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

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

по дисциплине «Введение в нереляционные базы данных»

Тема: Электронный журнал успеваемости (MongoDB)

Студенты гр. 6381 _____ Вергейчик Г.Л.

_____ Вероха В.Н.

_____ Ширяев Я.А.

Преподаватель _____ Заславский М.М.

Санкт-Петербург
2019

ЗАДАНИЕ

Студенты Вергейчик Г.Л., Вероха В.Н., Ширяев Я.А.

Группа 6381

Тема проекта: Разработка приложения для электронного журнала успеваемости

Исходные данные:
Необходимо реализовать приложение, использующее СУБД MongoDB

Содержание пояснительной записки:
«Содержание», «Введение», «Качественные требования к решению», «Сценарий использования», «Модель данных», «Разработанное приложение», «Заключение», «Список использованных источников»

Предполагаемый объем пояснительной записки:
Не менее 10 страниц.

Дата выдачи задания:

Дата сдачи ИДЗ:

Дата защиты ИДЗ:

АННОТАЦИЯ

В рамках данного курса необходимо было реализовать приложение на одну из поставленных тем. Была выбрана тема для создания приложения, которое хранит данные об успеваемости учащихся. В приложении должна осуществляться функция импорта/ экспорта данных.

SUMMARY

As part of this course, it was necessary to implement the application on one of the topics posed. A theme was chosen to create an application that stores student performance data. The application should carry out the function of import / export of data.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы – создать приложение электронного журнала для хранения и отслеживания информации об успеваемости учащихся.

Было решено разработать веб-приложение для просмотра списка студентов, редактирования, добавления и удаления информации, а также просмотра статистики.

1. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

Требуется реализовать веб-приложение, использующее СУБД MongoDB.

2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

2.1. Макеты пользовательского интерфейса

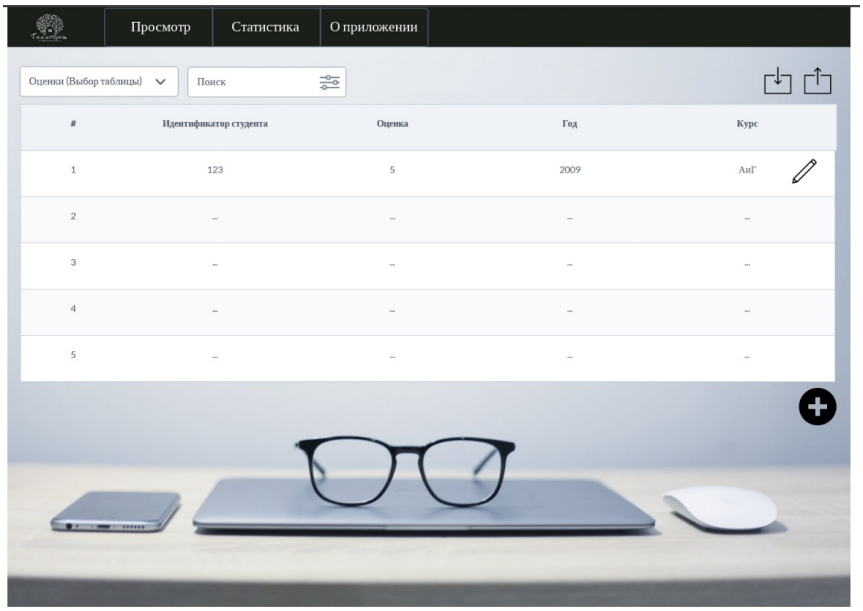

Рисунок 1 — Страница «Просмотр»

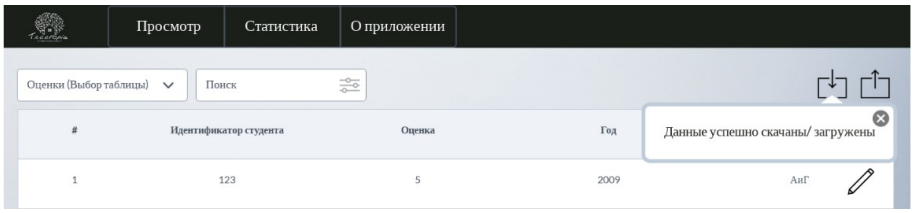

Рисунок 2 — Интерфейс работы кнопки «Импорт/ экспорт» на странице «Просмотр»

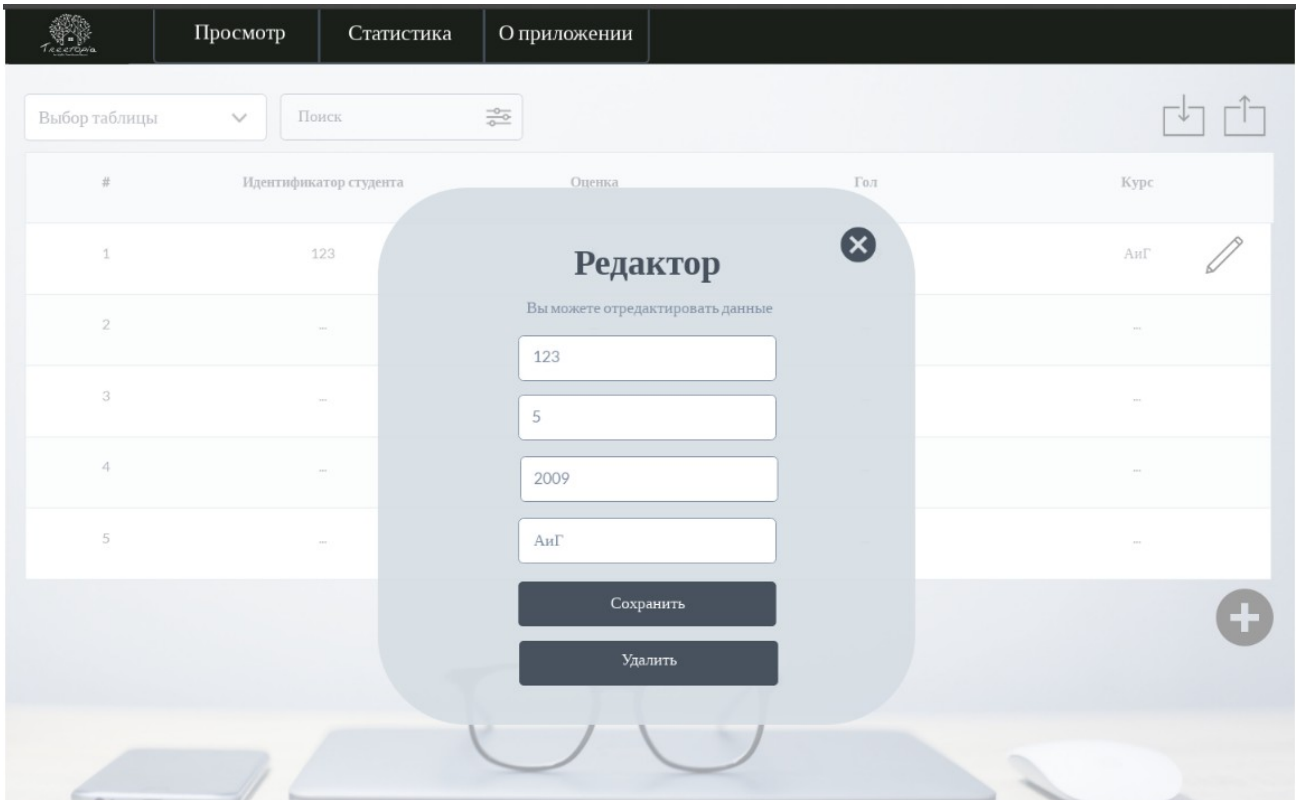

Рисунок 3 — Окно редактирования записи на странице «Просмотр»

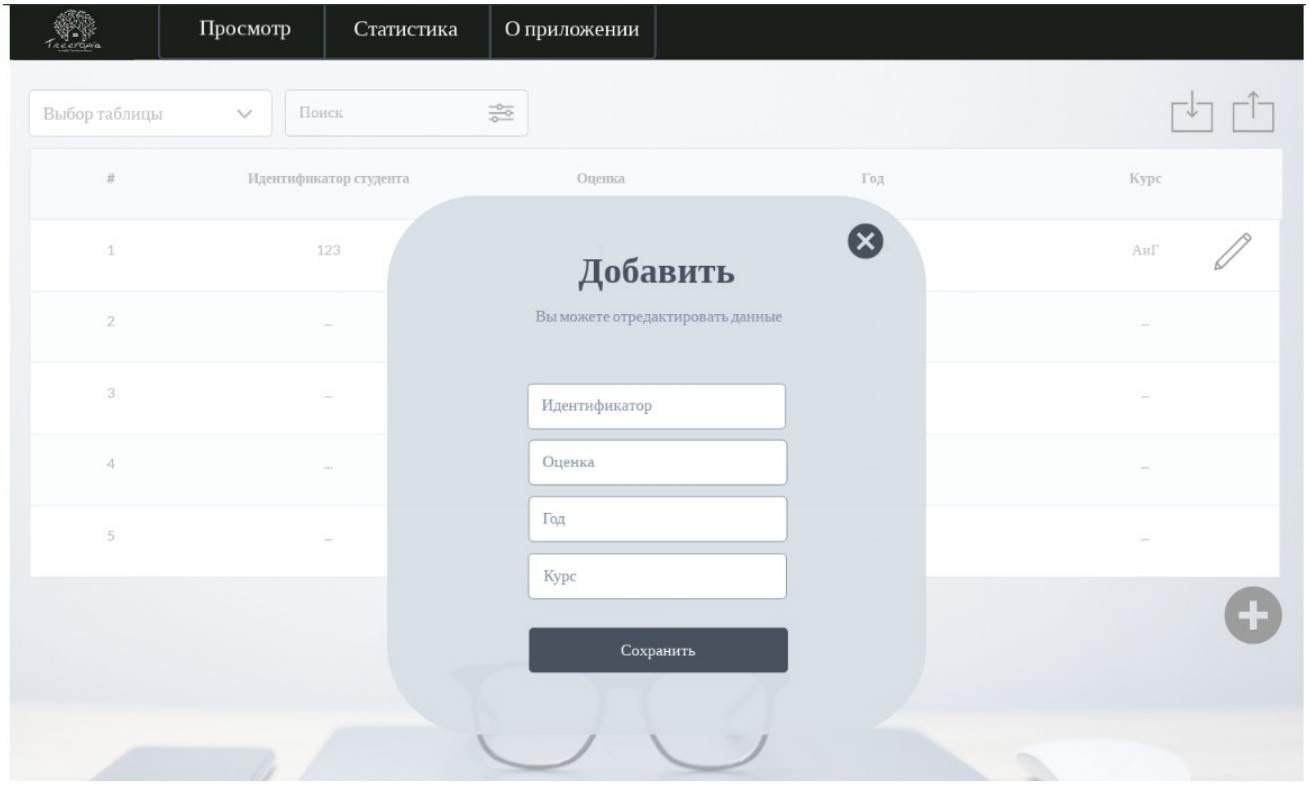

Рисунок 4 — Окно добавления записи на странице «Просмотр»

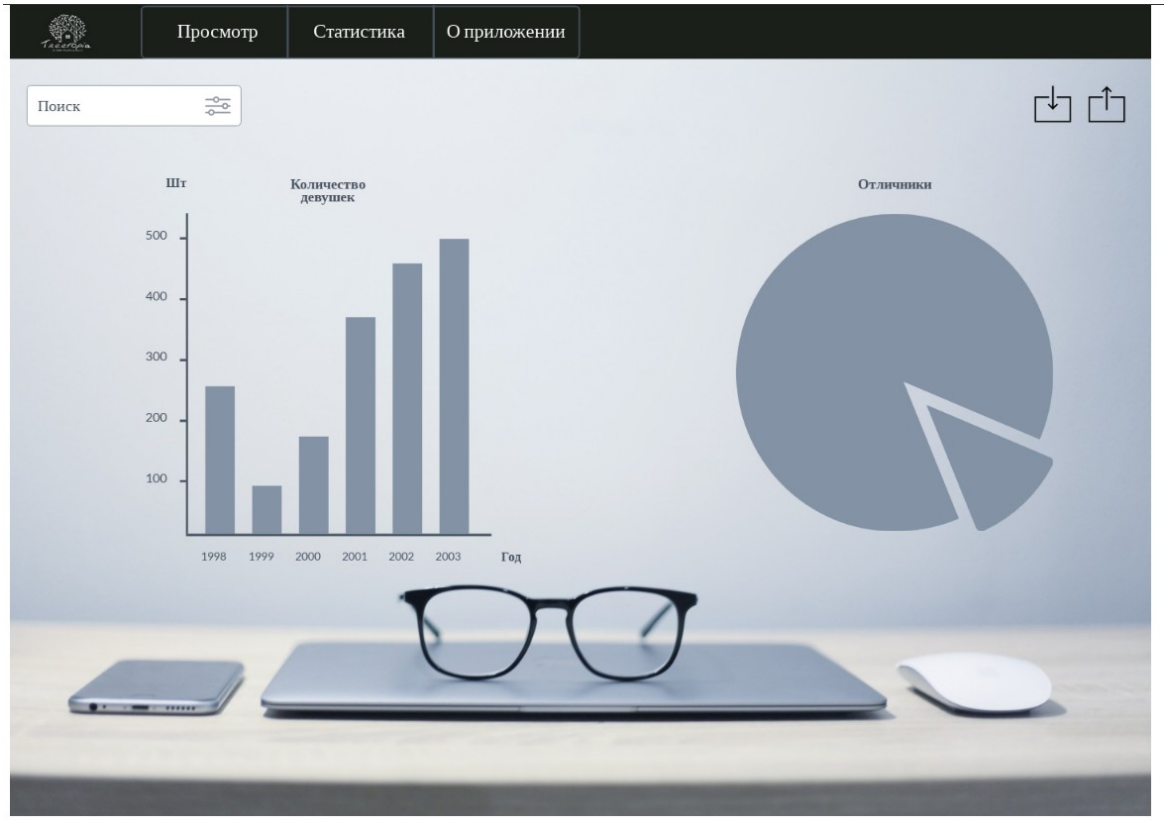

Рисунок 5 — Страница «Статистика»

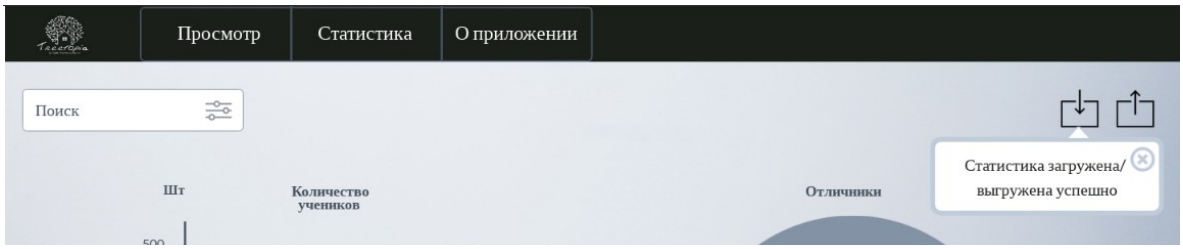

Рисунок 6 — Интерфейс работы кнопки «Импорт/ экспорт» на странице «Статистика»

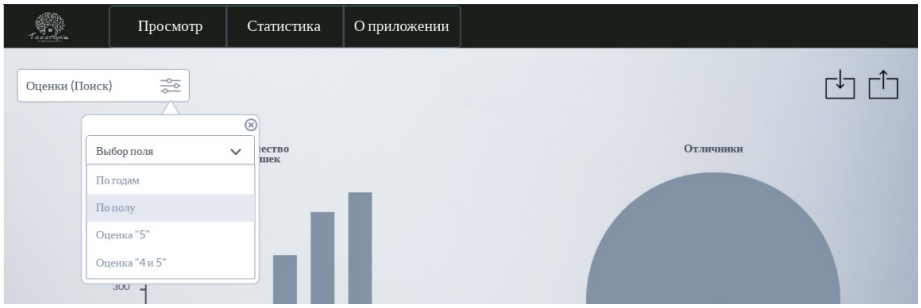

Рисунок 7 — Интерфейс настройки фильтров

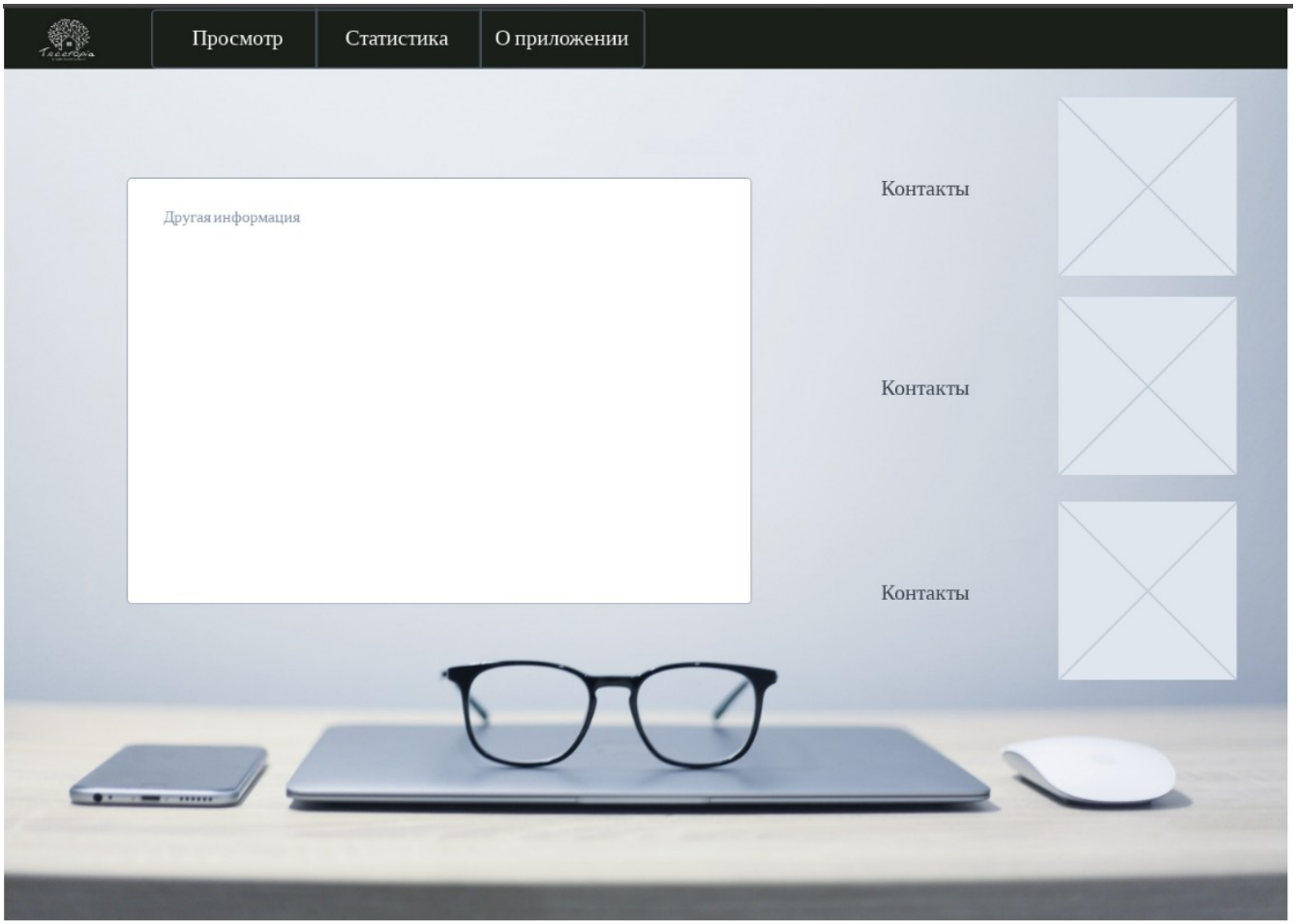

Рисунок 8 — Страница «О приложении»

2.2. Описание сценариев использования

При использовании приложения пользователь делает выбор страницы: «Просмотр», «Статистика» или «О приложении». Результат: переход на страницы «Просмотр», «Статистика» или «О приложении» соответственно.

2.2.1. Сценарий использования страницы «Просмотр»

1). Пользователь делает выбор таблицы из выпадающего списка. Результат: пользователь на рабочем пространстве видит содержание выбранной таблицы.

2). Далее пользователь может совершать действия над таблицей:

(1) Пользователь совершает поиск в строке «Поиск», также поиск может осуществляться с фильтрацией, то есть настройкой фильтров на усмотрение пользователя, что облегчит восприятия содержимого таблицы. Результат: пользователь ускоряет процесс поиска информации путем стороки «Поиск» и настройки фильтров.

(2) Пользователь совершает импорт таблицы. Результат: загруженная страница.

(3) Пользователь совершает экпорт таблицы. Результат: выгруженная страница.

(4) Пользователь совершает редактирование строк в таблице с помощью кнопки «Редактировать», сохраняя изменения. Также, открыв редактор, пользователь может удалить выбранную строку полностью. Результат: изменения содержимого строк в таблице после сохранения или удаление строки.

(5) Пользователь совершает добавдение строки с заполнением строк и сохранением. Результат: новая строка в таблице.

2.2.2. Сценарий использования страницы «Статистика»

1). Пользователь делает выбор статистики из выпадающего списка. Результат: пользователь на рабочем пространстве видит содержание выбранной статистики.

2). Пользователь может задать фильтры для статистик. Результат: показана выборка данных по заданным фильтрам.

3). Пользователь совершает импорт статистики. Результат: загруженная статистика.

4). Пользователь совершает экпорт статистики. Результат: выгруженная статистика.

2.2.3. Сценарий использования страницы «О приложении»

1). Пользователь может ознакомиться с информацией о приложении и контактах для связи. Результат: получение основной информации.

3. МОДЕЛЬ ДАННЫХ

3.1. NoSQL модель данных

3.1.1. Графическое представление

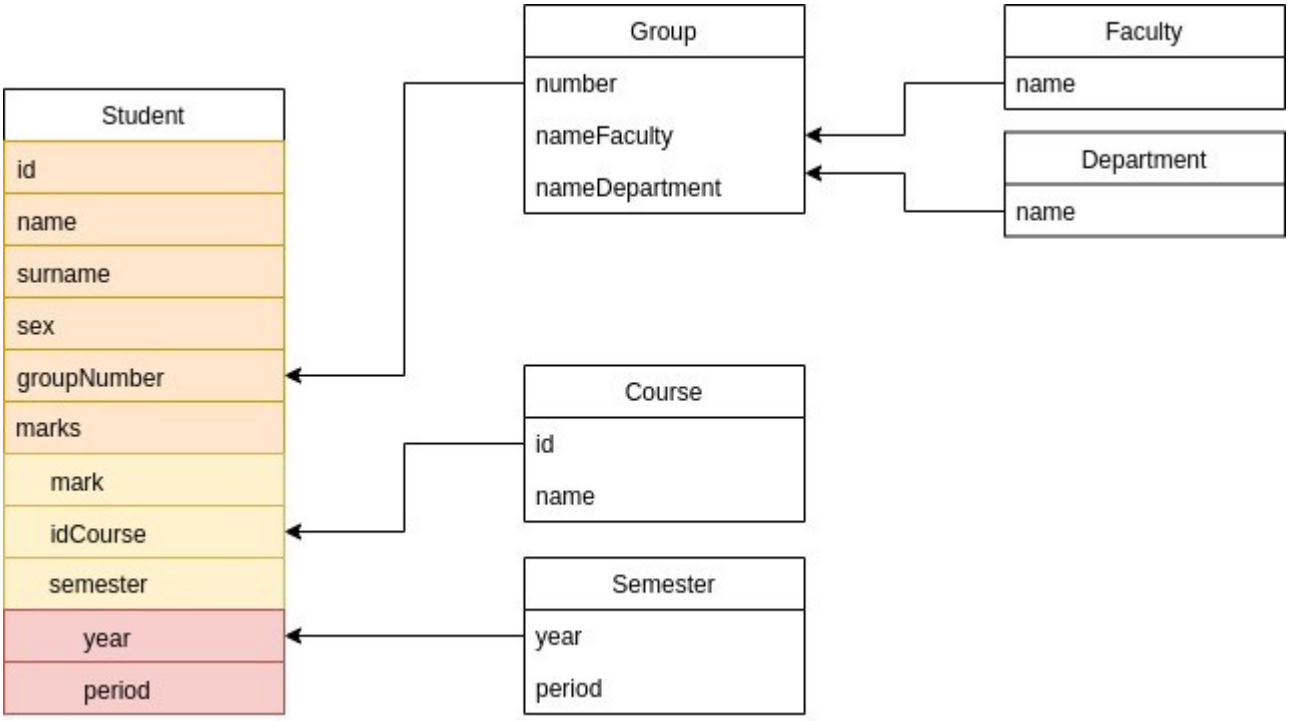

Рисунок 9 — Графическое представление NoSQL модели

3.1.2. Подробное описание и расчёт объема

В нереляционной базе данных MongoDB содержится 6 коллекций:

1).Students- коллекция для хранения студентов и их оценок. Каждый элемент коллекции Students содержит следующие поля:

- id: int - уникальный идентификатор студента.
- name: string - имя студента.
- surname: string - фамилия студента.
- sex: bool - пол студента.
- groups: array[int] - список групп, в которых учится студент (содержит только уникальные номера групп; кафедра и факультет, которым принадлежит группа, хранятся в коллекции Groups).

- marks: array[Mark] - оценки студента. Каждая оценка включает в себя саму оценку (поле mark: int), семестр (поля year: int и period: int), уникальный идентификатор курса (поле idCourse: int).

2). Groups- коллекция для хранения групп. Каждый элемент коллекции Groups содержит следующие поля:

- number: int - уникальный номер группы.
- faculty: string - факультет, которому принадлежит группа.
- departments: string - кафедра, которой принадлежит группа.

3). Faculties- факультеты. Коллекция представлена одним полем - name: string, которое содержит название факультета. Коллекция используется при добавлении новых групп. Группа, принадлежащая определенному факультету, может быть добавлена только в том случае, если факультет содержится в данной коллекции.

4). Departments- кафедры. Коллекция представлена одним полем - name: string, которое содержит название кафедры. Коллекция используется при добавлении новых групп. Группа, принадлежащая определенной кафедре, может быть добавлена только в том случае, если кафедра содержится в данной коллекции.

5). Courses- предметы. Содержит поля:

- id: int - уникальный идентификатор предмета.
- name: string - название предмета Оценка по определенному предмету может быть добавлена только в том случае, когда предмет есть в данной коллекции.

6). Semester- семестры. Содержит поля:

- year: int — год.
- period: string - период года (осень / весна). Позволяет определить весенний или осенний семестр. Оценка в определенном семестре может быть добавлена только в том случае, когда семестр содержится в данной коллекции.

Пусть на факультет приходится 12 студентов, на кафедру - 6, на группу - 3. На семестр и курс - общее кол-во студентов. У каждого студента по 1 оценке. Взяты следующие значения: int = 4Б, string = 50Б, DBRef = 10 Б, bool= 1 Б. Тогда получено:

- Faculty:{“name” : string} = 5012S Б.
- Department:{“name” : string} = 506S Б.
- Group:{“number”: int, “nameFaculty” : string + DBRef, “nameDepartment” : string + DBRef} = (4 + 100 + 210)3S = 1243*S Б.
- Semester:{“year” : int, "period" : int} = 8*S Б.
- Course:{“id” : int, “name” : string} = 54*S Б.
- Student:{ “id” : int, “name” : string, “surname” : string, “sex” : bool, “groupNumber” : int + DBRef, “marks” : [{“mark” : int, “idCourse” : int + DBRef, “semester” : {"year" : int + DBRef,"period" : string}}]} = 4 + 250 + 1 + 4 + 10 + 4 + 4 + 10 + 4 + 10 + 50 = 171 + 310 = 201*S Б.

«Чистый» объем: 5012S + 506S + 1043S + 8S + 54S + 171S = 1445S Б.

Фактический объем: 5012S + 506S + 1243S + 8S + 54S + 201S = 1535S Б.

Избыточность модели: (1535S)/(1445S)≈1.06.

3.1.3. Примеры запросов

```
`db.getCollection("students").insesrtOne({“id” : 1, “name” : “Иван”, “surname” : “Петров”, “sex” : true, “groups” : [6381], “marks” : []})`
```

Рисунок 10 — Запрос на добавление нового студента

```
`db.getCollection("students").update({“id” : 1},
{$push: {“marks” : {“mark” : 5, “semester” : {"year" : "2019", "period": "spring"}, “idCourse” : “1”}}})`
```

Рисунок 11 — Запрос на добавление оценки студенту

```
`db.getCollection("students").find({id : 1})`
```

Рисунок 12 — Запрос на получение информации о студенте и всех его оценок

3.2. SQL модель данных

3.2.1. Графическое представление

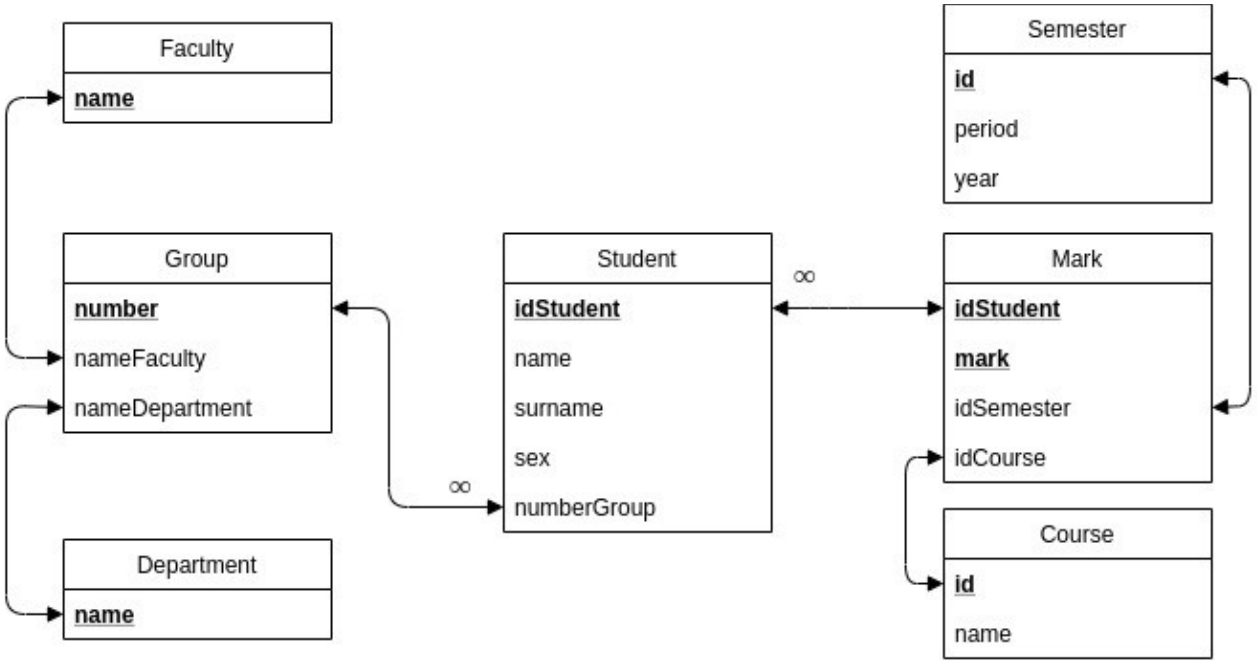

Рисунок 13 — Графическое представление SQL модели

3.2.2. Подробное описание и расчёт объема

В качестве реляционной СУБД использована MySQL[2]. Все сущности аналогичны сущностям из нереляционной модели, но для представления их в реляционной модели требуется 8 таблиц. Создана таблица Marks, используемая для хранения оценок, в таблицу вводится дополнительное поле idStudent: int для определения студента, который получил данную оценку. Создана таблица Semester с дополнительным полем id: int для однозначного определения семестра. Использование двух дополнительных таблиц обосновывается невозможностью хранить в одном столбце таблицы сразу нескольких полей.

Пусть на факультет приходится 12 студентов, на кафедру - 6, на группу - 3. На семестр и курс - общее кол-во студентов. У каждого студента по 1 оценке. Взяты следующие значения: int = 4Б, string = 50Б, DBRef = 10 Б, bool= 1 Б. Тогда получено:

- Faculty:{“name” : string} = 5012S Б.
- Department:{“name” : string} = 506S Б.

- Group:{“number”: int, “nameFaculty” : string + DBRef, “nameDepartment” : string + DBRef} = (4 + 100 + 210)3S = 1243*S Б.
- Semester:{“year” : int, "period" : int} = 8*S Б.
- Course:{“id” : int, “name” : string} = 54*S Б.
- Student:{ “id” : int, “name” : string, “surname” : string, “sex” : bool, “groupNumber” : int + DBRef, “marks” : [{“mark” : int, “idCourse” : int + DBRef, “semester” : {"year" : int + DBRef,"period" : string}}]} = 4 + 250 + 1 + 4 + 10 + 4 + 4 + 10 + 4 + 10 + 50 = 171 + 310 = 201*S Б.

«Чистый" объем»: 5012S + 506S + 1043S + 8S + 54S + 171S = 1445S Б.

Фактический объем: 5012S + 506S + 1243S + 8S + 54S + 201S = 1535S Б.

Избыточность модели: (1535S)/(1445S)≈1.06.

3.2.3. Примеры запросов

```
INSERT INTO student (id, name, surname, sex) VALUES (1, "Иван", "Петров", true)
```

Рисунок 14 — Запрос на добавление нового студента

```
INSERT INTO mark (studentId, mark, semesterId, courseId)
```

Рисунок 15 — Запрос на добавление оценки студенту

```
SELECT * FROM student
INNER JOIN mark ON student.id=mark.studentId
INNER JOIN semester ON mark.semesterId=semester.id
```

Рисунок 16 — Запрос на получение информации о студенте и всех его оценок

3.3. Сравнение NoSQL и SQL моделей данных

Сравним NoSQL и SQL по следующим пунктам:

- Количество запросов для совершения юзкейсов в зависимости от числа объектов в БД и прочих параметров. Количество запросов одинаково, однако в реляционной модели запрос на выбор студента и его оценок сложнее аналогичного запроса в нереляционной модели, так как требует выбора данных сразу из трех таблиц (коллекций). В нереляционной модели оценки хранятся вместе со студентом, поэтому нужен всего один запрос.

- Количество задействованных коллекций. В реляционной модели потребуется на одну таблицу (коллекцию) больше, так как в нереляционной модели каждая оценка студента хранится вместе со студентом, а в реляционной модели это невозможно, поэтому создается отдельная таблица для оценок.

Объем памяти нереляционной модели базы данных незначительно (на 20% для рассчитанного фактического объема) превышает объем реляционной модели. Для реализации проекта "Журнал успеваемости" была выбрана нереляционная модель данных из-за возможности хранить одни объекты внутри других — вложенность ускоряет процесс получения информации, находящейся в базе данных.

4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

4.1. Краткое описание

Для упрощения процесса разработки было принято решение разделить приложение на frontend и backend части.

Frontend-часть реализована с использованием Vue.js[3]. Vue.js — это фреймворк для создания пользовательских интерфейсов интерфейсов. Легко интегрируется в проекты с использованием других JavaScript-библиотек. Может функционировать как веб-фреймворк для разработки одностраничных приложений в реактивном стиле.

Backend-часть приложения реализована с использованием Scala[4]. Scala – мультипарадигмальный язык программирования, сочетающий возможности функционального и объектно-ориентированного программирования.

4.2. Схема экранов приложения

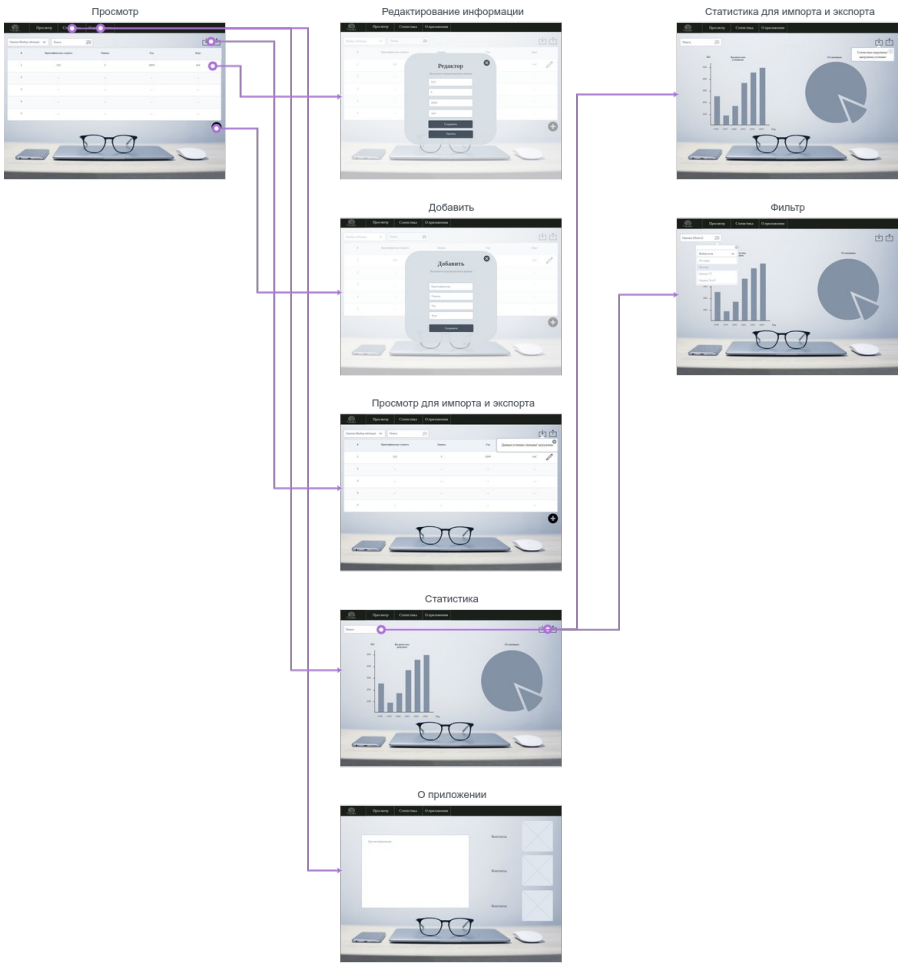

Рисунок 17 — Схема экранов приложения

4.3. Использованные технологии

БД: MongoDB.

Backend: Scala.

Frontend: Vue.js.

4.4. Ссылка на приложение

Ссылка на приложение доступна в разделе «Список использованных источников» [5].

ЗАКЛЮЧЕНИЕ

В ходе работы было реализовано приложение, в котором можно просмотреть информацию о студентах и их успехах, также реализованы функции редактирования, добавления и удаления. В приложении есть функции импорта/экспорта данных. Кроме того, пользователь может просмотреть различную статистику, выбирая необходимые фильтры. Таким образом, цель, поставленная перед началом работы, достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация MongoDB: https://docs.mongodb.com/manual/

2. Документация MySQL: https://dev.mysql.com/doc/

3. Документация Vue.js: https://vuejs.org/

4. Документация Scala: https://docs.scala-lang.org/

5. Исходный код приложения: https://github.com/moevm/nosql2h19-marks-mongo

ПРИЛОЖЕНИЕ А

ИНСТРУКЦИЯ ПО СБОРКЕ И ЗАПУСКУ

1. Установка mongo.
2. Установка jdk.
3. Установка sbt.
4. Запуск серверной части командой sbt run (в директории с файлом build.sbt).
5. Установка Node.js.
6. Установка пакетов с помощью npm.
7. Запуск frontend-части с помощью npm.