**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра MO ЭВМ**

**ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ**

**по дисциплине «Введение в нереляционные базы данных»**

**Тема: Приложение для мониторинга пациентов**

|  |  |  |
| --- | --- | --- |
| Студент гр. 9382 |  | Дерюгин Д.А. |
| Студент гр. 9383 |  | Вербин К.М. |
| Студент гр. 9383 |  | Гордон Д.А. |
| Преподаватель |  | Заславский М.М. |

Санкт-Петербург

2023**ЗАДАНИЕ**

|  |
| --- |
| Студенты  Дерюгин Д.А.  Вербин К.М.  Гордон Д.А. |
| Группа 9382, 9382 |
| Тема работы: Приложение для мониторинга пациентов. |
| Исходные данные:  Необходимо реализовать приложение для СУБД(MongoDB). |
| Содержание пояснительной записки:  «Содержание»  «Введение»  «Качественные требования к решению»  «Сценарий использования»  «Модель данных»  «Разработка приложения»  «Вывод» |
| Предполагаемый объем пояснительной записки:  Не менее 20 страниц. |
| Дата выдачи задания: |
| Дата сдачи реферата: |
| Дата защиты реферата: |

|  |  |  |
| --- | --- | --- |
| Студент гр. 9382 |  | Дерюгин Д.А. |
| Студент гр. 9383 |  | Вербин К.М. |
| Студент гр. 9383 |  | Гордон Д.А. |
| Преподаватель |  | Заславский М.М. |

**АННОТАЦИЯ**

В данной работе было разработано приложение для мониторинга пациентов. Для разработки использовались MongoDB в качестве базы данных, Nodejs для серверной части React для клиентской части приложения.

**SUMMARY**

In this work, an application for patient monitoring was developed. MongoDB was used as the database, Nodejs for the server-side, and React for the client-side of the application.

**СОДЕРЖАНИЕ**

|  |  |  |
| --- | --- | --- |
| 1. | Введение | 6 |
| 2. | Качественные требования к решению | 6 |
| 3. | Сценарии использования | 6 |
| 4. | Модель данных | 16 |
| 5. | Разработанное приложение | 30 |
| 6. | Вывод | 33 |

**1. Введение**

Цель работы - создать веб-приложение для предоставления докторам возможности наблюдать за состоянием пациентов.

**2. Качественные требования к решению**

Требуется разработать веб-приложение с использованием СУБД MongoDB.

**3. Сценарии использования**

**Макет UI:**

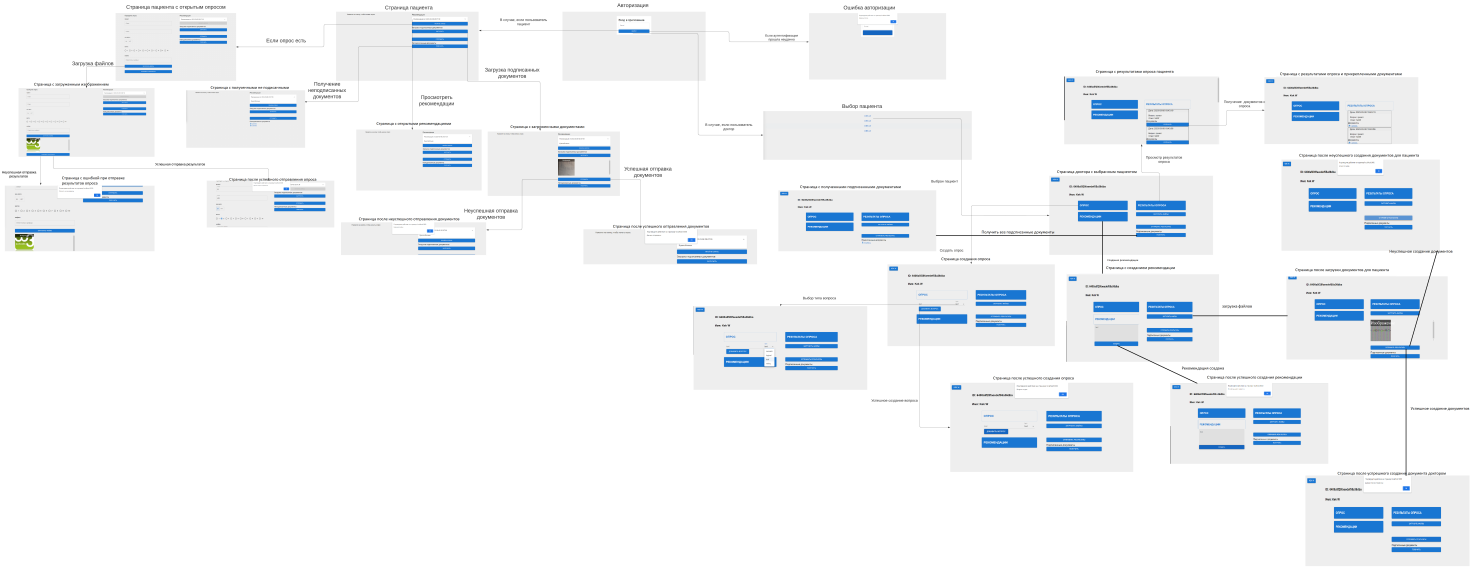
****

Рисунок 1 –Use Case

**Описание сценариев использования:**

**Сценарий использования – "Регистрация"**

Действующее лицо: Пользователь.

Начальные условия: Пользователь зашёл на сайт приложения (экран «Авторизация»).

Основной сценарий:

Пользователь нажимает на кнопку "Регистрация" (экран 1)

Пользователь вводит данные для входа, нажимает кнопку "Войти" (экран «Авторизация»)

Если данные введены корректно и пользователь - Пациент, то пользователь входит на сайт (экран «Страница пациента»)

Если данные введены корректно и пользователь - Доктор, то пользователь входит на сайт (экран «Выбор пациента»)

Альтернативный сценарий:

В случае некорректно введённых данных выдается ошибка (экран «Ошибка авторизации»)

**Сценарий использования – "Пройти опрос"**

Действующее лицо: Пациент.

Начальные условия: Пользователь авторизован на сайт приложения как пациент (экран «Страница пациента»).

Основной сценарий:

Пациент нажимает на кнопку «пройти опрос» (экран «Страница пациента»)

Открывается доступный опрос (экран «Страница пациента с открытым опросом»)

Пациент отвечает на все доступные вопросы(экран «Страница пациента с открытым опросом»)

Пациент нажимает на кнопку «загрузить файлы»

Пациент загружает файлы (экран «Страница с загруженным изображением»)

Пациент нажимает на кнопку «отправить результаты»(экран «Страница с загруженным изображением»)

Происходит успешная отправка результатов(экран «Страница после успешного отправления опроса»)

Альтернативный сценарий:

Пациент нажимает на кнопку «пройти опрос» (экран «Страница пациента»)

Открывается доступный опрос (экран «Страница пациента с открытым опросом»)

Пациент отвечает не на все доступные вопросы(экран «Страница пациента с открытым опросом»)

Пациент нажимает на кнопку «отправить результаты»(экран «Страница с загруженным изображением»)

Происходит ошибка отправка результатов(экран «Страница с ошибкой при отправке результата»)

**Сценарий использования – "Просмотр рекомендаций врача"**

Действующее лицо: Пациент.

Начальные условия: Пользователь авторизован на сайт приложения как пациент (экран «Страница пациента»). Рекомендации есть.

Основной сценарий:

Пациент нажимает на рекомендацию(экран «Страница пациента»)

Появляется текст рекомендации(экран «Страница с открытыми рекомендациями»)

**Сценарий использования – "Загрузка подписанных документов"**

Действующее лицо: Пациент.

Начальные условия: Пользователь авторизован на сайт приложения как пациент (экран «Страница пациента»). Рекомендации есть.

Основной сценарий:

Пациент нажимает кнопку "загрузить" (экран «страница пациента»)

Пациент загружает подписанные документы (экран «страница с загруженными документами»)

Пациент нажимает на кнопку «отправить»(экран «страница с загруженными документами»)

Пациент переходит на страницу с успешным отправлением документов(экран «страница после успешного отправления документов»

Альтернативный сценарий:

Пациент нажимает кнопку "загрузить" (экран «страница пациента»)

Пациент не загружает подписанные документы (экран «страница с загруженными документами»)

Пациент нажимает на кнопку «отправить»(экран «страница с загруженными документами»)

Пациент переходит на страницу с ошибкой отправления документов (экран «страница после неуспешного отправления документов»)

**Сценарий использования – "Получение неподписанных документов"**

Действующее лицо: Пациент.

Начальные условия: Пользователь авторизован на сайт приложения как пациент (экран «Страница пациента»).

Основной сценарий:

Пациент нажимает кнопку "получить " (экран экран «Страница пациента»)

Пациент нажимает на кнопку «скачать» (экран «страница с полученными неподписанными документами»)

**Сценарий использования – "Выбор пациента"**

Действующее лицо: Доктор.

Начальные условия: Пользователь авторизован на сайт приложения как пациент (экран «Выбор пациента»).

Основной сценарий:

Доктор выбирает из списка пациента (экран «Выбор пациента»)

Доктор переходит к странице пациента(экран «Страница доктора с выбранным пациентом»)

**Сценарий использования – "Создания опроса"**

Действующее лицо: Доктор.

Начальные условия: Пользователь авторизован на сайт приложения как пациент (экран «Выбор пациента»).

Основной сценарий:

Доктор нажимает кнопку "опрос" (экран «страница доктора с выбранным пациентом»)

Открывается страница с созданием опроса (экран «страница создания опроса»)

Доктор нажимает на поле выбора типа вопроса(экран «страница создания опроса»)

Появляется выпадающее меню с типами вопросов(экран «выбор типа вопроса»)

Доктор пишет опрос (экран «страница создания опроса»)

Доктор нажимает на кнопку «добавить вопрос» (экран «страница создания опроса»)

Доктор переходит на страницу с успешным добавлением вопроса (экран «страница после успешного создания опроса»)

**Сценарий использования – "Создание рекомендации"**

Действующее лицо: Доктор.

Начальные условия: Пользователь авторизован на сайт приложения как пациент (экран «Выбор пациента»).

Основной сценарий:

Доктор нажимает кнопку "рекомендации" (экран «страница доктора с выбранным пациентом»)

Открывается страница с созданием рекомендации(экран «страница с созданием рекомендации»)

Доктор пишет рекомендацию(экран «страница с созданием рекомендации»)

Доктор нажимает на кнопку «создать» (экран «страница с созданием рекомендации»)

Доктор переходит на страницу с успешным созданием рекомендации(экран «страница после успешного создания рекомендации»)

**Сценарий использования – "Просмотр ответов на опрос"**

Действующее лицо: Доктор.

Начальные условия: Пользователь авторизован на сайт приложения как пациент (экран «Выбор пациента»).

Основной сценарий:

Доктор нажимает кнопку "результаты опроса" (экран «страница доктора с выбранным пациентом»)

Открывается страница с результатами опросов(экран «страница с результатами опроса пациента»)

Доктор нажимает на кнопку «получить»(экран «страница с результатами опроса пациента»)

Доктор получает возможность сказать прикрепленные к опросу файлы(экран «Страница с результатами опроса и прикрепленными документами»)

**Сценарий использования – "Загрузка документов для подписания"**

Действующее лицо: Доктор.

Начальные условия: Пользователь авторизован на сайт приложения как пациент (экран «Выбор пациента»).

Основной сценарий:

Доктор нажимает кнопку "загрузить файлы" (экран «страница доктора с выбранным пациентом»)

Доктор загружает документы на подписание(экран «страница после загрузки документов для пациента»)

Доктор нажимает на кнопку «отправить результаты»(экран «страница после загрузки документов для пациента»)

Доктор переходит на страницу с успешным созданием документов(экран «страница после успешного создания документа доктором»

Альтернативный сценарий:

Доктор нажимает кнопку "загрузить файлы" (экран «страница доктора с выбранным пациентом»)

Доктор не загружает документы на подписание (экран «страница доктора с выбранным пациентом»)

Доктор нажимает на кнопку «отправить результаты» (экран «страница доктора с выбранным пациентом»)

Доктор переходит на страницу с ошибкой отправления документов(экран «страница после неуспешного создания документов для пациента»

**Сценарий использования – "Загрузка документов для подписания"**

Действующее лицо: Доктор.

Начальные условия: Пользователь авторизован на сайт приложения как пациент (экран «Выбор пациента»).

Основной сценарий:

Доктор нажимает кнопку "загрузить файлы" (экран «страница доктора с выбранным пациентом»)

Доктор загружает документы на подписание(экран «страница после загрузки документов для пациента»)

Доктор нажимает на кнопку «отправить результаты»(экран «страница после загрузки документов для пациента»)

Доктор переходит на страницу с успешным созданием документов(экран «страница после успешного создания документа доктором»

Альтернативный сценарий:

Доктор нажимает кнопку "загрузить файлы" (экран «страница доктора с выбранным пациентом»)

Доктор не загружает документы на подписание (экран «страница доктора с выбранным пациентом»)

Доктор нажимает на кнопку «отправить результаты» (экран «страница доктора с выбранным пациентом»)

Доктор переходит на страницу с ошибкой отправления документов(экран «страница после неуспешного создания документов для пациента»

**Сценарий использования – "Получение подписанных документов"**

Действующее лицо: Доктор.

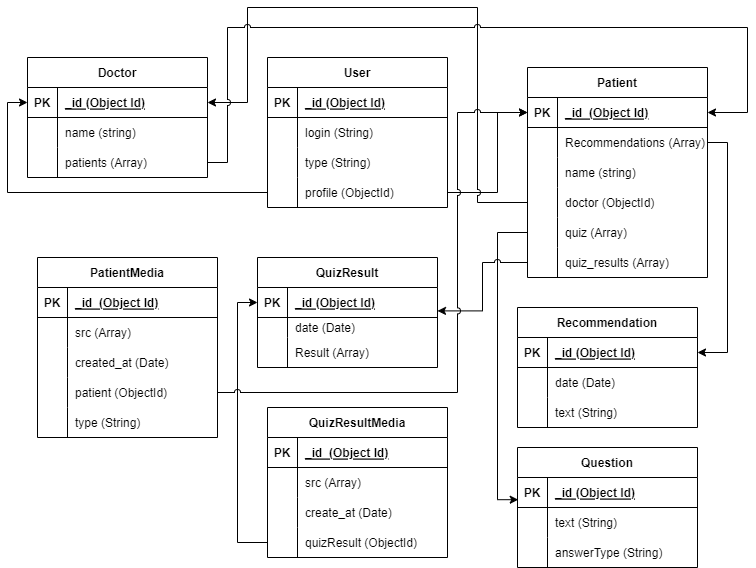
Начальные условия: Пользователь авторизован на сайт приложения как пациент (экран «Выбор пациента»).

Основной сценарий:

Доктор нажимает кнопку "получить" (экран «страница доктора с выбранным пациентом»)

Доктор нажимает на кнопку «скачать» (экран «страница с полученными подписанными документами»)

1. **Модель данных**
2. Графическое представление



1. Описание коллекций

|  |  |  |
| --- | --- | --- |
| **User** | | |
| **Наименование** | **Тип** | **Размеры (байты)** |
| login | String | [3; 15] |
| role | String | 6 или 7 |
| profile | ObjectId | 12 |

Хранит логин, роль и ссылку на доктора/пациента.

Память: 15+7+12 = 34 байт

|  |  |  |
| --- | --- | --- |
| **Doctor** | | |
| **Наименование** | **Тип** | **Размеры** |
| name | String | 20 |
| patients | Array of ObjectIds | N\*12 |

Хранит информацию о имени доктора и его пациентах.

Память: 20+N\*12 байт

|  |  |  |
| --- | --- | --- |
| **Patient** | | |
| **Наименование** | **Тип** | **Размеры** |
| name | String | 20 |
| doctor | ObjectId | 12 |
| quiz | Array of ObjectIds | N\*12 |
| quiz\_results | Array of ObjectIds | N\*12 |
| recommedations | Array of ObjectIds | N\*12 |

Хранит имя, лечащего врача, опрос, результат опроса и рекомендации пациента.

Память: байт

|  |  |  |
| --- | --- | --- |
| **PatientMedia** | | |
| **Наименование** | **Тип** | **Размеры** |
| src | Array of String | N\*28 |
| created\_at | Date | 29 |
| patient | ObjectId | 12 |
| type | String | 6 или 8 |

Хранит путь к файлам, дату создания, ссылку на пациента и подписаны ли файлы.

Память: байт

|  |  |  |
| --- | --- | --- |
| **Question** | | |
| **Наименование** | **Тип** | **Размеры** |
| answerType | String | 7 |
| text | String | 20 |

Хранит тип ответов и текст вопроса.

Память:

|  |  |  |
| --- | --- | --- |
| **QuizResult** | | |
| **Наименование** | **Тип** | **Размеры** |
| date | Date | 29 |
| Result | Array of Object ({answer: String, questionId: ObjectId}) | N\*(25+12) |

Хранит дату ответа на опрос и результат (ответ и ссылку на вопрос).

Память: байт

|  |  |  |
| --- | --- | --- |
| **QuizResultMedia** | | |
| **Наименование** | **Тип** | **Размеры** |
| src | Array of String | N\*28 |
| create\_at | Date | 29 |
| quizResult | ObjectId | 12 |

Хранит приложенные к опросу файлы (пути, дата создания и ссылку на ответ).

Память: байт

|  |  |  |
| --- | --- | --- |
| **Recommendation** | | |
| **Наименование** | **Тип** | **Размеры** |
| date | Date | 29 |
| text | String | 50 |

Хранит информацию о рекомендации (дата создания и текст рекомендации).

Память:

1. Оценка удельного объема информации, хранимой в модели.

|  |  |  |
| --- | --- | --- |
| Коллекция | Верхняя граница по памяти (байты) для одного экземпляра | Зависимость размера |
| User | 34 | Линейная зависимость от кол-ва докторов и пациентов |
| Doctor |  | Линейно зависит от кол-во пользователей |
| Patient |  | Линейной зависит от кол-ва вопросов, от рекомендаций, от ответа на опрос |
| PatientMedia |  | Линейно зависит от кол-ва файлов |
| QuizResult |  | Линейно зависит от кол-ва ответов |
| QuizResultMedia |  | Линейно зависит от кол-ва фалов |
| Question |  | Линейно зависит от кол-ва вопросов |
| Recommendation |  | Линейно зависит от кол-ва рекомендаций |

Кол-во Users = кол-во Doctors + кол-во Patients.

Пусть будет два пациента и два врача. У каждого по врачу.

Коллекция Users увеличится на 34\*4 байт.

Коллекция Doctors на 2\*(20+12) байт.

У пациента может быть записан только один доктор. При создании опроса к пациенту добавляются в массив Id вопросов, а после ответа на них – ещё и Id ответов (все ответы на один опрос – один экземпляр QuizResult). На каждый ответ – рекомендация.

Получается, что если в опросе – 12 вопросов и нужно загрузить 2 файла, то коллекция Patients увеличится в объеме на 2(кол-во пациентов)\*(32 + 10\*12 (Id вопросов) + 12(Id ответа) + 12(Id рекомендации)) байт, QuizResults увеличится на 2\*(29 + 10\*12(ответы)) байт, Recommendations на 2\*79 байт, Question на 2\*27\*12 байт, QuizResultMedia на 2\*(41 + 2\*28) байт. Итого:

байт.

При отправке каждому пациенту пяти документов на подпись увеличится коллекция PatientMedias на байт.

Итого: кбайта.

1. Избыточность модели (отношение между фактическим объемом модели и “чистым” объемом данных).

Vf = Фактический объем = 2396 байт.

Vc = Чистый объем = 2228 байт.

Vc/Vf = 0.92988313856

1. Направление роста модели при увеличении количества объектов каждой сущности.

|  |  |  |
| --- | --- | --- |
| Коллекция | Увеличение БД, байты | Объяснение |
| User | 34 | Зависит от Patient и Doctor |
| Doctor | При | Объем БД увеличивается на документ в *Users,* на документ в *Doctors* и на 12 байт (добавление к пациенту) |
| Patient | При  937 | Объем БД увеличивается на документ в *Users*, на документ в *Patients*, на документов в *Questions*, на документов в *Recommendations* и *QuizResults*, на документ в *QuizResultMedia*, на документ в *PatientMedias*, на 12\*4 байт (добавление к доктору в массив patients, в массивы пациента) |
| PatientMedia | 49+5\*28=189 | Объём БД увеличивается на документ в *PatientMedias* |
| QuizResult | При  29+5\*37+41+5\*28+12\*2+79  =498 | Объём БД увеличивается на документ в *QuizResults,* *QuizResultMedias,* *Recommendations,* на 12\*2 байт(добавление в массивы пациента), (подразумеваем, что вопросов уже есть и не учитываем) |
| QuizResultMedia | При  41+5\*28=181 | Объём БД увеличивается на документ в *QuizResultMedias* (добавление в *QuizResultMedias* подразумевает существование связного документа в *QuizResults*, поэтому не учитываем его) |
| Question | 5\*27+29+5\*37+41+5\*28+79  =609 | Объём БД увеличивается на документов в *Questions*, на документ в *QuizResults,* *QuizResultMedias,* *Recommendations*, на байтов (добавление в массивы пациента) |
| Recommendation |  | Объём БД увеличивается на документ в Recommendations и на 12 байт (ObjectId в массив recommendations пациента) |

V = 2228 байт

|  |  |
| --- | --- |
| Коллекция | Увеличение БД, % |
| User | 1.5 |
| Doctor |  |
| Patient |  |
| PatientMedia |  |
| QuizResult | 22.4 |
| QuizResultMedia |  |
| Question |  |
| Recommendation |  |

Самые ресурсоемкие – создание опроса (вопросы) и ответов на опрос (много связей с другими сущностями).

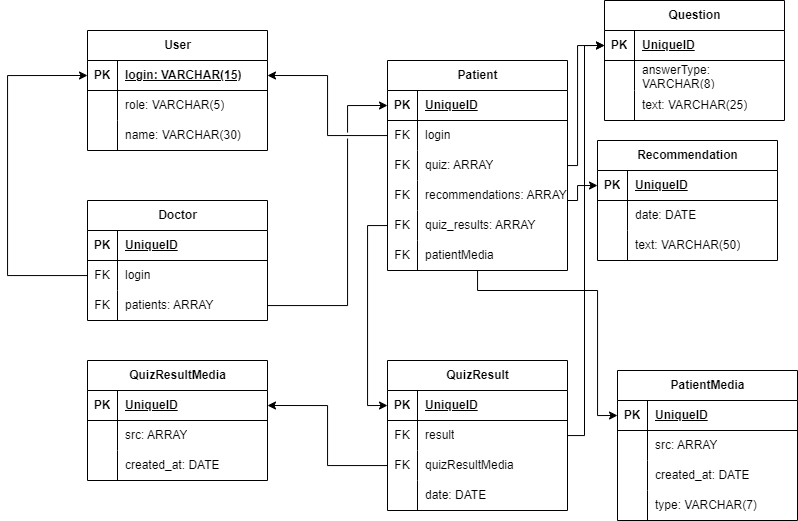
1. Запросы к модели, с помощью которых реализуются сценарии использования.
2. Текст запросов, кол-во задействованных коллекций

|  |  |  |
| --- | --- | --- |
| Описание | Текст | Кол-во задействованных коллекций |
| Получение пользователя по логину | Users.findOne({ login: body.login })  Для доктора:  Doctors.findById(user.profile)  Patients.find({ doctor: found.\_id }).select({ '\_id': 1, 'name': 1 })  Для пациента:  Patients.findById(user.profile)  Если есть назначенный доктор: Doctors.findById(found.doctor).select({ 'name': 1, '\_id': 1 })  Иначе: Doctors.find({}).select({ 'name': 1, '\_id': 1 }) | 3 |
| Получение пользователя | Patients.findById(patientId).populate('quiz').populate('recommendations').populate('quiz\_results') | 1 |
| Получение опроса по Id пациента | Patients.findById(patientId).populate('quiz') | 1 |
| Создание опроса | Получение пациента: Patients.findById(body.patientId)  Создание вопросов | 2 |
| Получение ответов(файлы) на опросы | QuizResultMedia.findOne({quizResult: QuizResultId}) | 1 |
| Создание ответа на опрос | Создание QuizResult, QuizResultMedia  Поиск пациента: Patients.findById(JSON.parse(patientId)) | 3 |
| Получение неподписанных файлов | PatientMedia.findOne({ patient: patientId, type: 'Unsigned' }) | 1 |
| Создание неподписанных документов | Создание PatientMedia | 1 |
| Получение подписанных файлов | PatientMedia.findOne({ patient: patientId, type: 'Signed' }) | 1 |
| Создание подписанных документов | Создание PatientMedia | 1 |
| Получение рекомендаций | Patients.findById(patientId).populate('recommendations') | 1 |
| Создание рекомендации | Создание Recommendation  Поиск пациента: Patients.findById(patientId) | 2 |
| Поменять доктора | Поиск пациента: Patients.findById(patientId) | 1 |

Пример хранения:



1. **SQL**
2. Графическое представление



1. Описание назначений таблиц

|  |  |  |
| --- | --- | --- |
| **User** | | |
| **Наименование** | **Тип** | **Размеры (байты)** |
| login | VARCHAR | [3; 15] |
| role | VARCHAR | 6 или 7 |
| name | VARCHAR | 20 |

Хранит логин, роль и ссылку на доктора/пациента.

Память: 15+7+20 = 42 байт

|  |  |  |
| --- | --- | --- |
| **Doctor** | | |
| **Наименование** | **Тип** | **Размеры** |
| login | VARCHAR | 15 |
| patients | Array | N\*8 |

Хранит информацию о имени доктора и его пациентах.

Память: 15+N\*8 байт

|  |  |  |
| --- | --- | --- |
| **Patient** | | |
| **Наименование** | **Тип** | **Размеры** |
| login | VARCHAR(15) | 15 |
| patientMedia | Foreign Key | 8 |
| quiz | Array | N\*8 |
| quiz\_results | Array | N\*8 |
| recommedations | Array | N\*8 |

Хранит имя, лечащего врача, опрос, результат опроса и рекомендации пациента.

Память: байт

|  |  |  |
| --- | --- | --- |
| **PatientMedia** | | |
| **Наименование** | **Тип** | **Размеры** |
| src | Array of VARCHAR(28) | N\*28 |
| created\_at | DATE | 3 |
| type | VARCHAR(8) | 6 или 8 |

Хранит путь к файлам, дату создания, ссылку на пациента и подписаны ли файлы.

Память: байт

|  |  |  |
| --- | --- | --- |
| **Question** | | |
| **Наименование** | **Тип** | **Размеры** |
| answerType | VARCHAR(8) | 7 |
| text | VARCHAR(20) | 20 |

Хранит тип ответов и текст вопроса.

Память:

|  |  |  |
| --- | --- | --- |
| **QuizResult** | | |
| **Наименование** | **Тип** | **Размеры** |
| date | DATE | 3 |
| Result | Array of Object ({answer: VARCHAR(25), questionId: Foreign Key}) | N\*(25+8) |
| quizResultMedia | Foreign Key | 8 |

Хранит дату ответа на опрос и результат (ответ и ссылку на вопрос).

Память: байт

|  |  |  |
| --- | --- | --- |
| **QuizResultMedia** | | |
| **Наименование** | **Тип** | **Размеры** |
| src | ARRAY of VARCHAR(28) | N\*28 |
| create\_at | DATE | 3 |

Хранит приложенные к опросу файлы (пути, дата создания и ссылку на ответ).

Память: байт

|  |  |  |
| --- | --- | --- |
| **Recommendation** | | |
| **Наименование** | **Тип** | **Размеры** |
| date | DATE | 3 |
| text | VARCHAR(50) | 50 |

Хранит информацию о рекомендации (дата создания и текст рекомендации).

Память:

1. Оценка удельного объема информации

Пусть будет 2 пациента, 2 врача (у каждого – по пациенту), каждому пациенту: 12 вопросов в опросе, 2 файла, приложенных к ответу, по рекомендации, 5 документов на подпись.

|  |  |
| --- | --- |
| Таблица | Объем, байты |
| User |  |
| Patient |  |
| Doctor |  |
| QuizResult |  |
| QuizResultMedia |  |
| PatientMedia |  |
| Question |  |
| Recommendation |  |
| Сумма, байты | |
|  | |

1. Избыточность модели

Vf = Фактический объем = 3324 байт.

Vc = Чистый объем = байт.

Vc/Vf = 0.759

1. Запросы к модели

|  |  |  |
| --- | --- | --- |
| Описание | Текст | Кол-во задействованных коллекций |
| Получение пользователя по логину | Для доктора:  SELECT id, name FROM DOCTORS WHERE login = body.login;  SELECT id, name FROM PATIENTS WHERE doctor = doctor.id;  Для пациента:  SELECT id, name FROM PATIENTS WHERE login = body.login;  Если есть назначенный доктор:  SELECT name, id from DOCTORS WHERE id = patient.doctor;  Иначе:  SELECT name, id from DOCTORS; | 2 |
| Получение пользователя | SELECT \* FROM PATIENTS ‘p’ WHERE id = patientId JOIN QUESTIONS ‘q’ on p.quiz = q.id  JOIN RECOMMENDATIONS ‘r’ on p.recommnedations = r.id  JOIN QUIZRESULTS ‘qr’ on p.quiz\_results = qr.id; | 4 |
| Получение опроса по Id пациента | SELECT \* FROM PATIENTS ‘p’ WHERE id = patientId JOIN QUIZZES ‘q’ on p.quiz = q.id; | 2 |
| Создание опроса | Получение пациента:  SELECT \* FROM PATIENTS ‘p’ WHERE id = patientId;  Создание вопросов:  INSERT INTO QUESTIONS VALUES (answerType, text) | 2 |
| Получение ответов(файлы) на опросы | SELECT \* FROM QUIZRESULTMEDIAS WHERE id = QuizResultId; | 1 |
| Создание ответа на опрос | Создание QuizResult, QuizResultMedia:  INSERT INTO QUIZRESULTS VALUES (date, result, quizResultMedia);  INSERT INTO QUIZRESULTMEDIAS VALUES (src, created\_at);  Поиск пациента: SELECT \* FROM PATIENTS ‘p’ WHERE id = patientId; | 3 |
| Получение неподписанных файлов | SELECT \* FROM PATIENTMEDIAS WHERE id = patientMedia AND type = ‘Unsigned’ | 1 |
| Создание неподписанных документов | INSERT INTO PATIENTMEDIAS VALUES (src, created\_at, type); | 1 |
| Получение подписанных файлов | SELECT \* FROM PATIENTMEDIAS WHERE id = patientMedia AND type = ‘Signed | 1 |
| Создание подписанных документов | INSERT INTO PATIENTMEDIAS VALUES (src, created\_at, type); | 1 |
| Получение рекомендаций | SELECT \* FROM PATIENTS ‘p’ WHERE id = patientId JOIN RECOMMENDATIONS ‘r’ on p.recommendations = r.id; | 1 |
| Создание рекомендации | INSERT INTO RECOMMENDATIONS VALUES (text, date);  Поиск пациента:  SELECT \* FROM PATIENTS ‘p’ WHERE id = patientId; | 2 |
| Поменять доктора | UPDATE PATIENTS  SET  doctor = doctorId  WHERE  id = patientId | 1 |

**Сравнение моделей.**

Сравнив объемы SQL и NoSQL баз данных, можно сказать, что во втором случае БД занимает меньше места. Коллекций использовано одинаковое количество. Из-за того, что SQL не обходится без JOIN, NoSQL выигрывает по времени. Однако NoSQL использует дублирование данных, в результате чего запросы на удаление/изменение могут также быть затратными по времени.

В результате, обе модели имеют свои преимущества и минусы. При наличии ограничений на типы данных и структуру хранящейся информации, следует выбрать реляционную модель. При высоких требованиях к скорости работы и нечетких требованиях к типам данных следует выбрать нереляционную модель.

**5. Разработанное приложение**

**Краткое описание**

Приложение реализовано на React+typescript+Nodejs, для с работы с MongoDB используется библиотека mongoose[3]. При помощи нее создаются подключение к бд, создание коллекций для хранения информации о пациентах, доктора, подписанных и неподписанных документах, а также об опросах и рекомендациях. Для загрузки файлов использовалась библиотека multer[4]. На фронтенде для создания интерфейса была использована библиотека MUI.

**Схема экранов приложения**

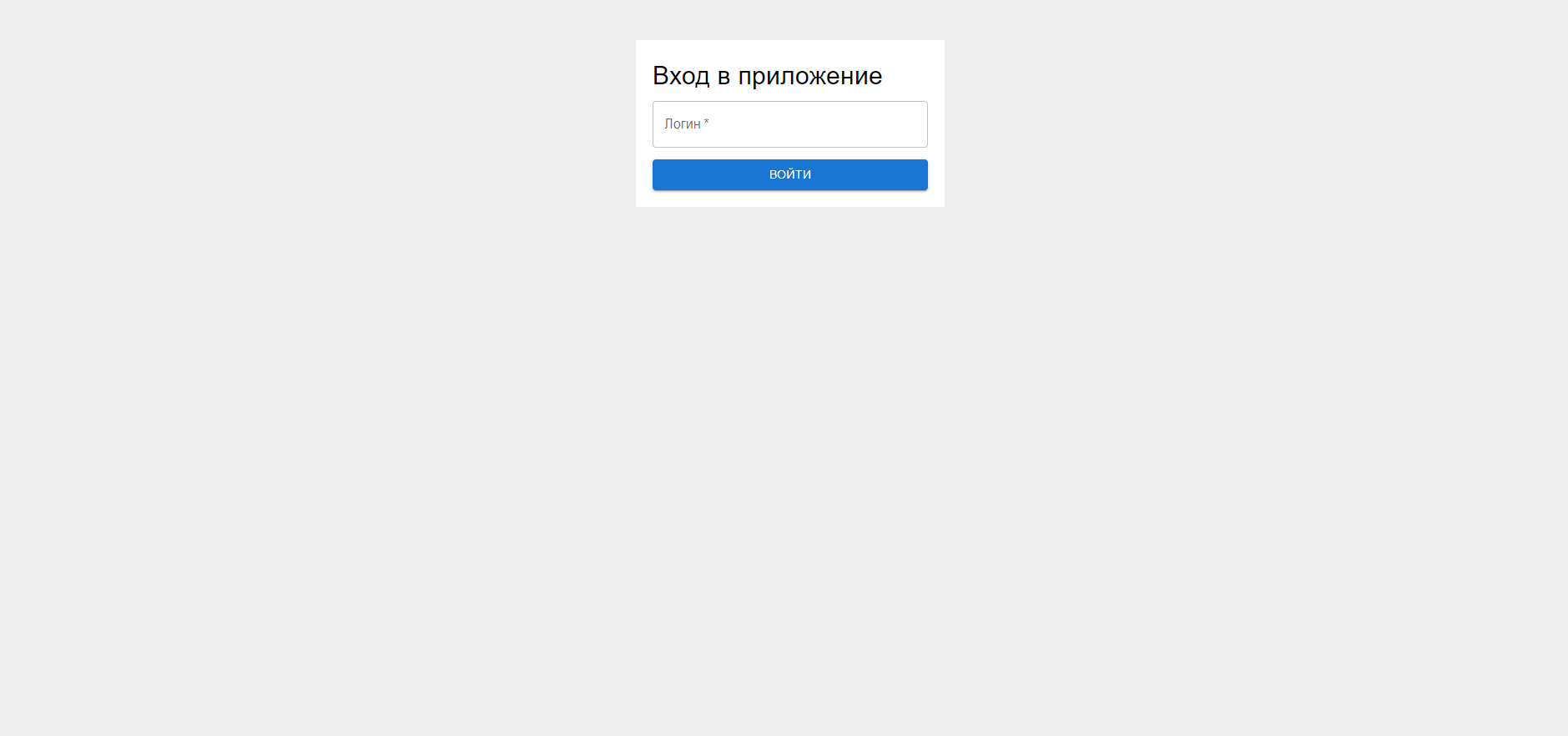


Рис. 2 Экран входа

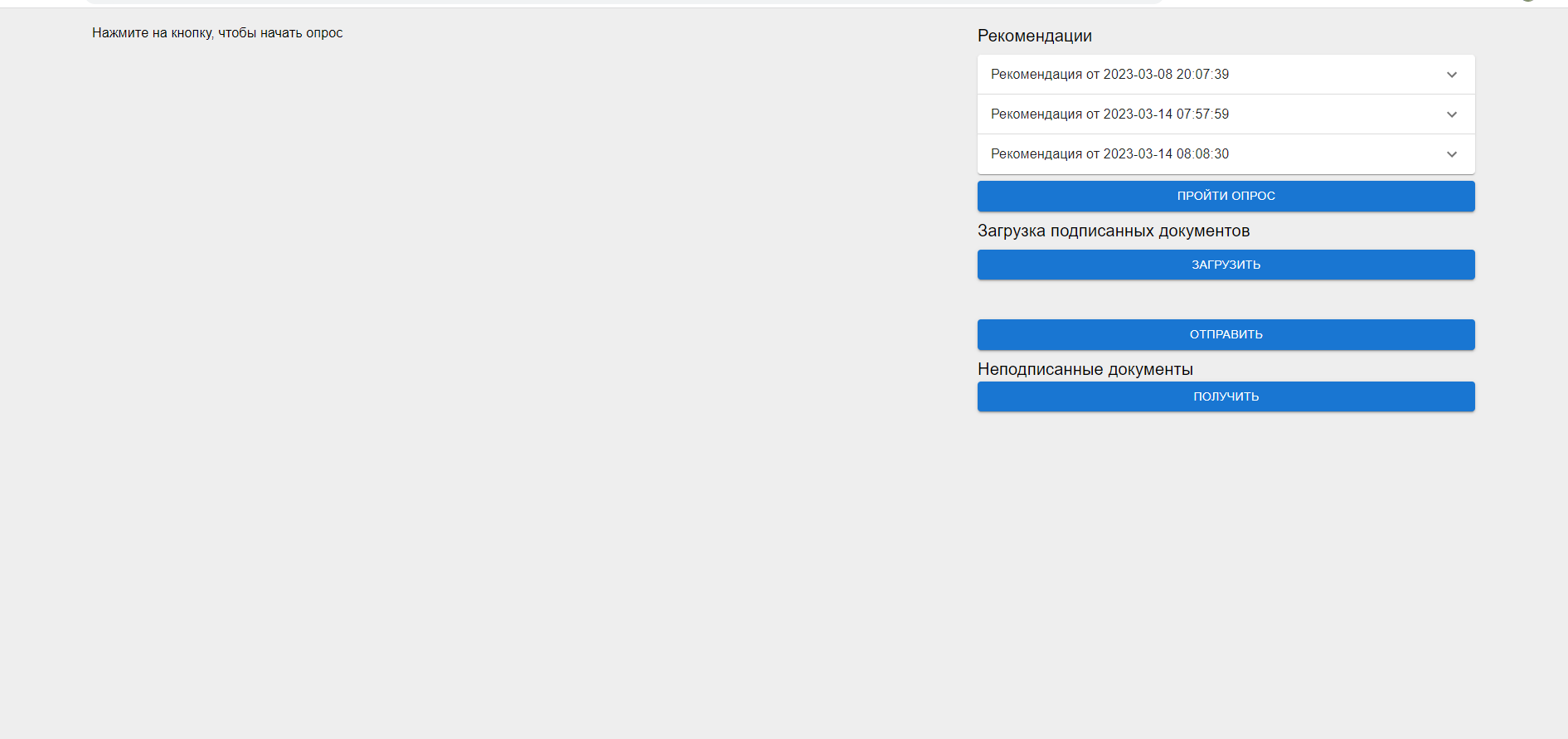


Рис. 3 Экран пациента

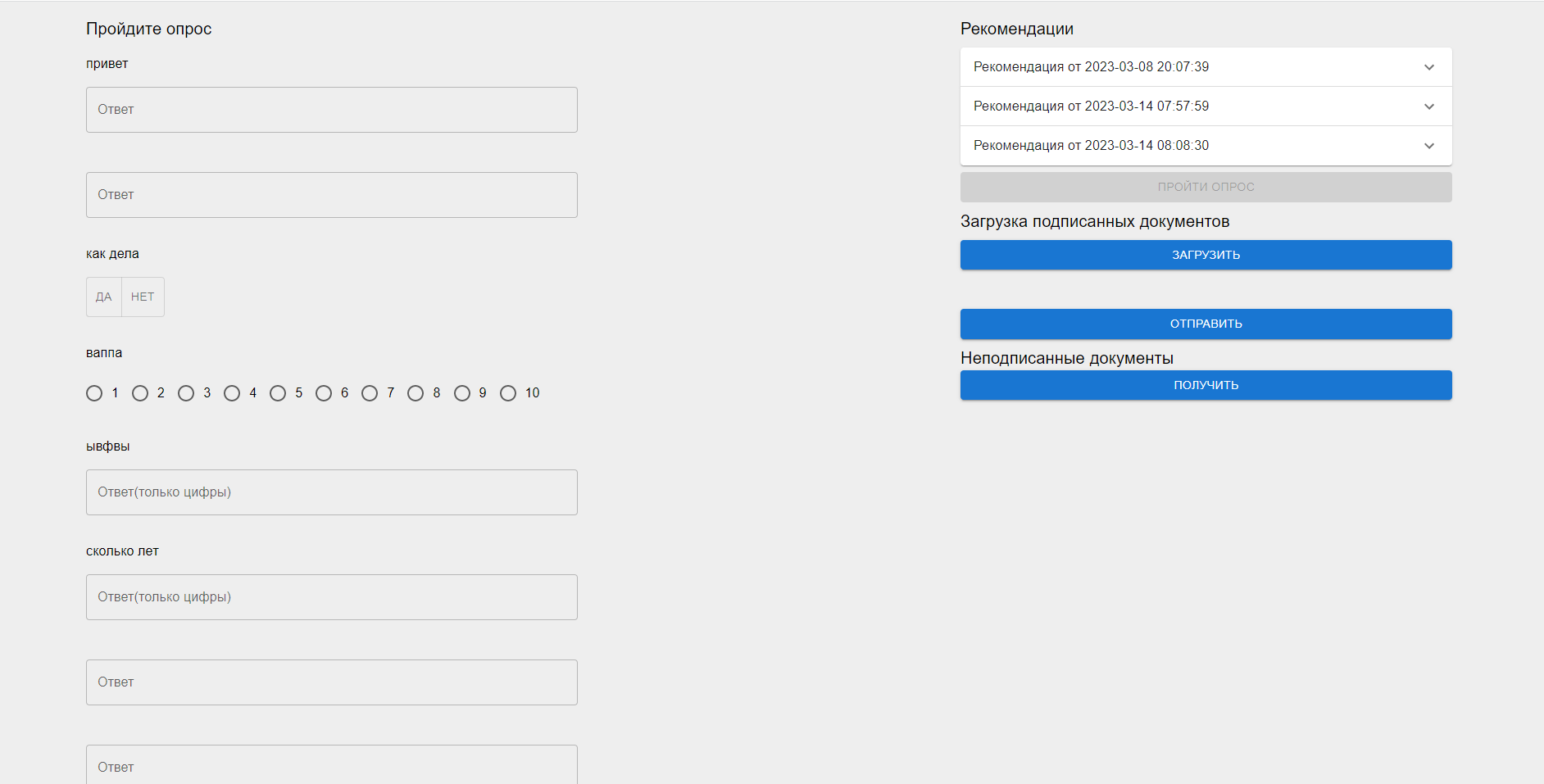


Рис. 4 Экран пациента с открытым опросом

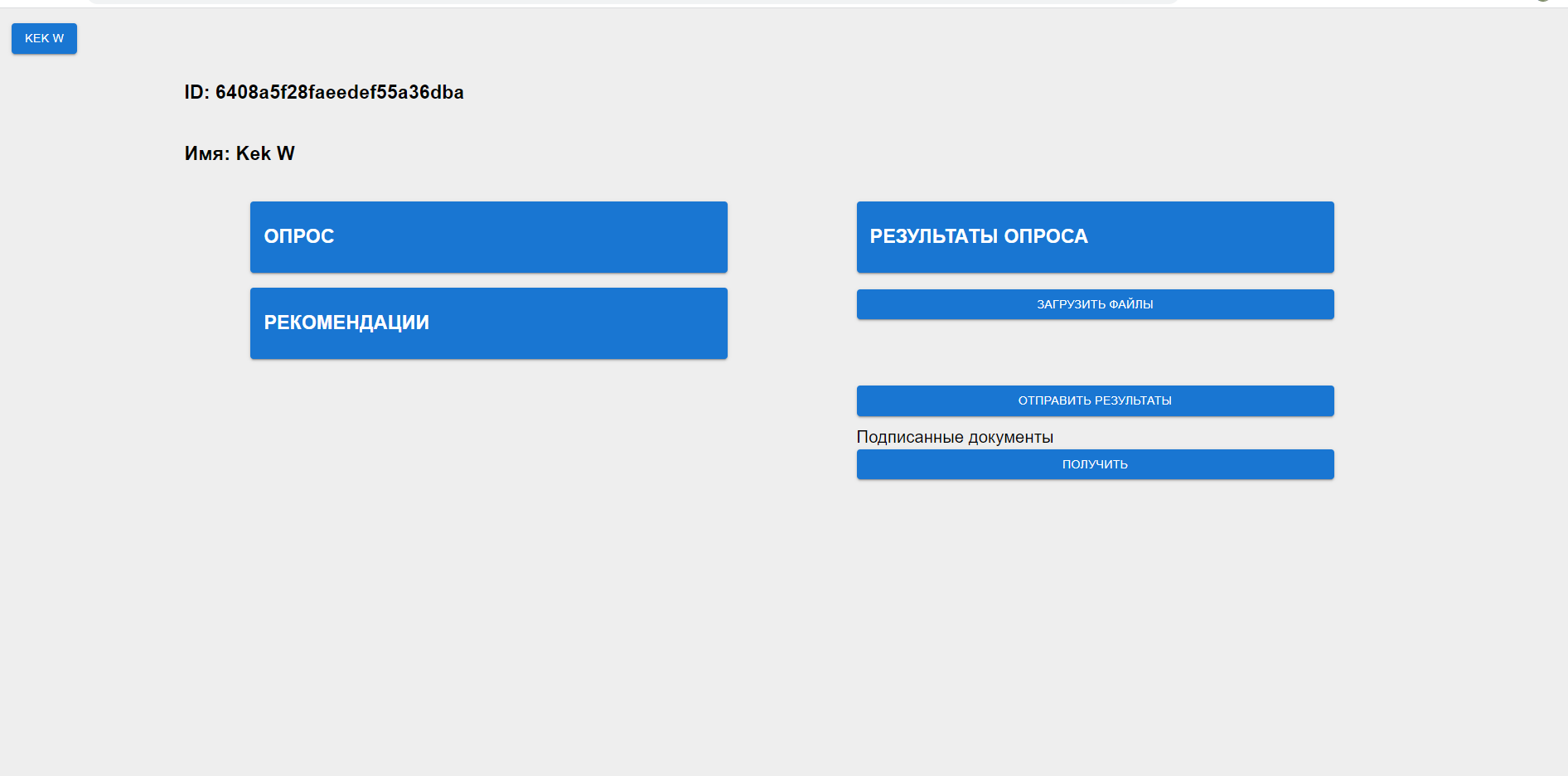


Рис. 5 Экран доктора

**Использованные технологии**

СУБД: MongoDB

Backend: Nodejs, Mongoose, multer

Frontend: React, typescript, mui, axios, redux-toolkit, sass .

**Ссылки на приложение**

Ссылка на github: https://github.com/moevm/nosql2h22-monitoring

**6.Вывод**

В ходе работы было разработано web-приложение мониторинга пациентов, позволяющее пользователям взаимодействовать с базой данных: просмотр содержимого СУБД, добавление новых элементов, как текстовых, так и файловых.

**Будущее развитие решения**

Планируется улучшить пользовательский интерфейс, добавить больше логики: статистика опросов, графики прохождения опрос, подписания документов при помощи цифровых подписей.