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: ИС для управления учебным процессом в школе

Студент гр. 9382 _____ Демин. В.В.

Студентка гр. 9303 _____ Отмахова М.А.

Студент гр. 9303 _____ Скворчевский Б.С.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2022

ЗАДАНИЕ

Студенты

Демин. В.В.
Группа 9382

Скворчевский Б.С.
Отмахова М.А.
Группа 303

Тема работы: Разработка веб-приложения для управления учебным процессом в школе.

Исходные данные:

Необходимо реализовать приложение для СУБД(MongoDB).

Содержание пояснительной записки:

«Содержание»

«Введение»

«Качественные требования к работе»

«Сценарии использования»

«Модель данных»

«Разработка приложения»

«Вывод»

«Приложение»

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания: 01.09.2022

Дата защиты реферата: 22.12.2022

Студент гр. 9382		Демин В.В.
Студентка гр. 9303		Отмахова М.А.
Студент гр. 9303		Скворчевский Б.С.
Преподаватель		Заславский М.М.

АННОТАЦИЯ

В данной работе была разработана ИС для для управления учебным процессом в школе. Для разработки использовались следующие технологии:

1. Основной язык программирования – Python [1].
2. В качестве базы данных использовалась СУБД MongoDB [2].
3. В качестве фреймворка для серверной части был использован фреймворк языка Python – Flask [3].
4. В качестве фреймворка для реализации клиентской части был использован фреймворк языка JavaScript [4] – React [5].

Был разработан первый прототип «Представление и хранение», в котором реализована возможность просматривать содержимое базы данных с помощью таблиц, а также добавлять новые данные.

SUMMARY

In this work, an IS was developed for managing the educational process at school. The following technologies were used for development:

1. The main programming language is Python [1].
2. The MongoDB DBMS [2] was used as a database.
3. As a framework for the server side, the Python language framework Flask [3] was used.
4. As a framework for the implementation of the client side, the JavaScript language framework [4] - React [5] was used.

The first prototype "View and Storage" was developed, which implemented the ability to view the contents of the database using tables, as well as add new data.

СОДЕРЖАНИЕ

Введение

Цель работы – создать веб-приложение для автоматизации учебного процесса.

1. Качественные требования к решению

Требуется разработать веб-приложение, используя СУБД MongoDB.

2. Сценарии использования

Макет пользовательского интерфейса представлен на рис.1.

Рисунок 1 – Пользовательский интерфейс

Use-case системы представлен на рис.2.

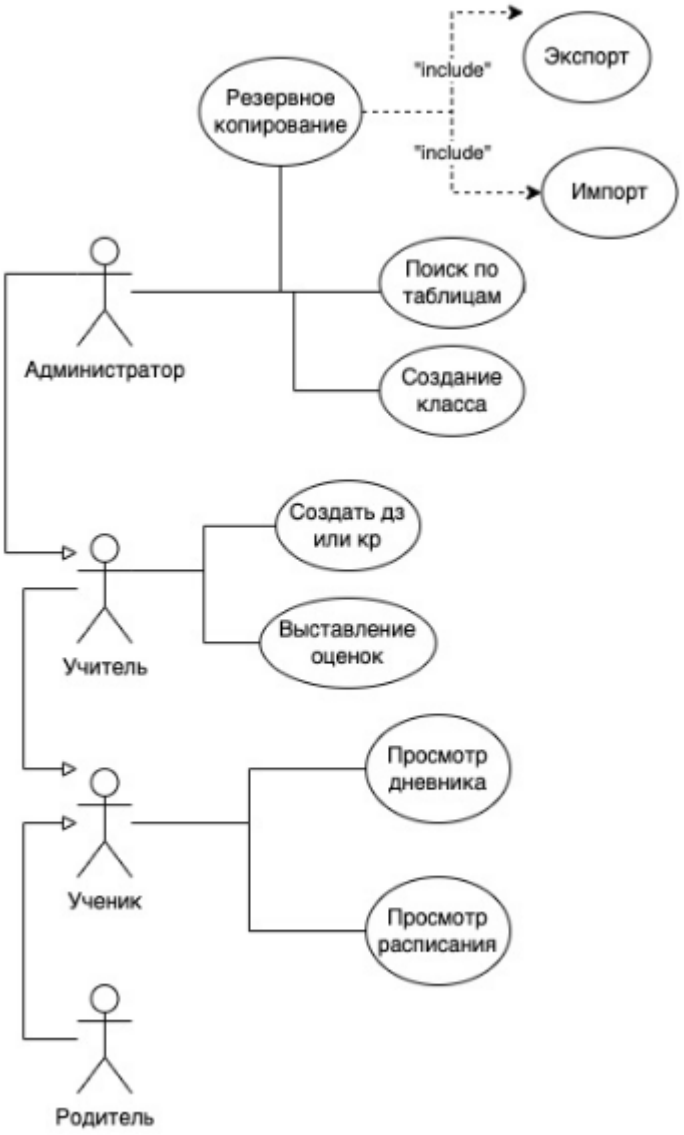

Рисунок 2 – Use-case

Сценарии использования системы представлены в табл.1.

Таблица 1 – Сценарии использования

Название сценария	Сценарий	Действующее лицо
0.1. Успешный вход в систему	1. Пользователь получает логин и пароль от администратора 2. Пользователь заходит на главную страницу сайта 3. Пользователь вводит свой логин и пароль 4. Пользователь нажимает кнопку войти **5. Вход в систему выполнен**	Учитель, ученик, родитель
0.2. Неуспешный вход в систему	4. Пользователь нажимает кнопку войти 5. Вход в систему не выполнен 6. Переход на шаг 3	Учитель, ученик, родитель
1. Просмотр дневника		

1.1. Просмотр личного дневника	0. Пользователь зашел в систему 1. Пользователь нажимает на кнопку “Дневник” 2. Пользователь видит окно со своими оценками за дз/кр и домашние задания 3. Есть возможность отфильтровать по типу оценки, по сроку	Ученик, Родитель
1.2. Просмотр дневников учеников	0. Пользователь зашел в систему 1. Пользователь нажимает на кнопку “Дневник” 2. Пользователь выбирает класс и ученика 3. Пользователь видит окно с оценками ученика и предстоящими домашними задания 4. Есть возможность отфильтровать по типу оценки, по сроку	Администратор, Учитель
2. Просмотр расписания	0. Пользователь зашел в систему 1. Пользователь нажимает на кнопку “Расписание” 2. Пользователь выбирает нужный класс и **просматривает расписание**	Ученик, Родитель, Учитель
3. Поиск по таблицам	0. Пользователь зашел в систему 1. Пользователь нажимает на кнопку “Администратор” 2. Пользователь выбирает нужную таблицу в фильтре 3. Пользователь в окне ввода запроса, вводит нужный запрос для поиска, нажимает Enter 4**. Появляется нужна таблица**	Администратор
4. Резервное копирование	0. Пользователь зашел в систему 1. Пользователь нажимает на кнопку “Администратор” 2. Пользователь нажимает на кнопку “Резервное копирование” 3. Пользователь выбирает нужную таблицу 4. Пользователь нажимает на нужную кнопку “Импорт”/”Экспорт” 5. При импорте у пользователя откроется системное окно для выбора файла	Администратор
5. Выставление оценок	0. Пользователь зашел в систему 1. Пользователь нажимает на кнопку “Все оценки” 2. Пользователь **вводит оценку ученику**	Учитель
6. Создание работы для учеников	0. Пользователь зашел в систему 1. Пользователь нажимает на кнопку “Расписание” 2. Пользователь выбирает нужную неделю и нужный день 3. Пользователь нажимает на свой предмет и вводит описание работы 4. После чего нажимает на кнопку “Добавить” 5. **Работа создана**	Учитель
7. Добавление класса	0. Пользователь зашел в систему 1. Пользователь нажимает на кнопку “Журнал” 2. Пользователь рядом с классами нажимает на кнопку + 3. Пользователь вводит учеников, после чего нажимает на кнопку “Создать” 4. Пользователь выбирает нужные предметы в классе,	Администратор

	после чего нажимает на кнопку “Создать” 5. **Класс создан**	

3. Модель данных

ER-диаграмма представлена на рис.3.

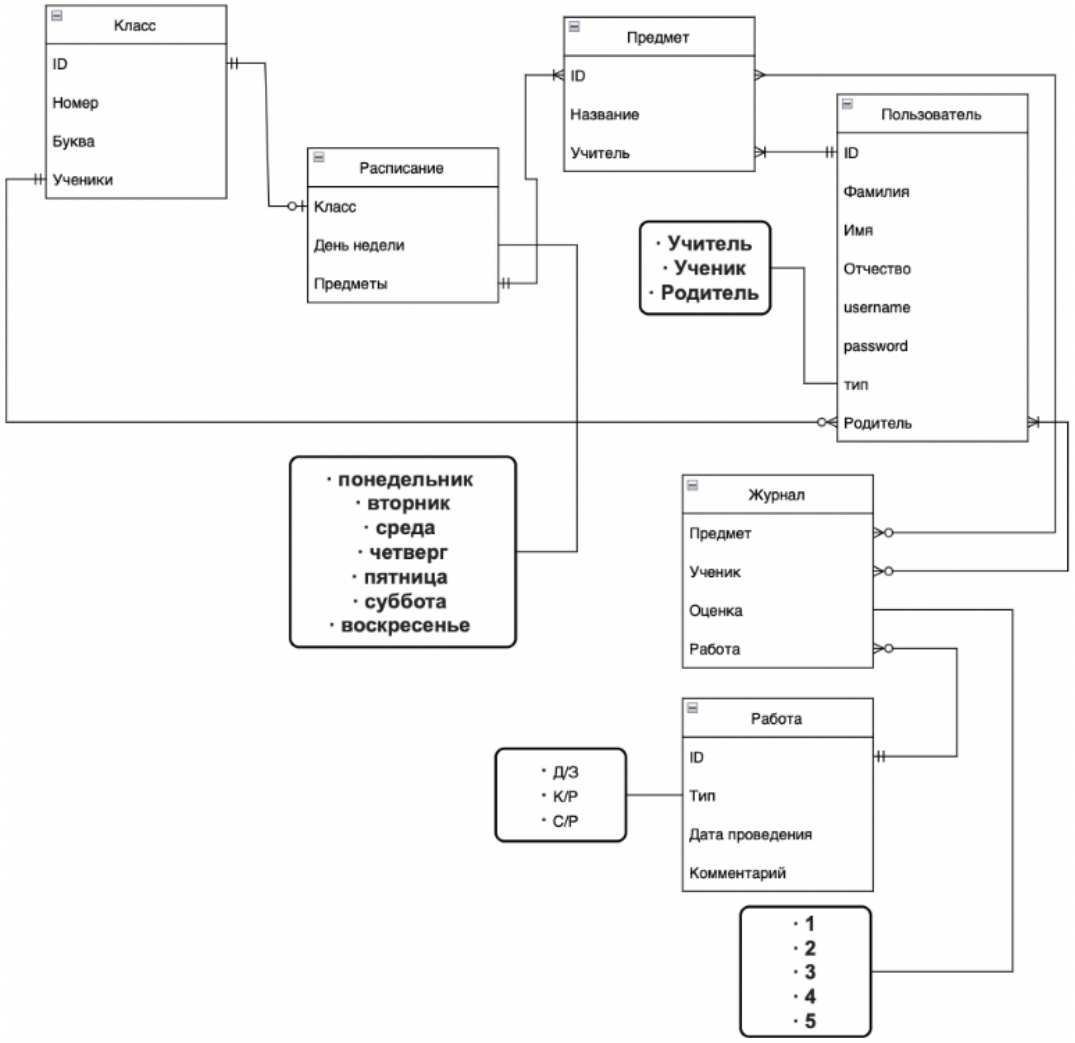

Рисунок 3 – ER-диаграмма

3.1. Нереляционная модель данных

Модель хранения данных представлена на рис.4.

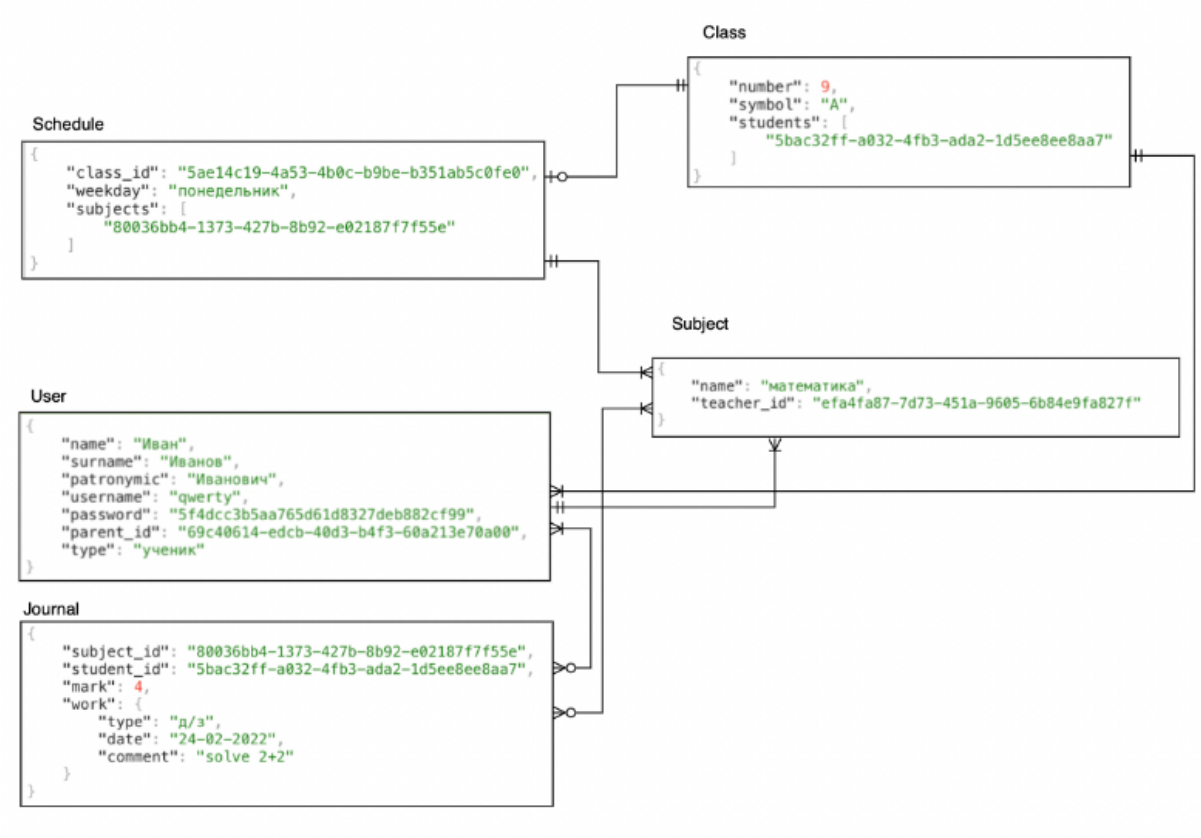

Рисунок 4 – Модель хранения данных

Оценка объема информации

Расписание (47 байт)

class_id: string - id класса (36 байт)

weekday: string - день недели (11 байт)

Предмет (86 байт)

name: string - название (50 байт)

teacher_id: string - id учителя (36 байт)

Класс (5 байт)

number: int - номер класса (4 байт)

symbol: string - буква класса (1 байт)

Пользователь (243 байт)

name: string - имя (50 байт)

surname: string - фаминия (50 байт)

patronymic: string - отчество (50 байт)

parent_id: string - id родителя (36 байт)

username: string - логин (15 байт)

password: string - пароль (32 байт)

type: string - тип пользователя (10 байт)

Журнал (89 байт)

subject_id: string - id предмета (36 байт)

student_id: string - id ученика (36 байт)

mark: int - оценка (4 байт)

work: object - работа (13 байт)

Работа (13 байт)

type: string - тип работы (3 байт)

date: string - дата (10 байт)

Запросы к БД для реализации сценариев использования

Вход в систему

Запрос:

```
def auth_user(self, auth_data):
    assert 'username' in auth_data and 'password' in auth_data
    auth_data['password'] =
hashlib.md5(auth_data['password'].encode('utf-8')).hexdigest()
    return self.user.find_one(auth_data)


auth_user({'username': '6wdyox7v9mj2t04', 'password': 'bh7mildje1w19en'})
```

Ответ:

```
{
    '_id': ObjectId('635e920293f0b39a5dcf025d'),
    'name': 'Иван',
    'password': 'bh7mildje1w19en',
    'patronymic': 'Иванович',
    'surname': 'Иванов',
    'type': 'student',
    'username': '6wdyox7v9mj2t04'
}
```

Просмотр дневника

Запрос:

```
def get_user_journal(self, user_id):
    return [day for day in self.journal.find({
        'student_id': ObjectId(user_id)
    })]


get_user_journal('635e920293f0b39a5dcf025d')
```

Ответ:

```
[{
```

```
    '_id': ObjectId('635e930d45fc814a7e48bc82'),
     'mark': 4,
     'student_id': ObjectId('635e930d45fc814a7e48bc7e'),
     'subject_id': ObjectId('635e930d45fc814a7e48bc7f'),
     'work': {'comment': 'solve 2+2', 'date': '24-02-2022', 'type': 'д/з'}
}]
```

Просмотр расписания

Запрос:

```
def get_user_schedule(self, user_id=None, class_id=None):
    assert user_id or class_id
    if user_id and not class_id:
        class_id = self.classes.find_one({
            'students': ObjectId(user_id)
        }).get('_id')
    return [
        self.serialize_object(day)
        for day in self.schedule.find({
            'class_id': ObjectId(class_id)
        })
    ]


get_user_schedule('635e920293f0b39a5dcf025d')
```

Ответ:

```
[{
    '_id': ObjectId('635e93d4b955342aa968c9f0'),
    'class_id': ObjectId('635e93d4b955342aa968c9ef'),
    'subjects': [{'_id': ObjectId('635e93d4b955342aa968c9ee'),
                  'name': 'математика',
                  'teacher_id': ObjectId('635e93d4b955342aa968c9ec')}],
    'weekday': 'понедельник'
```

```
}]
```

Поиск по таблицам

Запрос:

```
def search(self, table, text):
    search_result = []
    for data in self.__getattribute__(table).find():
        if text in str(data):
            search_result.append(data)
    return search_result


search('user', 'teacher')
```

Ответ:

```
[{
    '_id': ObjectId('635e7255302621fc9c0497f8'),
    'name': 'Иван',
    'password': '794d13ac3af1c44a49d72d3e5626fcbd',
    'patronymic': 'Иванович',
    'surname': 'Иванов',
    'type': 'teacher',
    'username': 'vzrfa3g68ti67ah'},
    {'_id': ObjectId('635e726ce4258fc0bd65503f'),
    'name': 'Иван',
    'password': '72deb04d47d4b1a15ff839c32c231ff7',
    'patronymic': 'Иванович',
    'surname': 'Иванов',
    'type': 'teacher',
    'username': 'x4h6jj7w5mc12n3'
}]
```

Резервное копирование

Запрос:

```
def export_table(self, table):
    return [data for data in self.__getattribute__(table).find()]

def import_data(self, table, data_list):
    table = self.__getattribute__(table)
    for data_obj in data_list:
        table.insert_one(data_obj)

export_table('user')
import_data('user', [{
    'name': 'Иван',
    'password': '1508903a47b86c5e38a8efcb4995f3da',
    'patronymic': 'Иванович',
    'surname': 'Иванов',
    'type': 'student',
    'username': '46b8e11c30arimq'
}])
```

Ответ:

```
[{
    '_id': ObjectId('635e7255302621fc9c0497f8'),
    'name': 'Иван',
    'password': '794d13ac3af1c44a49d72d3e5626fcbd',
    'patronymic': 'Иванович',
    'surname': 'Иванов',
    'type': 'teacher',
    'username': 'vzrfa3g68ti67ah'},
    {'_id': ObjectId('635e726ce4258fc0bd65503f'),
    'name': 'Иван',
    'password': '72deb04d47d4b1a15ff839c32c231ff7',
    'patronymic': 'Иванович',
```

```
    'surname': 'Иванов',
    'type': 'teacher',
    'username': 'x4h6jj7w5mc12n3'},
    {'_id': ObjectId('635e94cf9e8f2a065b624922'),
     'name': 'Иван',
     'password': '2812313350d765a5195b66450a797ad6',
     'patronymic': 'Иванович',
     'surname': 'Иванов',
     'type': 'student',
     'username': 'rofutbc1qc18jgz'}
]
```

Выставление оценок

Запрос:

```
def set_mark(self, journal_id, new_mark):
    self.journal.update_one({'_id': ObjectId(journal_id)}, {'$set': {'mark': new_mark}})

set_mark('635e8da8f95a6ae02c6bd24c', 5)
```

Ответ:

None

Создание работы для учеников

Запрос:

```
def create_new_work(self, work_data):
    assert all(work_data.get(param) for param in ['subject_id', 'comment', 'date', 'type'])
    assert work_data.get('student_id') or work_data.get('class_id')
    if work_data.get('student_id'):
        self.journal.insert_one({
            'mark': 0,
            'student_id': ObjectId(work_data.get('student_id')),
```

```
            'subject_id': ObjectId(work_data.get('subject_id')),
            'work': {
                'comment': work_data.get('comment'),
                'date': work_data.get('date'),
                'type': work_data.get('type')
            }
        })
    elif work_data.get('class_id'):
        self.journal.insert_many([
            {
                'mark': 0,
                'student_id': student_id,
                'subject_id': ObjectId(work_data.get('subject_id')),
                'work': {
                    'comment': work_data.get('comment'),
                    'date': work_data.get('date'),
                    'type': work_data.get('type')
                }
            } for student_id in self.classes.find_one({'_id':
ObjectId(work_data.get('class_id'))}).get('students')
        ])


create_new_work({
    'class_id': '635e752d8fd314e69ae84c0b',
    'subject_id': '635e8a4a2af02f90723c77e2',
    'comment': '2-2',
    'date': '12-12-2022',
    'type': 'д/з'
})
```

Ответ:

None

Добавление класса

Запрос:

```
def create_class(self, class_data):
    assert all(class_data.get(param) for param in ['number', 'students', 'symbol'])
    class_data['students'] = [ObjectId(student_id) for student_id in
class_data['students']]
    return self.classes.insert_one(class_data).inserted_id


create_class({
    'number': 9,
    'symbol': 'A',
    'students':  ['635e920293f0b39a5dcf025d']
})
```

Ответ:

None

Оценка удельного объема информации, хранимой в модели

Изучим предметную область

Неизвестные переменные:

Количество учеников - N

Тогда:

количество классов - N div 30

количество родителей - N (один аккаунт родителя на ученика)

в среднем на одну параллель из 4 классов необходимо около 8 учителей или 2 учителя на один класс

количество учителей - {количество классов} * 2

максимальное количество уникальных предметов - не больше 20

количество уроков на один класс в неделю в среднем - 6 дней * 6 уроков - 36

количество оценок у одного ученика в неделю, оценим как 1 оценка на один уникальный предмет(за неделю около 9) - 9
количество всех оценок у ученика за семестр - 15*9
количество всех оценок в журнале - 159 N

Оценка памяти для СУБД MongoDb

User = ({количество учеников} + {количество родителей} + {количество учителей})*size(User) = (N + N + (N div 30)*2) 243 = 468 (N + N div 30)
Class = {Количество классов} *size(Class) = (N div 30)*5
Subject = {количество учителей} size(Subject) = (N div 30) 2 86 = (N div 30) 172
Schedule = {Количество классов} {Количество предметов в неделю} {Количество недель в семестре} size(Schedule) = N div 30 36 15 47 = 25380 * (N div 30)

Journal = {количество всех оценок в журнале} size(Journal) = 159 N 89 = 12015

Итого: 468 *(N + N div 30) + (N div 30)*5 + (N div 30) 172 + 25380 (N div 30) + (N div 30) 5 + 12015 N = 26025 (N div 30) + 12493 N байт
В среднем в одной школе около 840 учеников
Тогда на такую школу потребуется - 728700 + 10494120 байт = 11222820 байт = 10 mb
Избыточность модели
Чистый объем данных:
User = size(User) * (N + N + (N div 30)*2) - 36 * N
Journal = 15*9 N 17
Class = {Количество классов} *size(Class) = (N div 30)*5
Schedule = N div 30* 36 15 (11)

Subject = (N div 30)* 2 * (86- 36)

Итого: 243 * (N + N + (N div 30)2) - 36 * N + 159 N 17 + (N div 30)5 + N div 30 36 15 (11) + (N div 30)* 2 * (86- 36) = 2745 * N + 6531 * (N div 30)

Избыточность модели: (26025 * (N div 30) + 12493* N) / (2745 * N + 6531 * (N div 30)) или (26025/30 + 12493)* N / (2745 + 6531/30) * N = 13360/2962 = 4.5

3.2. Реляционная модель данных

Модель хранения данных представлена на рис.5.

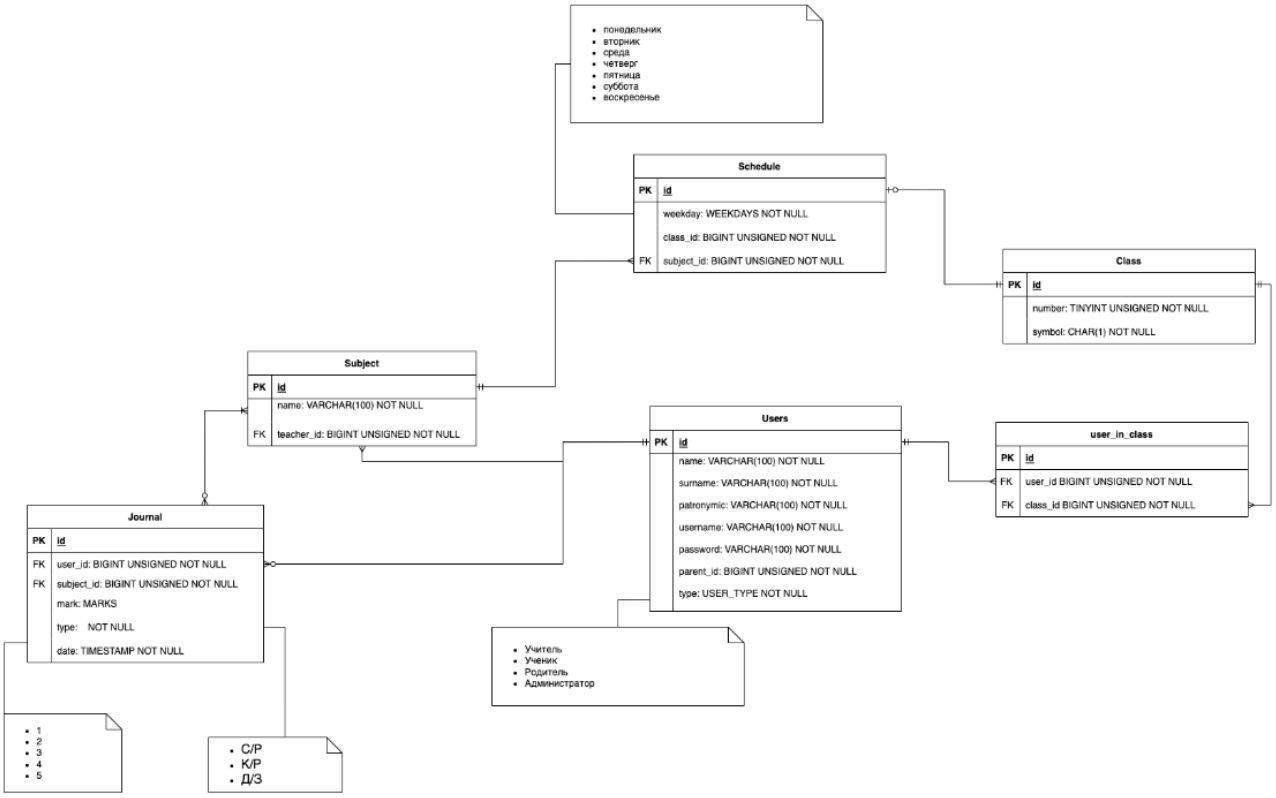

Рисунок 5 – Модель хранения данных

Оценка объема информации

Schedule (28 байт)

id: bigserial(8 байт)

weekday: enum WEEKDAYS (4 байт)

class_id: bigint(8 байт)

subject_id: bigint (8 байт)

Subject (116 байт)

id: bigserial(8 байт)

name: varchar(100 байт)

teacher_id: bigint(8 байт)

Class (13 байт)

id: bigserial(8 байт)

number: tinyint(4 байт)

symbol: char(1 байт)

Users (520 байт)

id: bigserial(8 байт)

name: varchar(100)(100 байт)

surname: varchar(100) (100 байт)

patronymic: varchar(100) (100 байт)

parent_id: bigint(8 байт)

username: varchar(100) (100 байт)

password: varchar(100)(100 байт)

type: enum USER_TYPE(4 байт)

Journal (40 байт)

id: bigserial(8 байт)

subject_id: bigint(8 байт)

user_id: bigint (8 байт)

mark: enum MARKS (4 байт)

type: enum Work (4 байт)

date: TIMESTAMP (8 байт)

user_in_class (24 байт)

id: bigserial(8 байт)

user_id: bigint - номер класса (8 байт)

class_id: bigint - буква класса (8 байт)

Оценка памяти для СУБД Posgresql

User = ({количество учеников} + {количество родителей} + {количество учителей})*size(User) = (N + N + (N div 30)*2) * 520 = 1040 * (N + N div 30)

user_in_class = {количество учеников} * size(user_in_class) = 24*N

Class = {Количество классов} *size(Class) = (N div 30)*13

Subject = {количество учителей} size(Subject) = (N div 30) 2 116 = (N div 30) 232

Schedule = {Количество классов} {Количество предметов в неделю} {Количество недель в семестре} size(Schedule) = N div 30 36 15 28 = 15120 * (N div 30)

Journal = {количество всех оценок в журнале} size(Journal) = 159 N 40 = 5400 N

Итого: 1040*(N + N div 30)+ 24* N + (N div 30)*13 + (N div 30)*232 + 15120 * (N div 30) + 5400N = 6464 N + (N div 30) * 15389 байт

В среднем в одной школе около 840 учеников

Тогда на такую школу потребуется - 5429769 + 430892 байт = 5860651 байт = 6 mb

Избыточность модели

Чистый объем данных:

User = (size(User) - 8(parent_id) - 8(id)) * (N + N + (N div 30)*2) = 1008 * (N + (N div 30))

Journal = (Size(Journal) - 8(id) - 8(user_id) - 8 (subject_id)) * 15*9 *N = 2160 * N

Class = (size(Class) - 8(id))* (N div 30) = 5 * (N div 30)

Schedule = (size(Schedule) - 8(id) - 8(subject_id))* N div 30* 36 *15 = 6480 * (N div 30)

Subject = (size(Subject) - 8(id)-8(teacher_id))* (N div 30)* 2 = 200 * (N div 30)

user_in_class = 0

Итого: 1008 * (N + (N div 30)) + 2160 * N + 5 * (N div 30) + 6480 * (N div 30) + 200 * (N div 30) = 7693 * (N div 30) + 3168 * N

Избыточность модели: 6464 N + (N div 30) * 15389 / 7693 * (N div 30) + 3168 * N или (15389/30 + 6464)* N / (3168 + 7693/30) * N = 10145/3424= 2.96

Сравнение моделей

MongoDB модель данных требует больше места. Поскольку необходимо хранить id сущностей в нескольких коллекциях, которые занимают куда больше памяти, чем id в SQL модели. Даже с учетом того, что необходимо выделять отдельную таблицу для связи класса и учеников. Что и можно увидеть по избыточности данных, в модели NoSql она больше.

4. Разработанное приложение

Было реализовано приложение, отвечающее требованиям. Экраны приложения представлены ны рис.6 - рис.12.

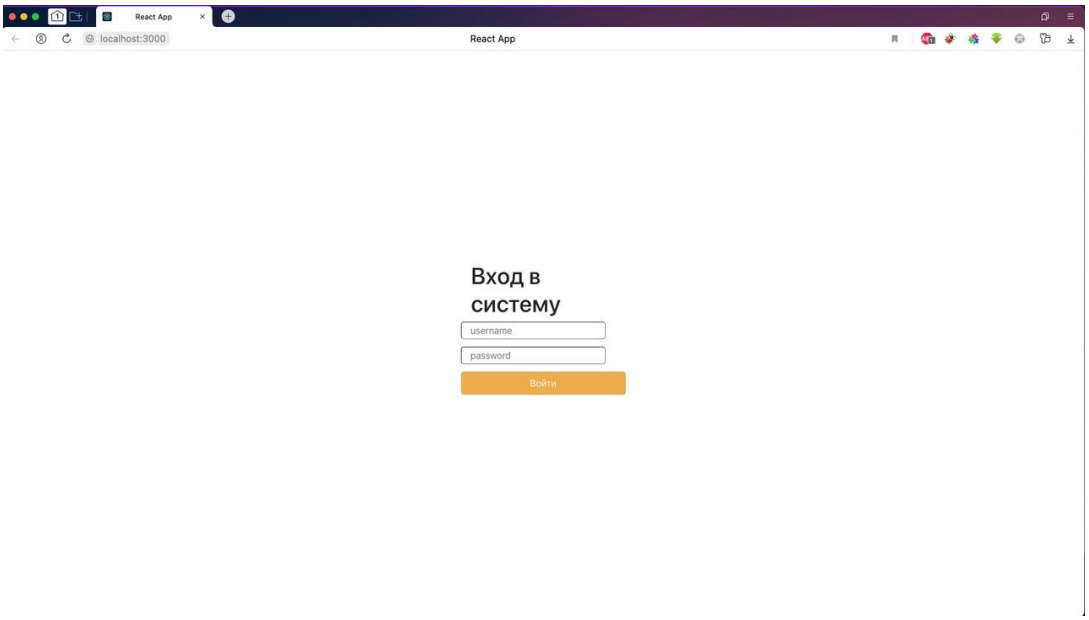

Рисунок 6 – Вход в систему

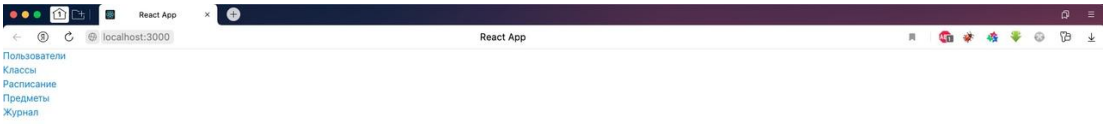

Рисунок 7 – Главная страница

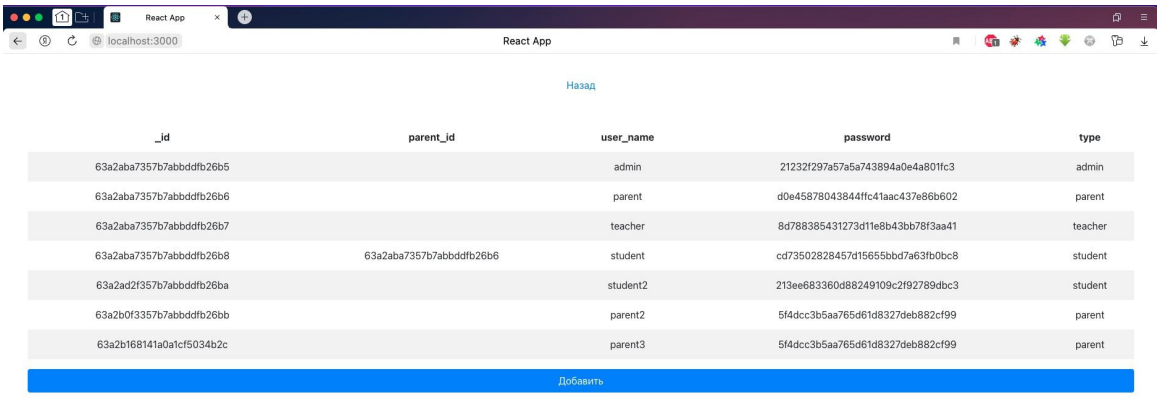

Рисунок 8 – Просмотр таблиц

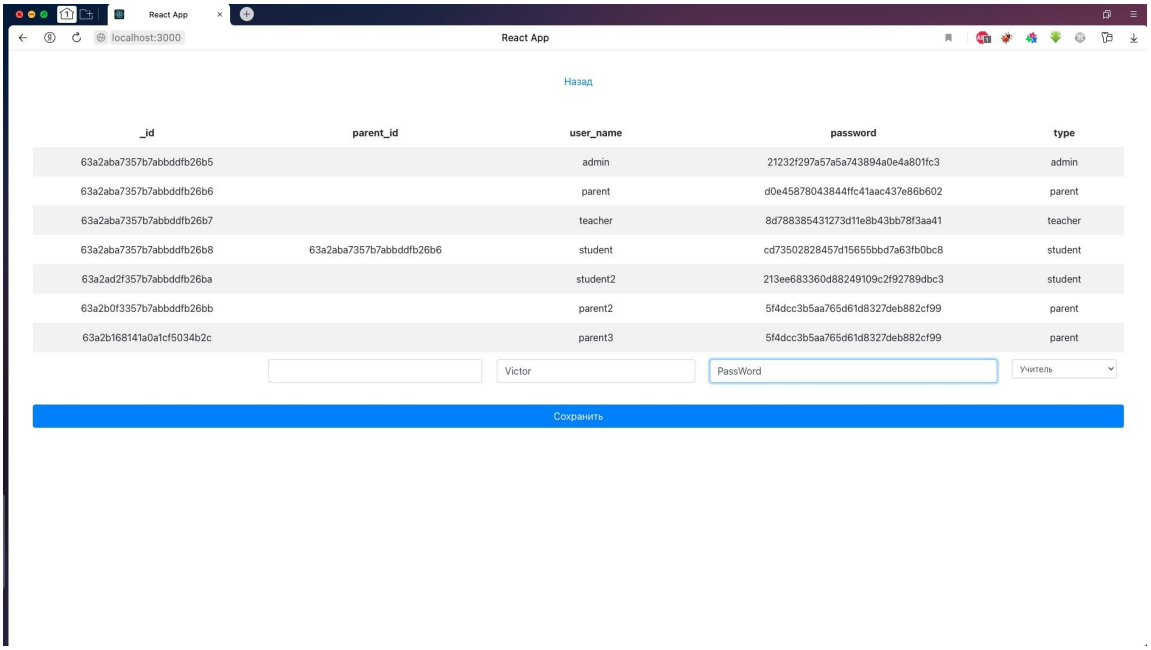

Рисунок 9 – Добавление пользователя

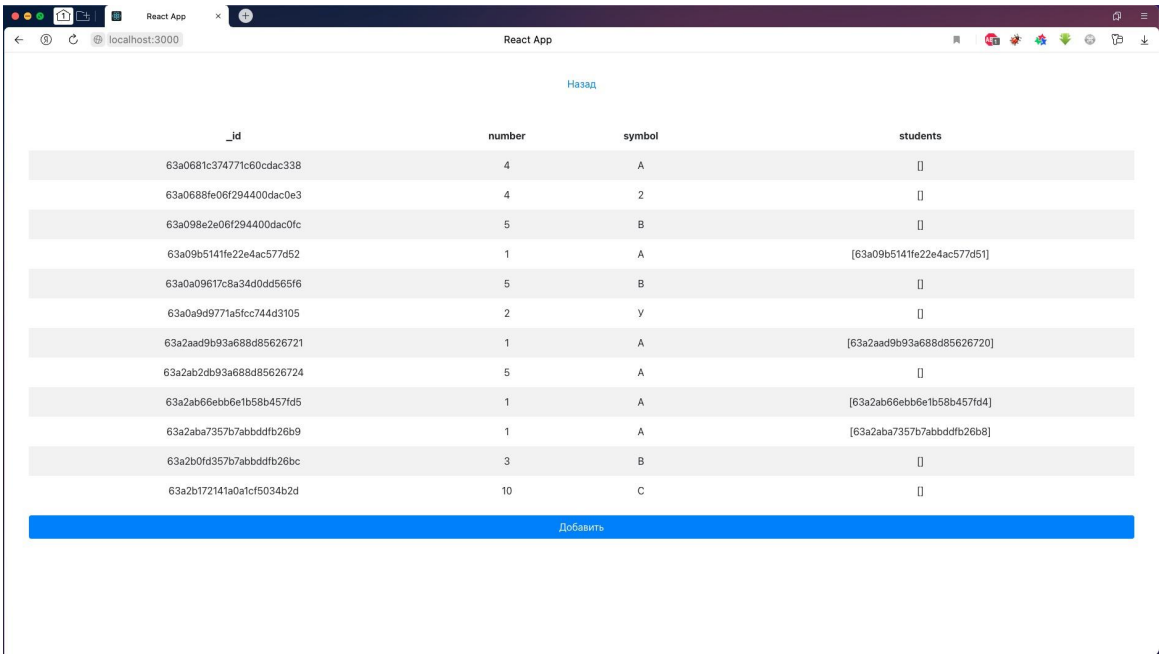

Рисунок 10 – Просмотр/добавление в таблицу

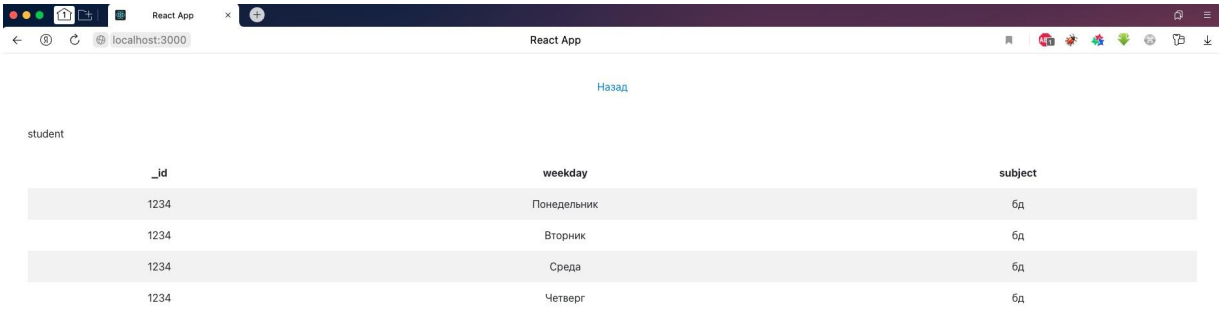

Рисунок 11 – Просмотр расписания

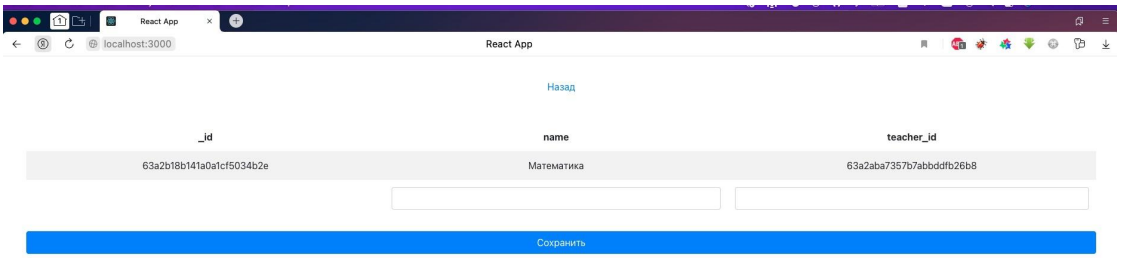

Рисунок 12 – Просмотр таблиц

Использованные технологии

СУБД: MongoDB

Backend: Python, Flask, PyMongo

Frontend: HTML, CSS

Ссылка на приложение

Ссылка на репозиторий проекта:

https://github.com/moevm/nosql2h22-school

ЗАКЛЮЧЕНИЕ

В ходе работы было разработано веб-приложение, позволяющее автоматизировать процесс обучения в школе. Приложение позволяет взаимодействовать с базой данных: просмотр содержимого, добавление элементов и экспорт/импорт данных, а также была реализована авторизация.

В данный момент интерфейс приложения не соответствует макету, а также не реализован поиск элементов. Также в приложении не был до конца реализован массовый экспорт и импорт данных.

В дальнейшем планируется исправить недостатки системы и реализовать интерфейс, соответствующий макету.

ПРИЛОЖЕНИЕ

Документация по сборке и развертыванию приложения

1. Скачать проект из репозитория;
2. npm start - запустить фронтэнд;

   python3 src/main.py - запустить бэкэнд.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Документация Python [Электронный ресурс] URL: https://docs.python.org/3/

[2] Документация MongoDB [Электронный ресурс] URL:
https://www.mongodb.com/docs/

[3] Документация Flask [Электронный ресурс] URL:
https://flask.palletsprojects.com/en/2.2.x/

[4] Документация JavaScript [Электронный ресурс] URL:
https://developer.mozilla.org/ru/docs/Web/JavaScript

[5] Документация React [Электронный ресурс] URL:
https://ru.reactjs.org/docs/getting-started.html