Министерство науки высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

(Университет ИТМО)

Факультет: инфокоммуникационных технологий

Образовательная программа: Интеллектуальные системы в гуманитарной сфере (Академический бакалавр, Очная ф/о)

Направление подготовки (специальность): 45.03.04 – Интеллектуальные системы в гуманитарной сфере

О Т Ч Е Т

по курсовой работе

по дисциплине «Основы Web-программирования»

Тема задания: Web-сервис администрирования образовательно процесса для завуча школы

Обучающийся: Шайдуллина Р. Р., группа К3342

Преподаватель дисциплины: Говоров А. И., ассистент факультета ИКТ Университета ИТМО

Оценка за курсовую работу _____

Подписи членов комиссии:

_____ (_____)

(подпись)

_____ (_____)

(подпись)

_____ (_____)

(подпись)

Дата _____

Санкт-Петербург

2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Курсовая работа посвящена созданию программной системы для завуча школы. В любом учебном заведении необходимо вести учет сведений об обучающихся, учителях, расписании и успеваемости. Все эти сведения необходимы завучу для модерирования образовательного процесса. Создаваемый сервис призван собрать в себе этот функционал и предложить понятную и полную систему администрирования школы.

Цель курсовой работы в рамках дисциплины «Основы Web-программирования»: овладеть практическими навыками и умениями реализации Web-сервисов средствами Django REST framework, Vue.js, Muse-UI.

Задачи:

1. Изучить предметную область.
2. Выделить функциональные характеристики.
3. Спроектировать модель данных и реализовать базу данных.
4. Реализовать серверную часть с помощью Django REST framework.
5. Реализовать клиентскую часть с помощью Vue.js и Muse-UI.

1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Предметная область

Предметной областью курсовой работы является управление образовательным процессом. Выполняет его завуч школы. Для этого требуются следующие сведения: о каждом учителе, классном руководстве, о предметах, которые они преподают в заданный период, номере закрепленного за ними кабинета, о расписании занятий. Также необходимо располагать информацией об учениках: фамилия и имя, в каком классе учится, какую оценку имеет в текущей четверти по каждому предмету. Одна из основных задач администрирования завучем процесса обучения – составление расписания, которое, помимо уже упомянутых сведений, требует учета существующих классов, кабинетов и преподаваемых предметов.

1.2. Функциональные требования

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:

1. Какой предмет будет в заданном классе, в заданный день недели на заданном уроке?
2. Сколько учителей преподает каждую из дисциплин в школе?
3. Список учителей, преподающих те же предметы, что и учитель, ведущий информатику в заданном классе.
4. Сколько мальчиков и девочек в каждом классе?
5. 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.

2. ОПИСАНИЕ СЕРВЕРНОЙ ЧАСТИ

2.1. Модель данных

Для реализации всего задуманного функционала в первую очередь разработана модель данных, отражающая все используемые сущности и связи между ними через атрибуты (рисунок 1).

Описание сущностей:

1. Teacher (Учитель). Атрибуты: первичный ключ id, имя, пол и опыт.
2. Class (Класс). Подразумевает учебный класс учеников – 5В, 11А. Атрибуты: первичный ключ name, внешний ключ учитель.
3. Pupil (Ученик). Атрибуты: первичный ключ id, имя, пол и класс.
4. Subject (Предмет, Дисциплина). Атрибуты: первичный ключ названия, тип (базовая/ профильная дисциплина).
5. Teaching (Обучение). Ассоциативная сущность, соединяет сущности Учитель и Предмет. Атрибуты: первичный ключ id, внешние ключи учитель и предмет.
6. Room (Кабинет). Атрибуты: первичный ключ номер, этаж, внешние ключи предмет и учитель.
7. Assessment (Оценивание). Содержит сведения об оценках учеников. Атрибуты: первичный ключ id, четверть, оценка, внешние ключи ученик и предмет.
8. Timetable (Расписание). Включает записи о занятиях в школе. Атрибуты: первичный ключ день недели, порядковый номер урока, внешние ключи класс, предмет, кабинет и учитель.

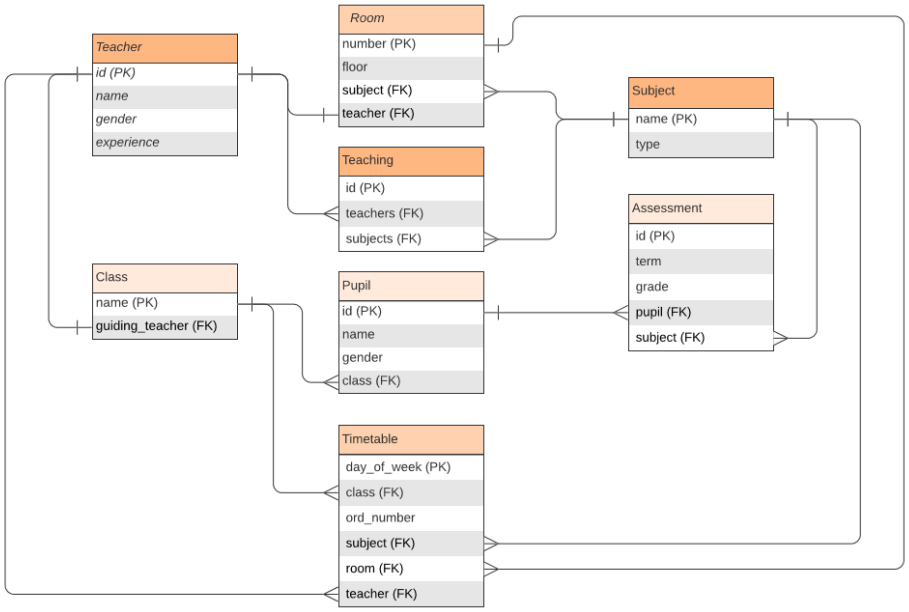


Рисунок 1 – Модель базы данных

2.2. Средства разработки

Реализация программного кода производилась в средах разработки PyCharm и Sublime на языке Python 3.8. Для хранения базы данных использовалась PostgreSQL 12. Бекэнд разрабатывался с использованием Django REST framework. [2]

2.3. Реализованные интерфейсы

Новый проект Django был создан в PyCharm. Сперва были подключены и поставлены все необходимые модули Django и другие библиотеки. Далее была подключена база данных PostgreSQL через задание специальных настроек. Следующим этапом стало создание всех описанных сущностей в файле models.py (рисунок 2) и их регистрация в админ-панели. Через нее также выполнялось заполнение созданных таблиц данными.

```
class Subject(models.Model):
    TYPES = (
        ('major', 'major'),
        ('minor', 'minor')
    )
    name = models.CharField(max_length=20, primary_key=True)
    sub_type = models.CharField(max_length=5, choices=TYPES)

    def __str__(self):
        return self.name
```

Рисунок 2 – Создание сущности Предмет – пример содержимого файла models.py

В файле views.py реализованы все функции, позволяющие вывести необходимую по заданию информацию. Сделаны они в виде классов rest_framework.views.APIView (рисунок 3). Для каждой сущности создан соответствующий класс и включены методы:

1. Для всех: GET – получение данных из данной модели.
2. Избирательно:
    a. PUT – добавление новой записи в модель.
    b. POST – получение данных из текущей модели по переданному параметру.
    c. DELETE – удалить запись из модели.

```
class Subjects(APIView):

    permission_classes = [permissions.IsAuthenticated, ]
    #permission_classes = [permissions.AllowAny, ]

    def get(self, request):
        subjects = Subject.objects.all()
        serializer = SubjectSerializers(subjects, many=True)
        return Response({'data': serializer.data})
```

Рисунок 3 – Методы класса Subjects

Для более полного вывода информации из моделей (например, по внешнему ключу учитель в сущности Кабинет помимо информации о Кабинете получить и данные об учителе) использован файл serializers.py, в котором указано, как выводить получаемую информацию на страницах (рисунок 4).

```
class RoomSerializers(serializers.ModelSerializer):
    #subject = SubjectSerializers()
    #teacher = TeacherSerializers()
    class Meta:
        model = Room
        fields = ('number', 'floor', 'subject', 'teacher')
```

Рисунок 4 – Сериалайзер сущности Кабинет

Для того, чтобы всю запрашиваемую и получаемую информацию вывести, есть файл urls.py, в котором прописаны пути на страницы, куда передается информация (рисунок 5).

```
urlpatterns = [
    path('subjects/', Subjects.as_view()),
    path('rooms/', Rooms.as_view()),
    # path('rooms_sub/', RoomsWithSubject.as_view()),
    path('teachers/', Teachers.as_view()),
    path('teacher/', TeacherOne.as_view()),
    path('pupils/', Pupils.as_view()),
    path('pupil/', PupilOne.as_view()),
    path('classes/', Classes.as_view()),
    path('assessments/', Assessments.as_view()),
    path('assessment/', AssessmentOne.as_view()),
    path('timetable/', Timetables.as_view()),
    path('teaching/', Teachings.as_view()),
    path('query1/', Query1.as_view()),
    path('query2/', Query2.as_view()),
    path('query3/', Query3.as_view()),
    path('query4/', Query4.as_view()),
    path('query5/', Query5.as_view()),
    path('assessments_reports/', Report.as_view())
]
```

Рисунок 5 – Пути передачи данных

Работа приложения запускается исполнением команды python manage.py runserver в интерфейсе командной строки. Приложение работает на локальном хосте 127.0.0.1:8000. На рисунке 6 представлен пример выводимой информации по учителям, где задействовано отображение и сериалайзер учителей.

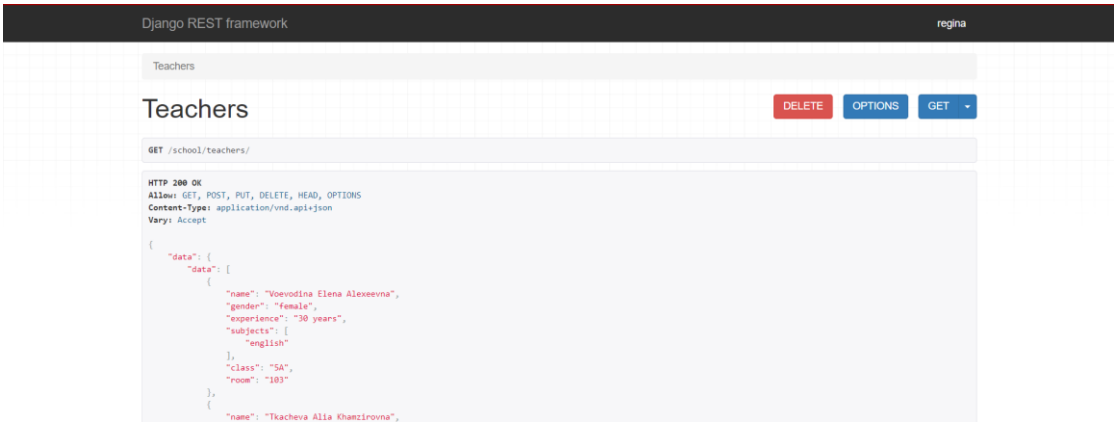


Рисунок 6 – Вывод информации по учителям

3. ОПИСАНИЕ КЛИЕНТСКОЙ ЧАСТИ

3.1. Средства разработки

Реализация фронтенда производилась с помощью Vue.js [3] и Muse-UI.

Был установлен Node.js и создана специальная папка для разработки клиентской части. Для сайта был создан ряд компонентов, каждый из которых отвечает за конкретную часть функционала (например, создание, изменение и удаление учеников, вывод отчета об успеваемости заданного класса и др.). Часть содержимого одного из компонентов представлено на рисунке 7.

```
<template>
    <mu-container><br>
        <mu-container class="button-wrapper2">
            <!-- <mu-button v-if="auth" color="#5c6bc0" textColor="white" @click="">Update</mu-button> -->
            <mu-button v-if="auth" color="#5c6bc0" textColor="white" @click="add">Add</mu-button>
            <mu-button v-if="auth" color="#5c6bc0" textColor="white" @click="del">Delete</mu-button><br><br>
        </mu-container>
        <mu-container>
            <mu-paper>
                <mu-data-table border :columns="columns" :data="timetable">
                    <template slot-scope="scope">
                        <td class="is-left">{{ scope.row.study_class }}</td>
                        <td class="is-left">{{ scope.row.day_of_week }}</td>
                        <td class="is-left">{{ scope.row.lesson_num }}</td>
                        <td class="is-left">{{ scope.row.subject }}</td>
                        <td class="is-left">{{ scope.row.room }}</td>
                        <td class="is-left">{{ scope.row.teacher }}</td>
                    </template>
                </mu-data-table>
            </mu-paper>
            <br><br>
        </mu-container>
    </mu-container>
</template>

<script>
/* eslint-disable */

export default {
    name: 'Timetable',
    data() {
        return {
            timetable: '',
            columns: [
                { title: 'Class', name: 'class', align: 'center' },
                { title: 'Day of Week', name: 'day', align: 'center' },
                { title: 'Lesson number', name: 'lesson_num', align: 'center' },
                { title: 'Subject', name: 'subject', align: 'center' },
                { title: 'Room', name: 'room', align: 'center' },
                { title: 'Teacher', name: 'teacher', width: '250', align: 'center' }
            ]
        }
    },
    computed: {
        auth() {
```

Рисунок 7 – Компонент Timetable.vue

3.2. Реализованные интерфейсы

Сайт выполнен в минималистичном бело-синем цвете, что в целом позволяет не отвлекаться на дизайн при работе с системой и сфокусироваться на важном. Все интерфейсы, а также примеры их работы представлены на рисунках 8-17. Рассмотрим их подробнее.

На рисунке 8 показан интерфейс главной страницы. Для неавторизованного пользователя доступна только кнопка входа, дабы сохранить конфиденциальность всех хранящихся данных об учителях, учениках и их оценках.

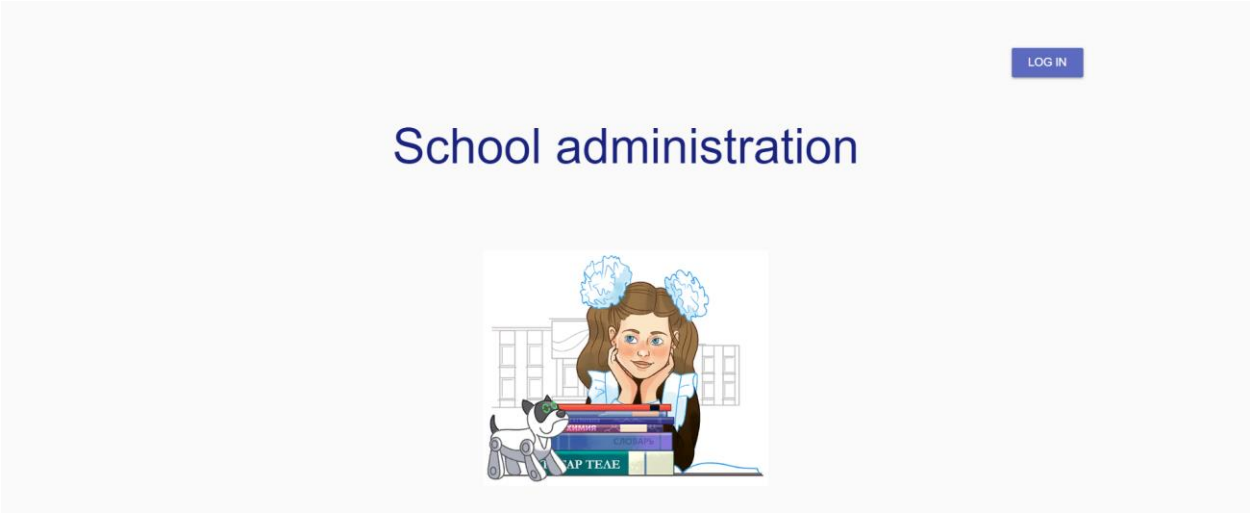


Рисунок 8 – Главная страница для неавторизованного пользователя

Чтобы получить возможность увидеть данные, необходимо войти по кнопке Log in. Поскольку система предназначена для завуча школы, его данные уже есть в базе. Завуч может не только видеть данные, но и редактироваться их. Сам процесс входа показан на рисунке 9.

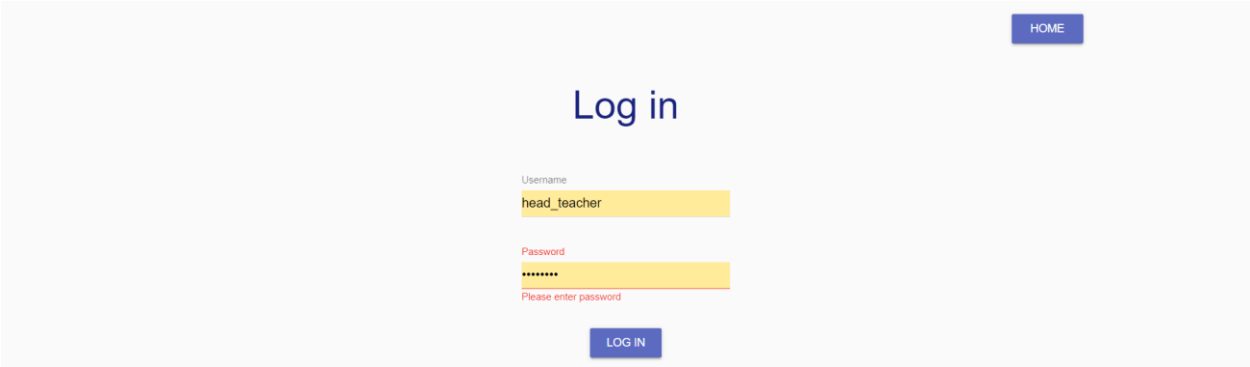


Рисунок 9 – Страница входа

После входа вниманию авторизованного пользователя предлагается картина, представленная на рисунке 10. Кнопка входа теперь становится кнопкой выхода. Главное меню состоит из вкладок посередине экрана, каждая из которых открывается при нажатии на нее на той же странице. Вкладки доступны следующие: «Расписание», «Учителя», «Ученики», «Оценки», общая вкладка «Классы, Предметы, Кабинеты», «Запросы» и вкладка «Отчет об успеваемости».

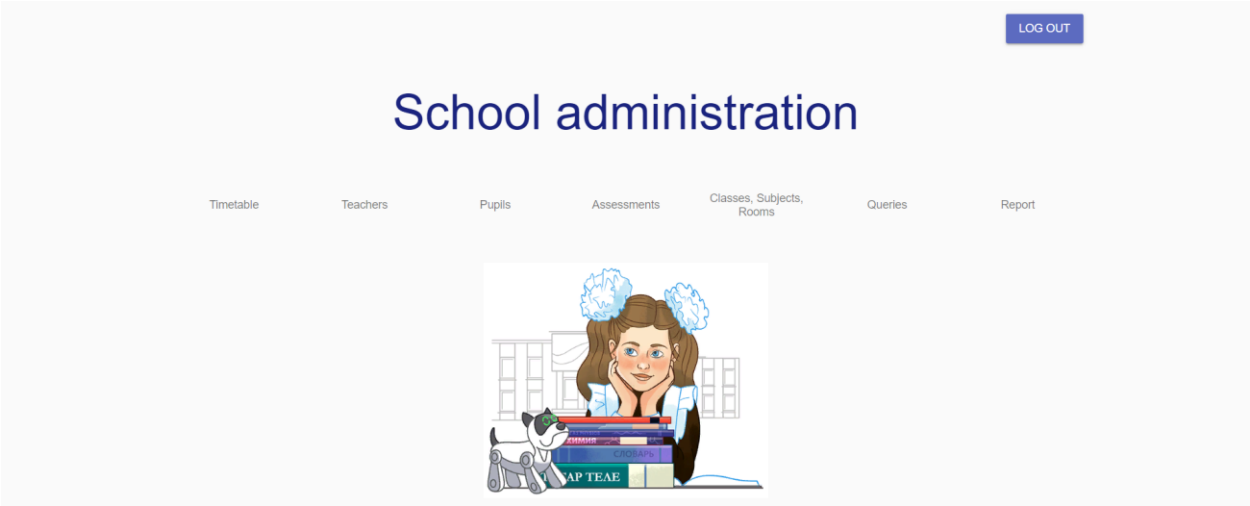
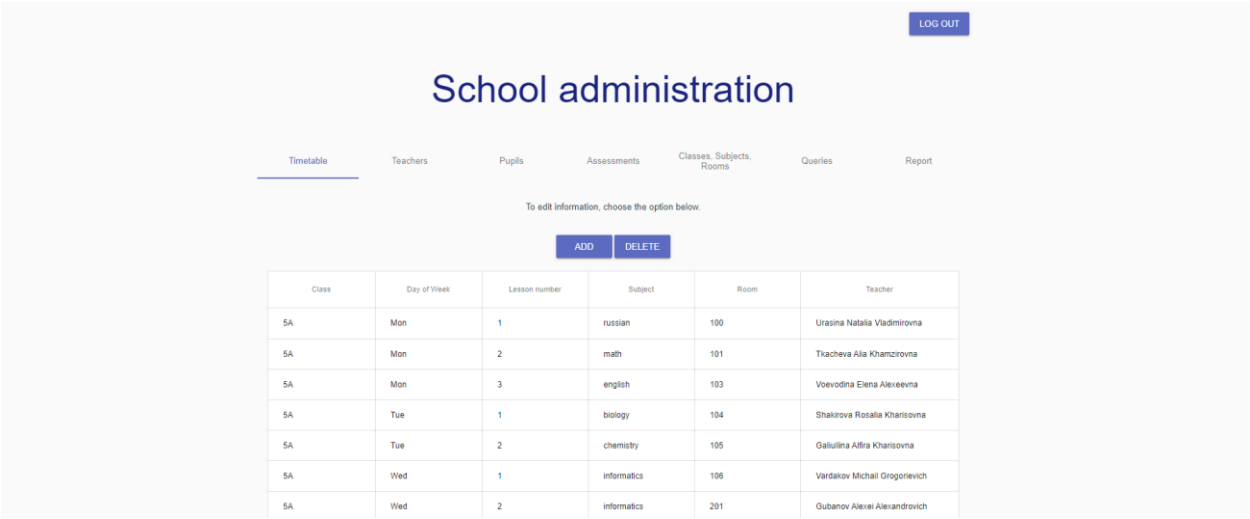


Рисунок 10 – Главная страница для авторизованного пользователя-учителя

Во вкладке «Расписание» можно в первую очередь посмотреть на текущее расписание для всех классов с указанием дня недели, номера урока, предмета, кабинета и учителя, а также перейти на страницы добавления и удаления записи в расписании. Добавление записи происходит путем ввода информации о каждом вышеописанном поле, а удаление происходит аналогично заполнение информации о строке в расписании, которую следует убрать. Оба процесса, а также сама страница Расписания представлены на рисунке 11. Следует отметить, что со страниц «Добавить» и «Удалить» (а также «Изменить», что будет введено позже в отчете) можно вернуться на главную страницу по кнопке HOME, она есть на всех подобных страницах.



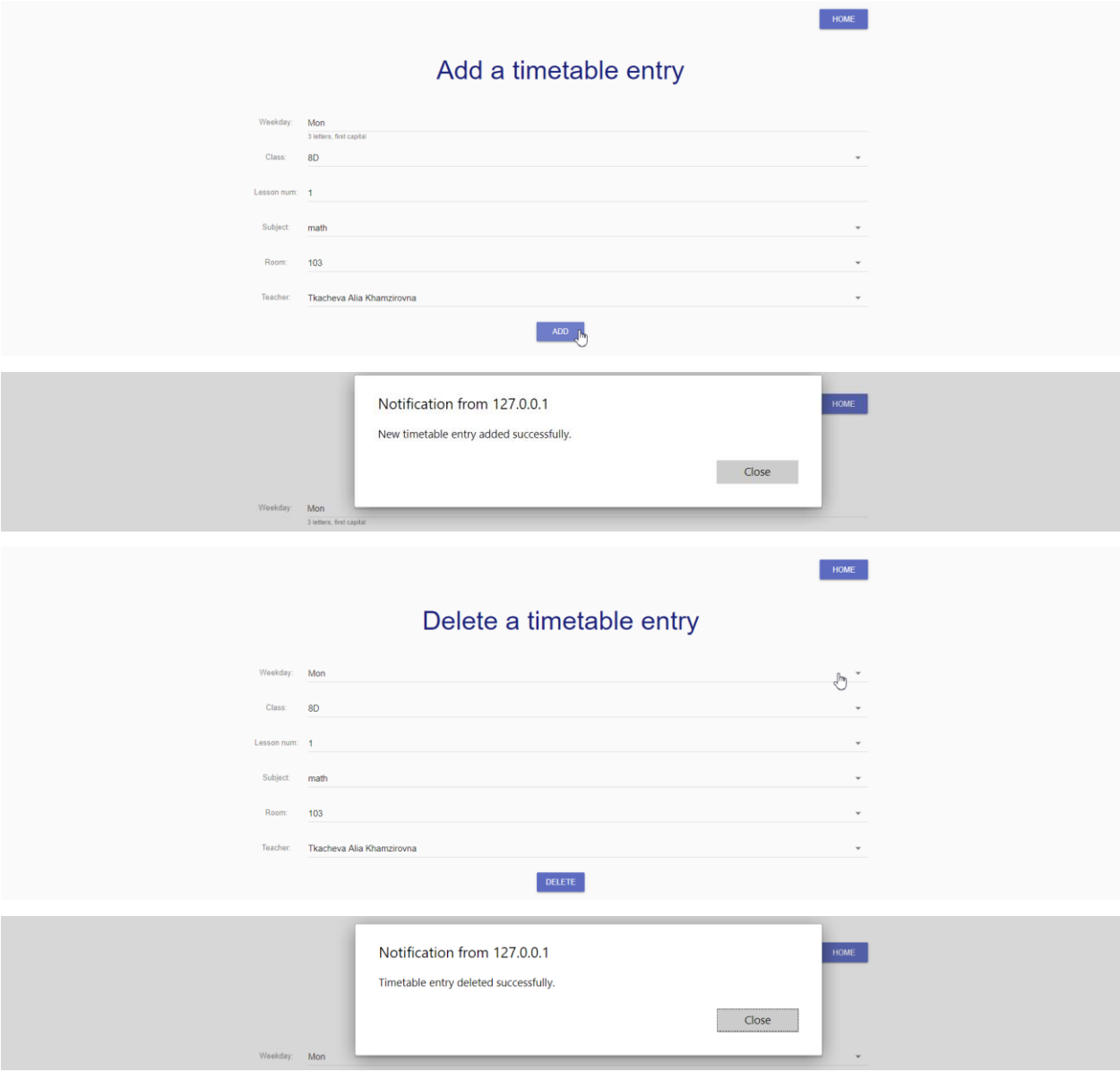
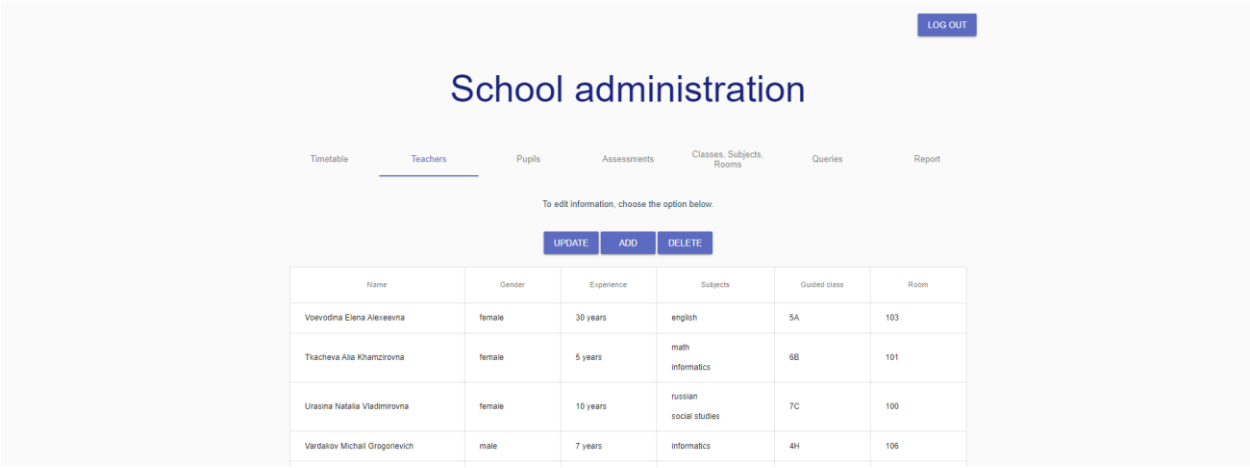
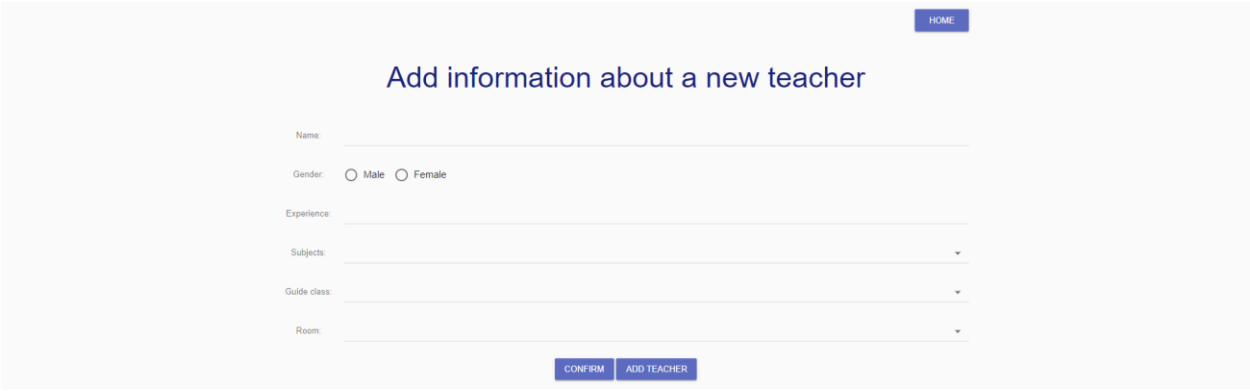
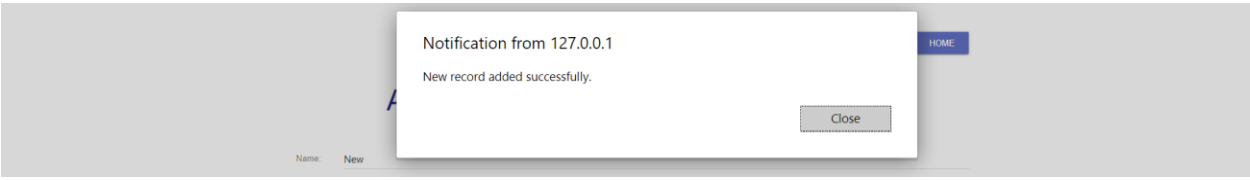


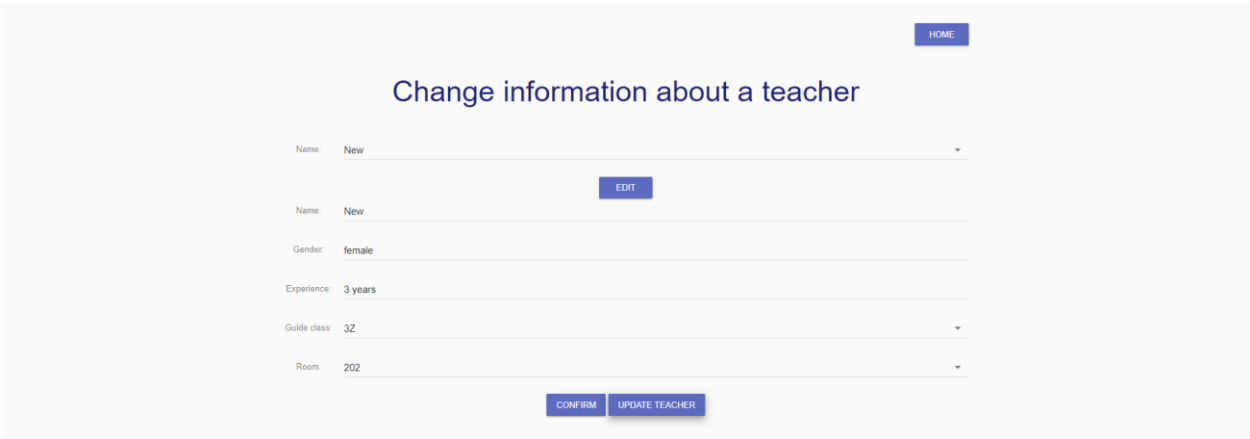
Рисунок 11 – Расписание

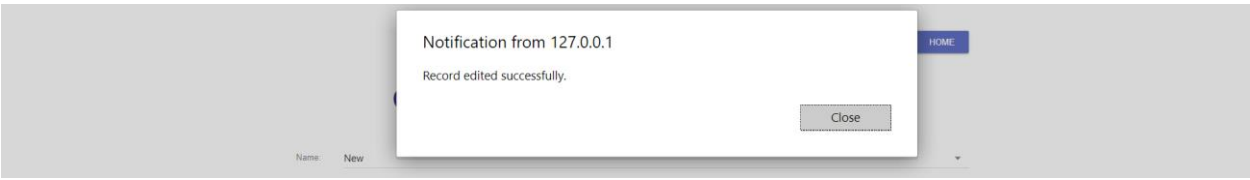
Во вкладке «Учителя» выводится список преподающих в данной школе учителей с указанием следующих сведений: имя, пол, опыт работы в годах, преподаваемые предметы, класс под классным руководством, закрепленный за ними кабинет. Здесь же можно выбрать одну из трех опций: добавить новую запись о нанятом учителе, изменить информацию о каком-либо преподавателе или удалить запись после увольнения. Добавление информации происходит путем ввода всей информации. Сперва мы подтверждаем ввод информации кнопкой CONFIRM. Три последних атрибута не содержатся в таблице Учитель, но учитель как внешний ключ есть в трех соответствующих таблицах. Поэтому сперва мы обновляем таблицу Учитель, а затем, используя новую созданную запись, обновляем три другие таблицы. При изменении нужно ввести имя учителя и внести необходимые правки, а удалить запись можно вводом только имени и нажатием кнопки. Все процессы показаны на рисунке 12.

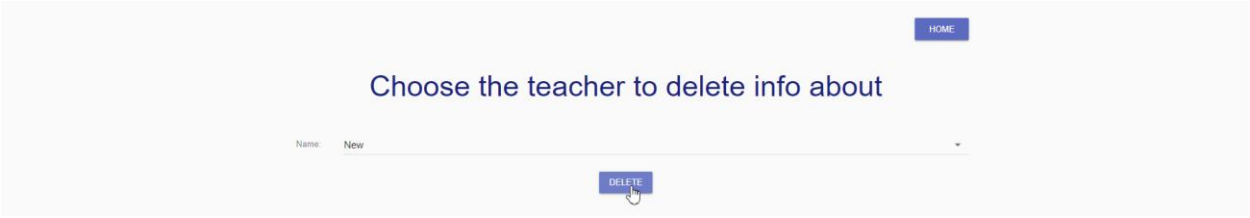

LOG OUT
School administration
Timetable
Teachers
Pupils
Assessments
Classes, Subjects, Rooms
Queries
Report
To edit information, choose the option below.
UPDATE
ADD
DELETE
Name
Gender
Experience
Subjects
Guided class
Room
Voevodina Elena Alexeevna
female
30 years
english
5A
103
Tkacheva Alia Khamzirovna
female
5 years
math
informatics
6B
101
Urasina Natalia Vladimirovna
female
10 years
russian
social studies
7C
100
Vardakov Michail Grogorievich
male
7 years
informatics
4H
106


HOME
Add information about a new teacher
Name:
Gender:
Male
Female
Experience:
Subjects:
Guide class:
Room:
CONFIRM
ADD TEACHER


Notification from 127.0.0.1
New record added successfully.
Close
HOME
Name:
New


HOME
Change information about a teacher
Name:
New
EDIT
Name:
New
Gender:
female
Experience:
3 years
Guide class:
3Z
Room:
202
CONFIRM
UPDATE TEACHER


Notification from 127.0.0.1
Record edited successfully.
Close
HOME
Name:
New



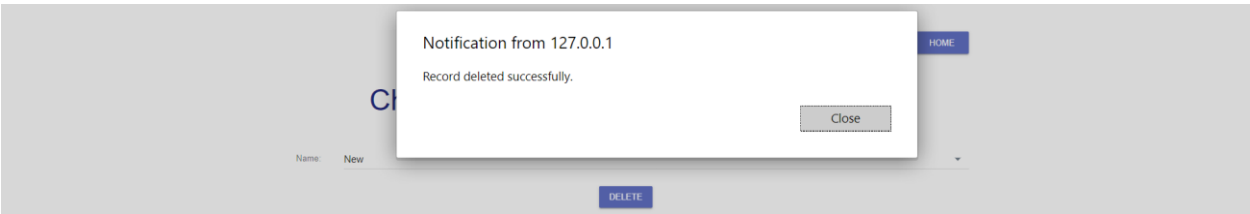
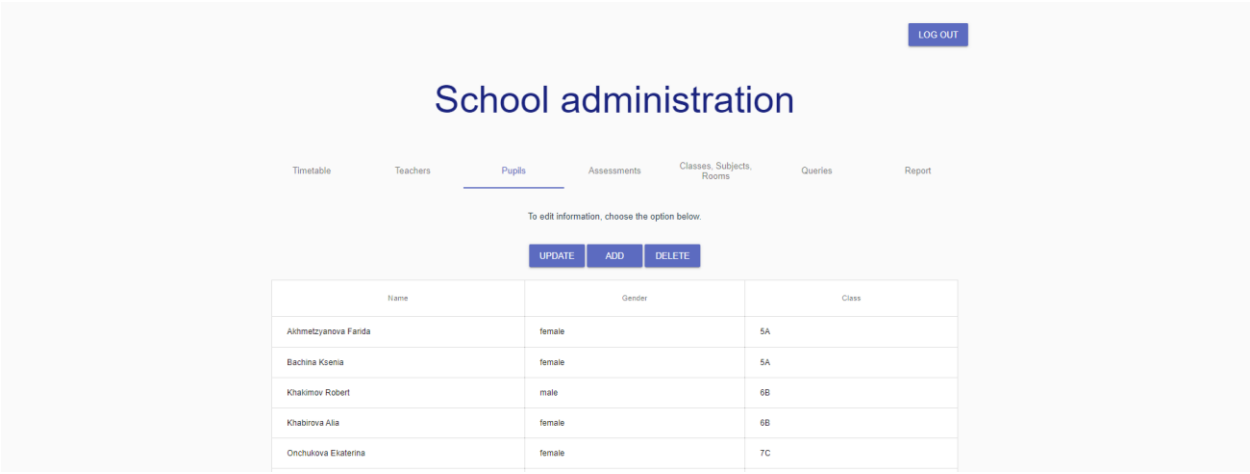
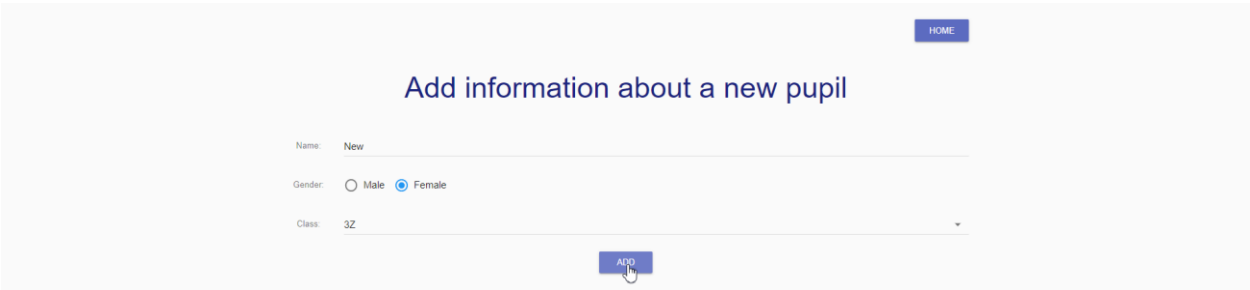
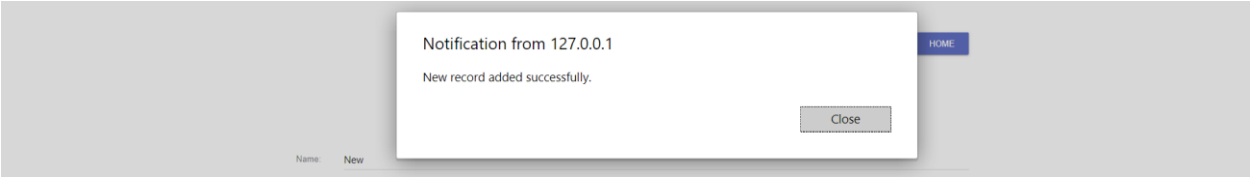


Рисунок 12 – Учителя

Вкладка «Ученики» содержит информацию об учениках, обучающихся в данной школе. Хранимая информация: имя, пол и класс. Здесь тоже есть процессы добавления, изменения и удаления записей. Все аналогично процессам с изменением состава учителей, но без дополнительных манипуляций с другими таблицами. Действия можно увидеть на рисунке 13.







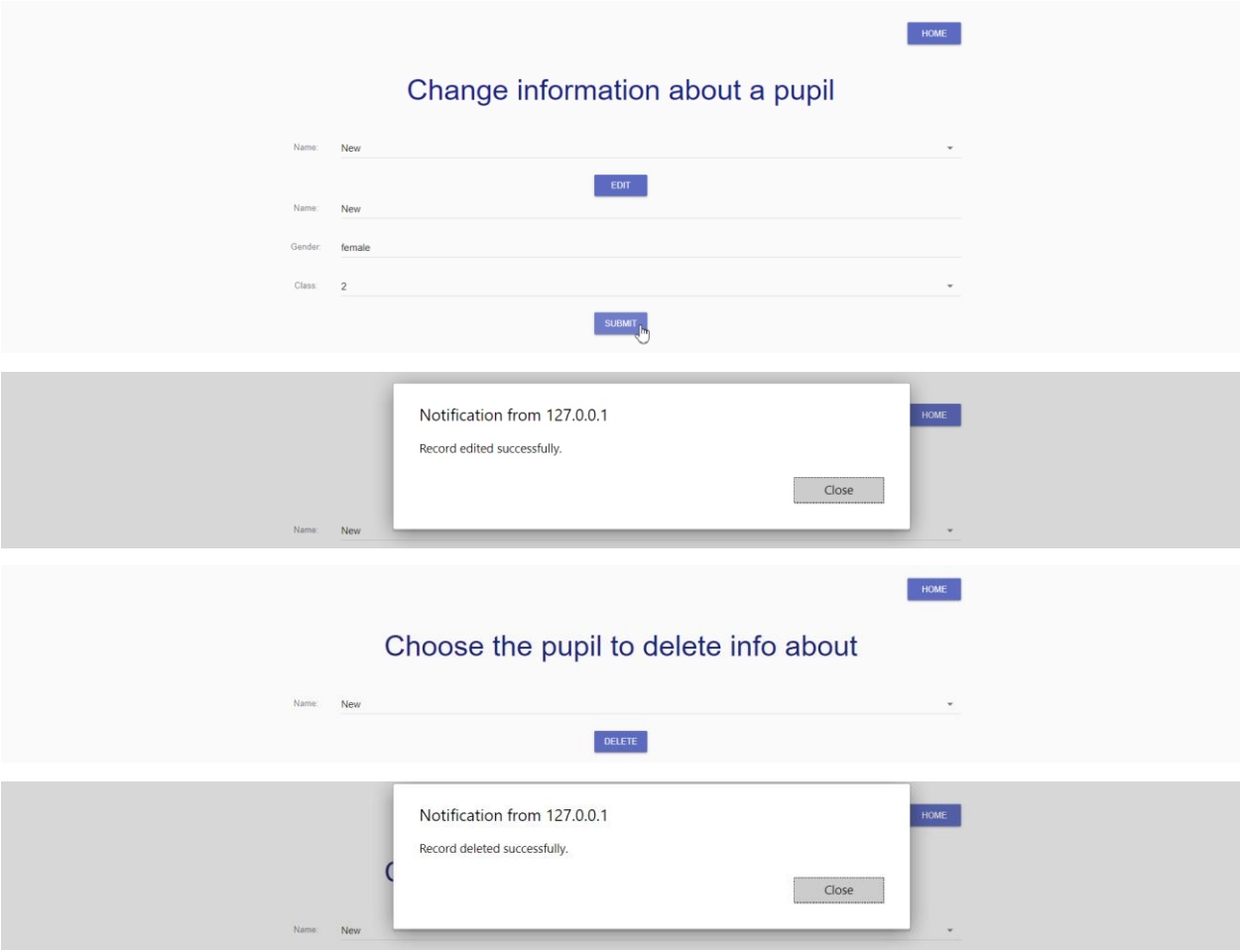
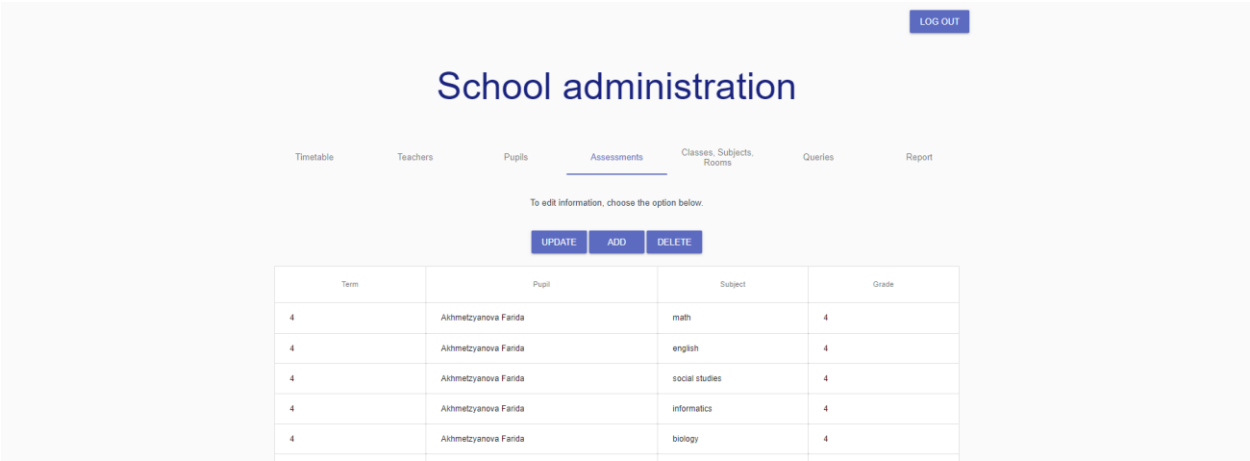


Рисунок 13 – Ученики

Во вкладке «Оценки» представлены все оценки учеников за каждую четверть по каждом предмету. Таблица состоит из полей четверть, имя ученика, предмета и самой отметки. Процессы изменения данных такие же, как и ранее, и представлены на рисунке 14.



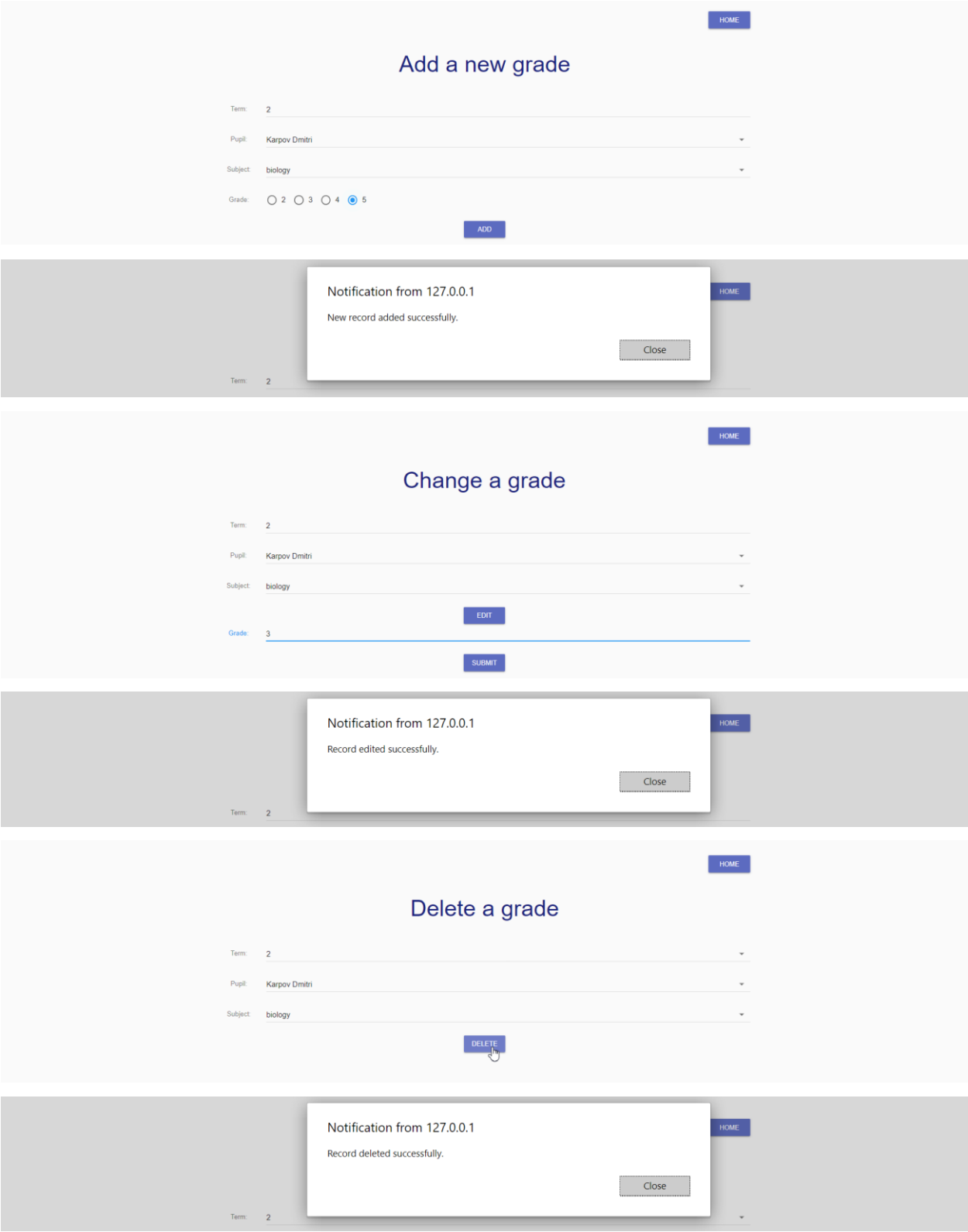


Рисунок 14 – Оценки

В общей вкладке «Классы, Предметы, Кабинеты» можно просто посмотреть, какие классы есть в школе, причем с указанием классного руководителя, какие дисциплины преподаются и их тип (базовый minor и профильный major), а также кабинеты, для которых указываются номер, этаж, предмет и закрепленный за ними учитель. Здесь не выполняются

дополнительные действия по добавлению, изменению или удалению записей – данные просто есть и представлены на рисунке 15.

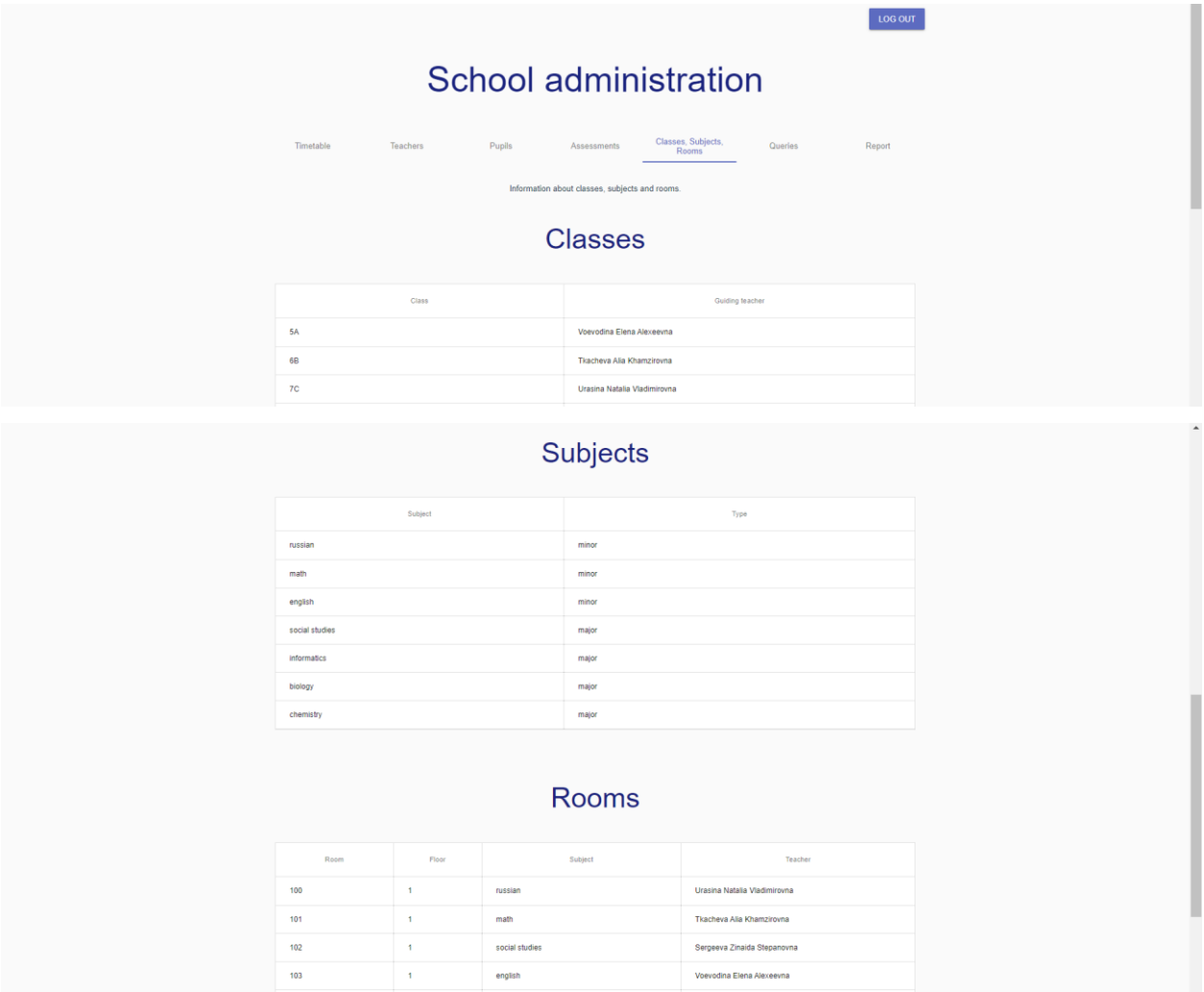


Рисунок 15 – Классы, Предметы и Кабинеты

Вкладка «Запросы» содержит 5 вопросов, ответы на которые могут понадобиться завучу. Все они есть на одной странице и открываются по нажатию. Рассмотрим подробнее эти запросы, представленные на рисунке 16.

«Какой предмет будет в заданном классе, в заданный день недели на заданном уроке?» Завуч может ввести класс обучения, день недели и номер урока и получить предмет, который соответствует введенным данным.

«Сколько учителей преподает каждую из дисциплин в школе?» Здесь информация в уже готовом виде представляется вниманию завуча в виде своего рода словаря формата «предмет: количество учителей».

«Список учителей, преподающих те же предметы, что и учитель, ведущий информатику в заданном классе.» Вводится искомый класс, в отчет выводятся все преподаватели, данные о которых соответствуют запросу.

«Сколько мальчиков и девочек в каждом классе?» Тоже готовый ответ формата словаря «класс – пол: количество учеников».

«Сколько кабинетов в школе для базовых и профильных дисциплин?» Готовый вывод двух строк: тип предмета и количество кабинетов, где они преподаются.

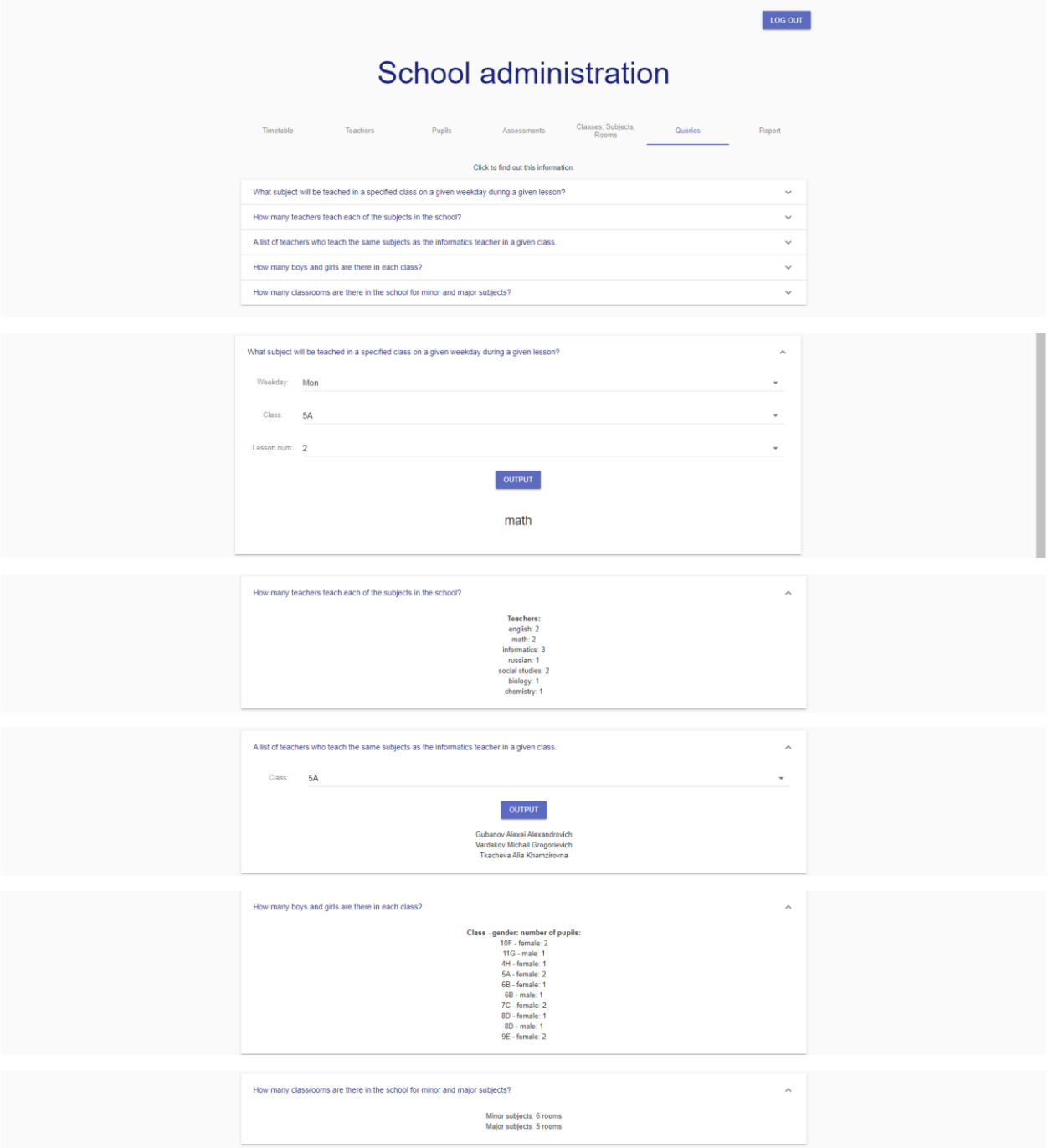


Рисунок 16 – Запросы

Наконец, во кладке «Отчет об успеваемости» предлагается выбрать класс, после ввода и нажатия кнопки выводится отчет. Он показывает среднюю оценку по каждому предмету в классе, среднюю оценку по классу вообще, количество учеников, классного руководителя и все оценки учеников класса в виде таблицы, аналогично вкладке «Оценки», но с фильтром по классу. Пример отчета представлен на рисунке 17.

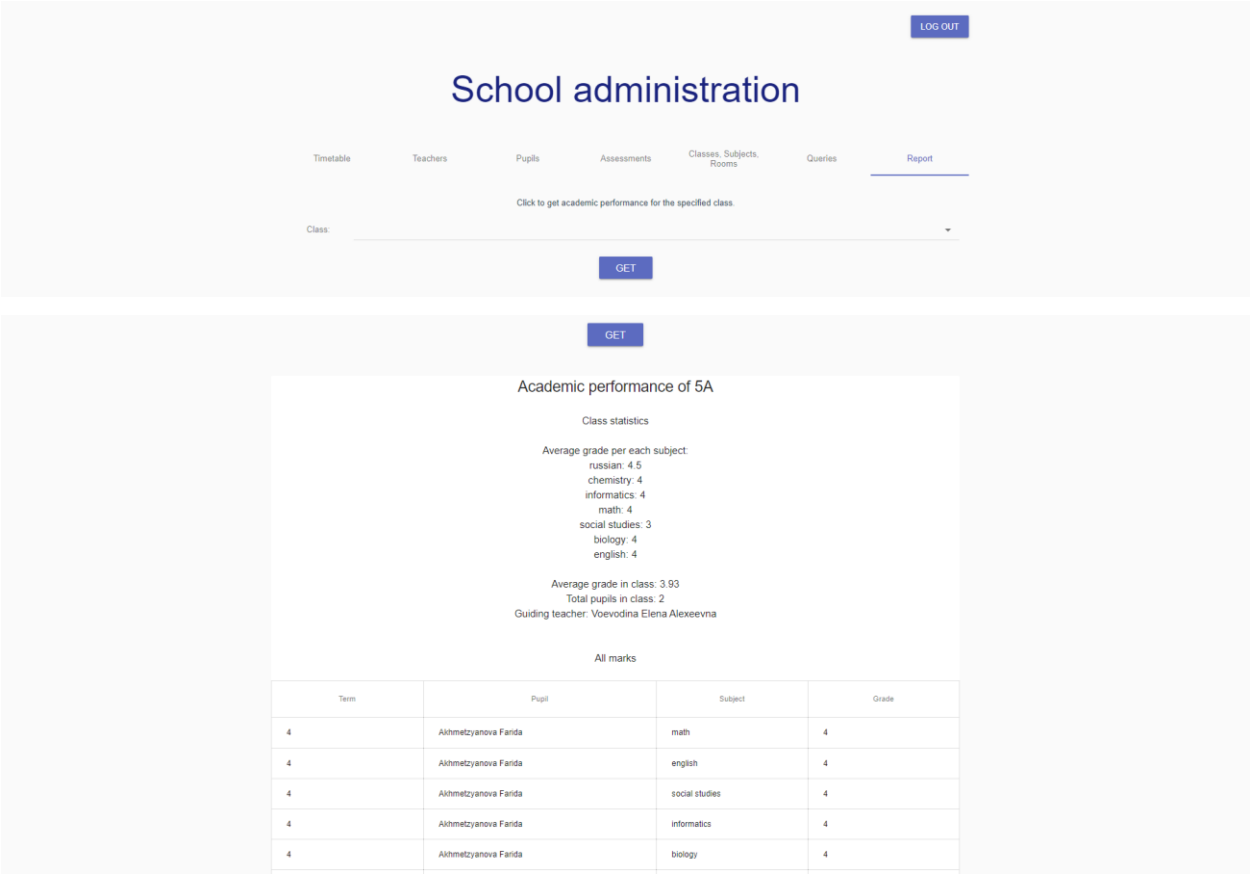


Рисунок 17 – Отчет

4. КОНТЕЙНЕРИЗАЦИЯ И ОРКЕСТРАЦИЯ

Docker – открытая платформа для разработки, доставки и эксплуатации приложений. Docker разработан для более быстрого выкладывания приложений. С его помощью можно отделить приложение от инфраструктуры и обращаться с ней как с управляемым приложением. В своем ядре Docker позволяет запускать практически любое приложение, безопасно изолированное в контейнере. Безопасная изоляция позволяет запускать на одном хосте много контейнеров одновременно.

Docker Compose – инструментальное средство, входящее в состав Docker. Оно предназначено для решения задач, связанных с развёртыванием проектов. [1]

При этом Docker применяется для управления отдельными контейнерами (сервисами), из которых состоит приложение, а Docker Compose используется для одновременного управления несколькими контейнерами, входящими в состав приложения. Этот инструмент предлагает те же возможности, что и Docker, но позволяет работать с более сложными приложениями.

Для обеспечения функционирования текущего проекта необходимо три сервиса: база данных, серверная часть и клиентская часть. Технология Docker Compose, если описывать её упрощённо, позволяет с помощью одной команды запускать множество сервисов.

Для использования Docker и Docker Compose были созданы отдельные папки для серверной (school-server) и клиентской части (school-vue). Файлы базы данных в данном случае не хранятся в локальной папке проекта, но подразумевается, что она есть, и контейнер для нее создается.

Для работы поверх этих локальных папок – на каталог выше – создается файл docker-compose.yml содержания, представленного на рисунке 18. В файле указываются названия сервисов, которые следует поместить в контейнеры, и прочая информация, которая необходима Docker, например, административные данные базы данных, папки с необходимыми файлами, команды для запуска сервисов.

```yaml
version: '3'

services:
        school_db:
                image: postgres
                ports:
                        - "5432:5432"
                environment:
                        - POSTGRES_DB=school
                        - POSTGRES_USER=postgres
                        - POSTGRES_PASSWORD=postgres

        backend:
                container_name: school_backend
                build:
                        context: ./school-server
                        dockerfile: Dockerfile

                command: bash -c "sleep 3 &&
                        python3 manage.py makemigrations && python3 manage.py migrate &&
                        python3 manage.py runserver 0.0.0.0:8000";
                volumes:
                        - ./school-server:/school-server
                ports:
                        - "8000:8000"
                depends_on:
                        - school_db

        frontend:
                container_name: school_frontend
                build:
                        context: ./school-vue
                        dockerfile: Dockerfile

                command: npm run dev
                volumes:
                        - ./school-vue:/school-vue
                ports:
                        - "8080:8080"
                depends_on:
                        - backend
```

Рисунок 18 – Содержимое файла docker-compose.yml

При этом в папках серверной и клиентской частей нужно создать файлы Dockerfile. Это файл Docker, который содержит инструкции, необходимые для создания окружения сервера. Примеры для обеих частей приложения представлены на рисунках 19 и 20.

Файл серверной части включает в себя версию Python, создание и обозначение рабочей директории в соответствующем контейнере, добавление всех файлов в созданную папку и установку используемых в бэкенде библиотек, список которых указан в файле requirements.txt.

```
FROM python:3.8.2

RUN mkdir /school-server

WORKDIR /school-server

ADD . /school-server

RUN pip install -r requirements.txt
```

Рисунок 19 – Dockerfile серверной части

Файл клиентской части содержит версию Node.js, обозначение рабочей директории в соответствующем контейнере, установку требуемых библиотек из файла package.json и копирование всех файлов из локальной директории в папку контейнера.

```
FROM node:12.16.3

WORKDIR /school-vue

COPY package*.json ./
CMD npm install
CMD npm start

COPY . .
```

Рисунок 20 – Dockerfile клиентской части

Также стоит отметить, что для работы сервисов в контейнерах нужно учесть следующие моменты:

1. Имя хоста базы данных, указанное в файле settings.py серверной части, должно совпадать с названием создаваемого контейнера.

2. Хост для работы клиентской части следует обозначить как «0.0.0.0» в конфигурационном файле index.js.

После того, как в docker-compose.yml и другие файлы внесены все необходимые инструкции, проект нужно собрать командой *docker-compose build*.

Когда проект собран, пришло время его запустить командой *docker-compose up*, работа которой представлена на рисунке 21.

```
(web-app-env) C:\Users\811840\Desktop\web\docker-tutorial-master\students\K3342\practical_works\Shaidullina_Regina\docker>docker-compose up
Starting docker_db_1 ... done
Recreating backend_school ... done
Recreating frontend_school ... done
Attaching to docker_db_1, backend_school, frontend_school
db_1          |
db_1          | PostgreSQL Database directory appears to contain a database; Skipping initialization
db_1          |
db_1          | 2020-06-25 07:11:10.550 UTC [1] LOG:  starting PostgreSQL 12.3 (Debian 12.3-1.pgdg100+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 8.3.0-6) 8.3.0, 64-bit
db_1          | 2020-06-25 07:11:10.552 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432
db_1          | 2020-06-25 07:11:10.552 UTC [1] LOG:  listening on IPv6 address "::", port 5432
db_1          | 2020-06-25 07:11:10.561 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
db_1          | 2020-06-25 07:11:10.609 UTC [25] LOG:  database system was shut down at 2020-06-24 18:16:26 UTC
db_1          | 2020-06-25 07:11:10.639 UTC [1] LOG:  database system is ready to accept connections
frontend_school |
frontend_school | > school-admin-vue@1.0.0 dev /school-vue
frontend_school | > webpack-dev-server --inline --progress --config build/webpack.dev.conf.js
frontend_school |
backend_school | No changes detected
backend_school | Operations to perform:
backend_school |   Apply all migrations: admin, auth, authtoken, contenttypes, django_summernote, school, sessions
backend_school | Running migrations:
backend_school |   No migrations to apply.
backend_school | [25/Jun/2020 07:13:08] "GET /school/rooms/?subject=english HTTP/1.1" 401 5283
```

Рисунок 21 – Запуск проекта через Docker Compose

После запуска таким образом проекта все сервисы одновременно запускаются в Docker на указанных портах.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы были получены навыки реализации Web-сервисов с выполнением всех этапов разработки: была выбрана и проанализирована предметная область, выявлены функциональные требования, которым должен соответствовать веб-сервис, спроектирована и реализована модель базы данных, реализованы серверная и клиентская стороны. Также была добавлена возможность запускать все сервисы одновременно через Docker и Docker Compose.

В ходе выполнения курсовой работы были применены навыки и умения, полученные во время выполнения практических и лабораторных работ по курсу «Основы Web-программирования». В частности, были изучены и опробованы на практике технологии Django REST framework, Vue.js, Muse-UI, а также использована с ними PostgreSQL. Результатом практической деятельности в ходе курса и выполнения курсовой работы является спроектированная и реализованная программная система для администрирования образовательного процесса завучем школы.

СПИСОК ЛИТЕРАТУРЫ

1. ru_vds (2019, 2 мая), Хабр, Руководство по Docker Compose для начинающих. [ЭИ] URL: https://habr.com/ru/company/ruvds/blog/450312/ (дата обращения: 26.06.2020).

2. Официальный сайт Django REST Framework. [ЭИ] URL: https://www.django-rest-framework.org (дата обращения: 10.06.2020).

3. Документация Vue.js. [ЭИ] URL: https://vuejs.org/v2/guide/ (дата обращения: 15.06.2020).