json'
```

GET http:///api/Image/:dicomId

http:///api/Image/:dicomId

HEADERS

Accept

text/plain, application/json, text/json

PATH VARIABLES

dicomId

{{dicomId}}

Example Request

http:///api/Image/:dicomId

```
curl --request GET \
  --url http:/api/Image/:dicomId \
  --header 'Accept: text/plain, application/json, text/json'
```

GET http:///api/Image/:dicomId&:sliceId

http:///api/Image/:dicomId&:sliceId

HEADERS

Accept

text/plain, application/json, text/json

PATH VARIABLES

dicomId

{{dicomId}}

sliceId

{{sliceId}}

Example Request

http:///api/Image/:dicomId&:sliceId

```
curl --request GET \
  --url 'http:/api/Image/:dicomId&:sliceId' \
  --header 'Accept: text/plain, application/json, text/json'
```

PUT http:///api/Image/:dicomId&:sliceId

http:///api/Image/:dicomId&:sliceId

HEADERS

Content-Type

application/json-patch+json

PATH VARIABLES

dicomId

{{dicomId}}

sliceId

{{sliceId}}

Example Request

http:///api/Image/:dicomId&:sliceId

```
curl --request PUT \
  --url 'http:/api/Image/:dicomId&:sliceId' \
  --header 'Content-Type: application/json-patch+json'
```

DEL http:///api/Image/:dicomId&:sliceId

http:///api/Image/:dicomId&:sliceId

PATH VARIABLES

dicomId

{{dicomId}}

sliceId

{{sliceId}}

Example Request

http:///api/Image/:dicomId&:sliceId

```
curl --request DELETE \
  --url 'http:/api/Image/:dicomId&:sliceId'
```

GET http:///api/Mask/:dicomId

http:///api/Mask/:dicomId

HEADERS

Accept

text/plain, application/json, text/json

PATH VARIABLES

dicomId

{{dicomId}}

Example Request

http:///api/Mask/:dicomId

```
curl --request GET \
  --url http:/api/Mask/:dicomId \
  --header 'Accept: text/plain, application/json, text/json'
```

GET http:///api/Mask/:dicomId&:sliceId

http:///api/Mask/:dicomId&:sliceId

HEADERS

Accept

text/plain, application/json, text/json

PATH VARIABLES

dicomId

{{dicomId}}

sliceId

{{sliceId}}

Example Request

http:///api/Mask/:dicomId&:sliceId

```
curl --request GET \
  --url 'http:/api/Mask/:dicomId&:sliceId' \
  --header 'Accept: text/plain, application/json, text/json'
```

PUT http:///api/Mask/:dicomId&:sliceId

http:///api/Mask/:dicomId&:sliceId

HEADERS

Content-Type

application/json-patch+json

PATH VARIABLES

dicomId

{{dicomId}}

sliceId

{{sliceId}}

Example Request

http:///api/Mask/:dicomId&:sliceId

```
curl --request PUT \
  --url 'http:/api/Mask/:dicomId&:sliceId' \
  --header 'Content-Type: application/json-patch+json'
```

DEL http:///api/Mask/:dicomId&:sliceId

http:///api/Mask/:dicomId&:sliceId

PATH VARIABLES

dicomId

{{dicomId}}

sliceId

{{sliceId}}

Example Request

http:///api/Mask/:dicomId&:sliceId

```
curl --request DELETE \
  --url 'http:/api/Mask/:dicomId&:sliceId'
```

POST http:///api/Mask/Recalculate/:dicomId&:sliceId

http:///api/Mask/Recalculate/:dicomId&:sliceId

PATH VARIABLES

dicomId

{{dicomId}}

sliceId

{{sliceId}}

Example Request

http:///api/Mask/Recalculate/:dicomId&:sliceId

```
curl --request POST \
  --url 'http:/api/Mask/Recalculate/:dicomId&:sliceId'
```

POST http:///api/NewDicom

http:///api/NewDicom

HEADERS

Accept

text/plain, application/json, text/json

Content-Type

application/json-patch+json

Example Request

http:///api/NewDicom

```
curl --request POST \
  --url http:/api/NewDicom \
  --header 'Accept: text/plain, application/json, text/json' \
  --header 'Content-Type: application/json-patch+json'
```

POST http:///api/NewDicom/:id

http:///api/NewDicom/:id

HEADERS

Content-Type

application/json-patch+json

PATH VARIABLES

id

{{id}}

Example Request

http:///api/NewDicom/:id

```
curl --request POST \
  --url http:/api/NewDicom/:id \
  --header 'Content-Type: application/json-patch+json'
```

POST http:///api/NewDicom/UploadList

http:///api/NewDicom/UploadList

HEADERS

Accept

text/plain, application/json, text/json

Content-Type

application/json-patch+json

Example Request

http:///api/NewDicom/UploadList

```
curl --request POST \
  --url http:/api/NewDicom/UploadList \
  --header 'Accept: text/plain, application/json, text/json' \
  --header 'Content-Type: application/json-patch+json'
```

GET http:///api/PatientData

http:///api/PatientData

HEADERS

Accept

text/plain, application/json, text/json

Example Request

http:///api/PatientData

```
curl --request GET \
  --url http:/api/PatientData \
  --header 'Accept: text/plain, application/json, text/json'
```

GET http:///api/PatientData/:dicomId

http:///api/PatientData/:dicomId

HEADERS

Accept

text/plain, application/json, text/json

PATH VARIABLES

dicomId

{{dicomId}}

Example Request

http:///api/PatientData/:dicomId

```
curl --request GET \
  --url http:/api/PatientData/:dicomId \
  --header 'Accept: text/plain, application/json, text/json'
```

PUT http:///api/PatientData/:dicomId

http:///api/PatientData/:dicomId

HEADERS

Content-Type

application/json-patch+json

PATH VARIABLES

dicomId

{{dicomId}}

Example Request

http:///api/PatientData/:dicomId

```
curl --request PUT \
  --url http:/api/PatientData/:dicomId \
  --header 'Content-Type: application/json-patch+json'
```

POST http:///api/PatientData/:dicomId

http:///api/PatientData/:dicomId

HEADERS

Content-Type

application/json-patch+json

PATH VARIABLES

dicomId

{{dicomId}}

Example Request

http:///api/PatientData/:dicomId

```
curl --request POST \
  --url http:/api/PatientData/:dicomId \
  --header 'Content-Type: application/json-patch+json'
```

DEL http:///api/PatientData/:dicomId

http:///api/PatientData/:dicomId

PATH VARIABLES

dicomId

{{dicomId}}

Example Request

http:///api/PatientData/:dicomId

```
curl --request DELETE \
  --url http:/api/PatientData/:dicomId
```

GET http:///api/Slice/:dicomId

http:///api/Slice/:dicomId

HEADERS

Accept

text/plain, application/json, text/json

PATH VARIABLES

dicomId

{{dicomId}}

Example Request

http:///api/Slice/:dicomId

```
curl --request GET \
  --url http:/api/Slice/:dicomId \
  --header 'Accept: text/plain, application/json, text/json'
```

GET http:///api/Slice/:dicomId&:sliceId

http:///api/Slice/:dicomId&:sliceId

HEADERS

Accept

text/plain, application/json, text/json

PATH VARIABLES

dicomId

{{dicomId}}

sliceId

{{sliceId}}

Example Request

http:///api/Slice/:dicomId&:sliceId

```
curl --request GET \
  --url 'http:/api/Slice/:dicomId&:sliceId' \
  --header 'Accept: text/plain, application/json, text/json'
```

POST http:///api/Slice/:dicomId&:sliceId

http:///api/Slice/:dicomId&:sliceId

HEADERS

Content-Type

application/json-patch+json

PATH VARIABLES

dicomId

{{dicomId}}

sliceId

{{sliceId}}

Example Request

http:///api/Slice/:dicomId&:sliceId

```
curl --request POST \
  --url 'http:/api/Slice/:dicomId&:sliceId' \
  --header 'Content-Type: application/json-patch+json'
```

DEL http:///api/Slice/:dicomId&:sliceId

http:///api/Slice/:dicomId&:sliceId

PATH VARIABLES

dicomId

{{dicomId}}

sliceId

{{sliceId}}

Example Request

http:///api/Slice/:dicomId&:sliceId

```
curl --request DELETE \
  --url 'http:/api/Slice/:dicomId&:sliceId'
```

PUT http:///api/Slice/:id

http:///api/Slice/:id

HEADERS

Content-Type

application/json-patch+json

PATH VARIABLES

id

{{id}}

Example Request

http:///api/Slice/:id

```
curl --request PUT \
  --url http:/api/Slice/:id \
  --header 'Content-Type: application/json-patch+json'
```

GET http:///api/Volume/Calculate/:id?dicomId={{dicomId}}

http:///api/Volume/Calculate/:id?dicomId={{dicomId}}

HEADERS

Accept

text/plain, application/json, text/json

PARAMS

dicomId

{{dicomId}}

PATH VARIABLES

id

{{id}}

Example Request

http:///api/Volume/Calculate/:id?dicomId={{dicomId}}

```
curl --request GET \
  --url 'http:/api/Volume/Calculate/:id?dicomId={{dicomId}}' \
  --header 'Accept: text/plain, application/json, text/json'
```

GET http:///api/Volume/:id?dicomId={{dicomId}}

http:///api/Volume/:id?dicomId={{dicomId}}

HEADERS

Accept

text/plain, application/json, text/json

PARAMS

dicomId

{{dicomId}}

PATH VARIABLES

id

{{id}}

Example Request

http:///api/Volume/:id?dicomId={{dicomId}}

```
curl --request GET \
  --url 'http:/api/Volume/:id?dicomId={{dicomId}}' \
  --header 'Accept: text/plain, application/json, text/json'
```