

Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,
МЕХАНИКИ И ОПТИКИ”

Факультет ИКТ

Образовательная программа 45.03.04 - Интеллектуальные системы в гуманитарной
сфере

Направление подготовки (специальность) 45.03.04 - Интеллектуальные системы в
гуманитарной сфере

О Т Ч Е Т

по курсовой работе

Тема задания: Реализация web-сервисов средствами Django REST framework, Vue.js, Muse-UI

Обучающийся Артамонова Валерия К3343

Руководитель: Говоров А.И.

Оценка ____

Дата ____

Санкт-Петербург
20 20

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
ГЛАВА 1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ И ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ	4
1.1 Описание варианта	4
1.2 Описание предметной области и функциональные требования.....	4
ГЛАВА 2. ОПИСАНИЕ СЕРВЕРНОЙ ЧАСТИ	6
2.1 Модель данных.....	6
2.2 Сериализация и создание отображений	7
2.3 Полученные интерфейсы в Django REST	8
ГЛАВА 3. ОПИСАНИЕ КЛИЕНТСКОЙ ЧАСТИ.....	12
3.1 Описание средств разработки клиентской части	12
3.2 Полученные интерфейсы.....	13
ГЛАВА 4. КОНТЕЙНЕРИЗАЦИЯ И ОРКЕСТРАЦИЯ	26
ЗАКЛЮЧЕНИЕ.....	27
СПИСОК ЛИТЕРАТУРЫ	28
ПРИЛОЖЕНИЯ	29

ВВЕДЕНИЕ

Разработка web-приложений является большой частью современной IT-индустрии. Сегодня практически все компьютеры (настольные и мобильные) подключены к сети Интернет и в основном используют один из ее сервисов – World Wide Web (Всемирную Паутину, web-сеть). Первоначально данный сервис (web-сеть) использовался только для связывания и предоставления статической информации. Однако в настоящее время он стал платформой для удаленного использования специальных прикладных программ – web-приложений. Если ранее, прежде чем использовать приложение на локальном компьютере, требовалось его устанавливать, то сейчас приложение можно запускать с помощью web-браузера (Интернет-обозревателя), просто указав его адрес. При этом само приложение выполняется на удаленном компьютере (сервере), а пользователь может работать с ним на своем компьютере с помощью web-браузера.

В связи с активным развитием сети Интернет, web-сети и изменением подхода к использованию приложений специалисты в области информационных технологий должны уметь разрабатывать такие web-приложения.

Создание web-приложений требует от специалистов по информационным технологиям, помимо умения программировать на каком-либо универсальном языке (C#, Java, Python, Ruby и т. п.), знать основные стандарты сети Интернет, такие как: URL, HTTP, HTML, CSS и JavaScript.

Целью данной курсовой работы является создание web-приложения согласно выбранному варианту с использованием современных средств web-разработки, таких как Django REST, Vue.js и Muse-ui.

ГЛАВА 1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ И ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

1.1 Описание варианта

Создать программную систему, предназначенную для завуча школы. Она должна обеспечивать хранение сведений о каждом учителе, классном руководстве, о предметах, которые он преподает в заданный период, номере закрепленного за ним кабинета, о расписании занятий. Существуют учителя, которые не имеют собственного кабинета. Об учениках должны храниться следующие сведения: фамилия и имя, в каком классе учится, какую оценку имеет в текущей четверти по каждому предмету.

Завуч должен иметь возможность добавить сведения о новом учителе или ученике, внести в базу данных четвертные оценки учеников каждого класса по каждому предмету, удалить данные об уволившемся учителе и отчисленном из школы ученике, внести изменения в данные об учителях и учениках, в том числе поменять оценку ученика по тому или иному предмету. В задачу завуча входит также составление расписания.

Завучу могут потребоваться следующие сведения:

- Какой предмет будет в заданном классе, в заданный день недели на заданном уроке?
- Сколько учителей преподает каждую из дисциплин в школе?
- Список учителей, преподающих те же предметы, что и учитель, ведущий информатику в заданном классе.
- Сколько мальчиков и девочек в каждом классе?
- Сколько кабинетов в школе для базовых и профильных дисциплин?

Необходимо предусмотреть возможность получения документа, представляющего собой отчет об успеваемости заданного класса. Отчет включает сведения об успеваемости за четверть по каждому предмету. Необходимо подсчитать средний балл по каждому предмету, по классу в целом, указать общее количество учеников в классе. Для класса указать классного руководителя.

1.2 Описание предметной области и функциональные требования

Согласно варианту, в качестве предметной области выбрана некая абстрактная школа. Информационная система для данной сферы занимается управлением стандартными для школы процессами: принятием и исключением учеников, наймом и увольнением учителей, составлением расписания и выставлением семестровых оценок. Данные об учениках и

учителях содержат личную информацию, которая не должна быть доступна любому пользователю web-приложения, следовательно, необходимо реализовать авторизацию и регистрацию пользователя-завуча в системе. Таким образом, нужно было разработать web-приложение, которое должно обладать следующим функционалом:

Для пользователя:

1. Авторизация и регистрация в web-приложении.
2. Просмотр информации об учителях, учениках, классах, расписании и оценках.
3. Возможность добавления/изменения/удаления учителя, ученика, записи в расписании и оценки.
4. Получение результатов запросов.
5. Получение отчета по выбранному классу.

Распределение всего функционала показано на use-case диаграмме, представленной на рисунке 1.

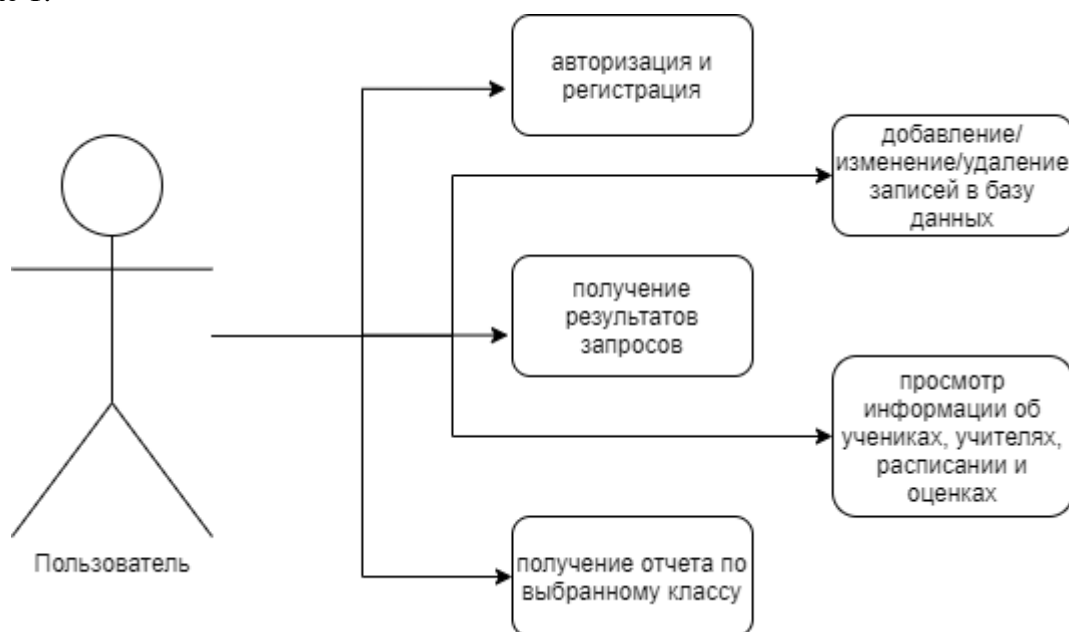


Рисунок 1 – Use-case диаграмма

ГЛАВА 2. ОПИСАНИЕ СЕРВЕРНОЙ ЧАСТИ

2.1 Модель данных

Согласно варианту и выявленным функциональным требованиям была создана модель данных, представленная на рисунке 2.

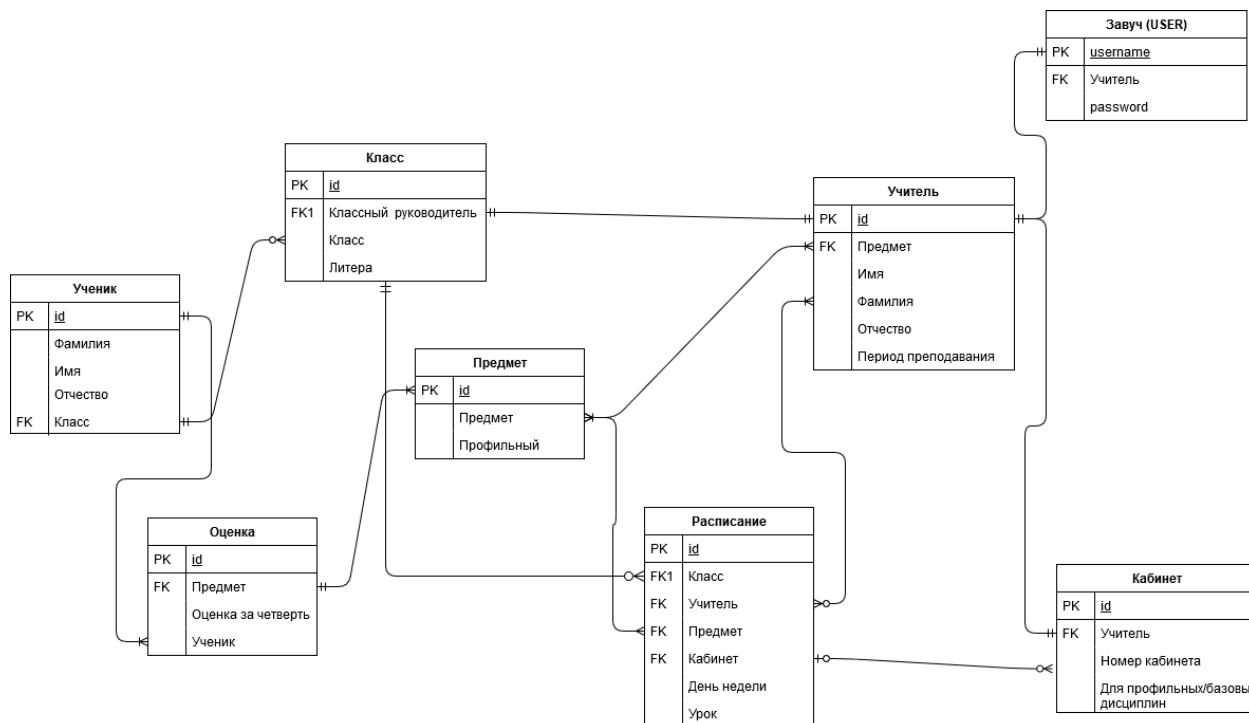


Рисунок 2 – модель данных

Данная модель данных содержит 8 сущностей, между которыми установлены отношения один-ко-многим, многие-ко-многим и один-к-одному.

Для управления базой данных была выбрана PostgreSQL – объектно-реляционная система управления базами данных (ОРСУБД, ORDBMS), основанная на POSTGRES, Version 4.2[1].

После подключения созданной на сервере Postgre базы данных к бэкенду Django были созданы следующие модели в файле models.py (листинг кода приведен в приложении 1):

- Class Subject – Предмет – данная модель содержит информацию о названии предмета и является он профильным, или базовым.
- Class Teacher – Учитель – данная модель содержит информацию об учителе: его имя, фамилию, отчество, предмет, который он преподает, и в течение какого периода он работает в школе.
- Class Cabinet – Кабинет – данная модель содержит информацию о кабинете, ответственном за него учителя и является он кабинетом для профильных, или базовых, дисциплин.

- Class Klass – Класс – данная модель содержит информацию о классе: его номер и литера; учителе, который является классным руководителем.
- Class Pupil – Ученик – данная модель содержит информацию об ученике: фамилию, имя, отчество, пол и в каком классе он учится.
- Class Grade – Оценка – данная модель содержит информацию о семестровых оценках каждого ученика по определенному предмету.
- Class Timetable – Расписание – данная модель содержит информацию о записи в расписании: какой учитель где и когда ведет какой предмет у какого класса.

2.2 Сериализация и создание отображений

Среда сериализации Django предоставляет механизм для «перевода» моделей Django в другие форматы. Обычно эти другие форматы основаны на тексте и используются для отправки данных Django, но сериализатор может обрабатывать любой формат (текстовый или нет) [2]. Использование сериализаторов в данной работе необходимо для того, чтобы серверная часть, написанная на Django REST, могла без проблем обмениваться данными с клиентской частью, написанной на Vue.js. Сериализация также помогает сделать вывод данных из моделей более понятным для пользователей, например, с помощью функции `related_name` можно заменить значение внешнего ключа, которое по умолчанию является `id`, на другое поле модели, которая является внешним ключом. Листинг кода всех сериализаторов приведен в приложении 2.

Отображения для запросов были созданы с помощью встроенной в платформу REST абстракции для работы с ViewSets, которая позволяет разработчику сконцентрироваться на моделировании состояния и взаимодействий API и оставить обработку URL-адреса автоматически, основываясь на общих соглашениях.

Класс `ViewSet` - это просто тип представления на основе классов, который не предоставляет никаких обработчиков методов, таких как `.get ()` или `.post ()`, и вместо этого предоставляет такие действия, как `.list ()` и `.create ()`. Обработчики метода для `ViewSet` связаны только с соответствующими действиями в момент завершения представления, используя метод `.as_view ()`. [3]. Использование классов `ViewSet` позволяет практически сразу получить CRUD(Create-Update-Delete) для любой модели с использованием одного, или нескольких, если это необходимо, сериализаторов.

Для создания отображений, отвечающих за выполнение запросов и предоставления отчета, был выбран класс `APIView`. Использование класса `APIView` во многом аналогично

использованию обычного класса View, как обычно, входящий запрос отправляется соответствующему методу-обработчику, например .get () или .post (). Кроме того, в классе может быть установлено несколько атрибутов, которые управляют различными аспектами политики API [4]. Листинг кода с созданием отображений приведен в приложении 3.

2.3 Полученные интерфейсы в Django REST

Далее представлены некоторые из полученных интерфейсов в Django REST.

1. Вывод всех учителей

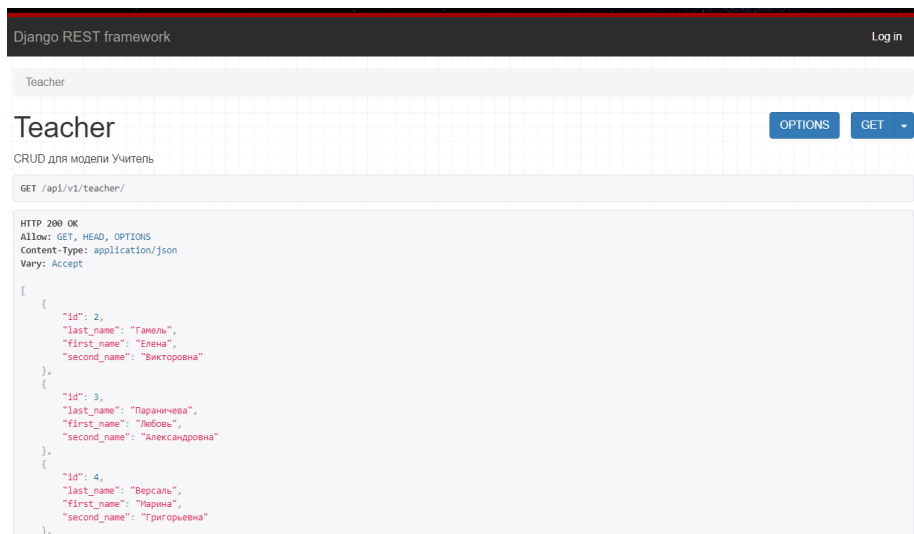


Рисунок 3 – вывод всех учителей

2. Добавление нового учителя

The screenshot shows the Django REST framework interface for the 'Teacher' model. The endpoint is GET /api/v1/teacher/add. The response is a 405 Method Not Allowed error:

```
HTTP 405 Method Not Allowed
Allow: POST, OPTIONS
Content-Type: application/json
Vary: Accept

{
  "detail": "Метод \"/api/v1/teacher/add/\" не разрешен."
}
```

Below the response is an HTML form for adding a new teacher. The form has the following fields:

- Фамилия: Петров
- Имя: Петр
- Отчество: Петрович
- Преподает до: 29.12.2029
- Subject: Информатика

The form also has a 'POST' button and tabs for 'Raw data' and 'HTML form'.

Рисунок 4 –добавление нового учителя

3. Изменение данных в модели Учитель

The screenshot shows the Django REST framework interface for the 'Teacher' model. The breadcrumb trail is 'Teacher / Teacher / Teacher'. The main title is 'Teacher' with an 'OPTIONS' button. Below it, the text 'CRUD для модели Учитель' is displayed. The selected endpoint is 'GET /api/v1/teacher/12/update'. The response status is 'HTTP 405 Method Not Allowed', with 'Allow: POST, OPTIONS' and 'Content-Type: application/json'. The response body is a JSON object:

```
{  "detail": "Метод \"GET\" не разрешен."}
```

. Below the response, there are tabs for 'Raw data' and 'HTML form'. The 'HTML form' tab is active, showing a form for the 'Klass' model. The form fields are: 'Фамилия' (text input), 'Имя' (text input), 'Отчество' (text input), and 'Преподают до' (date input with a calendar icon). A 'POST' button is at the bottom right of the form.

Рисунок 5 – изменение данных об определенном учителе

4. Удаление учителя

The screenshot shows the Django REST framework interface for the 'Teacher' model. The breadcrumb trail is 'Teacher / Teacher / Teacher'. The main title is 'Teacher' with 'DELETE' and 'OPTIONS' buttons. Below it, the text 'CRUD для модели Учитель' is displayed. The selected endpoint is 'GET /api/v1/teacher/12/delete'.

Рисунок 6 – удаление выбранного учителя

5. Вывод всех учеников

The screenshot shows the Django REST framework interface for the 'Pupil' model. The breadcrumb trail is 'Pupil'. The main title is 'Pupil' with 'OPTIONS' and 'GET' buttons. Below it, the text 'CRUD для модели Ученик' is displayed. The selected endpoint is 'GET /api/v1/pupil/'. The response status is 'HTTP 200 OK', with 'Allow: GET, HEAD, OPTIONS' and 'Content-Type: application/json'. The response body is a JSON array of four pupil objects:

```
[  {    "id": 3,    "last_name": "Артамонова",    "first_name": "Валерия",    "second_name": "Евгеньевна"  },  {    "id": 1,    "last_name": "Магченко",    "first_name": "Анастасия",    "second_name": "Александровна"  },  {    "id": 4,    "last_name": "Цветиков",    "first_name": "Антон",    "second_name": "Борисович"  },  {    "id": 6,    "last_name": "Будкевич",    "first_name": "Полина",    "second_name": "Геннадьевна"  }]
```

Рисунок 7 – вывод всех учеников

6. Добавление нового ученика

Django REST framework

Log in

Pupil / Pupil

Pupil

OPTIONS

CRUD для модели Ученик

GET /api/v1/pupil/add

HTTP 405 Method Not Allowed

Allow: POST, OPTIONS

Content-Type: application/json

Vary: Accept

{

"detail": "Метод \"GET\" не разрешен."

}

Raw data

HTML form

Фамилия

Бармина

Имя

Арина

Отчество

Викторовна

Пол

Женский

▼

Klass

5 Б

▼

POST

Рисунок 8 – добавление нового ученика

7. Информация об определенном классе

Django REST framework

Log in

Klass

OPTIONS

GET ▼

CRUD для модели Класс

GET /api/v1/class/3/

HTTP 200 OK

Allow: GET, HEAD, OPTIONS

Content-Type: application/json

Vary: Accept

{

"id": 3,

"teacher": "Версаль",

"pupils": [

{

"id": 14,

"last_name": "Бармина",

"first_name": "Арина",

"second_name": "Викторовна"

}

],

"timetable": [

{

"id": 6,

"subject_name": "Русский язык",

"cabinet_number": "1",

"teacher_name": "Версаль",

"class_name": "5",

"lesson": "1-8:00-8:45",

"day": "Понедельник"

},

{

"id": 9,

"subject_name": "Информатика",

"cabinet_number": "2",

"teacher_name": "Попов",

"class_name": "5",

"lesson": "4-10:40-11:25",

"day": "Четверг"

}

],

"number": "5",

"litera": "Б"

}

Рисунок 9 – вывод информации об определенном классе

8. Добавление записи в расписание

Django REST framework

Log in

Timetable / Timetable

Timetable

OPTIONS

CRUD для модели Расписание

GET /api/v1/timetable/add

HTTP 405 Method Not Allowed

Allow: POST, OPTIONS

Content-Type: application/json

Vary: Accept

```
{
  "detail": "Метод \"GET\" не разрешен."
}
```

Raw data

HTML form

Урок

6-12-20-13:05

▼

День недели

Пятница

▼

Klass name

7 В

▼

Предмет

Информатика

▼

Учитель

Романова Марина Петровна

▼

Кабинет

7 кабинет

▼

POST

Рисунок 10 – добавление новой записи в расписание

ГЛАВА 3. ОПИСАНИЕ КЛИЕНТСКОЙ ЧАСТИ

3.1 Описание средств разработки клиентской части

Для создания клиентской части был использован JavaScript-фреймворк Vue.js и его библиотека Muse-ui. Vue.js — это JavaScript библиотека для создания веб-интерфейсов с использованием шаблона архитектуры MVVM (Model-View-ViewModel).

Поскольку Vue работает только на «уровне представления» и не используется для промежуточного программного обеспечения и бэкэнда, он может легко интегрироваться с другими проектами и библиотеками. Vue.js содержит широкую функциональность для уровня представлений и может использоваться для создания мощных одностраничных веб-приложений.

Библиотека Muse UI, имеющая около 5 тысяч звёзд на GitHub, представляет собой набор компонентов для Vue 2.0, использующих Material Design.

Перед непосредственным созданием клиентской части была спроектирована архитектура web-приложения, представленная на рисунке 11.

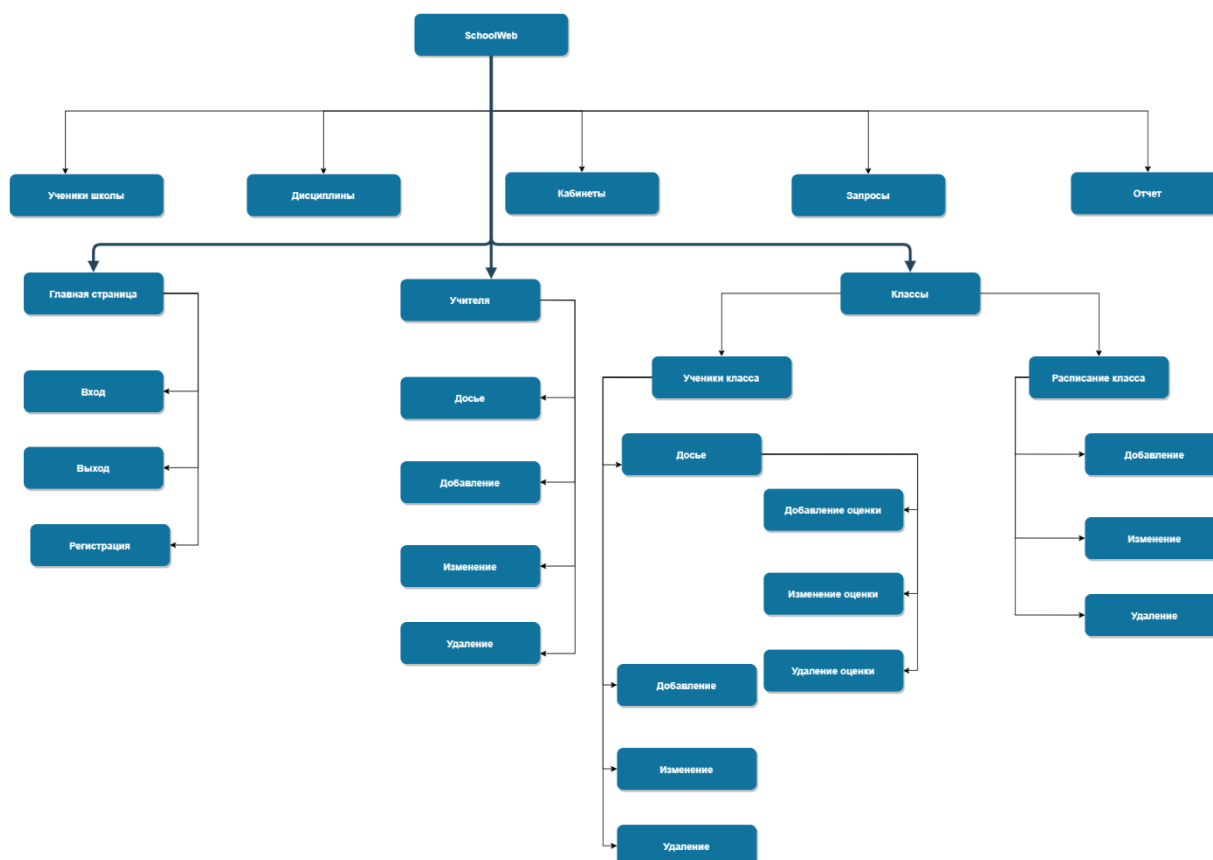


Рисунок 11 – архитектура web-приложения

3.2 Полученные интерфейсы

Согласно заданию и созданной архитектуре web-приложения, были созданы следующие интерфейсы Vue.

1. Главная страница

На главной странице присутствует карусель с изображениями школы и учеников. Скриншот представлен на рисунке 12.



Рисунок 12 –Главная страница

2. Список всех учителей школы

Данный интерфейс представляет собой список всех учителей школы. Каждая строка списка – это активная ссылка на досье выбранного учителя. При нажатии на кнопку «Добавить учителя» появляется модальное окно. Скриншот представлен на рисунке 13.

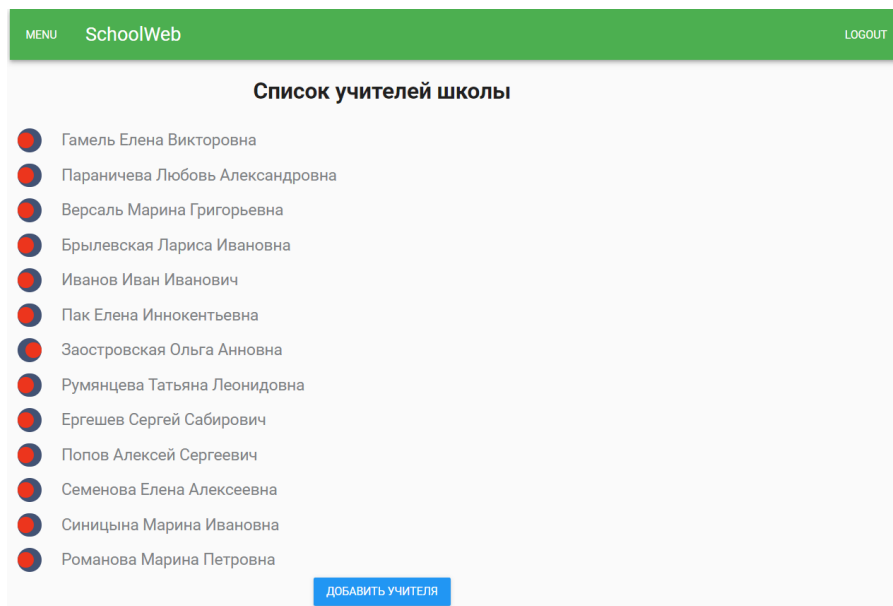
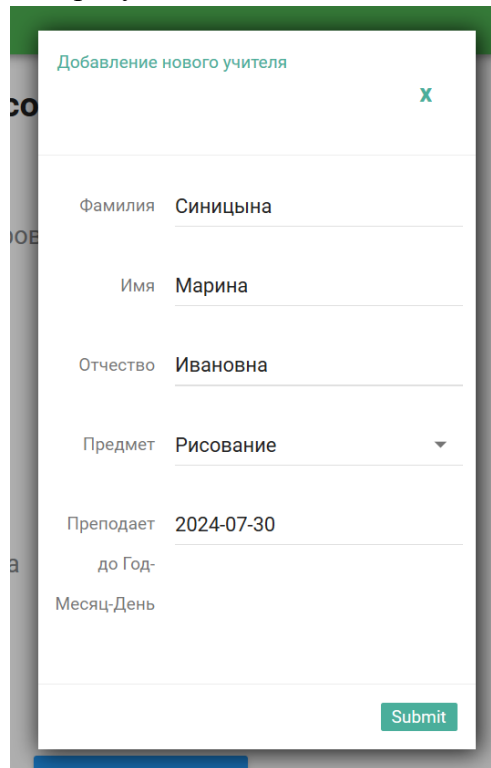


Рисунок 13 –Список всех учителей школы

3. Добавление нового учителя

Данное модельное окно имеет форму создания новой записи в модели «Учитель». Скриншот представлен на рисунке 14.



Добавление нового учителя

Фамилия Синицына

Имя Марина

Отчество Ивановна

Предмет Рисование

Преподает 2024-07-30

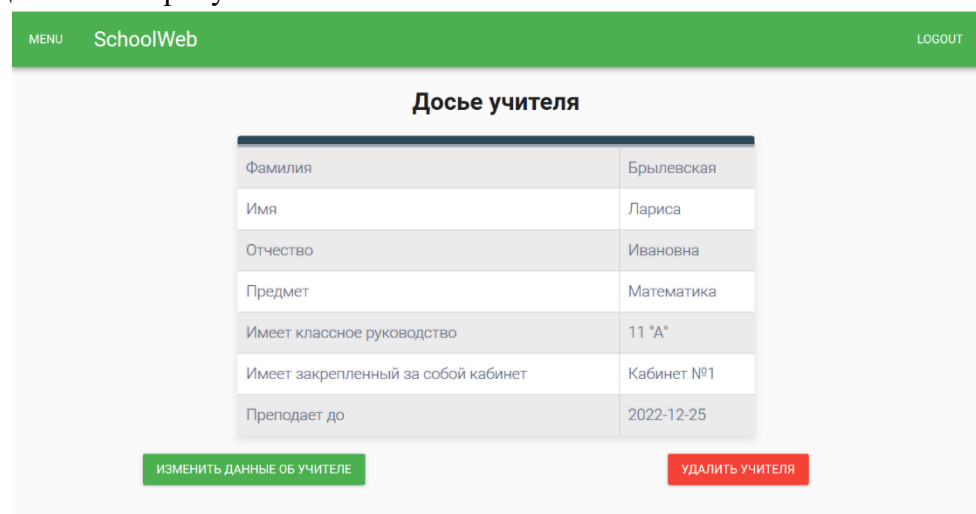
до Год-Месяц-День

Submit

Рисунок 14 –модальное окно «Добавление нового учителя»

4. Досье учителя

Данный интерфейс представляет собой таблицу с данными о выбранном учителе. При нажатии на кнопку «Удалить учителя» данный учитель удаляется из базы данных и происходит перенаправление на страницу «Учителя». Скриншот представлен на рисунке 15.



MENU SchoolWeb LOGOUT

Досье учителя

Фамилия	Брылевская
Имя	Лариса
Отчество	Ивановна
Предмет	Математика
Имеет классное руководство	11 "А"
Имеет закрепленный за собой кабинет	Кабинет №1
Преподает до	2022-12-25

ИЗМЕНИТЬ ДАННЫЕ ОБ УЧИТЕЛЕ УДАЛИТЬ УЧИТЕЛЯ

Рисунок 15 –Досье учителя

5. Изменение данных об учителе

При нажатии на кнопку «Изменить данные об учителе» в таблице с данными об учителе появляются поля ввода. При нажатии на кнопку «Внести изменения» данные о выбранном учителе изменяются и страница обновляется. Скриншот представлен на рисунке 16.

Досье учителя

Фамилия	Брылевская Семенова
Имя	Лариса Лариса
Отчество	Ивановна Ивановна
Предмет	Математика
Имеет классное руководство	11 "А"
Имеет закрепленный за собой кабинет	Кабинет №1
Преподаёт до	2022-12-25 2026-07-25

ВНЕСТИ ИЗМЕНЕНИЯ **УДАЛИТЬ УЧИТЕЛЯ**

Рисунок 16 –изменение данных об учителе

6. Список классов

Данный интерфейс представляет собой список всех классов школы. При нажатии на кнопку «Показать учеников» происходит перенаправление на страницу «Ученики», при нажатии на «Показать расписание» - на страницу «Расписание». Скриншот представлен на рисунке 17.

Список классов

- 11 "А"
ПОКАЗАТЬ УЧЕНИКОВ **ПОКАЗАТЬ РАСПИСАНИЕ**
- 5 "Б"
ПОКАЗАТЬ УЧЕНИКОВ **ПОКАЗАТЬ РАСПИСАНИЕ**
- 7 "В"
ПОКАЗАТЬ УЧЕНИКОВ **ПОКАЗАТЬ РАСПИСАНИЕ**
- 3 "Д"
ПОКАЗАТЬ УЧЕНИКОВ **ПОКАЗАТЬ РАСПИСАНИЕ**
- 8 "Е"
ПОКАЗАТЬ УЧЕНИКОВ **ПОКАЗАТЬ РАСПИСАНИЕ**

Рисунок 17 –Список классов

7. Список учеников выбранного класса

Данный интерфейс представляет собой список учеников выбранного класса. Каждая строка списка – это активная ссылка на досье выбранного ученика. При нажатии на кнопку «Добавить ученика» появляется модальное окно Скриншот представлен на рисунке 18.

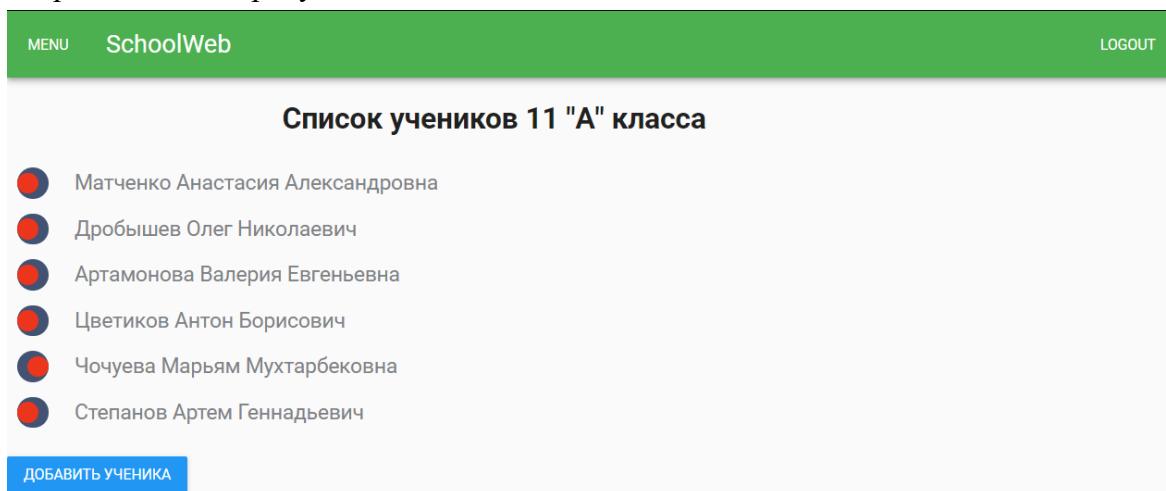


Рисунок 18 –Список учеников выбранного класса

8. Добавление нового ученика

Данное модельное окно имеет форму создания новой записи в модели «Ученик». Скриншот представлен на рисунке 19.

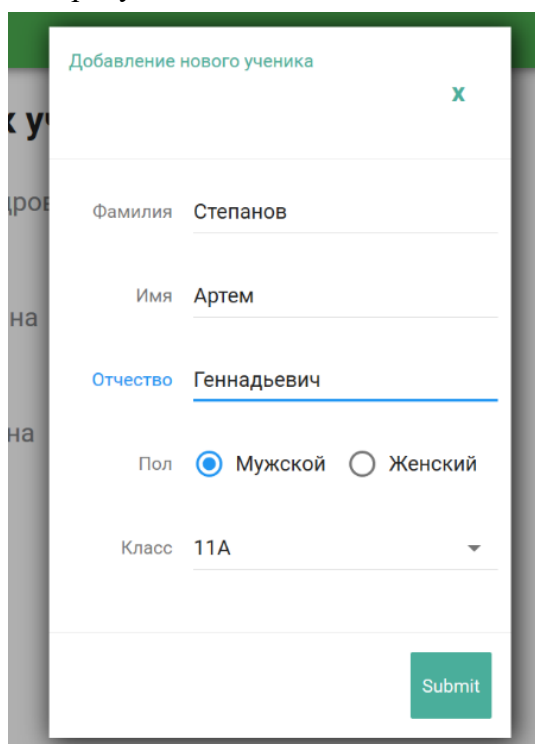


Рисунок 19 –модальное окно «Добавление нового ученика»

9. Досье ученика

Данный интерфейс представляет собой таблицу с данными о выбранном ученике. При нажатии на кнопку «Удалить ученика» данный ученик удаляется из базы данных и происходит перенаправление на страницу «Ученики школы». Скриншот представлен на рисунке 20.

Досье ученика	
Фамилия	Матченко
Имя	Анастасия
Отчество	Александровна
Пол	Женский
Класс	11
Оценки за четверть	
ДОБАВИТЬ ОЦЕНКУ ИЗМЕНИТЬ ОЦЕНКУ УДАЛИТЬ ОДНУ ИЗ ОЦЕНОК	
Математика	5
Русский язык	4
История	4
Химия	5
ИЗМЕНИТЬ ДАННЫЕ ОБ УЧЕНИКЕ УДАЛИТЬ УЧЕНИКА	

Рисунок 20 –Досье ученика

10. Изменение данных об ученике

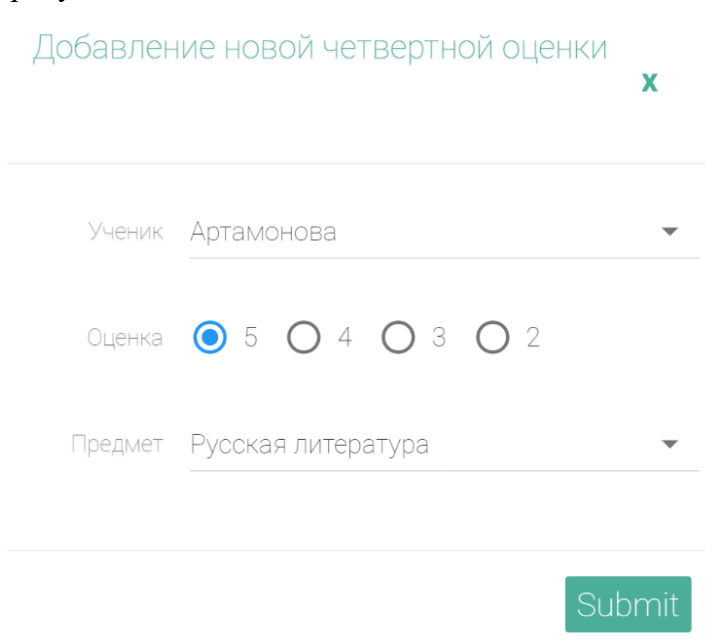
При нажатии на кнопку «Изменить данные об ученике» в таблице с данными об ученике появляются поля ввода. При нажатии на кнопку «Внести изменения» данные о выбранном ученике изменяются и страница обновляется. Скриншот представлен на рисунке 21.

Досье ученика	
Фамилия	Матченко Дробышева
Имя	Анастасия Анастасия
Отчество	Александровна Александровна
Пол	Женский
Класс	11
Оценки за четверть	
ДОБАВИТЬ ОЦЕНКУ ИЗМЕНИТЬ ОЦЕНКУ УДАЛИТЬ ОДНУ ИЗ ОЦЕНОК	
Математика	5
Русский язык	4
История	4
Химия	5
ВНЕСТИ ИЗМЕНЕНИЯ УДАЛИТЬ УЧЕНИКА	

Рисунок 21 –Изменение данных об ученике

11. Добавление оценки ученику

При нажатии на кнопку «Добавить оценку» в интерфейсе «Ученик» появляется модальное окно с формой добавления записи в модель «Оценка». Скриншот представлен на рисунке 22.



Добавление новой четвертной оценки

Ученик: Артамонова

Оценка: ☒ 5 ☐ 4 ☐ 3 ☐ 2

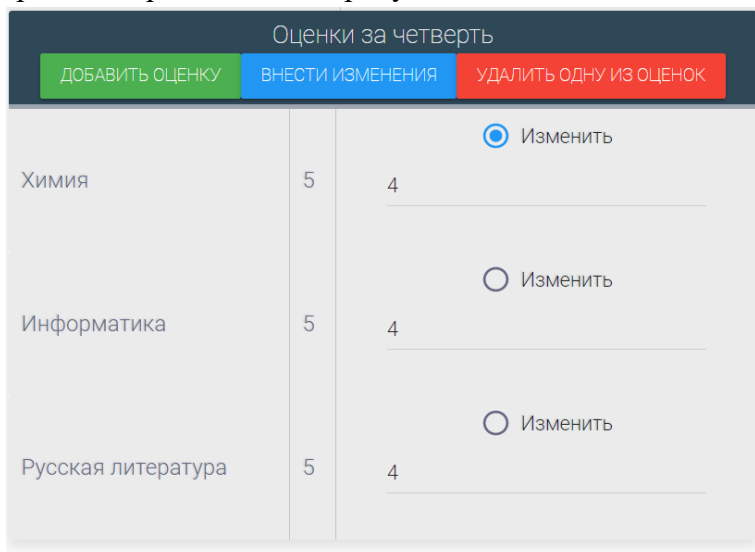
Предмет: Русская литература

Submit

Рисунок 22 –добавление оценки ученику

12. Изменение оценки

При нажатии на кнопку «Изменить оценку» в интерфейсе «Ученик» в таблице с оценками появляются поля ввода и опция выбора оценки. После нажатия на кнопку «Внести изменения» изменения вносятся в базу данных и страница обновляется. Скриншот представлен на рисунке 23.



Оценки за четверть		
<input type="button" value="ДОБАВИТЬ ОЦЕНКУ"/> <input type="button" value="ВНЕСТИ ИЗМЕНЕНИЯ"/> <input type="button" value="УДАЛИТЬ ОДНУ ИЗ ОЦЕНОК"/>		
Химия	5	<input checked="" type="radio"/> Изменить 4
Информатика	5	<input type="radio"/> Изменить 4
Русская литература	5	<input type="radio"/> Изменить 4

Рисунок 23 –изменение оценки

13. Удаление оценки

При нажатии на кнопку «Удалить одну из оценок» в интерфейсе «Ученик» рядом с каждой оценкой появляется опция выбора, при нажатии на кнопку «Подтвердить» выбранная оценка удаляется и страница обновляется. Скриншот представлен на рисунке 24.

Досье ученика	
Фамилия	Артамонова
Имя	Валерия
Отчество	Евгеньевна
Пол	Женский
Класс	11

Оценки за четверть		
<div>ДОБАВИТЬ ОЦЕНКУ ИЗМЕНИТЬ ОЦЕНКУ ПОДТВЕРДИТЬ</div>		
Информатика	5	<input type="radio"/> Удалить
Русская литература	5	<input type="radio"/> Удалить
Химия	4	<input checked="" type="radio"/> Удалить

Рисунок 24 –удаление оценки

14. Список всех учеников школы

Данный интерфейс представляет собой список всех учеников школы. Каждая строка списка – это активная ссылка на досье выбранного ученика. При нажатии на кнопку «Добавить ученика» появляется модальное окно, скриншот которого представлен на рисунке 19. Скриншот самого интерфейса представлен на рисунке 25.

MENU SchoolWeb LOGOUT

Список учеников школы

☒

 Артамонова Валерия Евгеньевна

☒

 Матченко Анастасия Александровна

☒

 Цветиков Антон Борисович

☒

 Будкевич Полина Геннадьевна

☒

 Иванова Иванка Ивановна

☒

 Red Hot Chili Peppers

☒

 Дробышев Олег Николаевич

☒

 Чочуева Марьям Мухтарбековна

☒

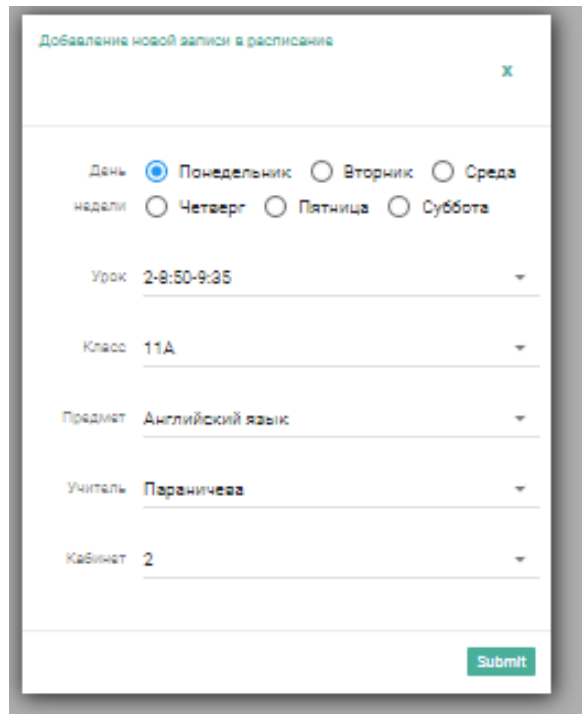
 Степанов Артем Геннадьевич

ДОБАВИТЬ УЧЕНИКА

Рисунок 25 –Список всех учеников школы

15. Добавление записи в расписание

При нажатии на кнопку «Добавить запись в расписание» в интерфейсе «Расписание класса» появляется модальное окно с формой добавления записи в модель «Расписание». Скриншот представлен на рисунке 26.

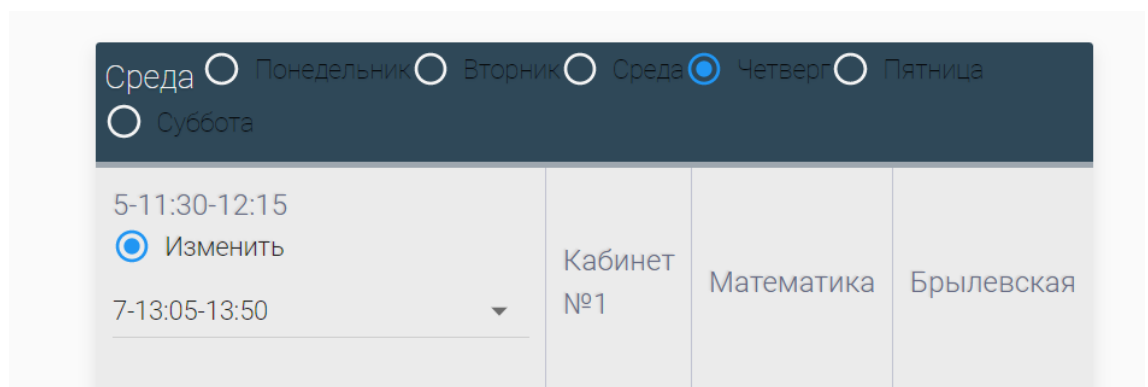


The screenshot shows a modal window titled "Добавление новой записи в расписание" (Adding a new record to the schedule). It contains several form fields with dropdown menus and radio buttons. The fields are: "День недели" (Day of the week) with radio buttons for "Понедельник" (Monday), "Вторник" (Tuesday), "Среда" (Wednesday), "Четверг" (Thursday), "Пятница" (Friday), and "Суббота" (Saturday); "Урок" (Lesson) with a dropdown menu showing "2-8:50-9:35"; "Класс" (Class) with a dropdown menu showing "11А"; "Предмет" (Subject) with a dropdown menu showing "Английский язык" (English); "Учитель" (Teacher) with a dropdown menu showing "Параничева"; and "Кабинет" (Cabin) with a dropdown menu showing "2". A green "Submit" button is located at the bottom right of the form.

Рисунок 26 –модальное окно «Добавление новой записи в расписание»

16. Изменение записи в расписании

При нажатии на кнопку «Изменить одну из записей» в интерфейсе «Расписание класса» в таблице появляются поля для ввода и опция выбора записи для изменения. После нажатия на кнопку «Внести изменения» изменения вносятся в базу данных и страница обновляется. Скриншот представлен на рисунке 27.



The screenshot shows a table with a header row for days of the week: "Среда", "Понедельник", "Вторник", "Среда", "Четверг", "Пятница", and "Суббота". The "Четверг" (Thursday) column is selected. Below the header, there is a table with four columns. The first column contains lesson times and an "Изменить" (Edit) button. The second column contains the cabin number. The third column contains the subject. The fourth column contains the teacher's name. The data in the first row is: "5-11:30-12:15", "Изменить", "Кабинет №1", "Математика", and "Брылевская". The second row shows "7-13:05-13:50" and a dropdown arrow.

Рисунок 27 –изменение одной из записей в расписании

17. Удаление записи из расписания

При нажатии на кнопку «Удалить одну из записей» в интерфейсе «Расписание класса» рядом с каждой записью появляется опция выбора, при нажатии на кнопку «Подтвердить» выбранная запись удаляется из базы данных и страница обновляется. Скриншот представлен на рисунке 28.

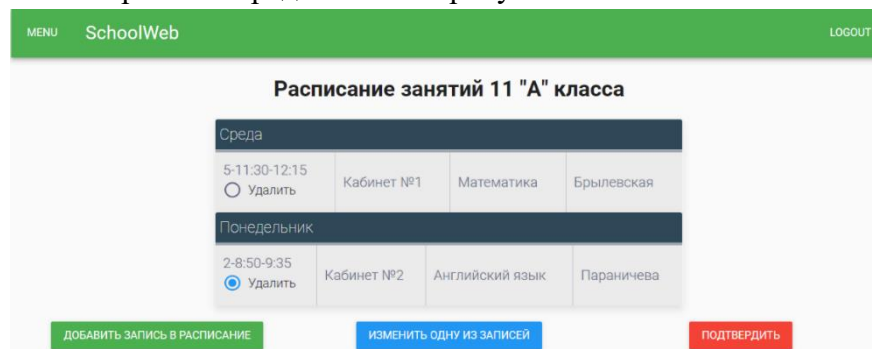


Рисунок 28 –удаление одной из записей в расписании

18. Список дисциплин, преподаваемых в школе

Данный интерфейс представляет собой список дисциплин, преподаваемых в школе. Скриншот представлен на рисунке 29.

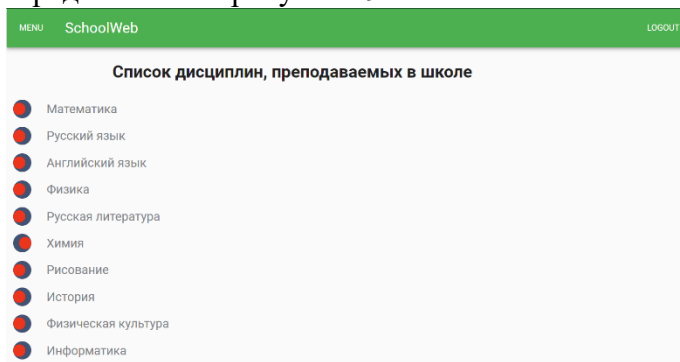


Рисунок 29 –Список дисциплин, преподаваемых в школе

19. Список кабинетов школы

Данный интерфейс представляет собой список кабинетов школы. Скриншот представлен на рисунке 30.

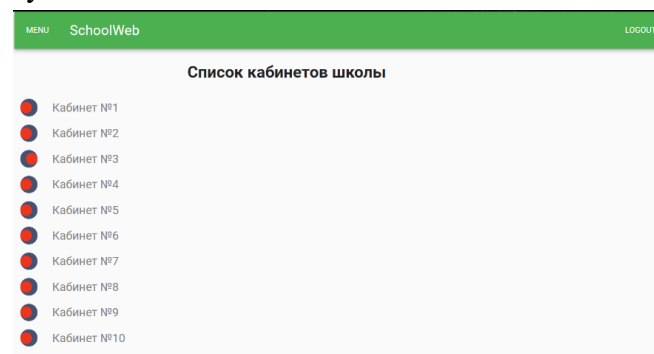


Рисунок 30 –Список кабинетов школы

20. Вход

Данный интерфейс представляет собой форму входа. В случае некорректного ввода логина и/или пароля появится соответствующее предупреждение. При нажатии на активную ссылку «зарегистрироваться» пользователь будет перенаправлен на страницу регистрации. Скриншот представлен на рисунке 31.

MENU SchoolWeb LOGIN

UserName
user
A character with a username of 6-12 length

Password

Please enter password

ВОЙТИ

Нет логина и пароля? Тогда Вам необходимо [зарегистрироваться](#)

Рисунок 31 –Вход

21. Регистрация

Интерфейс представляет собой форму регистрации нового пользователя. В случае успешного прохождения регистрации пользователь будет перенаправлен на страницу входа. Скриншот представлен на рисунке 32.

MENU SchoolWeb LOGIN

Пожалуйста, заполните следующие поля

Поля должны содержать не менее 8 символов.

99879vjgv

Password
***** visibility

Password again
***** visibility

SIGN UP

Рисунок 32 –Регистрация

22. Отчет по классу

Данный интерфейс выполняет функцию получения отчета по выбранному классу. Скриншот представлен на рисунке 33.

MENU SchoolWeb LOGOUT

Класс
11A

ПОЛУЧИТЬ ОТЧЕТ ПО КЛАССУ

Количество человек в классе 6

Классный руководитель Брылевская

Рисунок 33 –Отчеты

23. Реализация запросов

Интерфейс «Запросы» представляет собой список запросов, в котором при нажатии на каждую строку списка появляется форма или результат выполнения выбранного запроса. Скриншот представлен на рисунке 34.

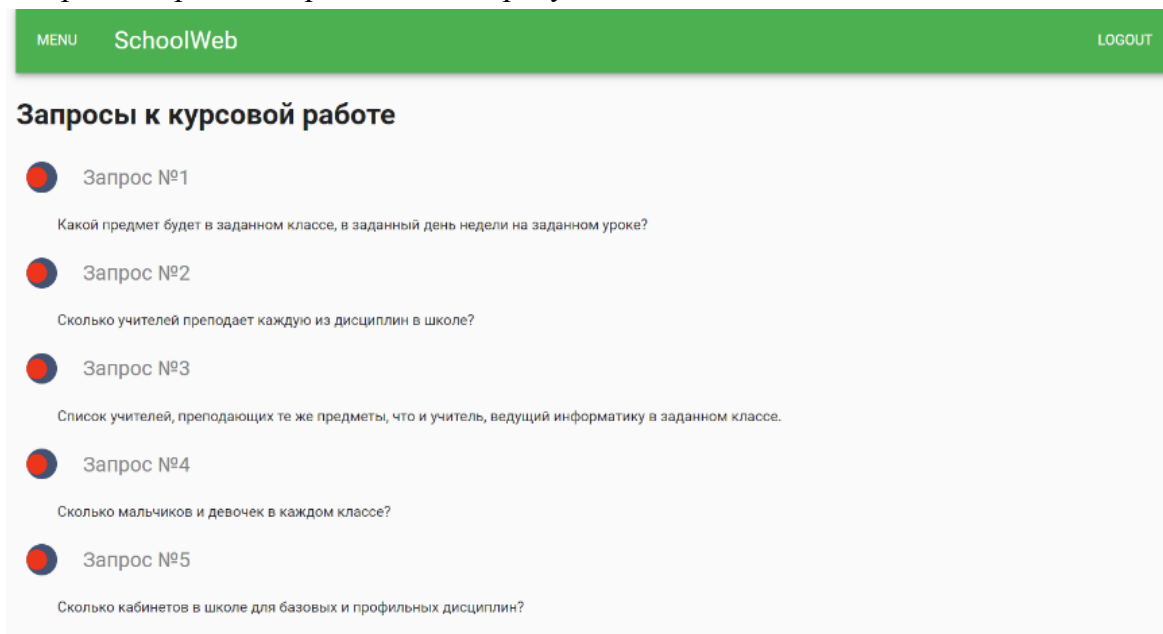


Рисунок 34 –Запросы к курсовой работе

а. Запрос 1

Какой предмет будет в заданном классе, в заданный день недели на заданном уроке?

Результат выполнения запроса представлен на рисунке 35

Запрос №1
Какой предмет будет в заданном классе, в заданный день недели на заданном уроке?

Пожалуйста, выберите данные для выполнения запроса

День недели
☐ Понедельник ☐ Вторник ☐ Среда
☒ Четверг ☐ Пятница ☐ Суббота

Урок
4-10:40-11:25

Класс
5Б

ВЫПОЛНИТЬ ЗАПРОС

Результат выполнения запроса

9	id
Информатика	subject_name
2	cabinet_number
Попов	teacher_name
5	klass_name
4-10:40-11:25	lesson
Четверг	day

Рисунок 35 –Запрос 1

б. Запрос 2

Сколько учителей преподает каждую из дисциплин в школе? Результат выполнения запроса представлен на рисунке 36.

Запрос №2

Сколько учителей преподает каждую из дисциплин в школе?

Результат выполнения запроса	
Предмет	Количество учителей
Математика	2
Английский язык	1
Русский язык	1
Физика	1
Химия	1
Русская литература	1
Рисование	2
Физическая культура	1
Информатика	2
История	1

Рисунок 36 –Запрос 2

с. Запрос 3

Список учителей, преподающих те же предметы, что и учитель, ведущий информатику в заданном классе. Результат выполнения запроса представлен на рисунке 37.

Запрос №3

Список учителей, преподающих те же предметы, что и учитель, ведущий информатику в заданном классе.

Класс

5Б

Предмет

Информатика

ВЫПОЛНИТЬ ЗАПРОС

Результат выполнения запроса

9

Информатика

2

Попов

Рисунок 37 –Запрос 3

d. Запрос 4

Сколько мальчиков и девочек в каждом классе? Результат выполнения запроса представлен на рисунке 38.

 Запрос №4


Сколько мальчиков и девочек в каждом классе?

Результат выполнения запроса		
Класс	Пол	Количество учеников
1	Женский	3
1	Мужской	3
4	Женский	1
5	Мужской	1
6	Женский	1

Рисунок 38 –Запрос 4

e. Запрос 5

Сколько кабинетов в школе для базовых и профильных дисциплин? Результат выполнения запроса представлен на рисунке 39.

 Запрос №5

Сколько кабинетов в школе для базовых и профильных дисциплин?

Результат выполнения запроса	
Тип кабинета	Количество кабинетов
Кабинеты для профильных дисциплин	0
Кабинеты для базовых дисциплин	7

Рисунок 39 –Запрос 5

ГЛАВА 4. КОНТЕЙНЕРИЗАЦИЯ И ОРКЕСТРАЦИЯ

Контейнеризация — это подход к разработке программного обеспечения, при котором приложение или служба, их зависимости и конфигурация (абстрактные файлы манифеста развертывания) упаковываются вместе в образ контейнера. Контейнеризованное приложение может быть протестировано как модуль и развернуто в виде экземпляра контейнера в операционной системе (ОС) текущего узла. Docker — это проект с открытым исходным кодом для автоматизации развертывания приложений в виде переносимых, самодостаточных контейнеров, которые могут работать в облаке или локально.

Оркестрация — это координация взаимодействия нескольких контейнеров. Конечно, можно работать и без оркестрации — никто не запрещает создать контейнер, в котором будут запущены все необходимые процессы. Однако в этом случае приложение лишается гибкости, масштабируемости, а также возникнут вопросы безопасности, поскольку запущенные в одном контейнере процессы не будут изолированы и смогут влиять друг на друга.

Оркестрация позволяет создавать информационные системы из множества контейнеров, каждый из которых отвечает только за одну определенную задачу, а общение осуществляется через сетевые порты и общие каталоги. При необходимости каждый такой контейнер можно заменить другим, что позволяет, например, быстро перейти на другую версию базы данных при необходимости [5].

Для того, чтобы созданное приложение запускалось с помощью оркестрации докер-контейнеров, весь бэкенд был помещен в отдельную папку `server`, а фронтенд — в папку `client`. В каждой папке были созданы соответствующие `Dockerfile`, представленные в приложении 4 для бэкенда и в приложении 5 для фронтенда. Оркестрация происходит с помощью файла `docker-compose`, содержание файла представлено в приложении 6.

ЗАКЛЮЧЕНИЕ

Данная работа показывает полученные в течение семестра навыки создания web-приложений с помощью web-фреймворка Django языка программирования Python, web-фреймворка Vue.js языка программирования JavaScript и библиотека Muse-UI.

В рамках выбранного варианта была создана программная система для завуча школы, обладающая всем необходимым заявленным функционалом:

- авторизация и регистрация в web-приложении.
- просмотр информации об учителях, учениках, классах, расписании и оценках.
- возможность добавления/изменения/удаления учителя, ученика, записи в расписании и оценки.
- получение результатов запросов.
- получение отчета по выбранному классу.

Заключительном этапом создания курсовой была работа с Docker – контейнеризация и оркестрация созданного web-приложения. Таким образом, конечный вариант web-приложения представляет собой образ, который можно развернуть на любой локальной машине, то есть теперь приложение находится уже на стадии разработки, а на стадии продакшена.

СПИСОК ЛИТЕРАТУРЫ

1. Документация PostgreSQL – <https://postgrespro.ru/docs/postgresql/9.6/intro-what-is> (дата обращения: 27.06.2020)
2. Serializing Django objects – <https://docs.djangoproject.com/en/3.0/topics/serialization/> (дата обращения: 27.06.2020)
3. ViewSets – <https://www.django-rest-framework.org/api-guide/viewsets/> (дата обращения: 27.06.2020)
4. Class-based Views - <https://www.django-rest-framework.org/api-guide/views/> (дата посещения 28.06.2020)
5. Что такое оркестрация контейнеров - <https://www.xelent.ru/blog/что-такое-orkestratsiya-konteynerov/> (дата посещения 03.07.2020)

ПРИЛОЖЕНИЯ

```

from django.db import models
from django.contrib.auth.models import User

class Subject(models.Model):
    subject = models.CharField("Предмет", max_length=100)
    profile_types = models.TextChoices('profile_types', 'Профильная_дисциплина Базовая_дисциплина')
    profile = models.CharField("Тип предмета", blank=True, choices=profile_types.choices, max_length=100)

    class Meta:
        verbose_name = "Предмет"
        verbose_name_plural = "Предметы"

    def __str__(self):
        return self.subject

class Teacher(models.Model):
    last_name = models.CharField("Фамилия", max_length=50)
    first_name = models.CharField("Имя", max_length=50)
    second_name = models.CharField("Отчество", max_length=50)
    subject = models.ForeignKey(Subject, on_delete=models.CASCADE)
    teaching_period = models.DateField("Преподает до")

    class Meta:
        verbose_name = "Учитель"
        verbose_name_plural = "Учителя"

    def __str__(self):
        first_name = self.first_name
        last_name = self.last_name
        second_name = self.second_name
        teacher = last_name + " " + first_name + " " + second_name
        return teacher

class UserProfile(models.Model):
    user_id = models.OneToOneField(User, on_delete=models.CASCADE)
    teacher_name = models.ForeignKey(Teacher, on_delete=models.CASCADE)

    def __str__(self):
        return self.teacher_name

class Cabinet(models.Model):
    number = models.CharField("Номер кабинета", max_length=4)
    teacher = models.ForeignKey(Teacher, on_delete=models.CASCADE, related_name='cabinet')
    profile_types = models.TextChoices('profile_types', 'Для_профильных_дисциплин Для_базовых_дисциплин')
    profile = models.CharField("Тип кабинета", blank=True, choices=profile_types.choices, max_length=100)

    class Meta:
        verbose_name = "Кабинет"
        verbose_name_plural = "Кабинеты"

    def __str__(self):
        cabinet = self.number + " кабинет"
        return cabinet

```

```

class Klass(models.Model):
    number_types = models.TextChoices('number_types', '1 2 3 4 5 6 7 8 9 10 11')
    number = models.CharField("Класс", blank=True, choices=number_types.choices, max_length=2)
    litera_types = models.TextChoices('litera_types', 'А Б В Г Д Е')
    litera = models.CharField("Литера", blank=True, choices=litera_types.choices, max_length=2)
    teacher = models.ForeignKey(Teacher, on_delete=models.CASCADE, related_name='klass')

    class Meta:
        verbose_name = "Класс"
        verbose_name_plural = "Классы"

    def __str__(self):
        klass = self.number + " " + self.litera
        return klass

class Pupil(models.Model):
    last_name = models.CharField("Фамилия", max_length=50)
    first_name = models.CharField("Имя", max_length=50)
    second_name = models.CharField("Отчество", max_length=50)
    gender_types = models.TextChoices('gender_types', 'Мужской Женский')
    gender = models.CharField("Пол", blank=True, choices=gender_types.choices, max_length=10)
    klass = models.ForeignKey(Klass, on_delete=models.CASCADE, related_name="pupils")

    class Meta:
        verbose_name = "Ученик"
        verbose_name_plural = "Ученики"

    def __str__(self):
        pupil = self.last_name + " " + self.first_name + " " + self.second_name
        return pupil

class Grade(models.Model):
    student = models.ForeignKey(Pupil, on_delete=models.CASCADE, related_name="grades")
    subject = models.ForeignKey(Subject, on_delete=models.CASCADE)
    grade_types = models.TextChoices('grade_types', '2 3 4 5')
    grade = models.CharField("Оценка за четверть", blank=True, choices=grade_types.choices, max_length=2)

    class Meta:
        unique_together = [
            ("student", "subject", "grade")
        ]
        verbose_name = "Четвертная оценка"
        verbose_name_plural = "Четвертные оценки"

    def __str__(self):
        grade = self.grade
        return grade

class Timetable(models.Model):
    klass_name = models.ForeignKey(Klass, on_delete=models.CASCADE, related_name="timetable")
    lesson_number = models.TextChoices('lesson_number', '1-8:00-8:45 2-8:50-9:35 3-9:40-10:25 4-10:40-11:25 5-11:30-12:15 6-12:20-13:05 7-13:05-13:50 8-14:00-14:45')
    lesson = models.CharField("Урок", blank=True, choices=lesson_number.choices, max_length=50)
    choose_day = models.TextChoices('choose_day', 'Понедельник Вторник Среда Четверг Пятница Суббота')

```

```

day = models.CharField("День недели", blank=True, choices=choose_day.choices, max_length=100)
subject_name = models.ForeignKey(Subject, verbose_name="Предмет", on_delete=models.CASCADE)
teacher_name = models.ForeignKey(Teacher, verbose_name="Учитель", on_delete=models.CASCADE)
cabinet_number = models.ForeignKey(Cabinet, verbose_name="Кабинет", on_delete=models.CASCADE)

class Meta:
    unique_together = [
        ("klass_name", "lesson", "day")
    ]
    verbose_name = "Расписание"
    verbose_name_plural = "Расписание"

def __str__(self):
    timetable = self.day + " " + self.lesson
    return timetable

```

Приложение 2. Файл serializers.py

```

from rest_framework import serializers
from .models import Teacher, Timetable, Klass, Pupil, Cabinet, Subject, Grade

class TeacherSerializer(serializers.ModelSerializer):
    """Список учителей"""
    class Meta:
        model = Teacher
        fields = ("id", "last_name", "first_name", "second_name", "teaching_period")

class TeacherAddSerializer(serializers.ModelSerializer):
    """Добавление учителя"""
    class Meta:
        model = Teacher
        fields = "__all__"

class PupilSerializer(serializers.ModelSerializer):
    """Список учеников"""
    class Meta:
        model = Pupil
        fields = ("id", "last_name", "first_name", "second_name")

class GradeCreateSerializer(serializers.ModelSerializer):
    """Добавление оценки"""
    class Meta:
        model = Grade
        fields = "__all__"

class GradeSerializer(serializers.ModelSerializer):
    """Вывод оценок"""
    subject = serializers.SlugRelatedField(slug_field="subject", read_only=True)

    class Meta:
        model = Grade
        fields = "__all__"

```



```

class PupilDetailSerializer(serializers.ModelSerializer):
    """Досье ученика"""
    klass = serializers.SlugRelatedField(slug_field="number", read_only=True)
    grades = GradeSerializer(many=True)

    class Meta:
        model = Pupil
        fields = "__all__"

class PupilAddSerializer(serializers.ModelSerializer):
    """Добавление ученика"""
    class Meta:
        model = Pupil
        fields = "__all__"

class TimetableAddSerializer(serializers.ModelSerializer):
    """Добавление расписания"""
    class Meta:
        model = Timetable
        fields = "__all__"

class TimetableSerializer(serializers.ModelSerializer):
    """Вывод расписания"""
    subject_name = serializers.SlugRelatedField(slug_field="subject", read_only=True)
    cabinet_number = serializers.SlugRelatedField(slug_field="number", read_only=True)
    teacher_name = serializers.SlugRelatedField(slug_field="last_name", read_only=True)
    klass_name = serializers.SlugRelatedField(slug_field="number", read_only=True)

    class Meta:
        model = Timetable
        fields = "__all__"

class KlassSerializer(serializers.ModelSerializer):
    """Список классов"""
    teacher = serializers.SlugRelatedField(slug_field="last_name", read_only=True)
    class Meta:
        model = Klass
        fields = "__all__"

class KlassAddSerializer(serializers.ModelSerializer):
    """Добавление класса"""
    class Meta:
        model = Klass
        fields = "__all__"

class KlassDetailSerializer(serializers.ModelSerializer):
    """Описание класса"""
    teacher = serializers.SlugRelatedField(slug_field="last_name", read_only=True)
    pupils = PupilSerializer(many=True)
    timetable = TimetableSerializer(many=True)

```

```

class Meta:
    model = Klass
    fields = "__all__"

class SubjectSerializer(serializers.ModelSerializer):
    """Список предметов"""
    class Meta:
        model = Subject
        fields = "__all__"

class CabinetSerializer(serializers.ModelSerializer):
    """Список кабинетов"""
    teacher = serializers.SlugRelatedField(slug_field="last_name", read_only=True)

    class Meta:
        model = Cabinet
        fields = "__all__"

class TeacherDetailSerializer(serializers.ModelSerializer):
    """Досье учителя"""
    subject = serializers.SlugRelatedField(slug_field="subject", read_only=True)
    klass = KlassSerializer(many=True)
    cabinet = CabinetSerializer(many=True)

    class Meta:
        model = Teacher
        fields = "__all__"

```

Приложение 3. Файл views.py

```

from rest_framework import generics, permissions, viewsets, renderers
from django_filters.rest_framework import DjangoFilterBackend
from .service import TimetableFilter
from rest_framework.views import APIView
from rest_framework.response import Response
from collections import Counter
from django.db.models import Count, Avg
from .models import Teacher, Timetable, Klass, Pupil, Cabinet, Subject, Grade
from .serializers import (TeacherSerializer, TeacherDetailSerializer, TeacherAddSerializer, PupilSerializer,
                           PupilDetailSerializer, GradeCreateSerializer, PupilAddSerializer, TimetableSerializer,
                           TimetableAddSerializer, KlassSerializer, SubjectSerializer, CabinetSerializer,
                           KlassDetailSerializer, KlassAddSerializer, GradeSerializer)

class TeacherViewSet(viewsets.ModelViewSet):
    """CRUD для модели Учитель"""
    queryset = Teacher.objects.all()

    def get_serializer_class(self):
        if self.action == 'list':
            return TeacherSerializer
        elif self.action == 'update':
            return TeacherSerializer
        elif self.action == 'create':
            return TeacherAddSerializer

```

```

        elif self.action != 'list':
            return TeacherDetailSerializer

class PupilViewSet(viewsets.ModelViewSet):
    """CRUD для модели Ученик"""
    queryset = Pupil.objects.all()

    def get_serializer_class(self):
        if self.action == 'list':
            return PupilSerializer
        elif self.action == 'update':
            return PupilSerializer
        elif self.action == 'create':
            return PupilAddSerializer
        elif self.action != 'list':
            return PupilDetailSerializer

class GradeViewSet(viewsets.ModelViewSet):
    """CRUD для модели Оценка"""
    queryset = Grade.objects.all()

    def get_serializer_class(self):
        if self.action == 'create':
            return GradeCreateSerializer
        elif self.action != 'create':
            return GradeSerializer

class TimetableViewSet(viewsets.ModelViewSet):
    """CRUD для модели Расписание"""
    queryset = Timetable.objects.all()
    filter_backends = (DjangoFilterBackend,
                       )
    filterset_class = TimetableFilter

    def get_serializer_class(self):
        if self.action == 'list':
            return TimetableSerializer
        elif self.action == 'retrieve':
            return TimetableSerializer
        elif self.action == 'update':
            return TimetableSerializer
        elif self.action == 'create':
            return TimetableAddSerializer

class KlassViewSet(viewsets.ModelViewSet):
    """CRUD для модели Класс"""
    queryset = Klass.objects.all()

    def get_serializer_class(self):
        if self.action == 'list':
            return KlassSerializer
        elif self.action == 'create':
            return KlassAddSerializer

```

```

        elif self.action != 'list':
            return KlassDetailSerializer

class CabinetViewSet(viewsets.ModelViewSet):
    """Отображение для модели Кабинет"""
    queryset = Cabinet.objects.all()
    serializer_class = CabinetSerializer

class SubjectViewSet(viewsets.ModelViewSet):
    """Отображение для модели Предмет"""
    queryset = Subject.objects.all()
    serializer_class = SubjectSerializer

"""Запросы к курсовой работе"""

class Query1(APIView):
    """Какой предмет будет в заданном классе, в заданный день недели на заданном уроке?"""

    def get(self, request):
        klass = request.GET.get('klass')
        #klass = '1'
        day = request.GET.get('day')
        #day = 'Среда'
        lesson = request.GET.get('lesson')
        #lesson = '5-11:30-12:15'
        timetable = Timetable.objects.get(day=day, klass_name=klass, lesson=lesson)
        serializer = TimetableSerializer(timetable, many=False)
        return Response({'data': serializer.data})

class Query2(APIView):
    """Сколько учителей преподает каждую из дисциплин в школе?"""

    def get(self, request):
        teachers = Teacher.objects.all()
        subjects = dict(Counter([teacher.subject.subject for teacher in teachers]))
        return Response({'data': subjects})

class Query3(APIView):
    """Список учителей, преподающих те же предметы, что и учитель, ведущий информатику в заданном классе"""

    def get(self, request):
        klass = request.GET.get('klass')
        #klass = '1'
        #subject = '1'
        subject = request.GET.get('subject')
        timetable = Timetable.objects.get(klass_name=klass, subject_name=subject)
        serializer = TimetableSerializer(timetable, many=False)
        """subjects = Teacher.objects.filter(id=teacher_name)
        subj = [s.subject.id for s in subjects]
        teachers = Teacher.objects.filter(subject=subj)
        teacher = set([t.teacher.last_name for t in teachers])"""

```

```

        return Response({'data': serializer.data})

class Query4(APIView):
    """Сколько мальчиков и девочек в каждом классе?"""

    def get(self, request):
        results = Pupil.objects.values('klass', 'gender').order_by('klass').annotate(Count('gender'))
        return Response({'data': results})

class Query5(APIView):
    """Сколько кабинетов в школе для базовых и профильных дисциплин?"""

    def get(self, request):
        base = Cabinet.objects.filter(profile='Для базовых дисциплин').count()
        for_profile = Cabinet.objects.filter(profile='Для профильных дисциплин').count()
        results = {'Кабинеты для профильных дисциплин': for_profile, 'Кабинеты для базовых дисциплин':
base}
        return Response({'data': results})

class KlassReport(APIView):
    """Получение отчета по классу"""

    def get(self, request):
        klass = request.GET.get('klass')
        #klass = '3'
        teacher = Class.objects.get(id=klass).teacher.last_name
        pupils = Pupil.objects.filter(klass=klass)
        pupils_cnt = pupils.count()
        results = {
            'Количество учеников в классе': pupils_cnt,
            'Классный руководитель': teacher
        }
        return Response({'data': results})

```

Приложение 4 Файл Dockerfile для бэкенда

```

FROM python:3.8.2

ENV PYTHONUNBUFFERED 1

RUN mkdir /server

WORKDIR /server

COPY . /server

RUN pip install -r req.txt

```

Приложение 5. Файл Dockerfile для фронтенда

```

FROM node:12

```

```
WORKDIR /client

COPY package*.json ./

RUN npm install --silent

CMD ["npm", "start"]

COPY ..
```

Приложение 6. Файл docker-compose.yml

```
version: '3'

services:
  db:
    image: postgres
    ports:
      - "5436:5432"
    environment:
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=1234jovi
      - POSTGRES_DB=school
    volumes:
      - ./dbs/postgres-data:/var/lib/postgresql
  backend:
    container_name: school_backend_container
    build: ./server
    command: bash -c "
      sleep 3 &&
      python3 manage.py makemigrations && python3 manage.py migrate &&
      python3 manage.py runserver --insecure 0.0.0.0:8000";
    volumes:
      - ./server:/server
    ports:
      - "8000:8000"
    depends_on:
      - db
  frontend:
    container_name: school_frontend_container
    build:
      context: ./client
      dockerfile: Dockerfile
    command: npm start --start;
    volumes:
      - ./client:/client
      - /client/node_modules
    ports:
      - "8080:8080"
    depends_on:
      - backend
```