**МИНОБРНАУКИ РОССИИ**
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**
**Кафедра МОЭВМ**

# ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

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

**Тема: Информационная система стоматологической клиники**

Студент гр. 0303 _____ Мыратгелдиев А.М.

Студент гр. 0303 _____ Торопыгин А.С.

Студент гр. 0303 _____ Тишкин М.А.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2023

**ЗАДАНИЕ**

Студенты

Мыратгилдиев А.М.

Торопыгин А.С.

Тишкин М.А.

Группа 0303

Тема задания: Информационная система стоматологической клиники

Исходные данные:

Задача - сделать сервис для управления стоматологической клиникой. Пользователи - администраторы (те, которые сидят на reception), врачи, пациенты. Необходимые (но недостаточные фичи) - карты пациентов, статистика для бухгалтерии, статистика и динамика больных, хранение снимков, склад и обслуживание оборудования

Содержание пояснительной записки:

“Содержание”

“Введение”

“Сценарий использования”

“Модель данных”

“Разработанное приложения”

“Заключение”

“Приложение А. Документация по сборке и развертыванию приложения”

Предполагаемый объем пояснительной записки:

Не менее 20 страниц.

Дата выдачи задания: 21.12.2023

Дата сдачи реферата: 21.12.2023

Дата защиты реферата: 21.12.2023

| | | |
|---|---|---|
| Студент 0303 | _____ | Мыратгелдиев А.М. |
| Студент 0303 | _____ | Торопыгин А.С. |
| Студент 0303 | _____ | Тишкин М.А. |
| Преподаватель | _____ | Заславский М.М. |

## АННОТАЦИЯ

В рамках данного курса предполагалось разработать в команде приложение на одну из предложенных тем. Была выбрана тема: Информационная система стоматологической клиники. Для выполнения задания предлагается использовать СУБД MongoDB.

Найти исходный код и всю дополнительную информацию можно по ссылке: https://github.com/moevm/nosql2h23-dentist.

## SUMMARY

Within the framework of this course it was supposed to develop in a team an application on one of the proposed topics. The topic was chosen: Dental clinic information system. To perform the task it is suggested to use MongoDB DBMS.

You can find the source code and all additional information at the link: https://github.com/moevm/nosql2h23-dentist.

# СОДЕРЖАНИЕ


| | | |
|---|---|---|
| | Введение | 6 |
| 1. | Сценарии использования | 7 |
| 1.1. | Макет UI | 7 |
| 1.2. | Сценарии использования для задачи | 7 |
| 2. | Модель данных | 11 |
| 2.1. | Нереляционная модель данных | 11 |
| 2.2. | Аналог модели данных для SQL СУБД | 18 |
| 2.3. | Сравнение моделей | 25 |
| 3. | Разработанное приложение | 27 |
| 3.1. | Описание | 27 |
| 3.2. | Использованные технологии | 27 |
| 3.3. | Снимки экрана приложения | 27 |
| | Заключение | 32 |
| | Список использованных источников | 33 |
| | Приложение А. Документация по сборке и развертыванию приложения | 34 |

# ВВЕДЕНИЕ

Цель работы – создать высокопроизводительное и удобное решение для для управления стоматологической клиникой.

Было решено разработать веб-приложение, которое позволит хранить информацию о пациентах, информацию об их посещениях, докторах, у которых они проходят лечение и пр.

# 1. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

## 1.1. Макет UI

Разработанный макет приложения доступен по ссылке: https://www.figma.com/file/oZtj828pW3W2QQZWU8YgSs/nosql-course-project?type=design&node-id=0%3A1&mode=design&t=JWvdJ5YbkfhjpTKk-1

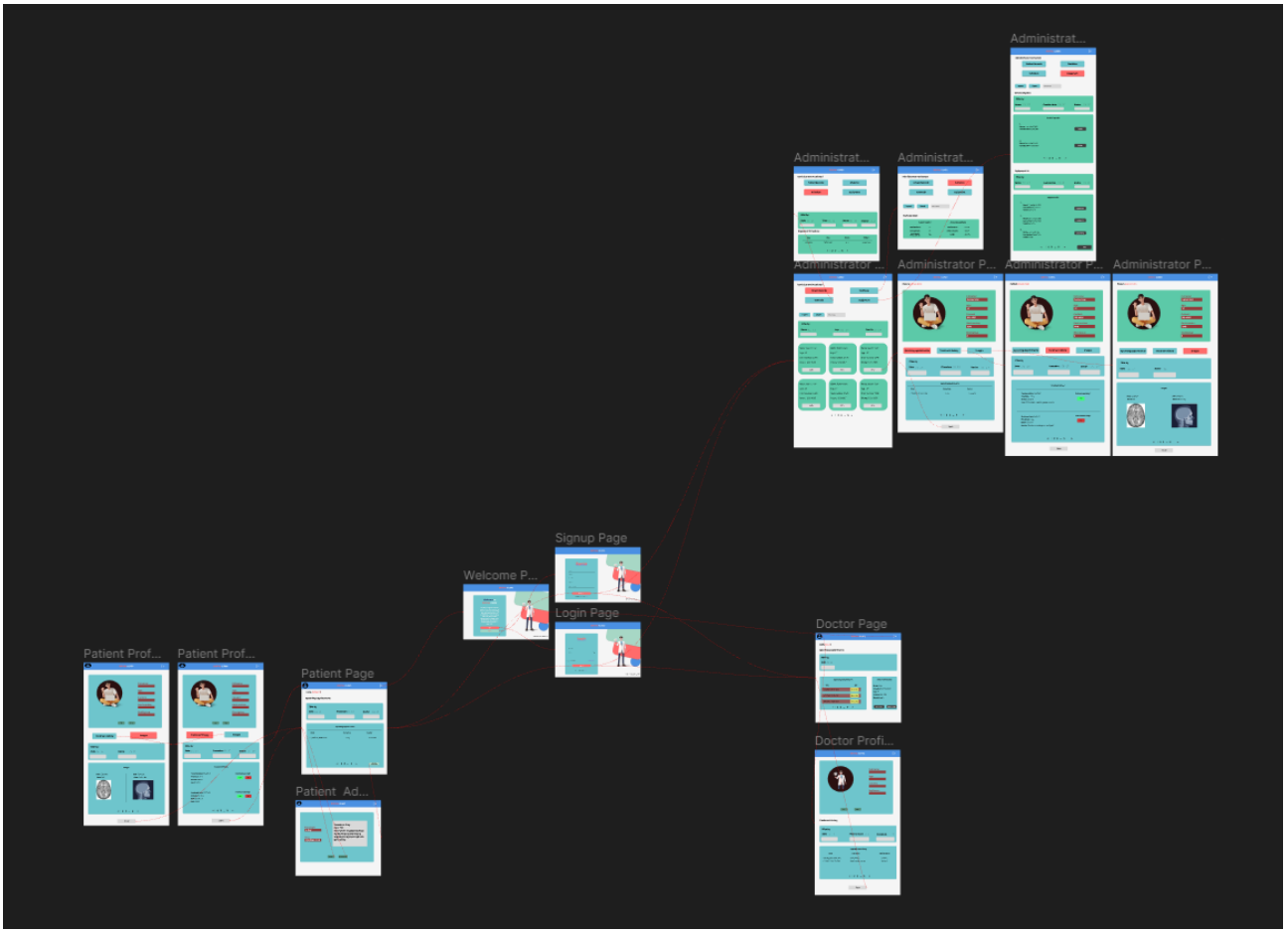


Рисунок 1 – Макет UI

## 1.2. Сценарии использования

Сценарий использования “Вход”.

Действующее лицо: Любой пользователь

Основной сценарий

Предусловие: пользователь не вошёл в систему.

1. Пользователь открывает главную страницу сайта
2. Пользователь нажимает на кнопку "Log in"
3. Пользователь попадает на страницу входа в профиль
4. Пользователь заполняет поля Username и Password и нажимает на кнопку Log In
5. Пользователь переходит на страницу клиники с уровнем прав, доступным пользователю (пациент, врач, администратор)

Альтернативный сценарий:

1. Пользователь ввёл неправильные данные (Username или Password)
2. Пользователь получает предупреждение об ошибке и ему предоставляется возможность авторизоваться ещё раз.

Сценарий использования “Выход”.

Действующее лицо: Любой пользователь

Предусловие: пользователь вошёл в систему.

1. Пользователь нажимает на кнопку выхода из системы
2. Пользователь попадает на главную страницу сайта

Сценарий использования “Просмотр предстоящих встреч”.

Действующее лицо: Пациент

1. Пациент входит на сайт и попадает на страницу пациента

2. На странице пациента есть информация о предстоящих встречах

Сценарий использования “Просмотр предстоящих встреч с пациентами”.

Действующее лицо: Доктор

1. Врач входит на сайт и попадает на страницу врача
2. На странице врача есть информация о предстоящих встречах

Сценарий использования “Просмотр предстоящих встреч”.

Действующее лицо: Администратор

1. Администратор входит на сайт и попадает на страницу Администратора
2. Администратор переключается во вкладку предстоящих записей
3. Администратор может видеть предстоящие записи

Сценарий использования “Просмотр карт пациентов”.

Действующее лицо: Администратор

1. Администратор входит на сайт и попадает на страницу Администратора
2. Администратор может найти информацию по пациентам, введя данные в поисковую строку
3. Администратор получает информацию по пациентам

# 2. МОДЕЛЬ ДАННЫХ

## 2.1. Нереляционная модель данных

**Описание назначений коллекций, типов данных и сущностей**

Коллекция patient

- _id типа ObjectId для хранения идентификационного номера пациента
- структура prof info - с полями email и password типа string для хранения логина пациента
- структура card - с полями name, surname, contact, bloodGroup типа string для хранения данных пациента, а также поле birthdate типа Date.
- массив images, хранящий структуры с полями date типа Date, name - string и img типа BinaryData

Коллекция doctor

- _id типа ObjectId для хранения идентификационного номера доктора
- поле name и surname типа String для хранения имени и фамилии доктора
- поле email и password типа String для хранения данных логина доктора
- поле contact типа String для хранения номера телефона доктора

Коллекция appointments

- _id типа ObjectId для хранения идентификационного номера посещения
- поле date типа Date для хранения даты посещения
- структура procedure - с полями name типа string и price типа Double для хранения данных процедуры.
- структура patientCard - с полями name, surname, contact, bloodGroup типа string для хранения данных пациента, а также поле birthdate типа Date.
- структура doctor - с полями name и surname типа String для хранения имени и фамилии доктора.

Коллекция administrator

- _id типа ObjectId для хранения идентификационного номера посещения
- поля name и surname типа String для хранения имени и фамилии администратора
- полям email и password типа string для хранения логина администратора

**Оценка удельного объема информации, хранимой в модели**

Коллекция patient. Допустим, что у каждого пациента в среднем 3 снимка и 5 записей в истории посещений. Тогда, для хранения x пациентов понадобится:

$$x * (12 + 42 + 52 + 3 * 5000000) = 15000106x$$

_id - тип ObjectID V = 12 байт

prof_info - тип Object V = 30 + 12 = 42

email - String V = 30 байт

password - String V = 12 байт

card - тип Object V = 15 + 15 + 8 + 12 + 2 = 52 байт

name - String V = 15 байт

surname - String V = 15 байт

birthdate - Date V = 8 байт

contact - String V = 12 байт

bloodGroup - String V = 2 байта

images - Array V_i = 8 + 20 + 5000000 = 5 МБайт

date - Date V = 8 байт

name - String V = 20 байт

img - Binary Data (допустим, что средний размер снимка 5 Мб) V = 5 МБайт

Коллекция doctor. Пусть на каждые 10 пациентов приходится 1 доктор. Тогда:

$$(x/10 + 1) * (12 + 94) = 106 * (x/10 + 1) = 106 + 68.6x = 106 + 69x$$

_id - ObjectID V = 12 байт

info - Object V = 15 + 15 + 30 + 12 + 12 + 10 = 94 байт

name - String V = 15 байт

surname - String V = 15 байт

email - String V = 30 байт

password - String V = 12 байт

contact - String V = 12 байт

experience - String V = 10 байт

Коллекция appointments. Пусть в среднем приходится 10 записей на каждого пациента x. Тогда

$$10x * (12 + 8 + 20 + 58 + 30) = 1280x$$

_id - ObjectID V = 12 байт

date - Date V = 8 байт

procedure - String V = 20 байт

patient - Object V = 15 + 15 + 8 + 12 + 6 + 2 = 58 байт

name - String V = 15 байт

surname - String V = 15 байт

birthdate - Date V = 8 байт

contact - String V = 12 байт

patientCard - String V = 6 байт

bloodGroup - V = 2 байта

doctor - Object V = 15 + 15 = 30 байт

name - String V = 15 байт

surname - String V = 15 байт

Коллекция administrator. Пусть в среднем 1 аккаунт администратора. Тогда:

$$12 + 15 + 15 + 30 + 12 = 84$$

_id - ObjectID V = 12 байт

name - String V = 15 байт

surname - String V = 15 байт

email - String V = 30 байт

password - String V = 12 байт

**Избыточность модели**

В БД дублируются данные о докторе о пациенте в историях лечений/посещений клиники, у новых пациентов список снимков может быть вообще пустым, а также некоторые пациенты могут не оставлять отзывов. Тогда получаем: V(x)= 7500452x + 106 + 69x + 420x + 1280x + 84 + 1440 + 940 + 280x + 800 = 7501716x + 2887

Отбросим свободный член, чтобы сравнить с фактическим объемом: V(x) = 7501716x. Отношение между фактическим и "чистым" объемом данных: $\frac{15002756x}{7502081} = 1.9$

**Запросы к модели, с помощью которых реализуются сценарии использования**

- **регистрация нового пациента**

```
db.patient.insert({
   _id: ObjectID(),
   prof_info: {
      email,
      password
   }
   card: {
      name,
```

```
        surname,
        birthdate,
        contact,
        bloodGroup
      },
      images: []
    })
```

- **поиск аккаунта пациента для входа в систему**

```
db.patient.find({
  prof_info: {
    email,
    password
  }
})
```

- **поиск всех предстоящих встреч**

```
db.appointments.find({})
```

- **поиск предстоящих встреч определенного пациента**

```
db.appointments.find({
  patientCard: {
    name,
    surname
  }
})
```

- **поиск предстоящих встреч определенного доктора**

```
db.appointments.find({
  doctor: {
    name,
    surname
```

```
    }
})
```

### 2.2. Аналог модели данных для SQL СУБД

Графическое представление модели изображено на рис. 2.

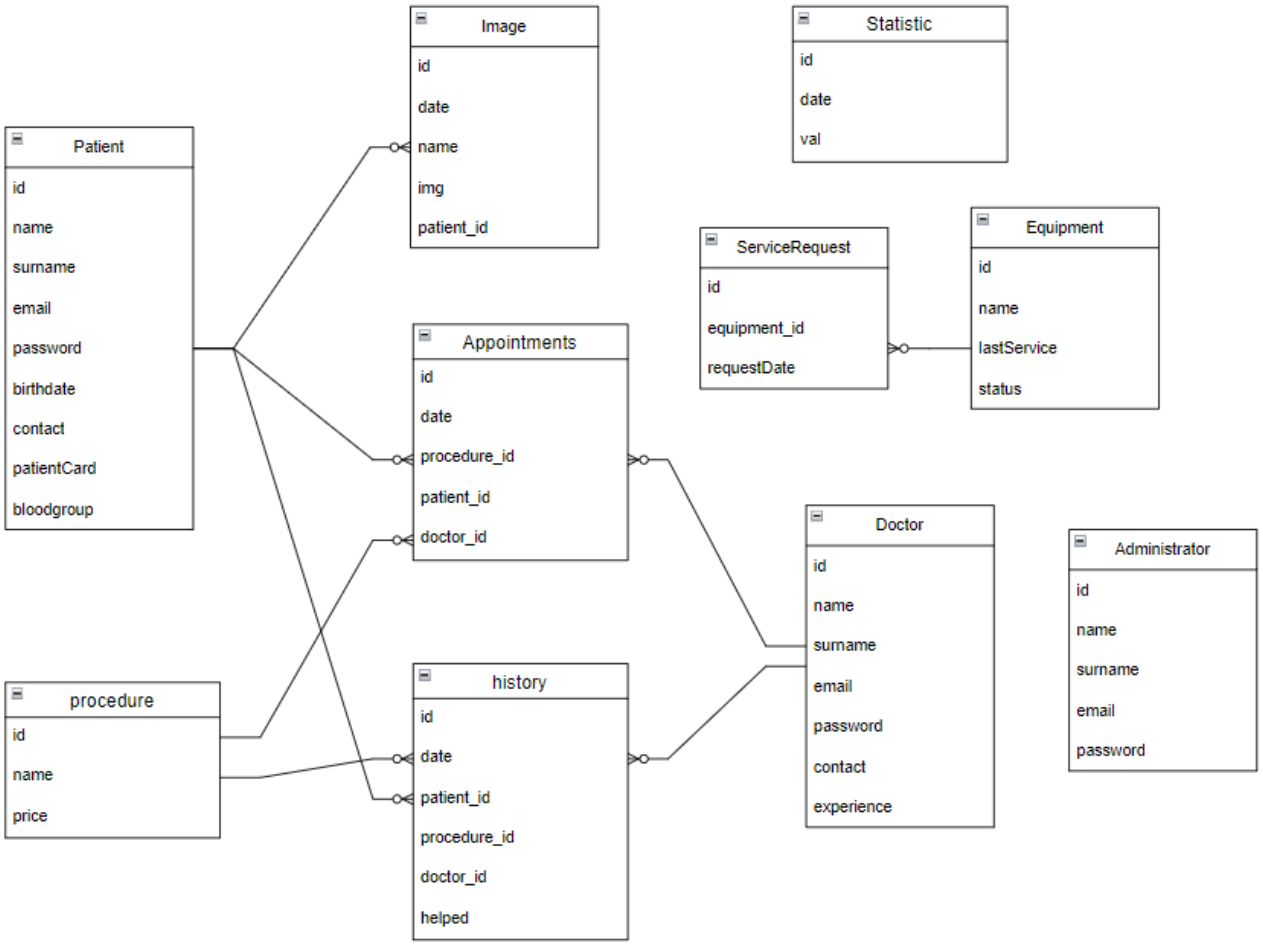


Рисунок 2 – модель реляционной базы данных

**Описание назначений коллекций, типов данных и сущностей**

Разница от нереляционной версии:

- наличие отдельной таблицы Image для хранения снимков
- отдельная таблица History для хранения истории

Поля и типы данных полей идентичны.

**Оценка удельного объема информации, хранимой в модели**

Таблица Patient. Оценим объем информации через x, где x - количество пациентов.

$$x * (8 + 15 + \ldots) = 102x$$

- _id - тип INT V = 8 байт
- name - VARCHAR V = 15 байт
- surname - VARCHAR V = 15 байт
- email - VARCHAR V = 30 байт
- password - VARCHAR V = 12 байт
- birthdate - Date V = 8 байт
- contact - VARCHAR V = 12 байт
- bloodGroup - VARCHAR V = 2 байта

Таблица Image. Пусть в среднем у каждого пациента 3 снимка. Тогда выразим через количество пациентов x

$$3x * (8 + 8 + 12 + 5000000 + 8) = 15000108x$$

- _id - тип INT V = 8 байт
- date - Date V = 8 байт
- name - VARCHAR V = 12 байт
- img - Binary Data (допустим, что средний размер снимка 5 Мб) V = 5 МБайт
- patient_id Int V = 8 байт

Таблица Appointments. Пусть в среднем 10 записей на каждого пациента x. Тогда

$$10x(8 + 8 + 8 + 8 + 8) = 450x$$

- _id - тип INT V = 8 байт
- date - Date V = 8 байт
- procedure_id - Int V = 8 байт
- patient_id - Int V = 8 байт
- doctor_id - Int V = 8 байт

Таблица Doctor. Пусть на каждые 10 пациентов приходится 1 доктор. Тогда

$$(x/10 + 1) * (8 + 15 + 15 + 30 + 12 + 8 + 12 + 8) = 108 + 10.5x = 11x + 108$$

- _id - тип INT V = 8 байт
- name - VARCHAR V = 15 байт
- surname - VARCHAR V = 15 байт
- email - VARCHAR V = 30 байт
- password - VARCHAR V = 12 байт
- birthdate - Date V = 8 байт
- contact - VARCHAR V = 12 байт
- experience - Int V = 8 байт

Таблица Administrator. Пусть в среднем 1 аккаунт админа. Тогда

$$8 + 15 + 15 + 30 + 12 = 80$$

- _id - тип INT V = 8 байт
- name - VARCHAR V = 15 байт
- surname - VARCHAR V = 15 байт
- email - VARCHAR V = 30 байт
- password - VARCHAR V = 12 байт

Тогда получим следующий объем данных, для хранения x пациентов. V(x) = 102x + 15000108x + 450x + 108 +11x + 80 = 15000671x + 188

**Запросы к модели, с помощью которых реализуются сценарии использования**

- Регистрация нового пациента

INSERT INTO Patient VALUES (1, "Ivan", "Ivanov", "qwe123@mail.ru", "qwe123asdfzx", "12.12.2000", "+74523418754", "1+")

- Поиск аккаунта пациента для входа в систему

SELECT * FROM Patient WHERE email=email, password=password;

- Поиск всех предстоящих встреч

SELECT * FROM Appointment;

- Поиск предстоящих встреч определенного пациента

```
SELECT *
 FROM Patient
        INNER JOIN Appointment ON Patient.id=Appointment.patient_id
WHERE Patient.name=name AND Patient.surname=surname;
```

- Поиск предстоящих встреч определенного доктора

```
SELECT *
FROM Doctor
   INNER JOIN Appointment ON Doctor.id=Appointment.doctor_id
WHERE Doctor.name=name AND Doctor.surname=surname;
```

- Подсчет прибыли

SELECT SUM(val) FROM Statistic;

- Просмотр оборудований клиники

SELECT * FROM Equipment;

- Просмотр оборудований, которые ремонтируются:

```
SELECT * FROM ServiceRequest
    JOIN Equipment ON Equipment.id=equipment_id
    JOIN Administrator ON Administrator.id=requester_id ;
```

### 2.3. Сравнение моделей

NoSQL требует больше памяти, по сравнению с SQL, так как в нем дублируются некоторые данные. SQL выигрывает по памяти, так как вместо того, чтобы хранить сами объекты целиком (как NoSQL), он хранит только id определенного элемента из другой таблицы.

По удобству запросов выигрывает NoSQL, так как ввиду дублирования данных в некоторых сущностях, нам не приходится JOIN-ить с другими коллекциями. Для некоторых запросов SQL приходится JOIN-ить несколько таблиц, что может сказать в скорости доступа.

# 3. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

## 3.1. Описание

Backend реализован при помощи фреймворка express на базе nodejs на языке программирования TypeScript с использованием базы данных MongoDB. Frontend реализован при помощи библиотеки React и CSS-фреймворка Tailwind. Для обработки запросов использована библиотека axios.

## 3.2. Использованные технологии

База данных: MongoDB

Backend: nodejs, express, TypeScript.

Frontend: React, Redux, axios, Tailwind CSS.

## 3.3. Снимки экрана приложения

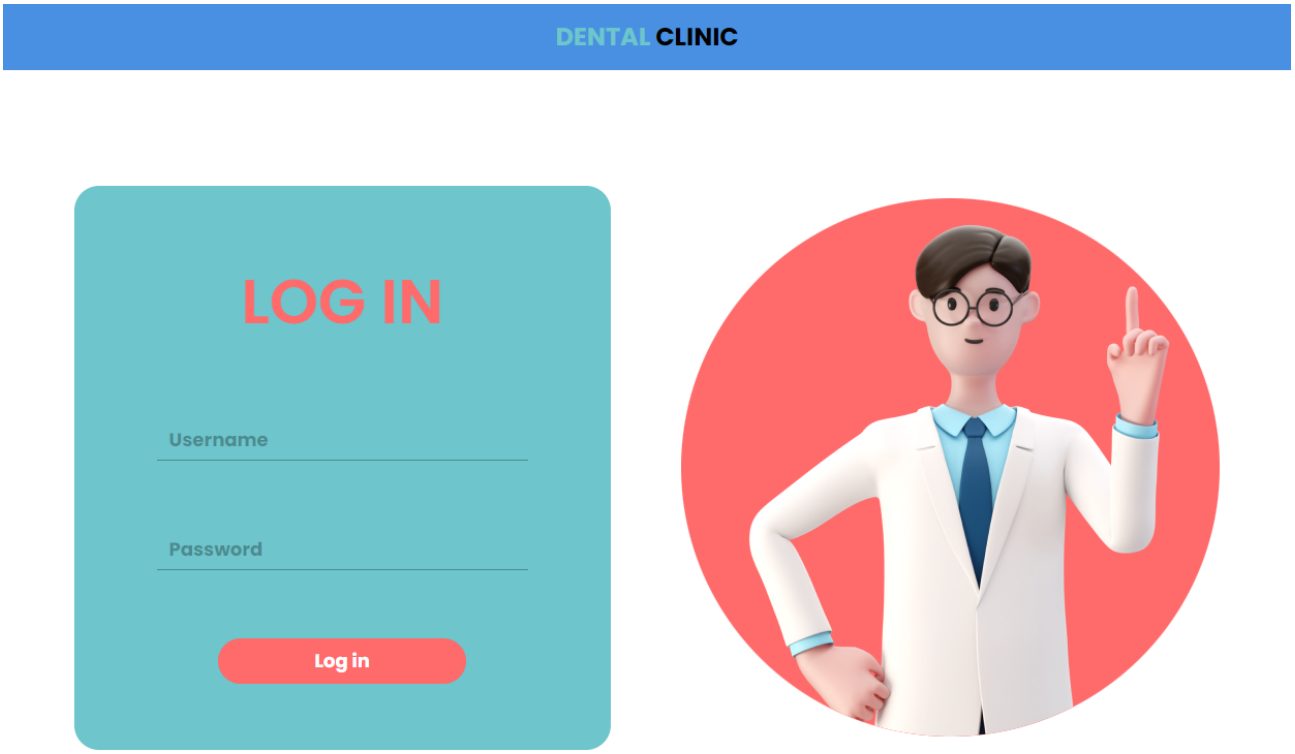

Рисунок 6 – Страница входа

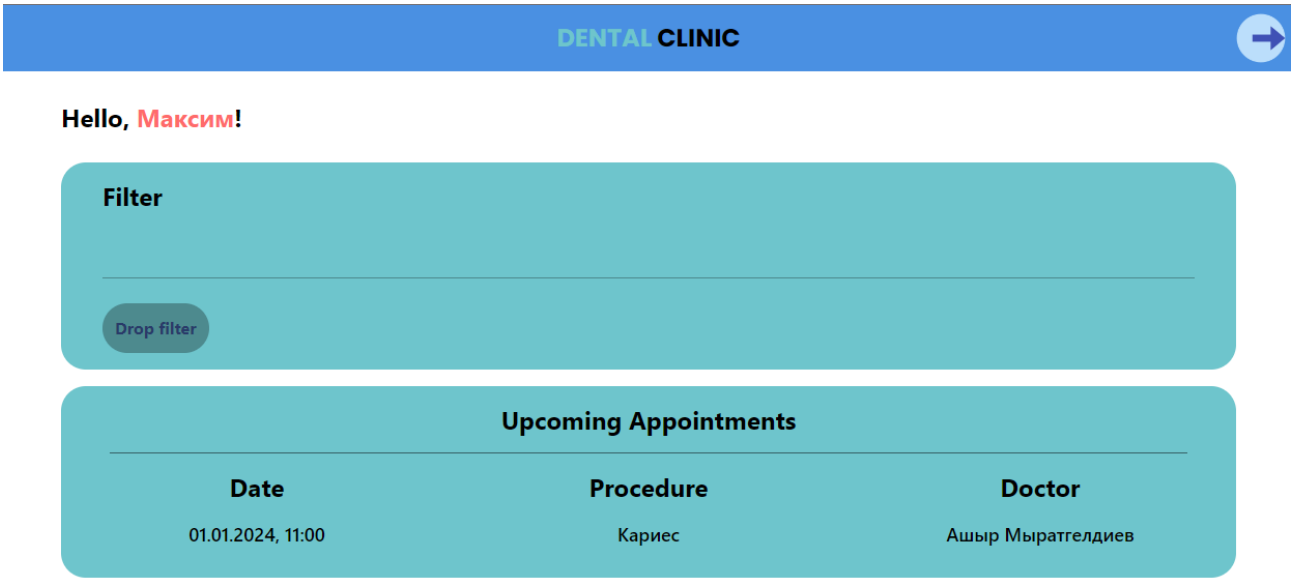


Рисунок 7 – Список предстоящих встреч пациента (страница пациента)

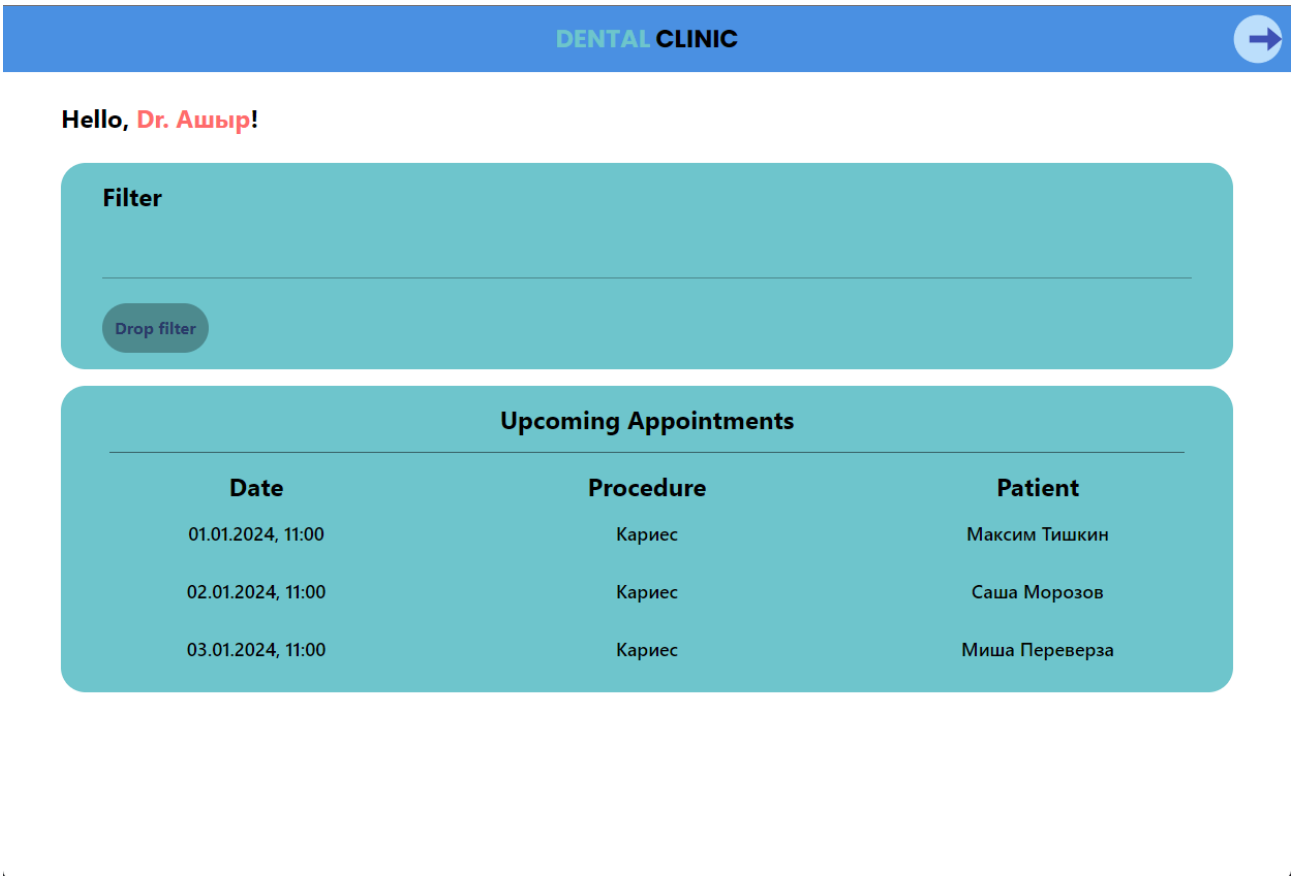


Рисунок 8 – Список предстоящих встреч доктора (страница доктора)

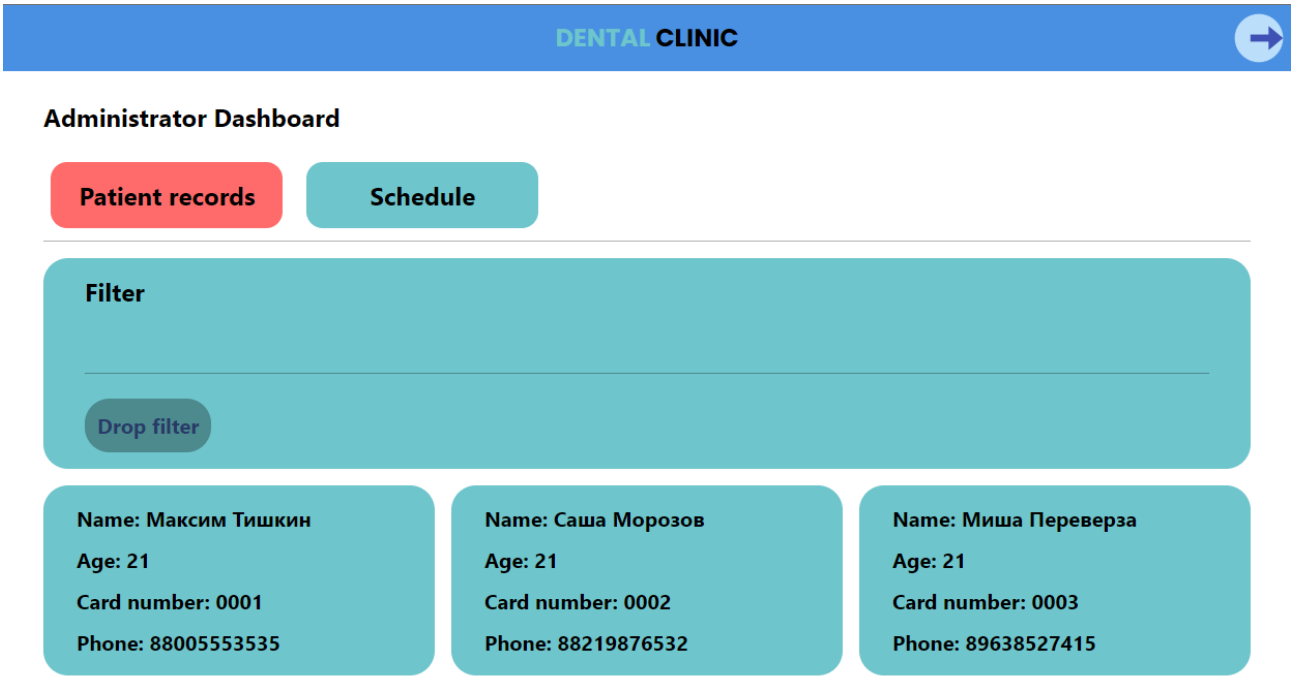

Рисунок 9 – Карточки пациентов (Страница админа)

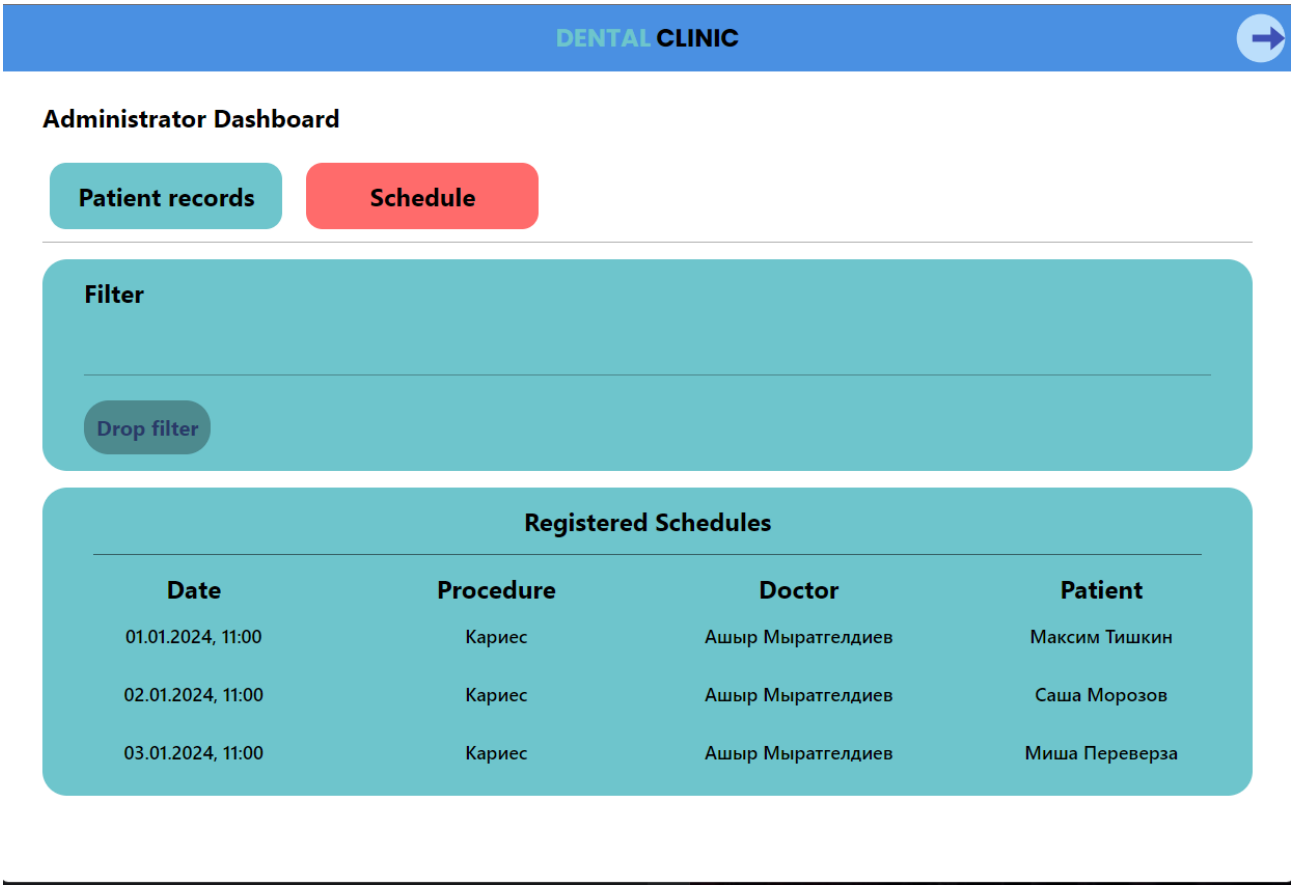


Рисунок 10 – Список предстоящих встреч (страница администратора)

# ЗАКЛЮЧЕНИЕ

В результате работы было разработано веб-приложение “Информационная система стоматологической клиники”, которое позволяет хранить информацию о предстоящих встречах пациентов, докторов и карточки пациентов.

a. Недостатки и пути для улучшения полученного решения

Добавление страницы для регистрации пользователей по ролям

Добавление возможности назначать встречу

Хранение снимков пациентов

Добавление страницы для просмотра истории лечений

b. Будущее развитие решения

Разработка мобильной версии приложения

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


1. Репозиторий веб-приложения: [электронный ресурс]. URL: https://github.com/moevm/nosql2h23-dentist
2. MongoDB The Developer Data Platform: [электронный ресурс]. URL: https://www.mongodb.com/

## ПРИЛОЖЕНИЕ А

## ДОКУМЕНТАЦИЯ ПО СБОРКЕ И РАЗВЕРТЫВАНИЮ ПРИЛОЖЕНИЯ

1. Скачать проект из репозитория (указан в ссылках на приложение)
2. Установить зависимости для клиентской части (в папке frontend) и для серверной части (в папке backend)
3. В папке backend запустить веб-сервер через команду npm run devStart
4. В папке frontend запустить клиентскую часть через команду npm run dev
5. Открыть приложение в браузере по адресу http://localhost:5173/