Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
“УНИВЕРСИТЕТ ИТМО”

Факультет ИКТ

Образовательная программа *09.03.02 Информационные системы и технологии*

Направление подготовки (специальность) *09.03.02 Информационные системы и технологии*

О Т Ч Е Т

по курсовой работе

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

Обучающийся Пахмурин Максим Андреевич, гр. К3340

Руководитель: Говоров А. И., ассистент факультета ИКТ Университета ИТМО

Оценка за курсовую работу ____

Подписи членов комиссии:

_____ ( )
(подпись)

_____ ( )
(подпись)

_____ ( )
(подпись)

Дата ____

Санкт-Петербург
20 20

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 3

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ 3

1.1. Описание предметной области 4
1.2. Выводы 4

2. ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ СЕРВИСА 5

2.1. Описание архитектуры сервиса 5
2.2. Модель данных 5
2.3. Выводы 6

3. РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ 7

3.1. Описание средств разработки серверной части 7
3.2. Реализация базы данных 7
3.3. Сериализация 8
3.4. Создание представлений 8
3.5. Настройка маршрутизации 8
3.6. Выводы 9

4. РАЗРАБОТКА КЛИЕНТСТКОЙ ЧАСТИ 10

4.1. Описание средств разработки клиентской части 10
4.2. Разработанные интерфейсы 10
4.2.1. Вход в систему 10
4.2.2. Интерфейсы системы 12
4.3. Выводы 16

ЗАКЛЮЧЕНИЕ 17

СПИСОК ЛИТЕРАТУРЫ 18

ВВЕДЕНИЕ

Интернет является важной составляющей современного общества. Невозможно переоценить влияние всемирной паутины на жизнь человека в наши дни. Поэтому специалист в области информационных технологий должен не только знать теоретические основы организации сети Интернет, но и иметь практические навыки реализации веб-сервисов.

Целью данной курсовой работы является разработка веб-сервиса согласно выбранному варианту. В рамках работы нужно было изучить и научиться применять средства создания веб-сервисов, которые используют в современных системах.

В ходе работы должны быть выполнены следующие задачи:

1. Изучение предметной области.
2. Анализ функциональных требований.
3. Проектирование архитектуры веб-сервиса.
4. Разработка серверной части системы.
5. Разработка клиентской части системы.
6. Контейнеризация проекта.

В первой главе проведен анализ предметной области и функциональных требований. Вторая глава посвящена проектированию архитектуры веб-сервиса. В третьей и четвертой главах рассмотрен подход к разработке серверной и клиентской частей системы соответственно.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

1.1. Описание предметной области

В качестве варианта был выбран вариант 12 – создание программной системы для администратора колледжа. Предметной областью является колледж. Основными пользователями являются администратор и диспетчер.

Информационная система для данной области представляет собой сервис, в котором реализованы стандартные процедуры для данной области: выставление оценок и составление расписания. Такая система должна обеспечивать хранение сведений о студентах, группах и преподавателях. О каждом студенте имеется следующая информация: группа, фамилия, имя, отчество. О преподавателях имеется следующая информация: фамилия, имя, отчество, номер закрепленного кабинета.

1.2. Выводы

Проведен анализ предметной области и функциональных требований системы. В результате было улучшено понимание данной области и того, как должна работать разрабатываемая система.

2. ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ СЕРВИСА

2.1. Описание архитектуры сервиса

Для веб-сервиса, разрабатываемого в рамках курсовой работы, была выбрана архитектура «клиент-сервер», представленная на рис. 1. В данной архитектуре сетевая нагрузка распределена между поставщиками услуг, серверами, и заказчиками услуг, клиентами.

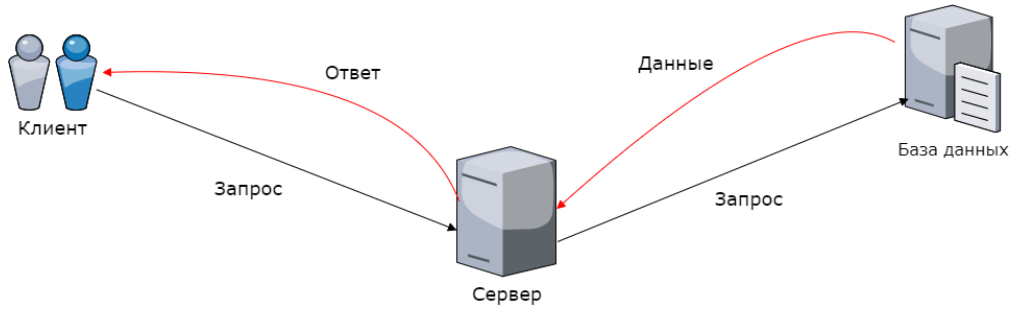

Рисунок 1 – Архитектура «Клиент-Сервер»

Сервером в данном случае считается абстрактная машина в сети, которая способна получить HTTP-запрос, обработать его и вернуть корректный ответ. Клиентом может считаться все, что способно сформировать и отправить HTTP-запрос. Сервер ожидает от клиента запрос и предоставляет свои ресурсы в виде данных или в виде сервисных функций. [6]

База данных представляет собой третье звено архитектуры. Она нужна для того, чтобы информация могла сохраняться даже при падении и рестарте системы. Наличие базы данных гарантирует облегченный поиск по данным и их сохранность.

Преимуществом использования данной архитектуры является отсутствие дублирования кода, так как сервер и база данных вынесены отдельно и, следовательно, нет необходимости в хранении одинакового кода по обработке логики системы на клиентских машинах. Еще одним преимуществом клиент-серверной архитектуры является повышенная безопасность системы, потому что клиент может видеть только доступную ему информацию.

2.2. Модель данных

В соответствии с вариантом задания и функциональными требованиями была создана модель данных, представленная на рис. 3.

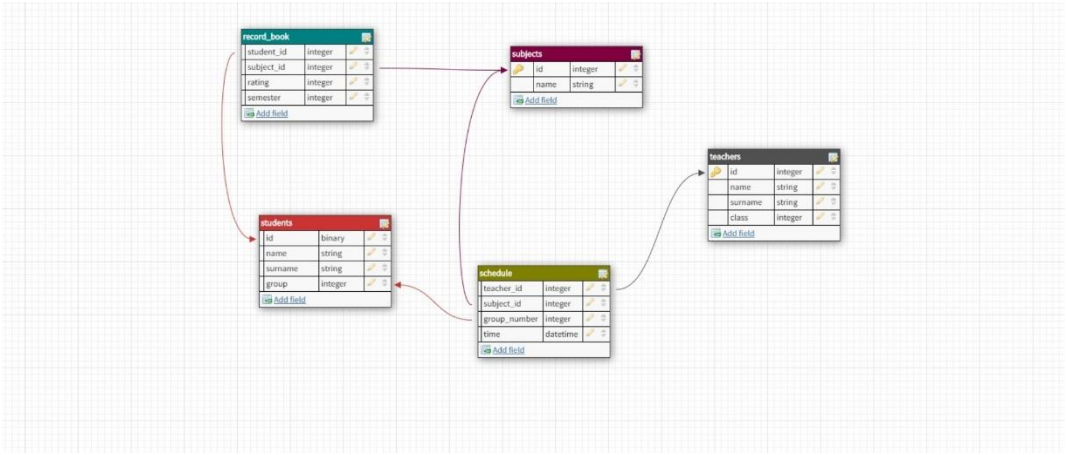

Рисунок 2 – Модель данных

Модель содержит 5 сущностей:

– Журнал с оценками.
– Предметы.
– Преподаватели.
– Студенты.
– Расписание.

Сущности соединены между собой связями Один-ко-многим и Многие-ко-многим.

2.3. Выводы

В ходе проектирования архитектуры сервиса был выбран тип архитектуры «Клиент-сервер». На основе функциональных требований к системе была создана архитектура веб-приложения и модель данных.

3. РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ

3.1. Описание средств разработки серверной части

Для реализации серверной части был использован фреймворк Django Rest, который является удобным инструментом, основанным на идеологии Django, для работы с rest.

Django – это высокоуровневая веб-инфраструктура языка Python, которая позволяет быстро создавать безопасные и поддерживаемые веб-сайты.

REST (сокр. англ. Representational State Transfer, «передача состояния представления») — стиль построения архитектуры распределенного приложения. Данные в REST должны передаваться в виде небольшого количества стандартных форматов (например HTML, XML, JSON). Сетевой протокол, как и HTTP, должен поддерживать кэширование, не должен зависеть от сетевого слоя, не должен сохранять информацию о состоянии между парами «запрос-ответ». Утверждается, что такой подход обеспечивает масштабируемость системы и позволяет ей эволюционировать с новыми требованиями. Кроме того, преимуществами использования REST являются надежность, производительность, прозрачность системы взаимодействия, простота интерфейсов и способность приспосабливаться к новым требованиям. [1]

3.2. Реализация базы данных

Представленная на рис. 3 модель данных была реализована с помощью СУБД PostgreSQL. PostgreSQL – это свободная объектно-реляционная система управления базами данных (СУБД). Преимуществами данной СУБД являются высокопроизводительные и надежные механизмы транзакций, расширенная система встроенных языков программирования, наследование и расширяемость [2].

Разработанная по представленной модели база данных содержит следующие таблицы:

– Client (клиент) – таблица содержит информацию о клиентах гостиницы.
– Room (номер) – таблица содержит информацию о номерах в гостинице.
– Checkin (заселение) – таблица содержит информацию о заселениях клиентов в номера гостиницы.
– Floor (этаж) – таблица содержит информацию об этажах в гостинице.
– Worker (работник) – таблица содержит информацию о служащих работниках гостиницы.
– Cleaning (уборка) – таблица содержит расписание уборок служащего по дням недели.
– Otchet (отчет об уборке) – таблица содержит информацию об уборках, проведенных работниками.

3.3. Сериализация

В программировании сериализация представляет собой процесс перевода какой-либо структуры данных в последовательность битов. Другими словами, это процесс создания потокового представления данных, которые можно передавать по сети. Обратным процессов является десериализация.

3.4. Создание представлений

Представление (view) – это функция обработчик запросов, которая получает HTTP-запросы и возвращает ответы. View имеет доступ к данным через модели, которые определяют структуру данных приложения и предоставляют механизмы для управления базой данных.

3.5. Настройка маршрутизации

Когда разработаны представления, нужно создать URL-адреса для того, чтобы система начала работать. В системе имеется список основных адресов, который включает адреса приложения и адреса для авторизации. URL-адреса приложения основаны на разработанных ранее представлениях. На рис. 6 представлен список URL-адресов веб-приложения.

```
urlpatterns = [
    path('', include(router.urls)),
    path('students/', views.StudentsListCreateView.as_view()),
    path('shedule/', views.SheduleListCreateView.as_view()),
    path('shedule/delete/<int:pk>', views.SheduleDeleteView.as_view()),
    path('teachers/delete/<int:pk>', views.TeachersDeleteView.as_view()),
    path('record/', views.RecordListCreateView.as_view()),
    path('subjects/', views.SubjectsListView.as_view()),
    path('teachers/update/<int:pk>', views.TeachersUpdateView.as_view()),
    path('students/update/<int:pk>', views.StudentsUpdateView.as_view()),
    path('students/delete/<int:pk>', views.StudentsDeleteView.as_view()),


]
```

Рисунок 3 – Список url-адресов приложения

3.6. Выводы

Средствами фреймворка Django REST был разработан бэкенд системы для управления гостиницей. Были созданы и описаны сериализаторы, представления и url-адреса веб-приложения.

4. РАЗРАБОТКА КЛИЕНТСТКОЙ ЧАСТИ

4.1. Описание средств разработки клиентской части

Для разработки клиентской части системы был использован фреймворк Vue.js и библиотека Muse UI.

Vue.js — это прогрессивный фреймворк для создания пользовательских интерфейсов. В отличие от фреймворков-монолитов Vue создан пригодным для постепенного внедрения. Его ядро в первую очередь решает задачи уровня представления (view), что упрощает интеграцию с другими библиотеками и существующими проектами. С другой стороны, Vue полностью подходит и для создания сложных одностраничных приложений (SPA, Single-Page Applications), если использовать его совместно с современными инструментами и дополнительными библиотеками. [3]

Библиотека Muse UI представляет собой набор компонентов для Vue, которые используют Material Design [4]. Это фреймворк для быстрого создания и запуска пользовательского интерфейса с приятным и удобным дизайном.

4.2. Разработанные интерфейсы

4.2.1. Вход в систему

При запуске системы открывается форма входа (рис.7). На ней пользователю предлагается войти в систему либо зарегистрироваться. При этом войти в систему можно как администратор или как диспетчер. Разные страницы входа направляют на разные сценарии работы с системой.

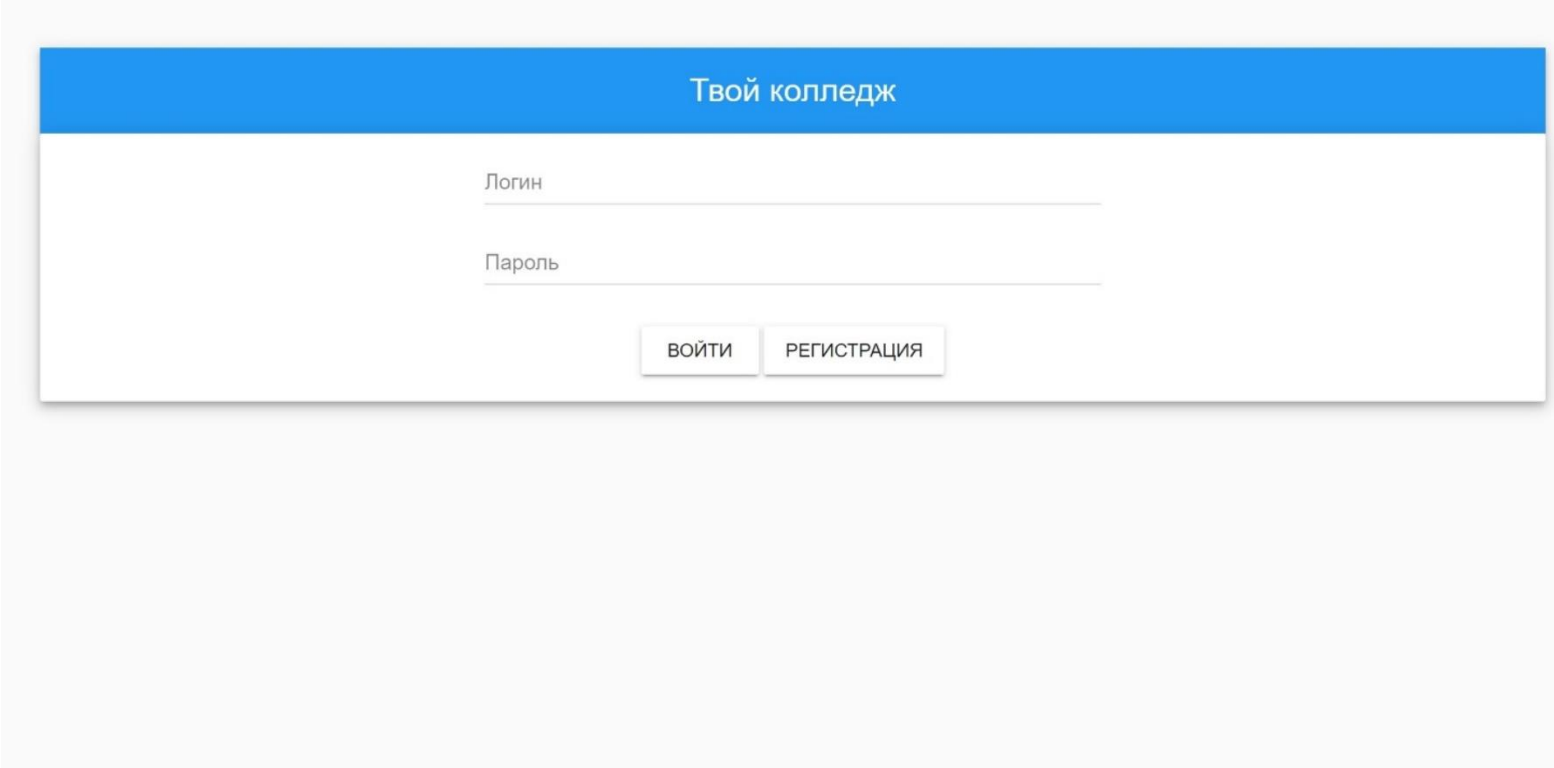

Рисунок 4 – Стартовая страница

Если пользователь еще не имеет своего аккаунта в системе, ему необходимо зарегистрироваться, нажав на кнопку регистрация.

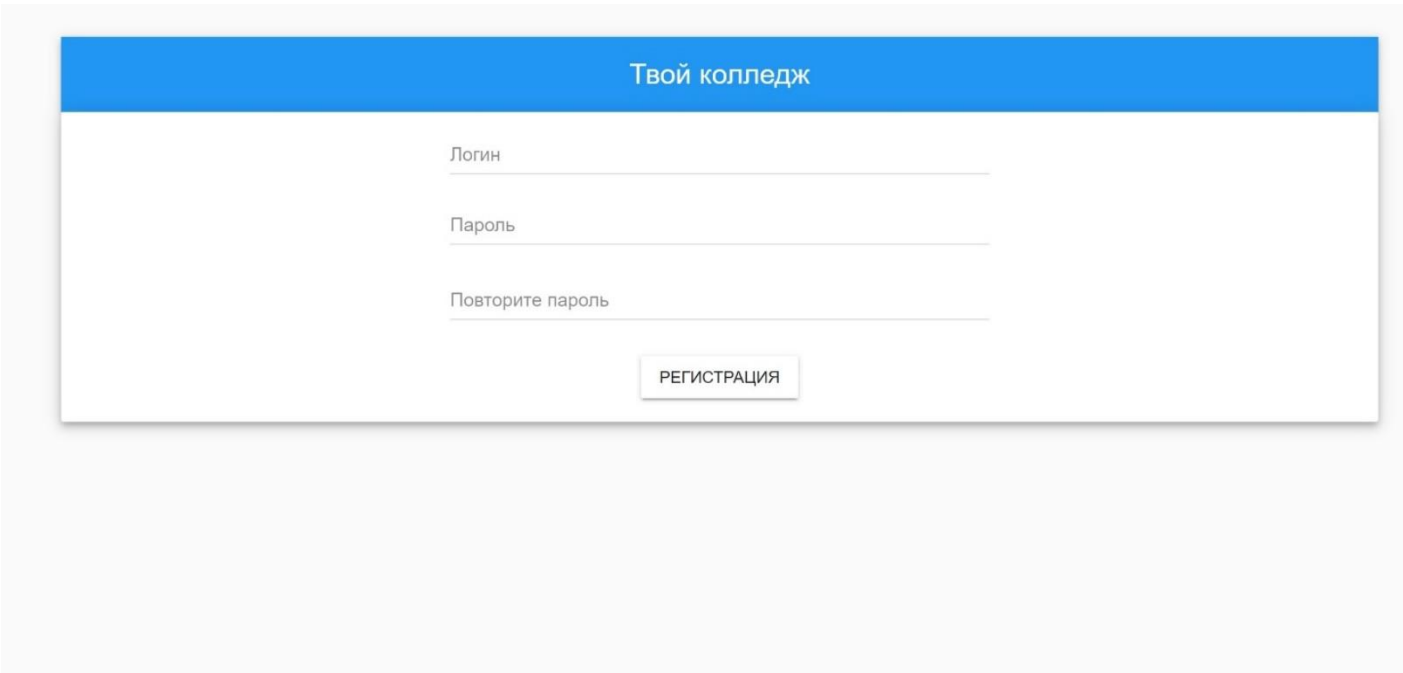

Рисунок 5 – Форма регистрации

4.2.2. Интерфейсы

Войдя в систему, пользователь попадает на страницу с расписанием (рис. 6).

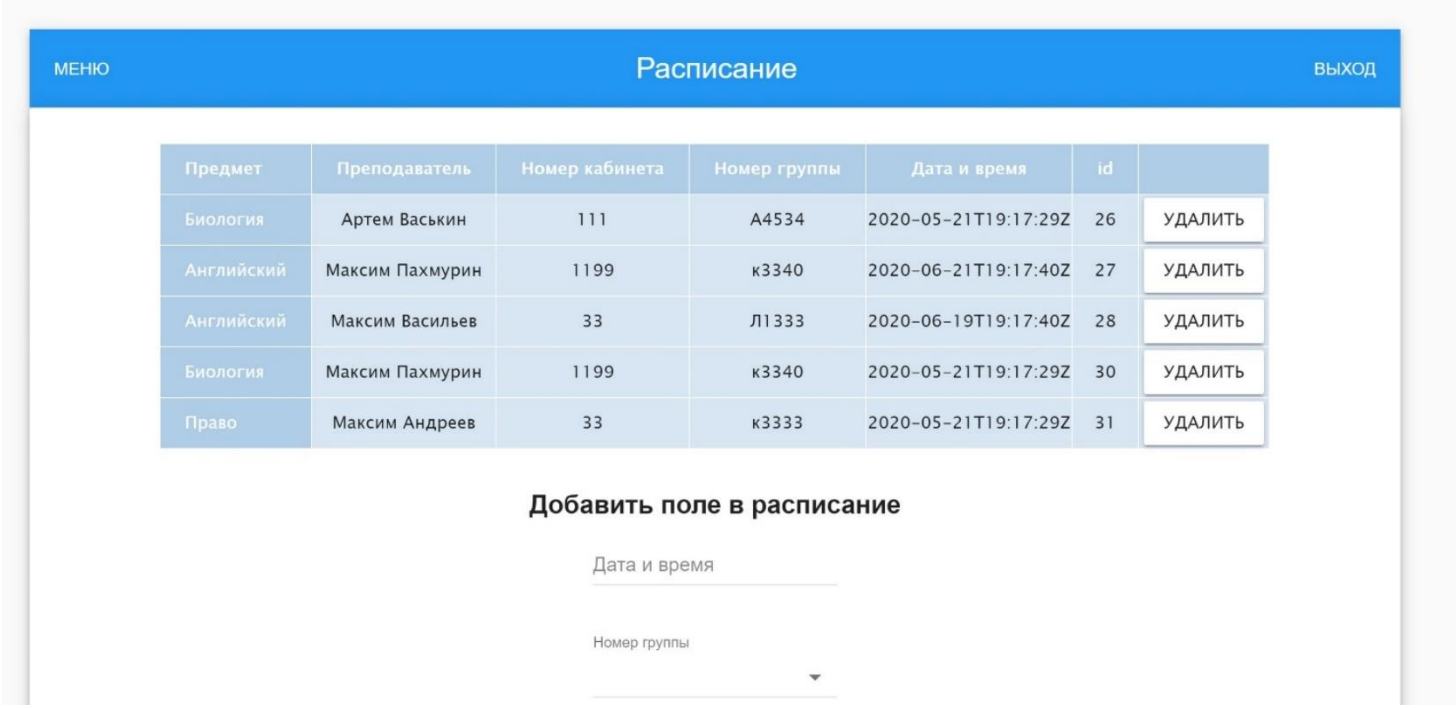

Рисунок 6 – Страница с расписанием

На данной странице выведен список всего составленного расписания. Ниже есть интерфейс добавления нового поля в расписание (рис. 7).

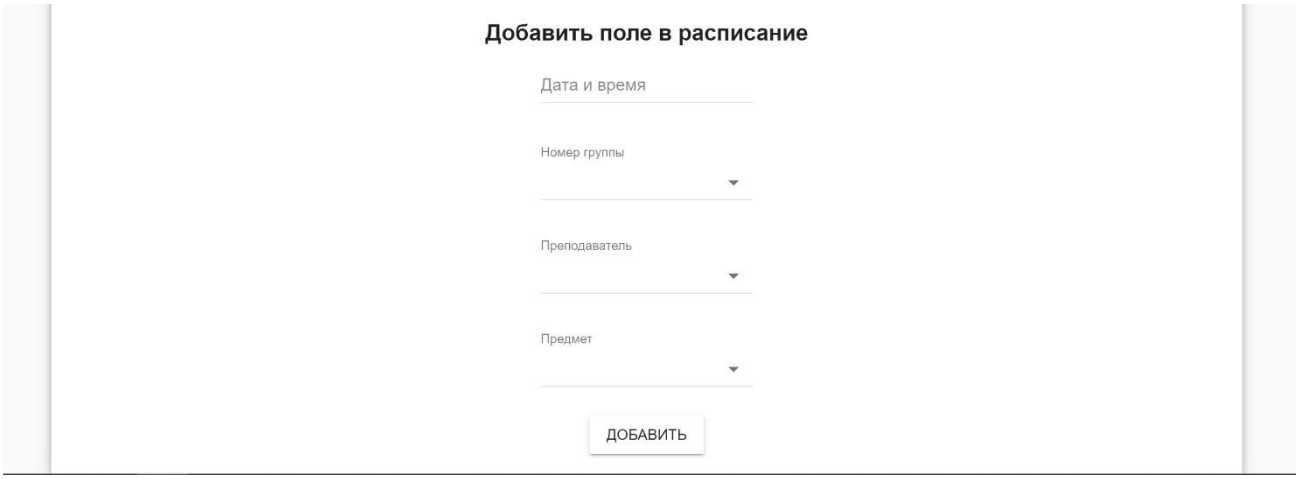

Рисунок 7 – Интерфейс добавления поля в расписание

На этой же странице имеются кнопки удаления полей расписания. На верхнем поле расположена кнопка выход, которая обеспечивает выход пользователя из системы и возвращение на страницу авторизации.

Также имеется интерфейс со списком преподавателей (рис. 8)

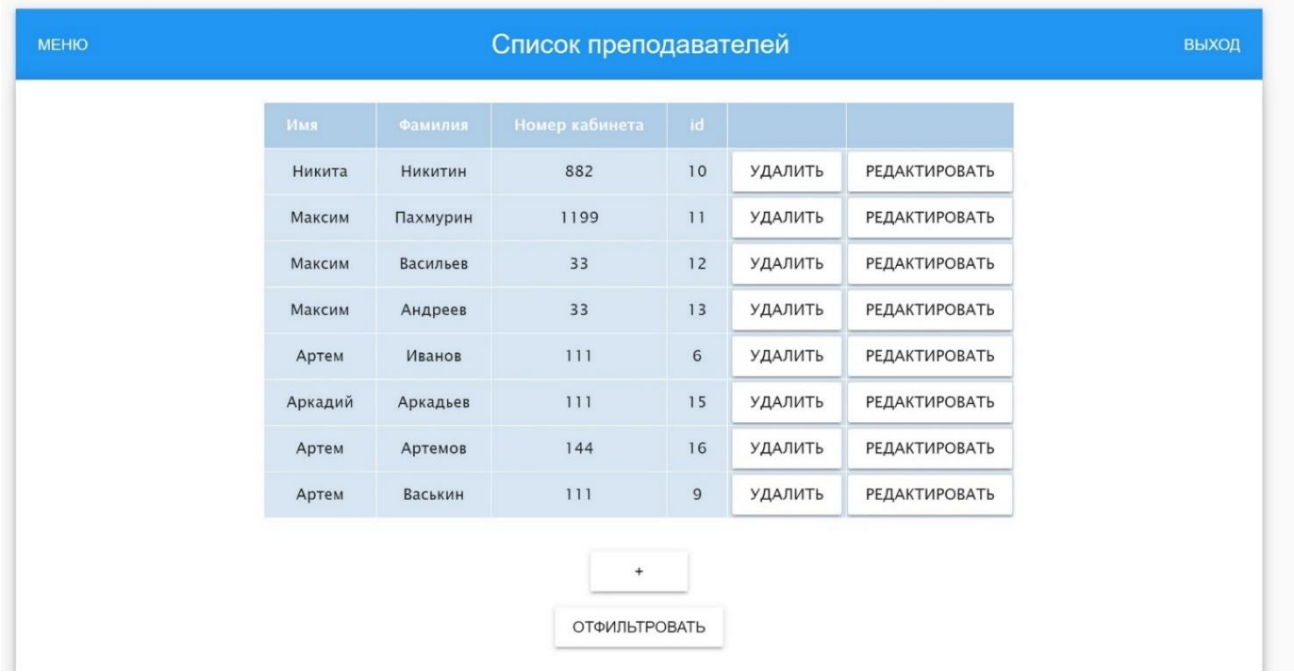

Рисунок 8 – Интерфейс список преподавателей

Также на странице просмотра преподавателей возможно добавление фильтров, есть возможность удаления и редактирования преподавателей. После нажатия на кнопку «редактировать», открывается окно редактирования (рис. 9)

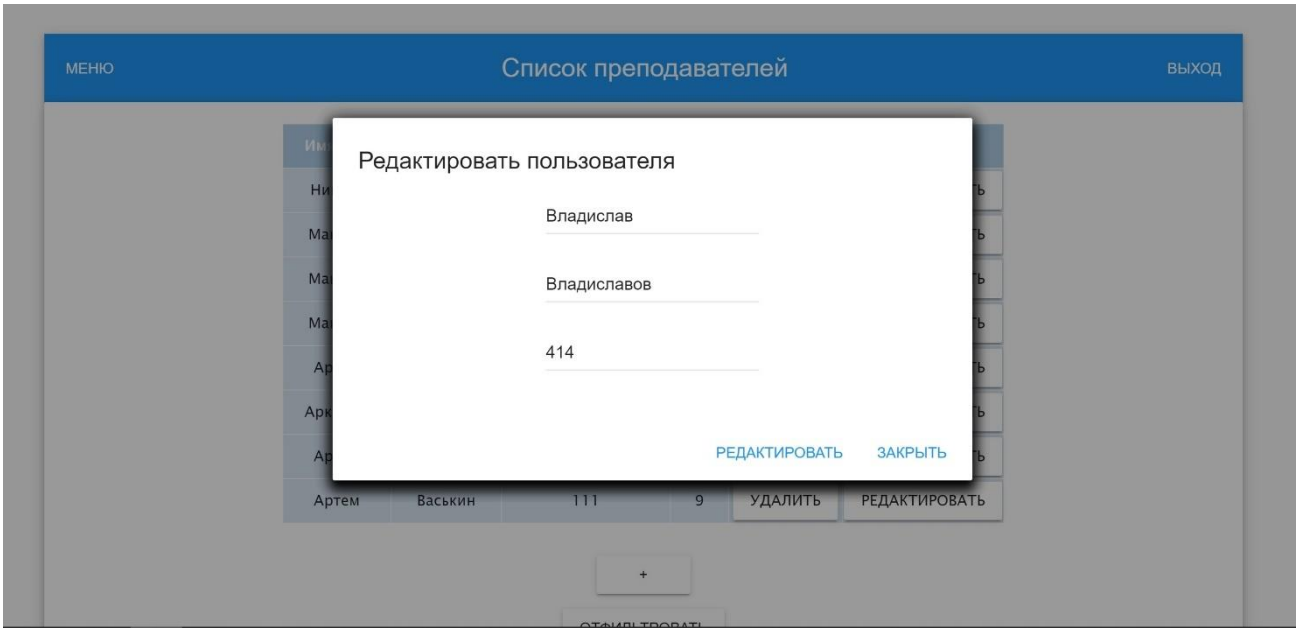

Рисунок 9 – Форма редактирования преподавателей

Ниже на странице «список преподавателей» расположена форма добавления нового преподавателя (рис. 10).

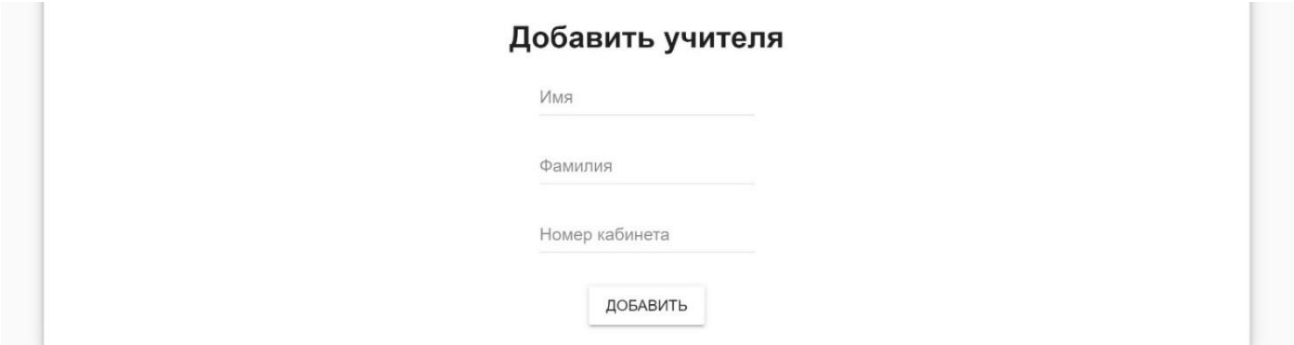

Рисунок 10 – Форма добавления учителей

Перейдем к странице просмотра студентов (рис. 11).

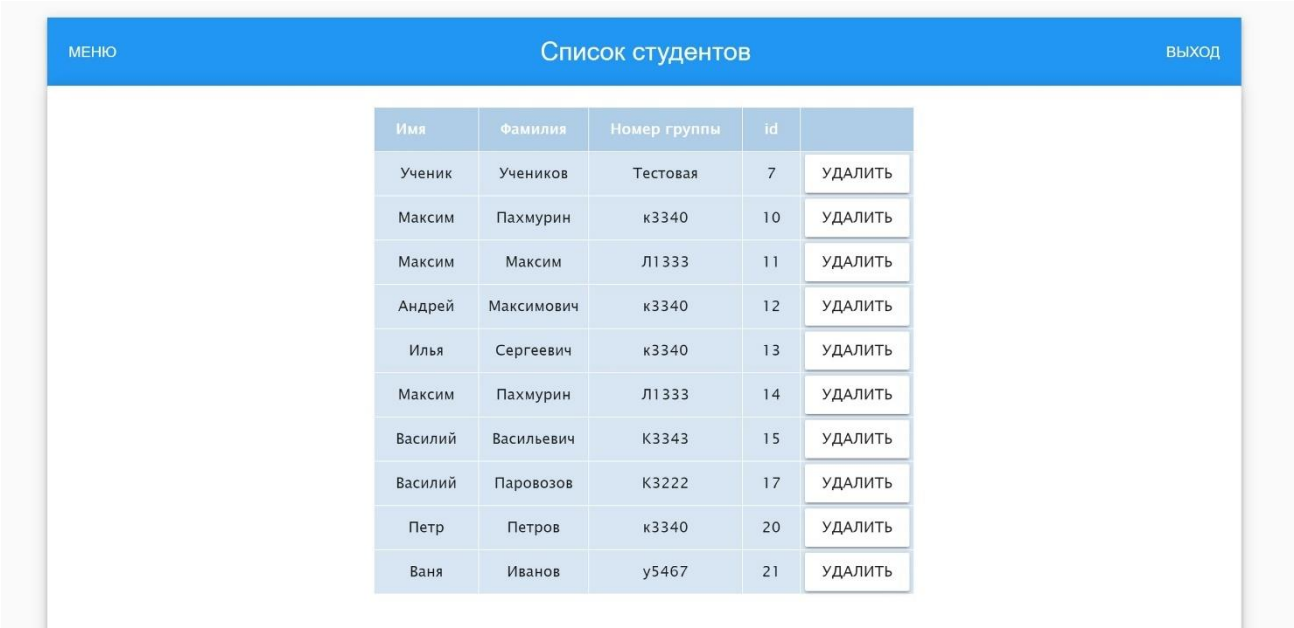

Рисунок 11 – Страница просмотра студентов

На данной странице имеется список студентов с их номерами групп. Также имеется возможность удаления студентов.

Ниже на странице имеется возможность добавления нового студента и создания новой группы(рис. 12).

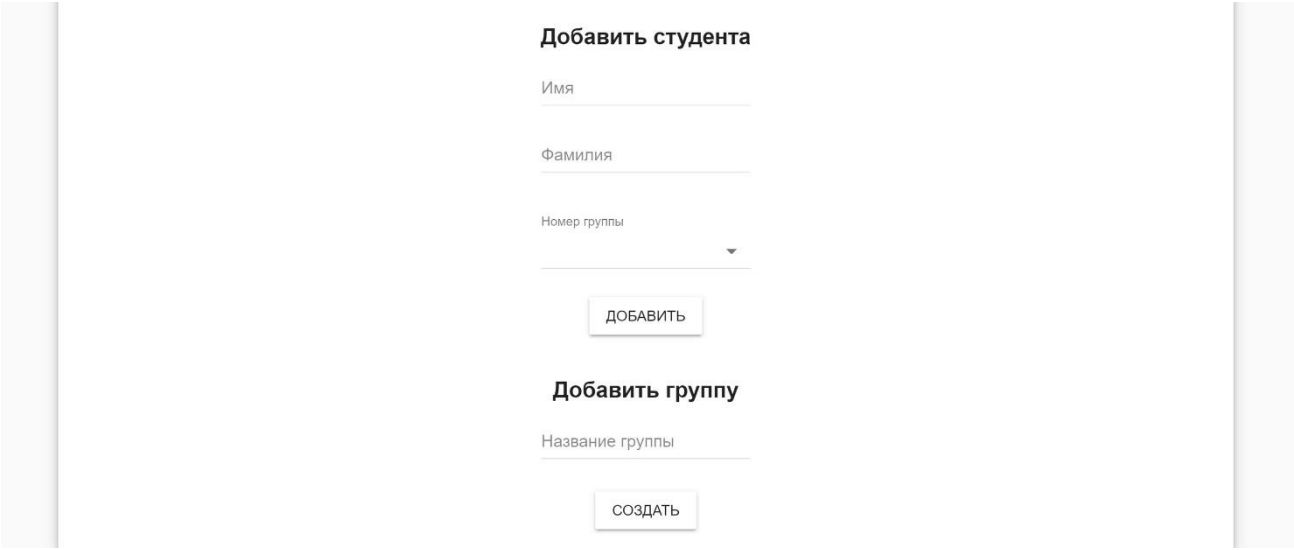

Рисунок 12 – Форма для добавления студентов и групп

Также имеется интерфейс «Журнал» (рис. 13), включающая в себя предмет, студент, оценка.

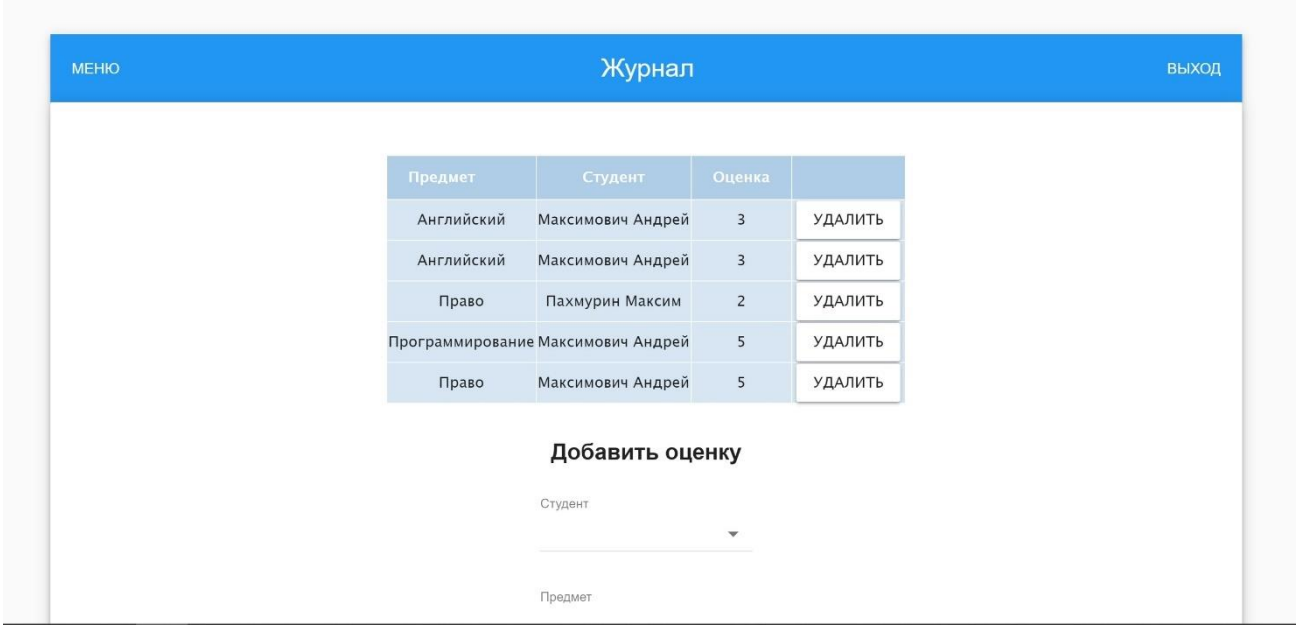

Рисунок 13 – Журнал

При этом также имеется возможность удаления полей. Ниже, на странице, имеется возможность выставить новую оценку (рис. 14).

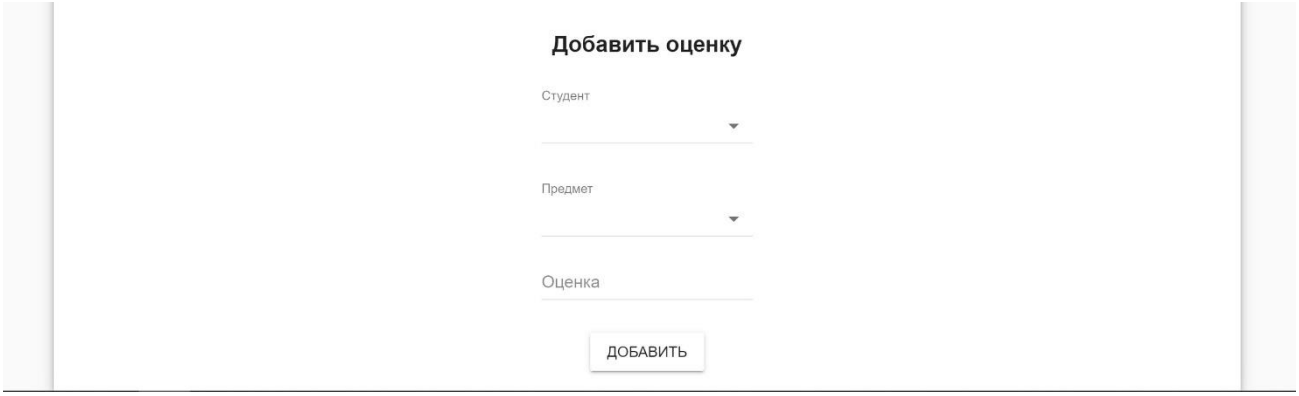

Рисунок 14 – Форма выставления оценки

На этом функционал системы окончен.

4.3. Выводы

Была разработана клиентская часть веб-сервиса. Разработаны интерфейсы для входа в систему, для просмотра и работы со студентами, преподавателями, группами, журналов и расписанием. Фреймворк Vue.js позволил сделать разработку быстрой и удобной. Благодаря использованию библиотеки Muse UI были получены приятные и стильные интерфейсы.

ЗАКЛЮЧЕНИЕ

По итогам данной работы были выполнены следующие задачи: проанализирована предметная область и функциональные требования, создана архитектура проекта, разработана серверная и клиентская части системы. В результате реализована система для управления колледжем, которая соответствует требованиям и обладает необходимым функционалом.

В рамках реализации задачи по созданию веб-сервиса были получены практические навыки работы с современными средствами разработки такими, как фреймворк Django REST, фреймворк Vue.js и библиотека Muse UI.

СПИСОК ЛИТЕРАТУРЫ

1. Документация Django Rest Framework [Электронный ресурс] — https://www.django-rest-framework.org/topics/documenting-your-api/. Дата обращения: 20.05.2020.
2. Документация PostgreSQL [Электронный ресурс] — https://www.postgresql.org/docs/. Дата обращения: 23.05.2020.
3. Документация Vue.js [Электронный ресурс] — https://ru.vuejs.org/v2/guide/. Дата обращения: 20.05.2020.
4. Документация Muse UI [Электронный ресурс] — https://muse-ui.org/#/en-US. Дата обращения: 30.05.2020.
5. Документация Docker [Электронный ресурс] — https://docs.docker.com/engine/install/. Дата обращения: 20.06.2020.
6. «Клиент-серверная архитектура в картинках» [Электронный ресурс] — https://habr.com/ru/post/495698/. Дата обращения: 20.05.2020.