САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

БУЕВ ПАВЕЛ ИВАНОВИЧ

Выпускная квалификационная работа

Разработка системы для разметки медицинских данных DICOM-формата

Уровень образования магистратура

Направление 01.04.02 «Прикладная математика и информатика»

Основная образовательная программа: ВМ.5691 «Прикладная математика и информатика в задачах медицинской диагностики»

Научный руководитель:
доктор физико-математических наук,
профессор, зав. кафедрой
диагностики функциональных систем
Котина Елена Дмитриевна

Санкт-Петербург

2023 год

Содержание

Введение

Стремительное развитие компьютерных технологий влияет на различные сферы науки и производства. Рассмотрим медицинскую отрасль, а именно – медицинские изображения. Современные радиологические центры или клинические отделения оснащены множеством цифровых устройств. Такая модернизация отделений в первую очередь обязана современным процедурам компьютерной томографии и магнитно-резонансной томографии [9]. Очевидно, что многочисленные аппараты для диагностики во всем мире генерируют множество медицинских изображений и документов, которые требуют систематизации.

Для решения этой задачи был разработан стандарт создания, хранения, передачи и визуализации цифровых медицинских изображений и документов обследованных пациентов – DICOM (Digital Imaging and Communications in Medicine). Медицинские изображения, использующие стандарт DICOM, называются DICOM изображениями.

Аппараты диагностики и рабочие электронно-вычислительные машины обмениваются информацией посредством PACS систем (Picture Archiving and Communication System). Такие системы передачи и архивации DICOM изображений предполагают создание специальных удалённых архивов на DICOM серверах, где хранятся DICOM архивы, которые доступны для поиска и просмотра с рабочих машин.

Для взаимодействия с прикладными WEB-приложениями был разработан стандарт DICOMweb, который использует привычные для приложений механизмы, такие как HTTP (HyperText Transfer Protocol – сетевой протокол передачи данных), JSON (JavaScript Object Notation – текстовый формат структурированных данных, основанный на JavaScript) и типы мультимедиа (например, "image/jpeg").

В связи с развитием машинного обучения стали появляться задачи, связанные с анализом медицинских данных, в том числе и исследований в формате DICOM. Сегментация изображений является одной из важных задач в анализе изображений, т.к. это один из первых шагов в распознавании изображений, и последующие шаги, такие как извлечение образов или классификация значительно зависят от результатов сегментации [20]. Подобные задачи рассматриваются в работах многих авторов, например, задачу сегментации медицинского изображения – автоматическое выделение легких на снимках КТ – решает автор работы [3]. Так же в [8] используются снимки МРТ сердца для автоматической сегментации левого желудочка и расчета фракции выброса. В статьях [2] и [4] медицинские изображения используются для тренировки сверточной нейронной сети типа U-Net, и последующего установления наличия заболевания. Одним из типов подобных задач является анализ предварительно размеченных контурами участков изображения. Такая задача может возникнуть, например, при выявлении корреляции диагноза и сегмента изображения, на котором присутствует конкретный орган. На данный момент присутствует дефицит инструментов, которые предоставляли бы инструментарий PACS системы и оконтуривания DICOM изображений, с возможностью экспорта данных и последующего редактирования контуров, сохраненных в системе. Это позволяет утверждать, что разработка такого инструмента является актуальной.

В данной работе планируется рассмотреть технологии DICOM, PACS и DICOMweb, а также разработать инструмент для работы с изображениями в DICOM формате с функционалом разметки изображений.

Постановка задачи

Цель данной работы – создание системы для разметки медицинских данных DICOM-формата. Создаваемая система должна поддерживать стандарт DICOMweb, предоставлять функционал разметки сегментов на медицинских изображениях, экспорта DICOM файлов и данных о разметке в стандартизированном виде.

Для достижения цели были поставлены следующие задачи:

1. Изучить существующие программные решения. Провести анализ функциональности. Выбрать заимствуемые функции;
2. Сформулировать требования к разрабатываемой системе. Определить необходимую функциональность;
3. Спроектировать программное обеспечение;
4. Реализовать программное обеспечение.

Обзор литературы

Задача разработки инструмента для разметки медицинских данных не нова. Существуют различные инструменты, которые так или иначе выполняют функцию разметки данных. Например, в сервисе CoreSlicer, описанном в статье [11], можно обводить участки изображения, и экспортировать обведенные участки, однако они сохраняются в растровом формате png, что затрудняет работу с оригинальным изображением. Другой инструмент, просмотрщик изображений medDream [7] от компании Softneta, позволяет создавать контуры в видео геометрических фигур и свободные контуры, рисуемые от руки, но не позволяет их экспортировать. Проект OHIF Viewer [14] от компании Open Health Imaging Foundation имеет большое разнообразие инструментов, в т.ч. разметку изображения контурами и позволяет экспортировать эти контуры в формате csv, но имеет ограничения: контуры могут быть лишь в видео геометрических фигур – эллипса и прямоугольника. Однако этот проект имеет большое разнообразие инструментов и находится в

открытом доступе, т.е. имеет открытый исходный код, благодаря чему на его основе можно разработать собственное решение.

Для написания этой работы были использованы статьи, описывающие назначение, использование и историю создания стандартов DICOM и DICOMweb [1, 12], а также сам текст стандартов [16]. Полный текст стандарта DICOM был использован по большей части во время разработки системы, т.к. в нем описываются принципы работы DICOM файла – поля исследования, типы данных в этих полях, а также принципы работы веб-сервисов DICOMweb – что должны из себя представлять HTTP запросы к серверу и как должны выглядеть ответы сервера. Также для проектирования разрабатываемой ифнормационной системы были использованы источники, описывающие работу, архитектуру и разработку PACS-систем [15, 18]. В исследованиях [2–4, 8], которые подтверждают актуальность работы, описываются задачи, связанные с обработкой медицинских изображений посредством нейронных сетей. Литература о разработке информационных систем и сопутствующих инструментах [5, 6, 10, 17, 19] была использована во время работы над третьей главой.

Глава 1. Технологии цифровизации медицинских данных

1.1 Стандарт DICOM

Когда в 1970-х годах началась разработка цифровых компьютеров и оборудования, идея использования оборудования для получения цифровых изображений в медицине стала реальностью. ACR (Американский колледж радиологии) и NEMA (Национальная ассоциация производителей электрооборудования) сформировала в 1983 году совместный комитет, задачей которого была разработка стандарта для подключения дисплеев и аналогичных устройств к медицинскому оборудованию для визуализации различных производителей. Первая версия под названием ACR/NEMA Standard Version 1.0 была опубликована в 1985 году.

В 1988 г. в новую редакцию стандарта был включен новый материал, и была создана версия 2.0. Обе версии 1.0 и 2.0 допускали соединение точка-точка, что представляло проблему для современных сетей связи, в которых не используются абсолютно выделенные каналы. Из-за этого пришлось представить новую редакцию стандарта. В 1993 году была выпущена новая версия стандарта под названием DICOM Version 3.0, который описывает использование сетевой среды, вместо соединения точка-точка. Комитет по стандартам DICOM в настоящее время создает и поддерживает международные стандарты для обмена цифровыми медицинскими изображениями и связанными с ними данными. Он стал ведущим стандартом, используемым всеми основными поставщиками диагностического медицинского оборудования [12].

Сегодня DICOM позволяет эффективно хранить и передавать медицинские изображения на большие географические территории. составляющие основу для систем архивирования и передачи изображений (PACS).

Каждый файл DICOM разработан как автономный - вся информация, необходимая для идентификации файла, встроена в каждый заголовок. Эта информация разделена на 4 уровня иерархии - пациент, исследование, серия и экземпляр. «Пациент» (Patient) – это человек, проходящий обследование. «Исследование» (Study) – это процедура диагностики, выполняемая в определенный день и время в медицинском учреждении. «Серия» (Series) – каждое исследование состоит из нескольких серий. Серии могут представлять пациента, которого физически сканируют несколько раз в одном исследовании, или могут быть виртуальными, когда пациента сканируют один раз и данные реконструируются разными способами. «Экземпляр» (Instance) – каждый фрагмент трехмерного изображения рассматривается как отдельный экземпляр. В этом контексте «экземпляр» является синонимом самого файла DICOM.

1.2 Стандарт DICOMweb

Стандарт DICOM, выпущенный впервые в 1985 году, претерпел множество изменений и был принят во всем мире. Требования к сетевым технологиям в IT индустрии росли, что требовало развития стандарта. Устройства, которые используют врачи, изменились: от ограниченного использования выделенных рабочих станций до свободного использования персональных компьютеров и мобильных устройств. Изменился способ доступа к информации: теперь он не ограничен внутри медицинского учреждения, а используется в масштабах всего предприятия или между предприятиями. Эти изменения потребовали новых требований к безопасности для работы с DICOM.

Начиная с 2003 года, начало формироваться расширение DICOM с поддержкой веб-доступа. Первоначально оно называлось WADO (Web Access to DICOM Objects) – веб-доступ к объектам DICOM. Это расширение определил метод получения объектов DICOM с использованием HTTP. Со

временем в DICOM были добавлены другие службы для расширения возможностей использования извлечения, такие как QIDO (Query based on ID for DICOM Objects) – поиск DICOM объектов и STOW (Store Over the Web) – сохранение DICOM объектов на сервере.

В совокупности эти технологии стали называться DICOMweb. DICOMweb добавляет в DICOM сервисы с поддержкой HTTP, смоделированные по стилю передачи репрезентативного состояния (REST) [1]. REST стал доминирующим стилем API, используемым компаниями и рынками по всему миру. REST позволяет создавать масштабируемые, отказоустойчивые, восстанавливаемые, безопасные и слабосвязанные системы. DICOMweb был разработан для дополнения веб-технологиями уже существующих систем работы с DICOM. Это важно, потому что, учитывая проникновение систем с поддержкой DICOM по всему миру сотнями компаний, производящих триллионы изображений DICOM, невозможно без сбоев перейти на совершенно другой стандарт. DICOMweb позволяет преобразовывать экземпляры DICOM в широко используемые удобные для потребителей форматы, подходящие для веб-браузеров как на настольных компьютерах, так и на мобильных устройствах, без необходимости массового преобразования существующих данных или информационной модели.

Глава 2. Средства разработки

2.1 Обоснование выбора и описание средств

Для обеспечения доступности разрабатываемой системы была выбрана среда веб-разработки. Создавая именно веб-приложение, а не концентрируясь на определенной платформе, обеспечивается доступность приложения с большего количества устройств. Используя архитектурный стиль REST разработку веб-приложений можно разделить на два больших раздела – на разработку программно-аппаратной части сервиса – бэкенд (серверная часть) и разработку клиентской стороны пользовательского интерфейса к программно-аппаратной части сервиса – фронтенд (клиентская часть). Эти термины появились в программной инженерии вследствие развития принципа разделения ответственности между внешним представлением и внутренней реализацией. Бэкенд создает некоторое API (программный интерфейс приложения), которое использует фронтенд.

Перед разработкой приложения с REST архитектурой необходимо определиться с выбором языков и фреймворков. Фронтенд часть приложения была разработана на языке JavaScript с использованием фреймворка React. Фреймворк – это заготовки, шаблоны для программной платформы, определяющие структуру программной системы. React является одним из самых используемых фронтенд фреймворков, разрабатываемый компанией Meta и сообществом отдельных разработчиков и корпораций. Серверная часть приложения была разработана на языке программирования Go. Go, или Golang, — это строго типизированный компилируемый многопоточный язык с открытым исходным кодом. Это высокопроизводительный язык общего назначения. Его часто используют для высоконагруженного ПО, исследования данных или задач, связанных с системным программированием.

В качестве СУБД была использована PostgreSQL. Эта СУБД работает с языком управлениями данных SQL. Сильными сторонами PostgreSQL являются высокая скорость и надёжность транзакций [17].

2.2 Дополнительные средства

Для хранения файлов проекта используется система контроля версий git. Система контроля версий помогает отслеживать изменения, внесённые в базу кода [6]. Более того, она записывает, кто внёс изменения и может восстановить стёртый или изменённый код. Сама разработанная система распространяется по лицензии MIT, что позволяет безвозмездно использовать эту систему без ограничений.

Во множестве современных проектов по разработке веб-приложений используется ПО для автоматизации развертывания, запуска и управления приложениями. Docker является одним из таковых. С помощью Docker можно «упаковать» приложение со всем его окружением и зависимостями в некую изолированную среду – контейнер, которая может быть перенесена на любую другую машину, поддерживающую систему Docker [10]. Использование такого ПО для виртуализации позволяет создать одинаковую рабочую среду приложения на всех машинах разработчиков и разворачивать приложение на сервере с помощью всего лишь нескольких команд.

В качестве веб-сервера используется Nginx – веб-сервер и почтовый прокси-сервер, работающий на Unix-подобных операционных системах. В проекте он используется как в качестве веб-сервера с отдельным Docker контейнером для обслуживания статических файлов, так и как обратный прокси-сервер для доступа к динамическим ресурсам бэкенд приложения.

Тестирование результатов работы информационной системы будет осуществляться, используя возможный сценарий использования системы – подготовка данных для дальнейшего анализа с помощью ИИ. Задачи, связанные с таким анализом, довольно часто выполняются на языке программирования Python, т.к. этот язык обладает довольно простым синтаксисом, что позволяет больше сосредоточиться на решении задачи. Таким образом вокруг языка Python построилось сообщество, разрабатывающее различные библиотеки, например NumPy, scikit-learn,

Pandas помогающие решить задачи манипуляции и анализа данных [13]. Именно поэтому для тестирования работы системы будет создана программа на языке Python, демонстрирующая как работать с результирующими данными.

Глава 3. Разработка информационной системы

3.1 Проектирование информационной системы

Проектирование системы было начато с выделения основных функциональных блоков. Архитектура PACS-систем может быть совершенно разной. Например, автор статьи [18] выделяет следующие основные функции: БД, форма для просмотра, ввода и редактирования информации, система отчетов для разбора и предоставления в вышестоящие организации и программы просмотра снимков. А в книге [15] в 13 главе, полностью посвященной архитектуре PACS, помимо прочих, описываются такие блоки, как HL7 сервер, оперирующий медицинскими данными, Modality Worklist сервер, блоки архивирования данных и другие. Проанализировав литературу и архитектуру уже существующих медицинских информационных систем (ИС), был выделен требуемый функционал разрабатываемой системы:

- Загрузка и хранение DICOM объектов в хранилище;
- Поиск DICOM объектов по атрибутам;
- Получение DICOM объектов;
- Просмотр DICOM объектов;
- Разметка DICOM изображений;
- Вывод данных DICOM и разметки из системы.

Первые три пункта образуют функционал, являющимся базовым для PACS систем. Используя стандарт DICOMweb эти пункты должны быть реализованы с использованием спецификаций STOW, WADO и QIDO соответственно. Эти спецификации описаны в стандарте DICOM PS3.18 в главах 10.5, 10.4 и 10.6 и доступны на веб-странице [16].

Для определения отношений между основными объектами системы – медицинскими исследованиями – была спроектирована с использованием языка UML [5] диаграмма классов. Диаграмма классов – статическая структурная диаграмма, описывающая структуру системы, демонстрирующая классы системы, их атрибуты, методы и зависимости между классами. В

проектируемой системе этот тип диаграмм использовался для моделирования классов моделей, работающих с сущностями базы данных. Диаграмма классов моделей изображена на рисунке 1.

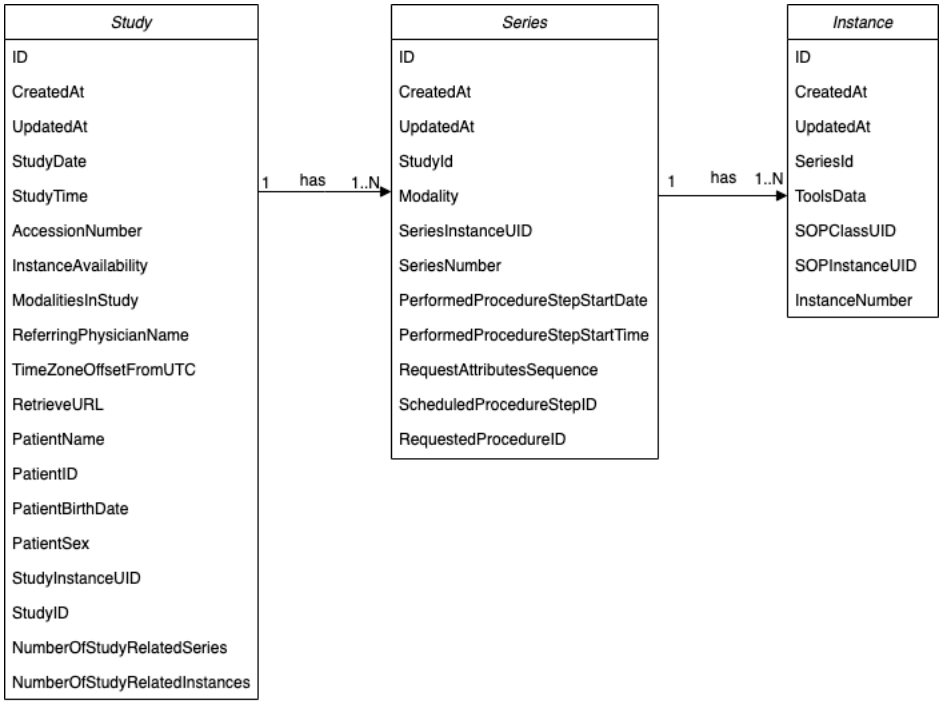

Рисунок 1 – Диаграмма классов.

На диаграмме видно, что у моделей присутствуют некоторые поля, идентичные атрибутам самих DICOM файлов. Это дублирование необходимо для быстрого доступа к ним в базе данных, т.к. в DICOMweb описывается работа именно с этими полями.

Диаграмма компонентов (component diagram) – статическая структурная диаграмма, показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами. В качестве компонентов могут выступать файлы, библиотеки, программные модули и т. п. Диаграмма компонентов разрабатываемой системы, изображенная на рисунке 2, позволяет увидеть, как пользователь и серверные процессы взаимодействуют между собой.

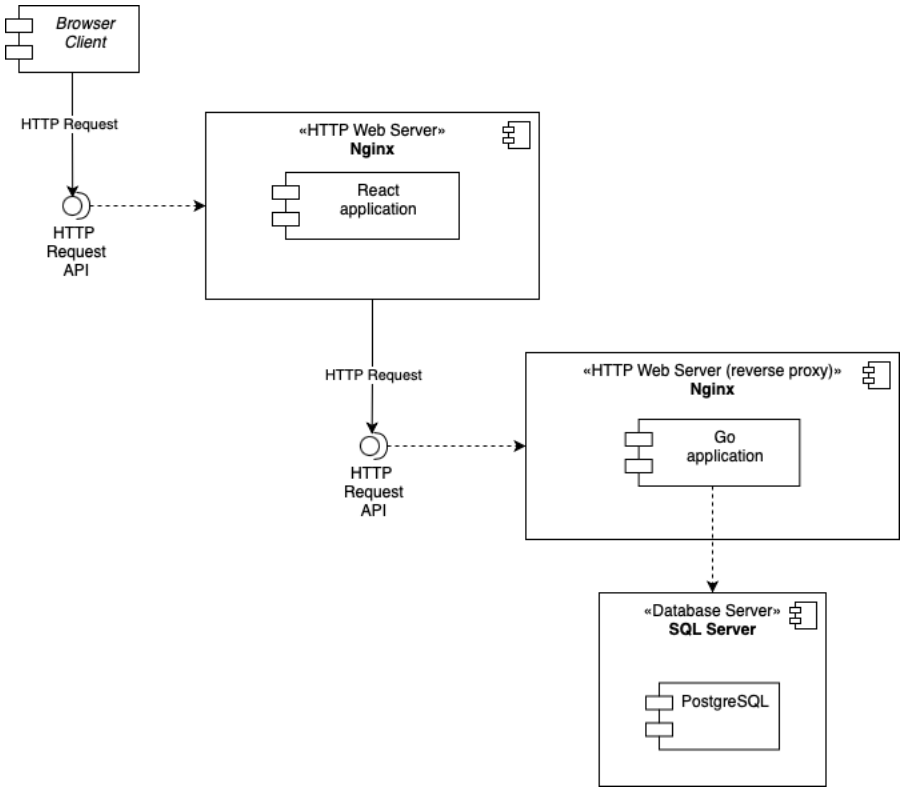

Рисунок 2 – Диаграмма компонентов

Диаграмма развертывания (deployment diagram, диаграмма размещения) – служит для моделирования работающих узлов (аппаратных средств) и артефактов, развёрнутых на них. На рисунке 3 изображена диаграмма развертывания приложения, отображающая два основных узла: клиент (веб-браузер) и система, состоящая из JavaScript и Go приложений, развернутых с помощью системы виртуализации Docker.

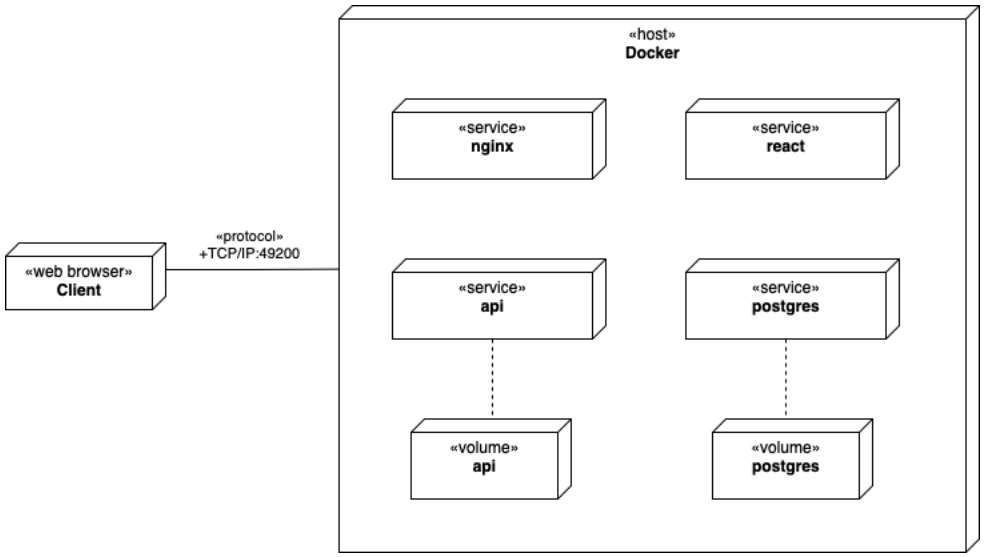

Рисунок 3 – Диаграмма развертывания

Как видно из диаграммы конечному пользователю не потребуется много усилий, чтобы установить систему, т.к. все сетевое взаимодействие и взаимодействие с файловой системой берет на себя система контейнеризации Docker. Конфигурационный файл, описывающий взаимосвязь Docker контейнеров, присутствует в приложении А.

3.2 Выбор архитектуры информационной системы

Архитектура информационной системы – концепция, определяющая модель, структуру, выполняемые функции и взаимосвязь компонентов информационной системы.

Многоуровневая архитектура клиент-сервер – разновидность архитектуры клиент-сервер, в которой функция обработки данных вынесена на один или несколько отдельных серверов. Это позволяет разделить функции хранения, обработки и представления данных для более эффективного использования возможностей серверов и клиентов [19].

Среди многоуровневой архитектуры клиент-сервер наиболее распространена трехуровневая архитектура предполагающая наличие следующих компонентов приложения: клиентское приложение (также возможны названия терминал или «тонкий клиент»), подключенное к серверу приложений, который в свою очередь подключен к серверу или серверам баз данных. Схематичное изображение трехуровневой архитектуры ТС представлено на рисунке 4.

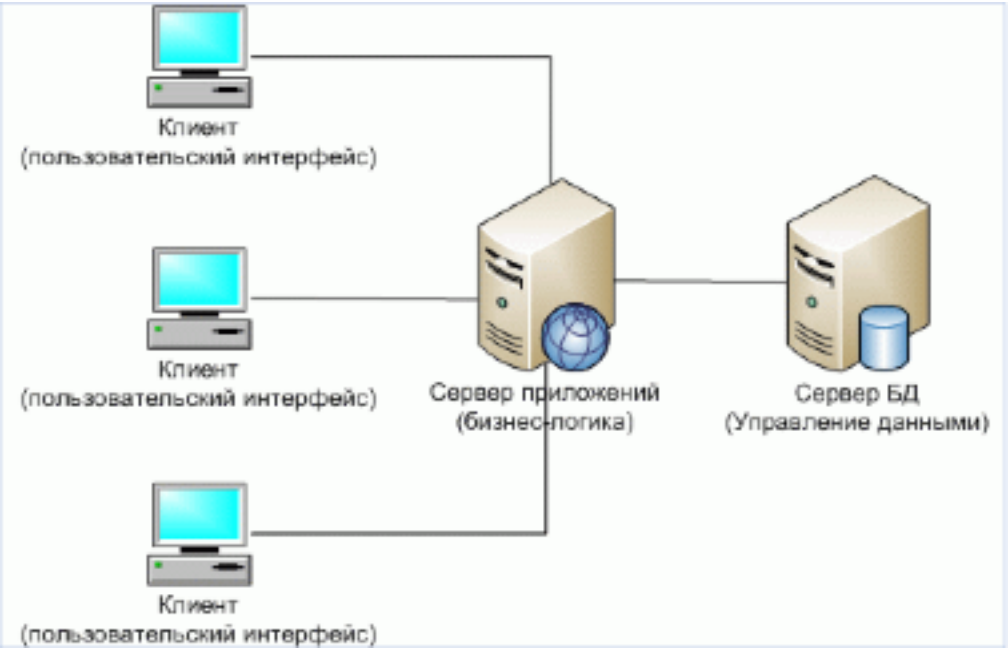

Рисунок 4 – Схема архитектуры «клиент-сервер»

Причины выбора данной архитектуры:

- Обработка данных на сервере позволяет обеспечить большую надежность данных. Вмешаться в работу системы на клиенте куда легче, чем на сервере.
- Использование тонкого клиента повысит скорость доступа к ресурсам приложения, что обеспечит большее удобство пользования.
- Такая модель архитектуры традиционно используется в разработке веб-приложений.

Принцип разделения работы приложения на уровни (уровень представления, уровень бизнес-логики и уровень управления данными) предоставляет большую гибкость при разработке новых компонентов, а также большую надежность при отказе одного из уровней.

3.3 Разработка клиентской части приложения

Одна из целей разработки – создание приложения, которым было бы удобно пользоваться как с веб-браузера настольного компьютера, так и с мобильного. Для достижения этой цели была использована библиотека разработки интерфейса – MaterialUI, предоставляющая возможности использовать блоки интерфейса, такие как «кнопка», «таблица» и другие, корректно отображающиеся на различных форматах экранов. Работа над клиентской частью велась с помощью интегрированной среды разработки GoLand и пакетного менеджера npm.

Были разработаны следующие модули системы:

Модуль просмотра списка объектов. Этот модуль был реализован как таблица, которая выводит все поля объектов, имеет настройки пагинации (настройка количества объектов на страницу, смена страниц). Над таблицей располагается навигационная цепочка (breadcrumbs), с помощью которого пользователь может возвращаться на предыдущие страницы объектов. Уровни навигационной цепочки разграничены по уровню иерархии DICOM объектов: «Исследование» > «Серия» > «Экземпляр». На рисунке 5 представлен модуль просмотра объектов на странице списка объектов «Исследование».

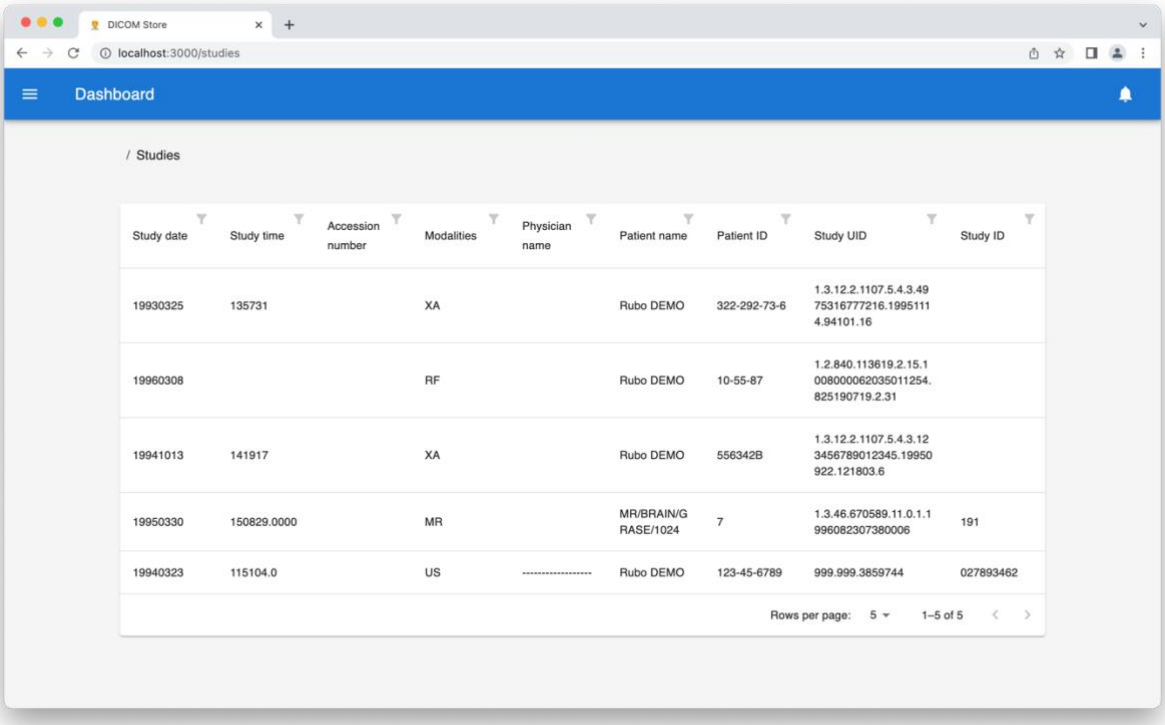

Рисунок 5 – Страница просмотра списка объектов «Исследование»

В заголовке таблицы у каждого столбца есть иконка добавления фильтра по значению поля объекта. При добавлении фильтра приложение отправляет QIDO запрос на сервер и отображает отфильтрованные записи. Процесс фильтрации объектов показан на рисунках 6 и 7.

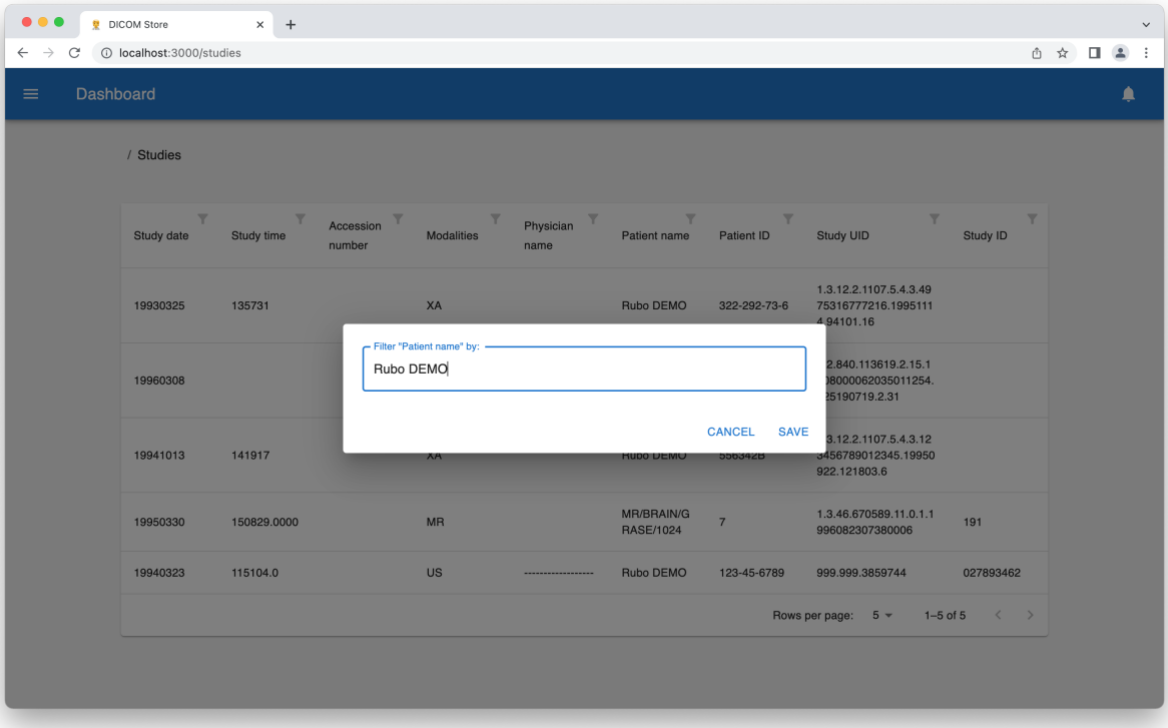

Рисунок 6 – Создание фильтра по полю «Patient name»

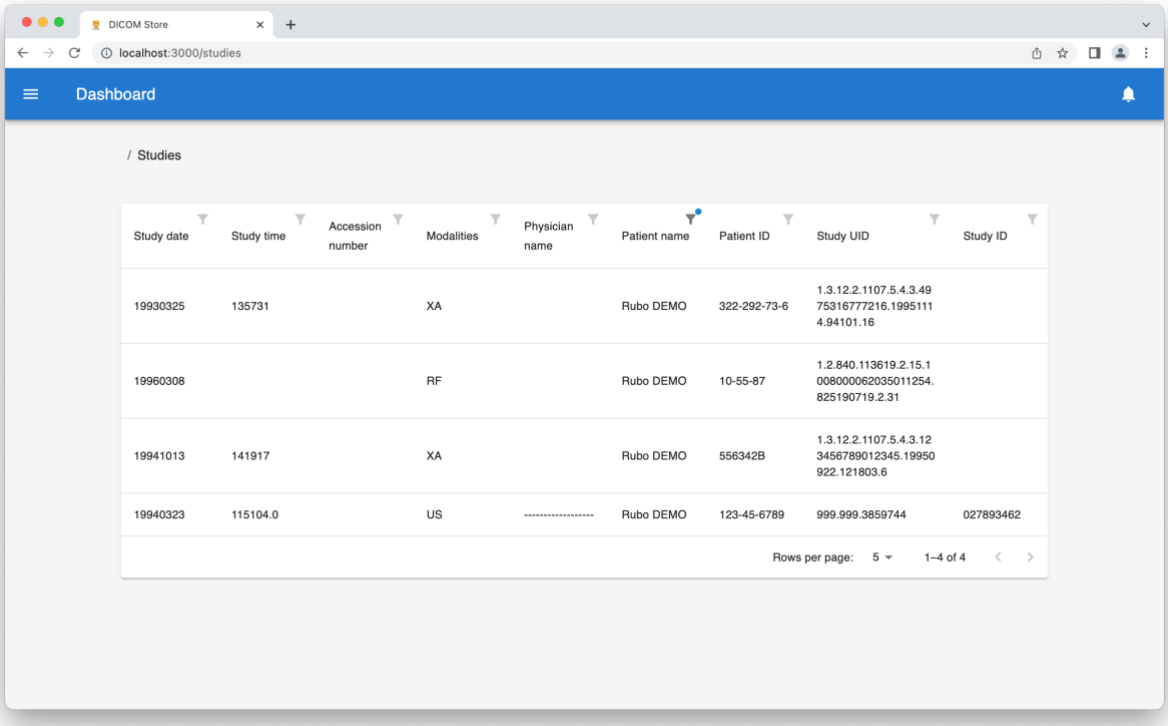

Рисунок 7 – Отфильтрованные записи по полю «Patient name» и значению «Rubo DEMO»

Модуль просмотра объекта. При клике на строку в таблице модуля просмотра списка объектов открывается страница просмотра объекта, который расположен в выбранной строке. Если выбранный объект содержит потомков (например, «Исследование» содержит «Серия», а «Серия» содержит «Экземпляр»), то на странице просмотра объекта отображается аналогичная таблица объектов – потомков. Также эта страница содержит блок всех полей объекта и кнопку для открытия страницы просмотра изображений. На рисунках 8 и 9 изображена страница просмотра объекта «Серия».

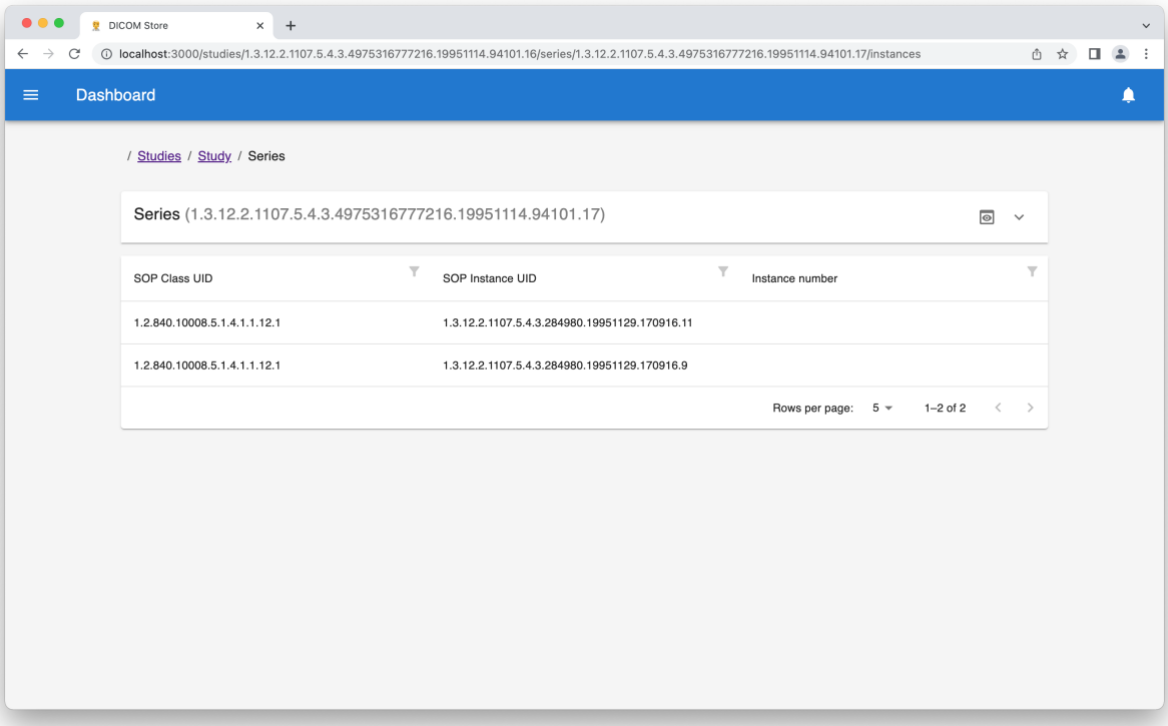

Рисунок 8 – Страница просмотра объекта «Серия»

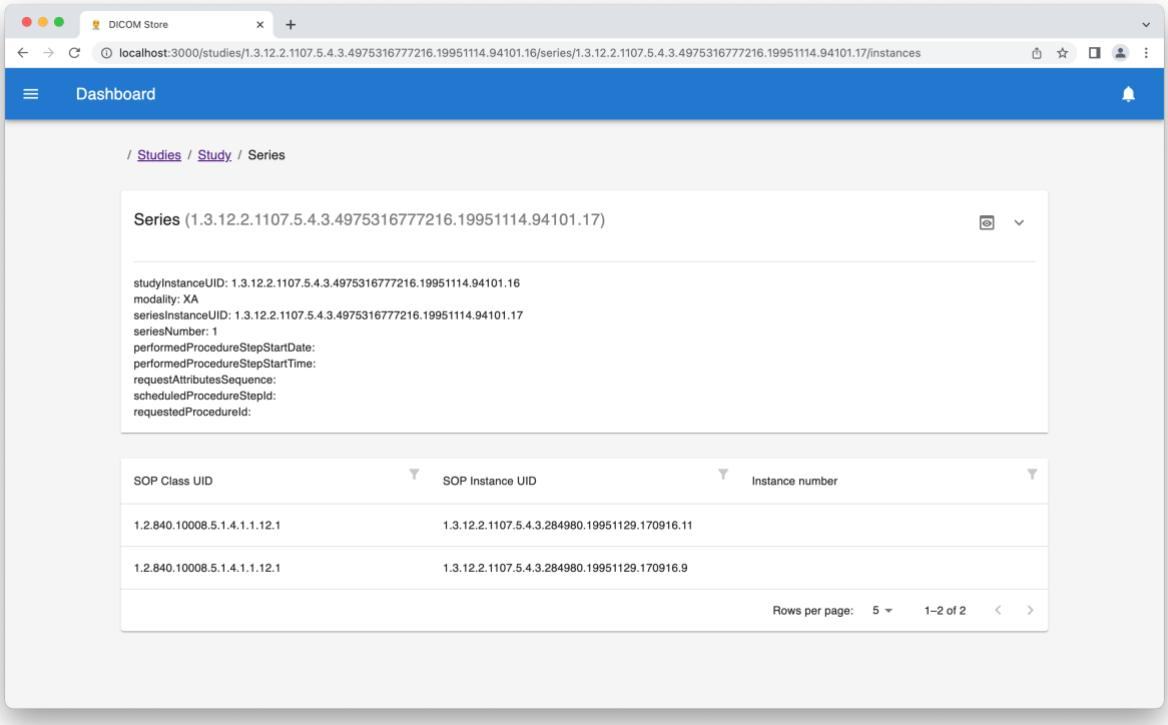

Рисунок 9 – Страница просмотра объекта «Серия» с открытым модулем полей объекта

Для объекта Instance модуль просмотра изображения изначально встроен в страницу просмотра объекта. На рисунке 10 изображена страница просмотра объекта «Экземпляр». Помимо модуля просмотра полей объекта на этой странице присутствует модуль просмотра DICOM изображений.

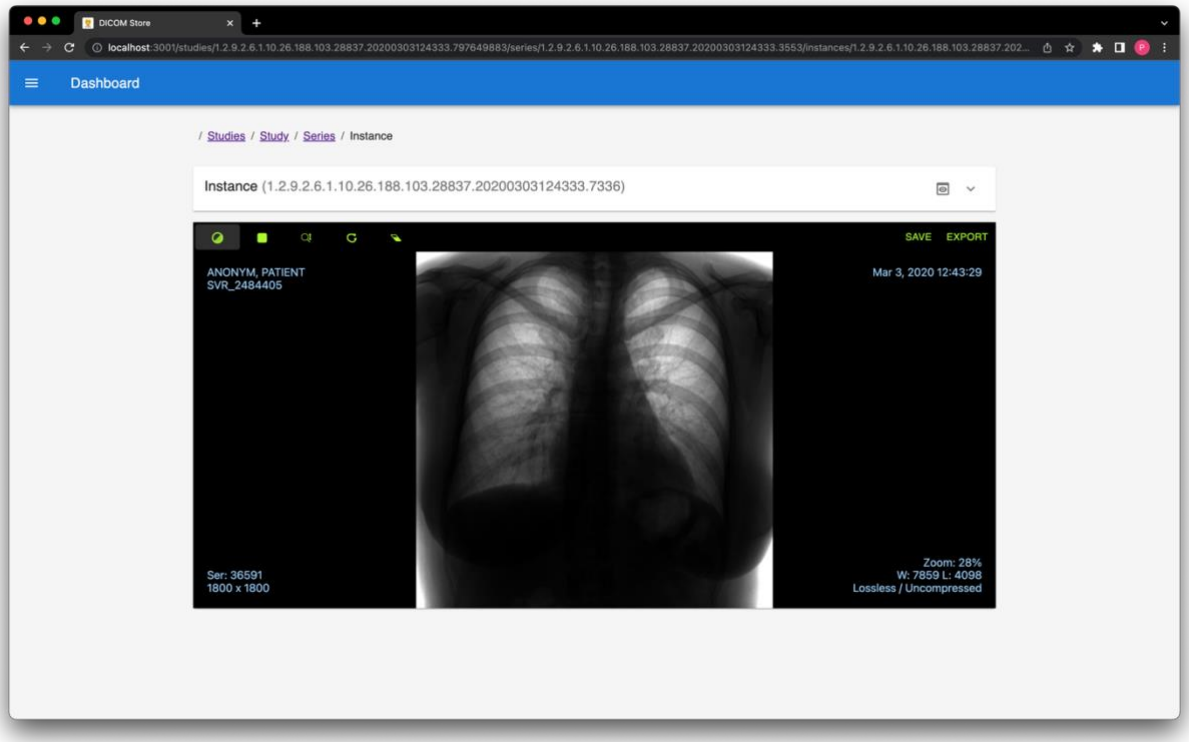

Рисунок 10 – Страница просмотра объекта «Экземпляр»

Модуль просмотра DICOM изображений. Для разработки этого модуля была использована библиотека Cornerstone.js. Этот модуль может получить DICOM изображения по стандарту WADO, отображать их, и предоставляет некоторые базовые инструменты для работы с изображениями: изменение масштаба, настройка ширины и центра окна, просмотр серии изображений. Также в этом модуле представлены инструменты работы с контурами – создание, удаление, сохранение и экспорт. Интерфейс модуля представлен на рисунке 11.

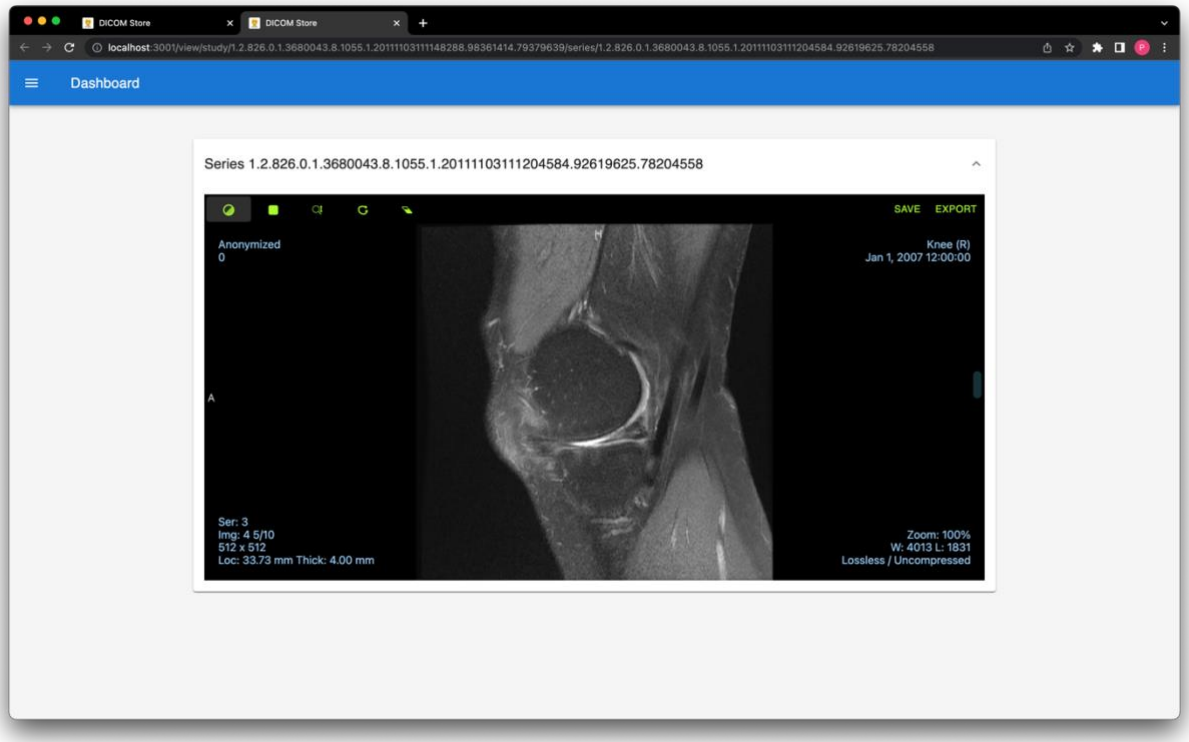

Рисунок 11 – Страница просмотра DICOM изображения

Модуль загрузки файлов DICOM. Данный модуль использует спецификацию STOW стандарта DICOMweb для загрузки исследований. Для каждого выбранного файла модуль отправляет соответствующий запрос на сервер. На каждый выбранный файл модуль показывает пользователю сообщение о статусе загрузки исследования. Страница загрузки файлов показана на рисунке 12.

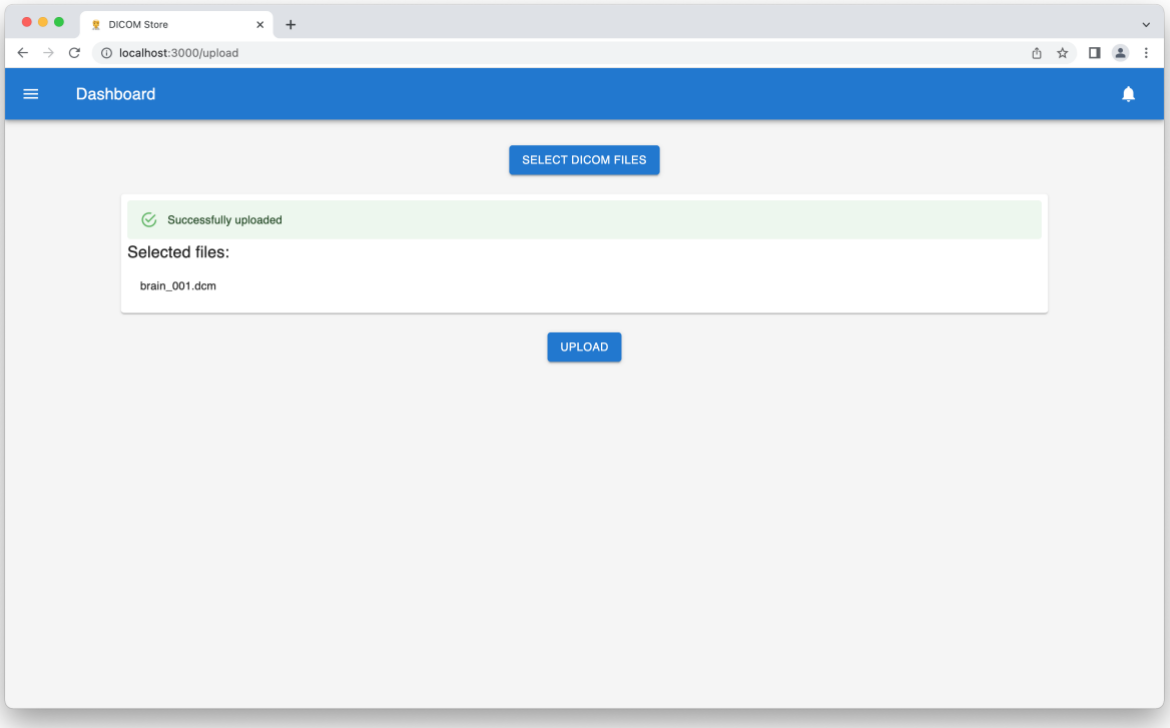

Рисунок 12 – Страница загрузки DICOM файлов

3.4 Разработка серверной части приложения

Как было описано в пункте 3.1 для взаимодействия между узлами приложения используется система контейнеризации Docker. Этот инструмент позволяет описывать конфигурацию нескольких сервисов, запускаемых вместе. Помимо того, что для каждого сервиса собирается свое изолированное окружение, Docker настраивает внутреннюю сеть между поднимаемыми сервисами. Так же Docker позволяет проксировать запросы в контейнер с помощью отображения порта с хост машины на порт внутри контейнера с определенным сервисом. Еще одна важная функция Docker системы – volumes. Volumes – это директории внутри контейнера, которые связаны с директориями на хост машине, что дает возможность изменить файлы на хост машине и считать изменения из сервиса, и наоборот – изменения, произведенные сервисом могут быть сохранены на хост машине. Так же механизм volumes позволяет не пересобирать контейнер при изменении кода приложения для применения внесенных изменений.

В конфигурационном файле docker-compose.yml (приложение А) описаны развернутые сервисы:

- nginx – сервис, содержащий web сервер, который обрабатывает запросы с хост машины и перенаправляет запросы в сервисы api и react;
- api – сервис, который содержит в себе окружение, необходимое для функционирования Go приложения;
- react – сервис, содержащий web сервер, который обрабатывает запросы из сервиса nginx и отвечает статичными файлами;
- postgres – сервис, содержащий РСУБД PostgreSQL.

Глава 4. Применение информационной системы

Современный тренд на создание систем и алгоритмов с использованием искусственного интеллекта (ИИ) порождает большое количество задач, связанных с обработкой DICOM. В этой главе будет показан возможный сценарий применения системы для дальнейшей обработки DICOM изображения с помощью ИИ. В первую очередь DICOM файл будет загружен в систему, после чего будет проведена разметка контура. Этот контур будет использован для создания маски изображения – фильтра, который оставит только выделенные в контуре данные DICOM изображения. Важно заметить, что полученные из системы файлы позволяют создать маску изображения без преобразования в растровый формат, при котором возможны потери данных. Например, в используемом КТ исследовании информация об изображении представлена в виде денситометрических показателей – единиц Хаунсфилда, которые могут принимать значения от -1024 до 3071. Перевод в растровый формат может обрезать часть данных (зависит от выбранной глубины цвета) или исказить данные, используя, например, интерполяцию для сжатия данных. Поэтому необходимо использовать исходные данные, закодированные в DICOM формате.

Для тестирования разработанной информационной системы были использованы файлы исследований различных модальностей, таких как: CT (компьютерная томография), MR (магнитно-резонансная томография), US (ультразвуковые исследования) и MG (маммография), однако этот список не является ограничивающим, и система может работать с другими модальностями.

4.1 Загрузка и разметка исследования

Запуск системы описан в файле README.md, находящемся в корневой директории системы. После запуска системы доступ к графическому интерфейсу системы осуществляется из веб-браузера. Стартовая страница системы, изображенная на рисунке 13 отображает количество исследований, модальностей и пациентов в базе данных.

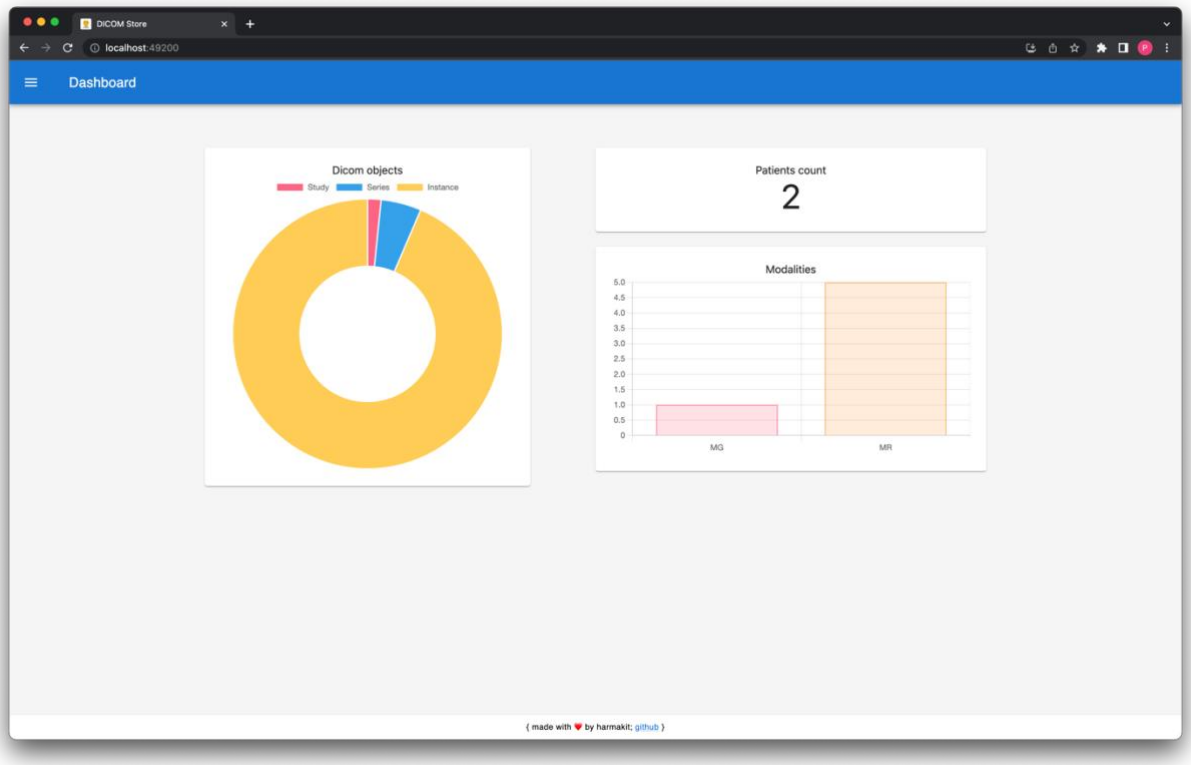

Рисунок 13 – Домашняя страница системы

С помощью модуля загрузки файлов DICOM добавляем заранее подготовленные файлы исследований в систему. Открываем список исследований (рисунок 14), после чего находим изображение, которое нам потребуется разметить (рисунок 15).

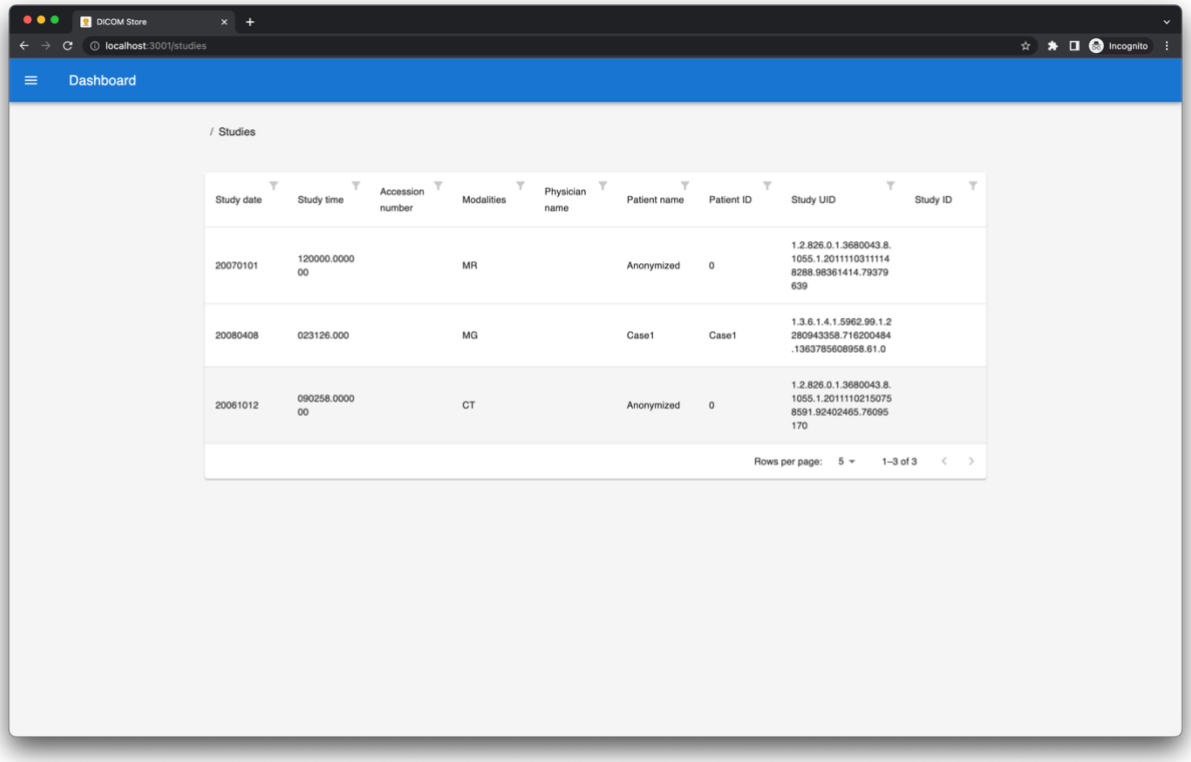

Рисунок 14 – Страница списка исследований

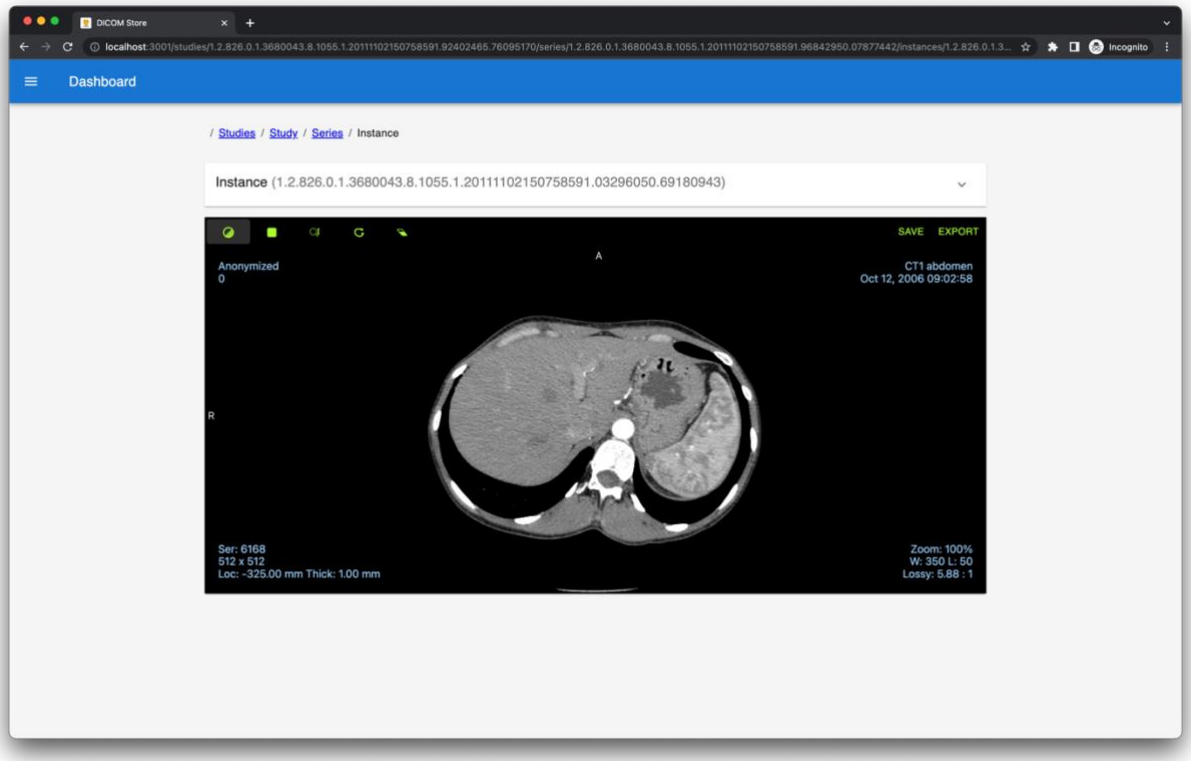

Рисунок 15 – Страница с выбранным снимком

С помощью инструмента, на иконку которого в меню инструментов указывает стрелка на рисунке 16, обведем требуемый сегмент изображения. В том же меню, с помощью кнопки «EXPORT», получим файл изображения в формате DICOM и файл с контуром в формате JSON.

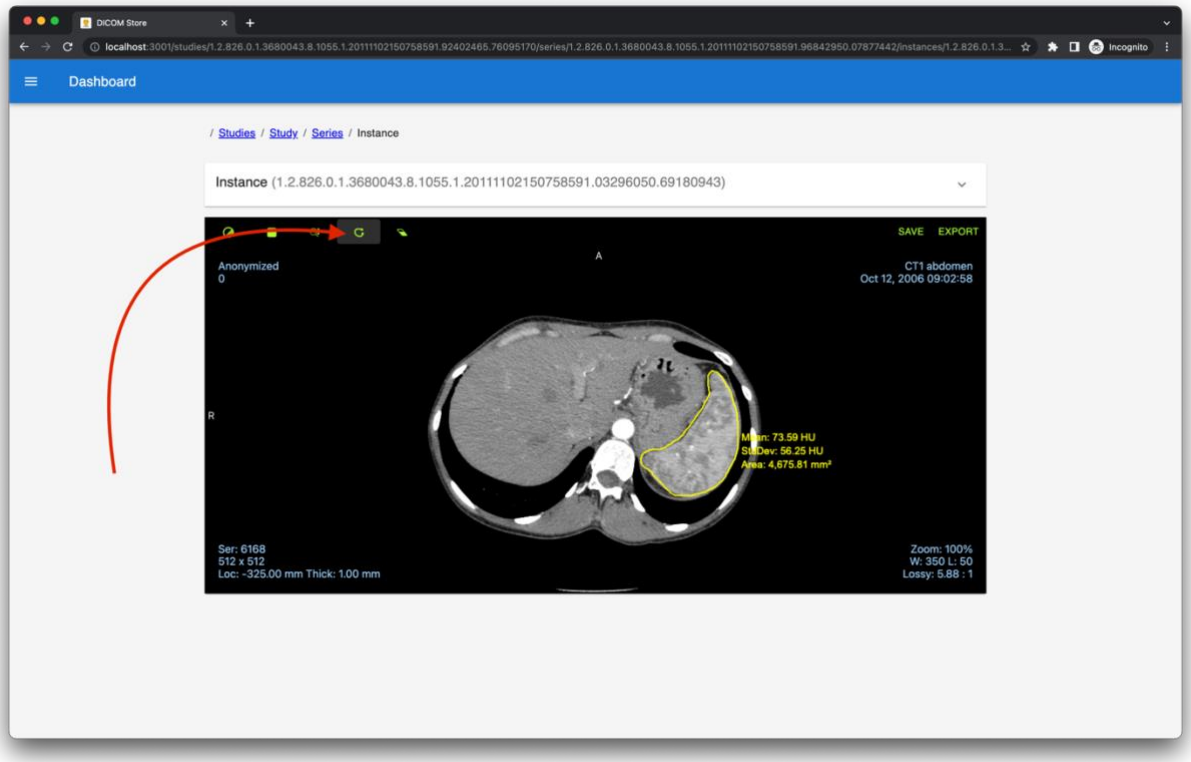

Рисунок 16 – Размеченный снимок

В следующем разделе будет описан вариант использования полученных файлов для извлечения размеченных данных.

4.2 Создание маски для обработки в Python

Полученные файлы снимка и контуров были переименованы в «dicom.dcm» и «contours.json0» соответственно для удобности использования. После чего была написана программа на языке Python, которая создает маску изображения, используя сохраненные контуры. На рисунках 17 и 18 изображены результаты работы программы. Полный файл программы прикреплен в приложении Б.

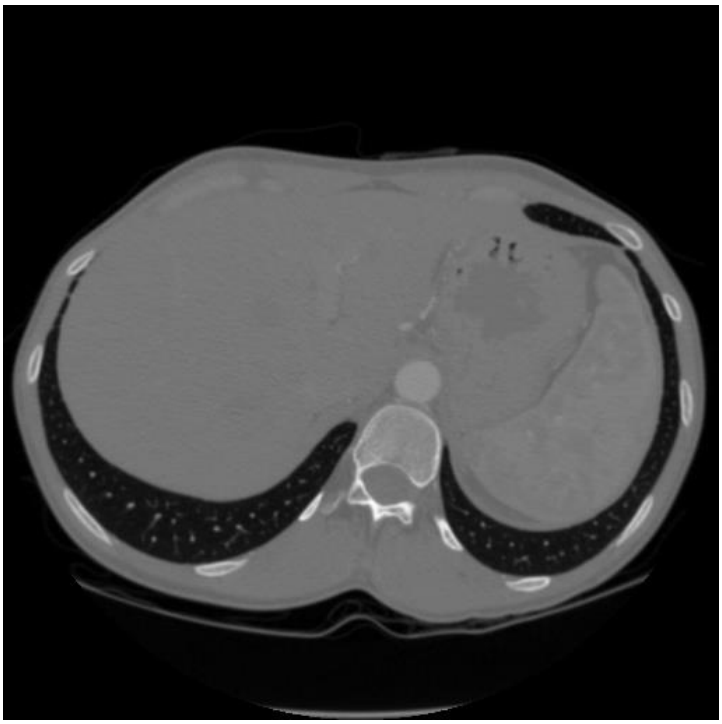

Рисунок 17 – Оригинальное изображение

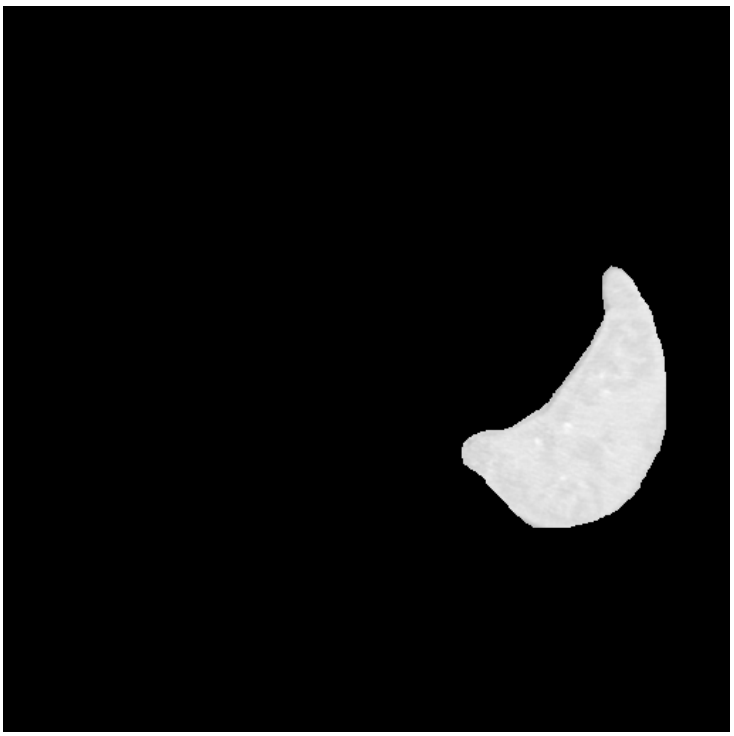

Рисунок 18 – Изображение, ограниченной маской

На результирующем изображении видно, что программа отфильтровала исходное изображение по маске, созданной с помощью контурной разметки. Таким образом разработанная информационная система была протестирована как инструмент, позволяющий подготовить данные для последующего анализа с помощью алгоритмов искусственного интеллекта.

Выводы

В результате работы были исследованы технологии, используемые в медицинской отрасли, такие как DICOM, PACS, DICOMweb. DICOM – широко используемый стандарт медицинского для управления медицинскими изображениями. Он разработан для стандартизации данных изображений и упрощения обмена между оборудованием разных производителей и рабочими станциями. DICOMweb – стандарт, позволяющий использовать DICOM системы с помощью веб-технологий. PACS-системы, работающие с веб-технологиями, позволяют работать с исследованиями в формате DICOM через веб-браузеры, что повышает доступность использования таких систем. Разработанный инструмент позволяет разметить DICOM изображения через веб-браузер и использовать размеченные контуры для последующих исследований без преобразования в растровый формат изображений, благодаря чему не будет происходить потеря данных.

Заключение

Была разработана система для разметки медицинских данных DICOM-формата. Задачи, поставленные для достижения цели, были выполнены: был проведен анализ существующих программных решений, на основе которого были сформулированы требования к разрабатываемой системе. Сама система была спроектирована и разработана со следующим функционалом: загрузка, хранение и поиск по исследованиям в системе, работа с исследованиями – просмотр исследований, создание, редактирование и экспортирование контуров. Функционал системы был протестирован – была создана дополнительная программа на языке Python, выполняющая маскирование изображения для возможного дальнейшего решения задач обработки с помощью нейронных сетей.

Список литературы

1. DICOMweb™: Background and Application of the Web Standard for Medical Imaging / B.W. Genereaux, D.K. Dennison, K. Ho [и др.] // Journal of Digital Imaging. – 2018. – № 31. – С. 321–326. – DOI 10.1007/s10278-018-0073-z
2. Ensemble classification and segmentation for intracranial metastatic tumors on MRI images based on 2D U-nets / L. Cheng-Chung, W. Meng-Yun, S. Ying-Chou. [и др.] // Scientific Reports. – 2021. – № 11. – С. 1-7. – ISSN 2045-2322 – DOI 10.1038/s41598-021-99984-5
3. Gaidel, A. Method of automatic ROI selection on lung CT images / A. Gaidel // Procedia Engineering. – 2017. – № 201. – С. 258-264. – DOI 10.1016/j.proeng.2017.09.612
4. Intelligent Decision Support System for Differential Diagnosis of Chronic Odontogenic Rhinosinusitis Based on U-Net Segmentation / V. Alekseeva, A. Nechyporenko, M. Frohme [и др.] // Electronics. – 2023. – № 12. – С. 1202. – ISSN 10.3390/electronics12051202
5. Larman, C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development / C. Larman. – Upper Saddle River, NJ : Prentice Hall PTR, 2004. – 703 с. – ISBN 978-0-13-148906-6.
6. Loeliger, J. Version Control with Git - Powerful Tools and Techniques for Collaborative Software Development / J. Loeliger. – Sebastopol, CA : O'Reilly Media, Inc., 2009. – 526 с. – ISBN 978-1492091196.
7. MedDream HTML5 DICOM Viewer | SOFTNETA : сайт. – URL: https://www.softneta.com/products/meddream-dicom-viewer/ (дата обращения: 05.05.2023)
8. Mehta, K. Heart Disease Diagnosis using Deep Learning / K. Mehta, K. Subramanian // 2022 IEEE India Council International Subsections Conference (INDISCON) . – Бхубанешвар, 2022. – С. 1-6. – DOI 10.1109/INDISCON54605.2022.9862847

9. Mildenberger, P. Introduction to the DICOM standard / P. Mildenberger, M. Eichelberg, E. Martin // European Radiology. – 2002. – № 12. – С. 920-927. – DOI 10.1007/s003300101100
10. Mouat, A. Using Docker: Developing and Deploying Software with Containers / A. Mouat. – Sebastopol, CA : O'Reilly Media, Inc., 2016. – 354 с. – ISBN 978-1491915769.
11. Mullie, L. CoreSlicer: a web toolkit for analytic morphomics / L. Mullie, J. Afilalo // BMC Med Imaging. – 2019. – № 19. – С. 15-29. – DOI 10.1186/s12880-019-0316-6
12. Mustra, M. Overview of the DICOM standard / M. Mustra, K. Delac, M. Grgic // 50th International Symposium ELMAR. – 2008. – № 1. – С. 39-44.
13. Nelli, F. Python data analytics With Pandas, NumPy, and Matplotlib / F. Nelli. – Berkeley, CA : Apress, 2018. – 569 с. – ISBN 978-1-4842-3912-4.
14. Open Health Imaging Foundation : сайт. – URL: https://ohif.org/ (дата обращения: 06.05.2023)
15. PACS: A guide to the digital revolution: Second edition / K. J. Dreyer, J. H. Thrall, D. S. Hirschorn, A. Mehta. – New York : Springer Science+Business Media, 2006. – 579 с. – ISBN 978-0-387-26010-5.
16. PS3.18 // DICOM NEMA : сайт. – URL: https://dicom.nema.org/medical/dicom/current/output/chtml/part18/PS3.18.html (дата обращения: 05.05.2023)
17. Николаенко М.А., Сидоренко А.С., Денисов И.А., Гребенник О.Г., Игрунова С.В. MySQL и PostgreSQL - сравнительный анализ // Экономика и социум. 2015. №1-1 (14). URL: https://cyberleninka.ru/article/n/mysql-i-postgresql-sravnitelnyy-analiz (дата обращения: 24.04.2023).
18. Сайко Е.В. Информационные технологии в медицинской диагностике. Pacs-системы и Рентгенологическая информационная система Ариадна - новейшие технологии хранения и обработки данных // Научные

исследования и разработки молодых ученых. 2016. №8. URL: https://cyberleninka.ru/article/n/informatsionnye-tehnologii-v-meditsinskoy-diagnostike-pacs-sistemy-i-rentgenologicheskaya-informatsionnaya-sistema-ariadna-noveyshie (дата обращения: 10.12.2021)

19. Сиразетдинов, Р. Р. Архитектура информационных систем / Р. Р. Сиразетдинов, Д. В. Белоус // Техника средств связи. – 2020. – № 3. – С. 64-68.
20. Эль-Хатиб С. А., Скобцов Ю. А. Компьютерная система сегментации медицинских изображений методом муравьиных колоний // Радіоелектроніка, інформатика, управління. 2015. №3 (34). URL: https://cyberleninka.ru/article/n/kompyuternaya-sistema-segmentatsii-meditsinskih-izobrazheniy-metodom-muravinyh-koloniy (дата обращения: 30.04.2023).

ПРИЛОЖЕНИЕ А

Конфигурационный файл docker-compose.yml

```
version: "3.8"

volumes:
  postgres:
  api:

services:
  nginx:
    image: nginx:alpine
    restart: unless-stopped
    tty: true
    ports:
      - "49200:80"
    volumes:
      - ./nginx/conf.d/:/etc/nginx/conf.d/

  api:
    build:
      context: ../dicomweb-pacs-api
    depends_on:
      - postgres
    volumes:
      - ./.data/api/uploads:/app/uploads
    ports:
      - "49201:3000"
    environment:
      LOG_LEVEL: debug
      LOG_TEXTLOGGING: "true"
      PORT: 3000
      DB_ADDR: postgres:5432
      DB_USER: postgres
      DB_PASSWORD: postgres
      DB_DATABASE: postgres
      ENABLE_CORS: "true"
    deploy:
      restart_policy:
        condition: on-failure
        delay: 1s
```

ОКОНЧАНИЕ ПРИЛОЖЕНИЯ А

Конфигурационный файл docker-compose.yml

```
react:
  stdin_open: true
  build:
    context: ../dicomweb-pacs-front
  volumes:
    - ../dicomweb-pacs-front:/root
  ports:
    - "49202:4000"


postgres:
  image: postgres:13
  restart: unless-stopped
  ports:
    - "49203:5432"
  volumes:
    - ./.data/postgres:/var/lib/postgresql/data
  environment:
    POSTGRES_PASSWORD: postgres
```

ПРИЛОЖЕНИЕ Б

Листинг программы использования контуров

```
import json

import cv2
from pydicom import dcmread


def main():
    path = './dicom.dcm'
    contours_path = './contours.json'
    ds = dcmread(path)
    contours = json.load(open(contours_path))  # array of
contours. each contour is an array of points {x, y}
    first_contour = contours[0]

    arr = ds.pixel_array
    # show plot of the original image
    import matplotlib.pyplot as plt
    plt.imshow(arr, cmap='gray')
    plt.show()

    # create cv2 polygon from contour
    import numpy as np
    # first_contour is an array of dicts {x, y}
    pts = np.array([[p['x'], p['y']] for p in first_contour])
    pts = pts.reshape((-1, 1, 2))
    # create mask
    mask = np.zeros(arr.shape, np.uint8)
    cv2.fillPoly(mask, np.int32([pts]), 255)
    # apply mask
    masked_arr = cv2.bitwise_and(arr, arr, mask=mask)
    # show plot of the result
    plt.imshow(masked_arr, cmap='gray')
    plt.show()


if __name__ == '__main__':
    main()
```