Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 20.Б11-мм

Шехаде Даниэль

Реализация сервиса обмена сообщениями и расширенного поиска по пользователям и врачам в веб-приложении

Отчёт по учебной практике
в форме «Решение»

Научный руководитель:
доцент кафедры системного программирования, к.т.н., Ю.В. Литвинов

Санкт-Петербург
2022

Оглавление

1. Введение

Второе мнение — услуга повторной консультации с врачом. Суть заключается в том, что пациент уже по имеющимся результатам проведенных медицинских анализов с целью уточнения диагноза, плана лечения и причины заболевания может обратиться к медицинскому специалисту за «вторым мнением». Данная процедура представляет собой консультативный прием пациента, и проводится опытным врачом узкой специализации.

В 2020 году глава Минздрава России Михаил Мурашко заявил, что ежегодно из-за ошибок врачей в стране страдают от осложнений заболевания порядка 70 тыс. человек [21]. Основным критерием врачебной ошибки является заблуждение врача, которое вытекает из определенных объективных условий и не содержит элементов халатности, небрежности и профессионального невежества. [19]

Сервис будет полезен, как пациентам, так и врачам. Пациенты смогут найти медицинского специалиста или же воспользоваться функцией автоматического анализа медицинских снимков, и таким образом получить “второе мнение”. Врачи смогут использовать наш сервис, как веб-форум для общения с коллегами, поиска клинических случаев для своих научных исследований, а также, как платформу, где можно спасти чью-то жизнь. Функциональность автоматического анализа реализована при помощи библиотеки MIRF, которая, используя алгоритмы машинного обучения, предоставляет возможность обрабатывать медицинские снимки.

Первоначально необходимо реализовать разделение зарегистрированных пользователей на роли: пользователь и врач. И для того, чтобы у пользователя была возможность найти и обратиться к нужному специалисту, необходимо реализовать сервис для обмена сообщениями, а также поиск по врачам, которые смогут оказать помощь на данной платформе, и по пользователям, чтобы была возможность всем людям, зарегистрированным на площадке, искать друг друга.

2. Постановка задач

Целью работы является реализация расширенного поиска по пользователям и врачам, предоставление возможности общения пользователей и врачей путем реализации личных сообщений в веб-приложении. Для ее выполнения были поставлены следующие задачи:

1. Для выполнения поставленных целей необходимо реализовать:
   (a) серверную часть решения, используя фреймворк Spring для Java-платформы.
   (b) клиентскую часть решения, используя JavaScript-библиотеку React для разработки пользовательских интерфейсов.
2. Спроектировать свое решение в уже сложившимся в данном проекте паттерне MVC (Model-View-Controller).
3. Протестировать серверную часть решения, используя фреймворки JUnit и spring-boot-test.
4. Настроить CI (Continuous Integration) на GitHub.

3. Обзор существующих решений

Была рассмотрена реализация поиска врачей и пользователей на различных медицинских сервисах. Внимание обращалось на реализацию UI (User interface) и UX (User experience).

1. Сайт медицинской клиники «Скандинавия» [23]. Из плюсов можно выделить возможность сортировки врачей в алфавитном порядке по фамилии, имени и отчеству. Удобный фильтр по адресам клиник, в которой работает врач, и по специальности врача. К минусам можно отнести то, что произвести поиск можно только по нажатию клавиши Enter, то есть не поддерживается поиск по нажатию кнопки в строке ввода.

2. Сайт сети медицинских центров «Поликлиника» [22]. Реализован фильтр поиска по адресам клиник и специальности врача, в стиле идентичном сайту "Скандинавия". Есть возможность произвести поиск по нажатию клавиши Enter и по нажатию кнопки "Поиск". Кнопка "Поиск" расположена в стороне от строки ввода (Input), было бы удобнее и практичнее поместить ее внутрь. Отсутствует возможность сортировки врачей в алфавитном порядке по фамилии и имени.

3. Сайт медицинского центра «XXI век» [20]. В разделе "Врачи" можно найти специалиста по его направлению и филиалу клиники, в которой он работает. Отсутствует строка ввода для поиска в разделе "Врачи". Чтобы произвести поиск по фамилии или имени врача, необходимо воспользоваться глобальным поиском, который относится ко всему сайту, что является неудобным.

4. Сайт медицинского центра «СМ-клиника» [7]. Реализован фильтр для поиска по специальности и месту работы врача в филиале. Найденные врачи сортируются в алфавитном порядке по фамилии, имени и отчеству. Из минусов можно выделить то, что отсутствует возможность воспроизведения поиска по клавише Enter.

При разработке UI и UX дизайна для мессенджера, были рассмотрены и проанализированы реализации таких популярных сервисов как:

1. WhatsApp [18]. Из минусов в данном мессенджере можно выделить то, что нет возможности добавлять новых пользователей в групповые чаты с сохранением истории переписки. Нет возможности редактирования сообщений, а также при удалении сообщения собеседник видит, что оно было удалено.

2. VKontakte [16]. Интерфейс веб версии VKontakte предназначен только для полноэкранного режима, то есть HTML компоненты в основном статичны и не реагируют на изменения размера экрана (плохо реализована адаптивная верстка).

3. Telegram [15]. В Telegram не было выявлено недостатков. Вся базовая функциональность мессенджера реализована хорошо.

4. Реализация

4.1. Фронтенд

Был реализован пользовательский интерфейс страницы поиска (представлен на Рис. 1) и страницы мессенджера (представлен на Рис. 2), при помощи JavaScript-библиотеки React и набора компонентов Material UI [4], с учетом проведенного анализа и выявленных недостатков в поисковиках и в сервисах обмена сообщениями на вышеупомянутых сайтах.

Для предоставления возможности поиска и идентификации пользователя по имени, фамилии и отчеству, а также для разделения пользователей на роли, на странице регистрации были добавлены обязательные поля ввода для фамилии и имени, опциональное поле для отчества и переключатель ролей пользователь-врач.

Дополнительно была реализована возможность загрузки аватаров в личный профиль для их последующего отображения в мессенджере.

Были написаны компоненты для отображения интерфейса пользователю и сервисы, из которых отправляются запросы и принимаются ответы от сервера.

Более подробно про поисковик: был реализован фильтр поиска пользователя по логину или фамилии-имени, по роли: пользователь или врач. Для поиска пользователя по логину, необходимо написать логин полностью и правильно, так как логин является уникальным идентификатором пользователя. Поиск пользователя по фамилии, имени и отчеству осуществляется по содержанию указанной подстроки в данных параметрах. Также была реализована возможность осуществления поиска по нажатию клавиши Enter или по нажатию кнопки ”Поиск”.

Более подробно про мессенджер: была реализована возможность отправления и получения сообщений, файлов, изображений в режиме реального времени. Реализован список контактов, которые сортируются по времени отправления последнего сообщения, с логикой вывода дат

и времени, присущей большинству популярных мессенджеров (”вчера”, ”ПН”, ”ВТ”, ...”ВС”, 19.05.2022, можно увидеть на Рис. 2). В связи с тем, что при использовании WebSocket API [17] изображения и файлы перед отправкой на сервер необходимо преобразовывать в Base64 [2] (данный процесс весьма ресурсоемкий), было реализовано ограничение на размер одного файла (до 50 МБ) и на количество файлов (до 6 единиц), загружаемых в мессенджер. В случае если пользователю необходимо будет передать файлы большего размера, он сможет предварительно загрузить файлы на SFTP-сервер [8] или в случае с DICOM [3] изображениями на Orthanc-сервер [5] по HTTP запросу.

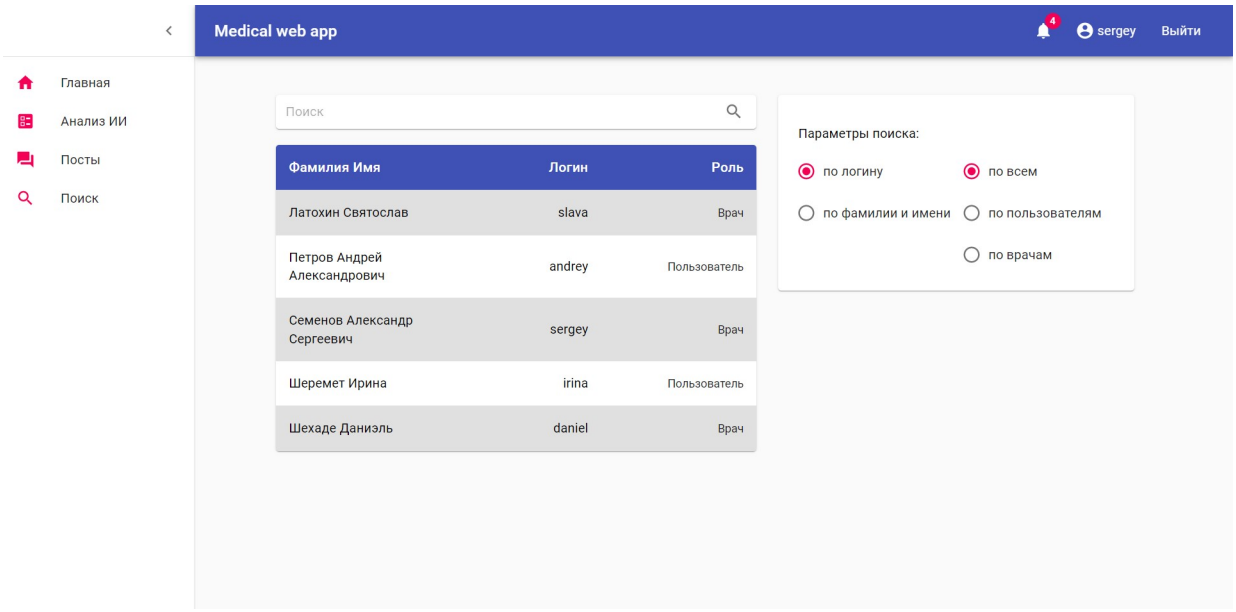

Рис. 1: Интерфейс поиска пользователей

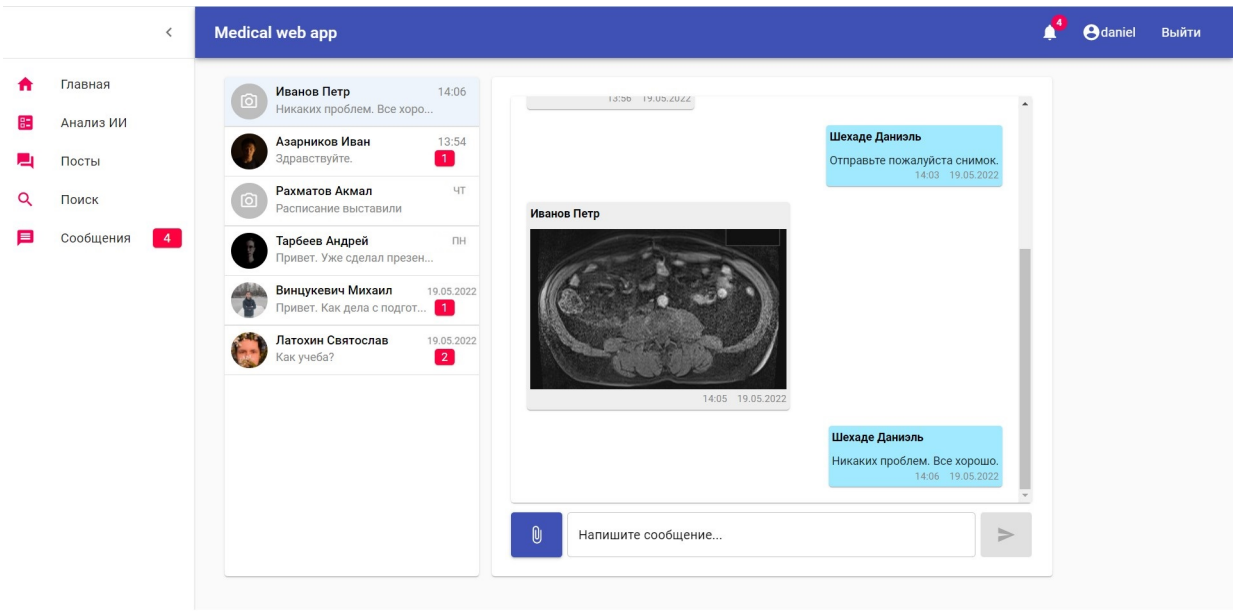

Рис. 2: Интерфейс мессенджера

4.2. Бэкенд

Серверная часть решения задачи написана на языке Java с использованием фреймворка Spring. В основе реализации бэкенда лежат технологии Spring Boot Web [11] и Spring Data JPA [12].

Были написаны контроллеры для обработки запросов, а также для отправки ответов клиенту, сервисы в которых реализована бизнес-логика и репозитории, в которых имеется возможность при помощи JPA посылать запросы для получения данных, используя ключевые слова. В класс модели, который используется для создания объектов User и дальнейшего сохранения пользователей в базу данных, были добавлены поля фамилии, имени и отчества.

4.3. API

Связь клиента с сервером реализована при помощи библиотеки Axios [1] — это HTTP-клиент для отправки запросов.

Двусторонняя связь и соединение (клиент<->сервер), используемые для обмена сообщениями, реализованы на основе WebSocket API [17]. В

качестве протокола для обмена сообщениями используется Stomp [14]. Для корректного соединения с сервером, на клиенте используется библиотека SockJS [9], которая предоставляет объект, соответствующий определениям и стандартам WebSocket. На сервере для установления соединения используется библиотека Spring WebSocket [13].

4.4. Тестирование

Были написаны интеграционные тесты на классы контроллеров, которые отправляют HTTP запросы. Обращение происходит к существующей базе данных, поэтому для подобных тестирований был создан отдельный контейнер на основе СУБД PostgreSQL [6].

Были написаны модульные тесты на классы сервисов. В данном случае, чтобы не происходило обращений к реальной базе данных, происходит подмена репозитория Mock объектом [10].

4.5. Continuous Integration

CI на GitHub настроен так, что перед тестами поднимаются контейнеры, необходимые для работы серверной части приложения, после чего запускаются интеграционные и модульные тесты.

4.6. Дополнительно

Кроме выполнения поставленных задач, также было произведено улучшение пользовательского интерфейса на страницах входа (Старый интерфейс представлен на Рис. 2, новая на Рис. 3), регистрации (Старый интерфейс представлен на Рис. 4, новая на Рис. 5), запуска конвейеров (Старый интерфейс представлен на Рис. 6, новая на Рис. 7) и загрузки файлов (Старый интерфейс представлена на Рис. 8, новая на Рис. 8).

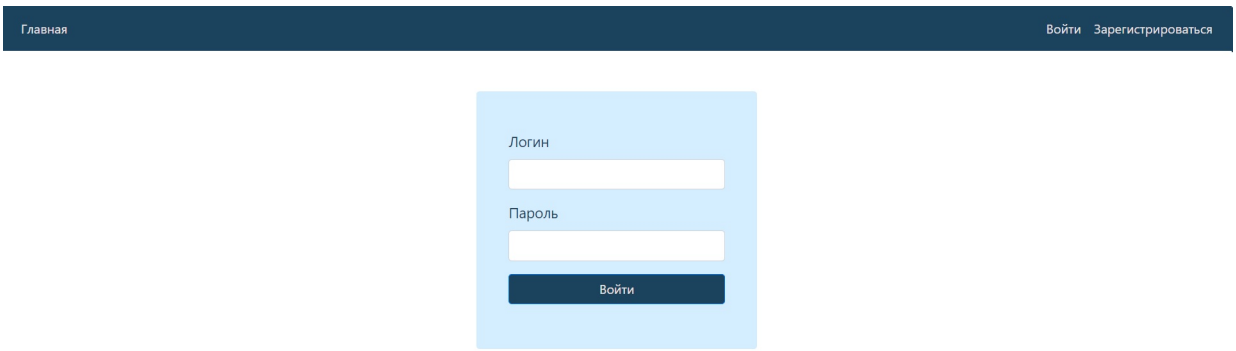

Рис. 2: Страница входа (старый UI)

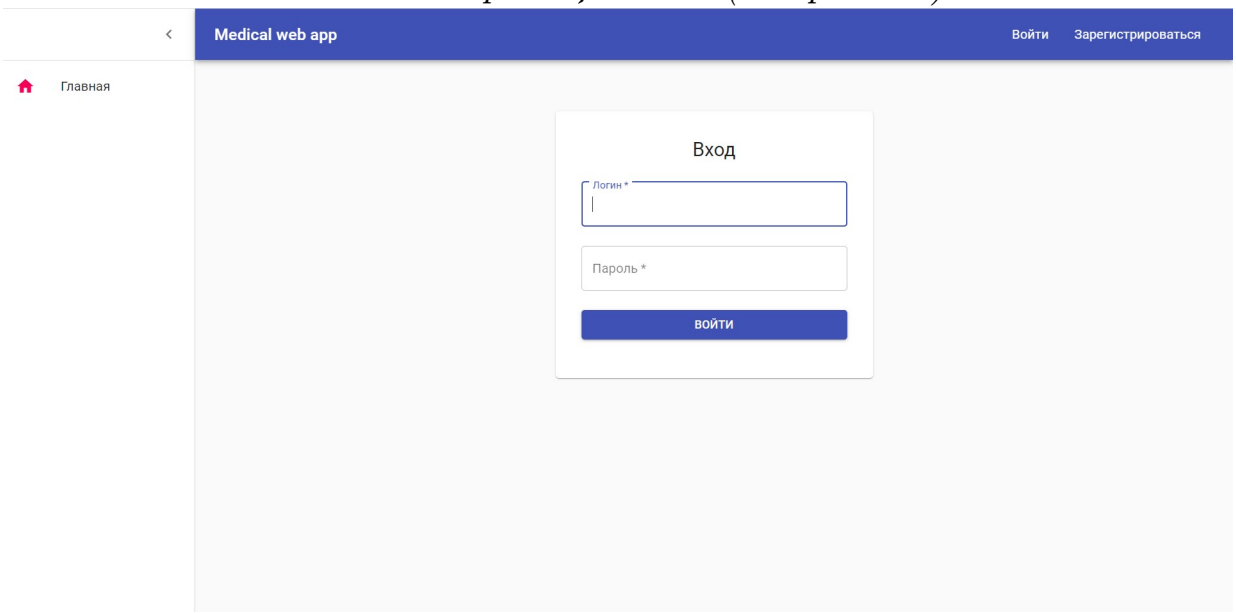

Рис. 3: Страница входа (новый UI)

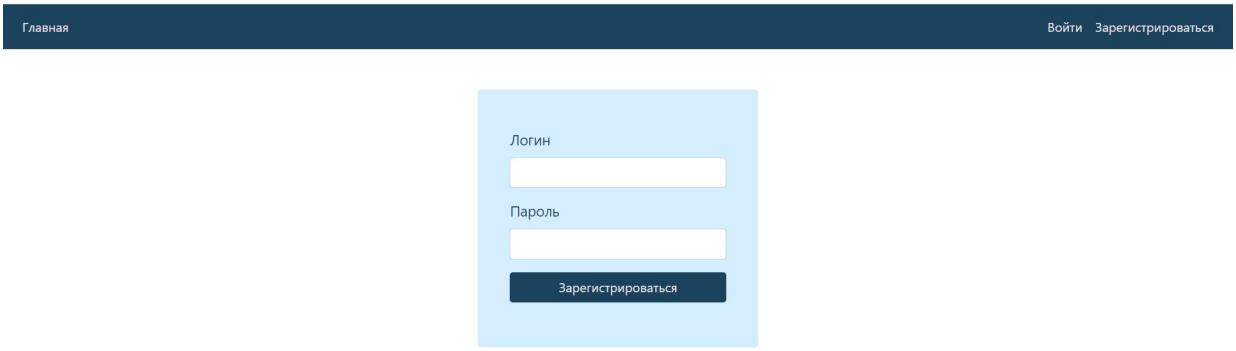

Рис. 4: Страница регистрации (старый UI)

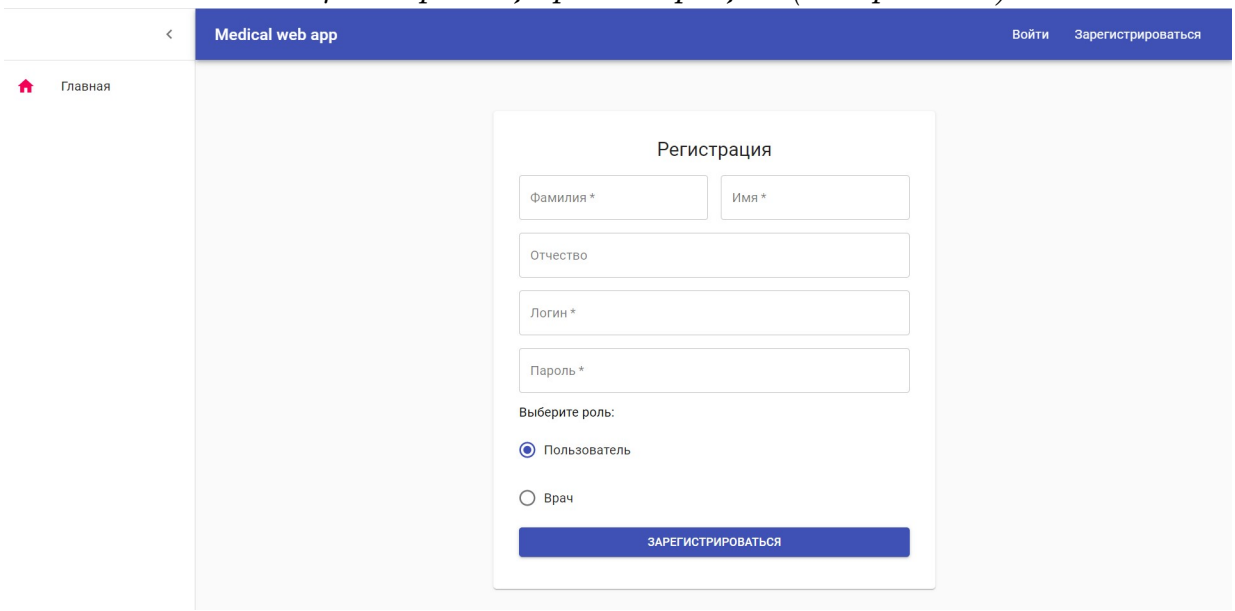

Рис. 5: Страница регистрации (новый UI)

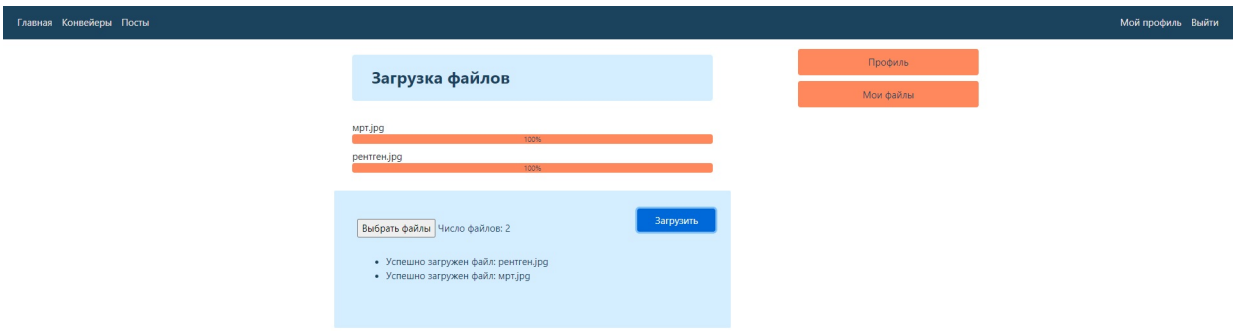

Рис. 6: Страница загрузки файлов (старый UI)

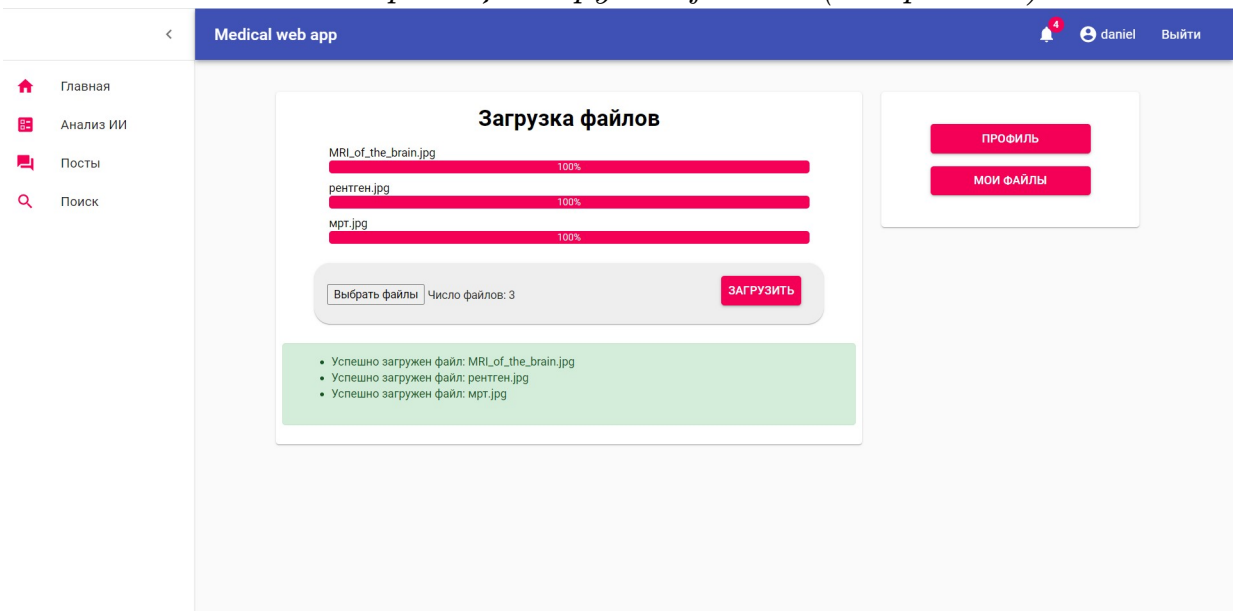

Рис. 7: Страница загрузки файлов (новый UI)

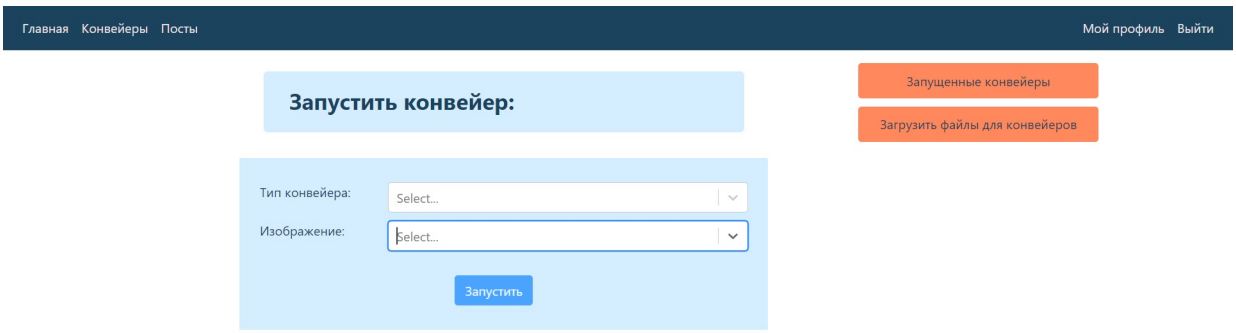

Рис. 8: Страница запуска конвейеров (старый UI)

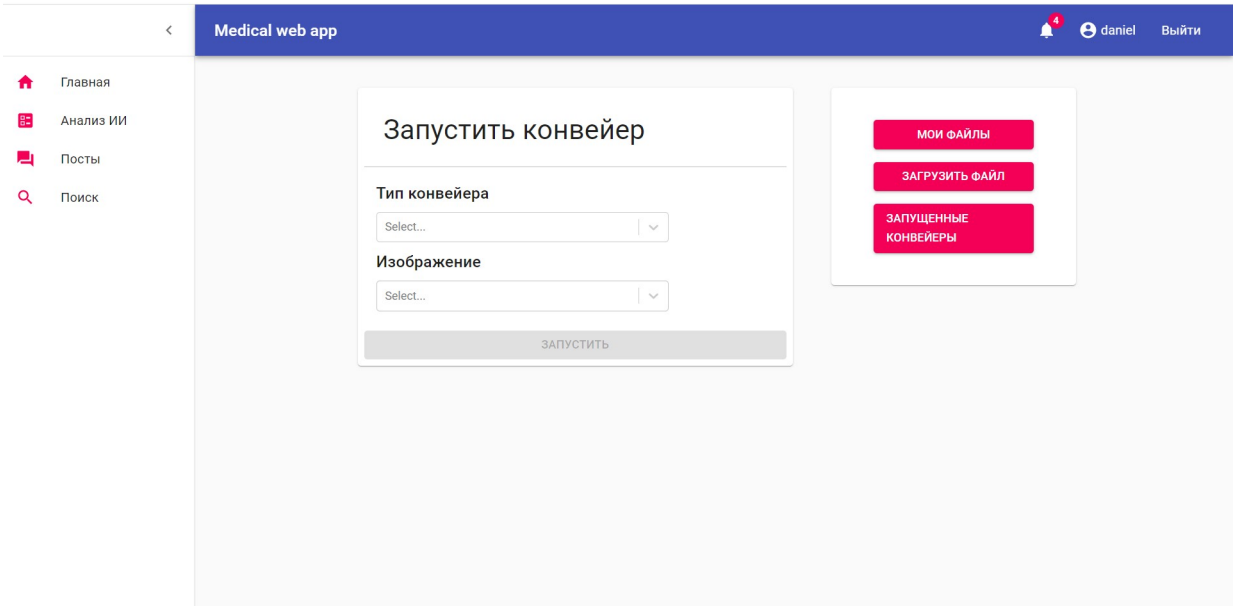

Рис. 9: Страница запуска конвейеров (новый UI)

5. Заключение

В ходе выполнения работы получены следующие результаты:

1. Сделан обзор существующих решений.
2. Реализована клиентская и серверная часть приложения, используя паттерн MVC (Model-View-Controller).
3. Протестирована серверная часть решения.
4. Настроен CI (Continuous Integration) в репозитории GitHub.

Код доступен в репозитории GitHub [1].

[1] https://github.com/MathAndMedLab/Medical-Web-App

Список литературы

[1] Axios. Promise based HTTP client for the browser and node.js. — https://axios-http.com/. — [Дата обращения 2022-05-22].

[2] Base64. — https://developer.mozilla.org/ru/docs/Glossary/Base64. — [Дата обращения 2022-05-22].

[3] Dicom. — https://www.dicomstandard.org/. — [Дата обращения 2022-05-22].

[4] MaterialUI. The React UI library. — https://mui.com/. — [Дата обращения 2022-05-22].

[5] Orthanc. — https://book.orthanc-server.com/users.html. — [Дата обращения 2022-05-22].

[6] PostgreSQL. The PostgreSQL object-relational database system provides reliability and data integrity. — https://registry.hub.docker.com/_/postgres. — [Дата обращения 2022-05-22].

[7] SM клиника. — https://www.smclinic.ru. — [Дата обращения 2022-05-22].

[8] Sftp. — https://www.ssh.com/academy/ssh/sftp. — [Дата обращения 2022-05-22].

[9] SockJs. — https://github.com/sockjs/sockjs-client. — [Дата обращения 2022-05-22].

[10] Spring. Annotation Type MockBean. — https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/test/mock/mockito/MockBean.html. — [Дата обращения 2022-05-22].

[11] Spring. Building a RESTful Web Service. — https://spring.io/guides/gs/rest-service/. — [Дата обращения 2022-05-22].

[12] Spring. Spring Data JPA. — https://spring.io/projects/spring-data-jpa#overview. — [Дата обращения 2022-05-22].

[13] Spring. Using WebSocket. — https://spring.io/guides/gs/messaging-stomp-websocket/. — [Дата обращения 2022-05-22].

[14] Stomp over WebSocket. — http://jmesnil.net/stomp-websocket/doc/. — [Дата обращения 2022-05-22].

[15] Telegram. — https://web.telegram.org/. — [Дата обращения 2022-05-22].

[16] VKontakte. — https://vk.com/. — [Дата обращения 2022-05-22].

[17] WebSocket API. — https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API. — [Дата обращения 2022-05-22].

[18] WhatsApp. — https://www.whatsapp.com/. — [Дата обращения 2022-05-22].

[19] Константиновна Краснопеева Марина. Современные врачебные ошибки, статистика летальных исходов в России. — https://cyberleninka.ru/article/n/sovremennye-vrachebnye-oshibki-statistika-letalnyh-ishodov-v-ros [Дата обращения 2022-05-22].

[20] Медицинский центр XXI век. — https://mc21.ru. — [Дата обращения 2022-05-22].

[21] Минздрав сообщил о 70 тыс. случаев осложнений в год из-за ошибок врачей. — https://www.interfax.ru/russia/694577. — [Дата обращения 2022-05-22].

[22] Поликлиника. — https://polyclinika.ru/. — [Дата обращения 2022-05-22].

[23] Скандинавия. — https://www.avaclinic.ru. — [Дата обращения 2022-05-22].