

**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 Интеллектуальные системы в гуманитарной сфере

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

**О Т Ч Е Т**

по курсовой работе

Тема задания: РЕАЛИЗАЦИЯ WEB-СЕРВИСОВ СРЕДСТВАМИ Django REST framework.

Обучающийся: Осипенкова Анна Викторовна К3342

Руководитель: Говоров А. И.

Подписи членов комиссии:

\_\_\_\_\_ Говоров А. И.  
(подпись)

\_\_\_\_\_ Чунаев А.В  
(подпись)

\_\_\_\_\_ Антонов М.Б.  
(подпись)

Дата \_\_\_\_\_

Санкт-Петербург

2020

# СОДЕРЖАНИЕ

СОДЕРЖАНИЕ .....	1
ВВЕДЕНИЕ .....	2
1. ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ СЕРВИСА.....	3
1.1. Архитектура web-сервиса.....	3
1.2. Архитектура базы данных .....	4
2. РЕАЛИЗАЦИЯ WEB-СЕРВИСА.....	5
2.1. Общая информация .....	5
2.2. Реализация интерфейса “Восхождения” .....	6
2.3. Реализация интерфейса “Альпинисты” .....	9
2.4. Реализация интерфейса “Группы” .....	13
2.5. Реализация интерфейса “Вершины” .....	17
2.5 Реализация интерфейса “ Восхождения/Активность ” .....	22
ЗАКЛЮЧЕНИЕ.....	24

## ВВЕДЕНИЕ

Цель курсовой работы заключалась в реализации web-сервиса Climber, являющегося программной системой, предназначенной для администратора альпинистского клуба.

Средствами Django Rest Framework был реализован бэкенд web-сервиса, Vue.js был выбран для реализации фронтенда, а также использована база данных - PostgreSQL.

Администратор должен иметь возможность:

- добавления сведений о новом альпинисте, новой вершине;
- изменении сведений об альпинистах и вершинах;
- формирования новых групп и внесения всей информации после завершения восхождения группой.

Перечень возможных запросов:

1. Показать список альпинистов, осуществлявших восхождение в указанный интервал дат;
2. Показать список восхождений (групп), которые осуществлялись в указанный пользователем период времени
3. Предоставить информацию о том, сколько альпинистов побывали на каждой горе.
4. Предоставить данные о вершинах, если на них не было восхождений
5. Показать информацию о количестве восхождений каждого альпиниста на каждую гору.

# 1. ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ СЕРВИСА

## 1.1. АРХИТЕКТУРА WEB-СЕРВИСА

Для реализации web-сервиса Climber за основу был взят Django Rest Framework. С помощью него реализовано сервисное отображение данных, выборка необходимой информации, добавление новой информации и исполнение запросов в соответствии с поставленными задачами функционала. В качестве модели взаимодействия пользователя с системой была выбрана клиент-серверная архитектура.

Vue.js с плагином Vuetify использовались для осуществления фронтенда приложения. Vuetify был выбран из соображений быстрой скорости сборки клиентской части, а также удобства и простоты дизайна. Представления web-сервиса отображены в папке Components и полностью удовлетворяют запрашиваемому функционалу.

В качестве базы данных была использована база данных PostgreSQL, настройка базы данных, а также последующая работа и сборка приложения производилась с помощью редактора PyCharm. Для удобства запуска, а также быстроты дальнейшей сборки на желаемом сервисе были добавлены Docker контейнеры для каждой части архитектуры приложения.

Так, диаграмму архитектуры разрабатываемого web-сервиса представлены на Рисунке 1.

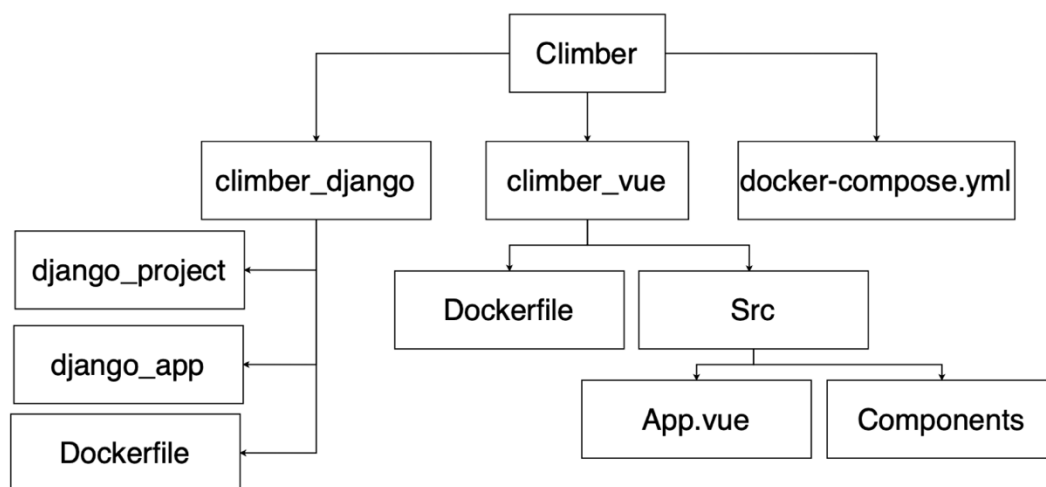


Рисунок 1 - Архитектура web-сервиса Climber

## 1.2. АРХИТЕКТУРА БАЗЫ ДАННЫХ

В качестве сущностей для разрабатываемого web-сервиса были выбраны вершины, восхождения, альпинисты, группы, клубы, владельцы клубов. Реализация вышеописанной базы данных представлена на Рисунке 2.

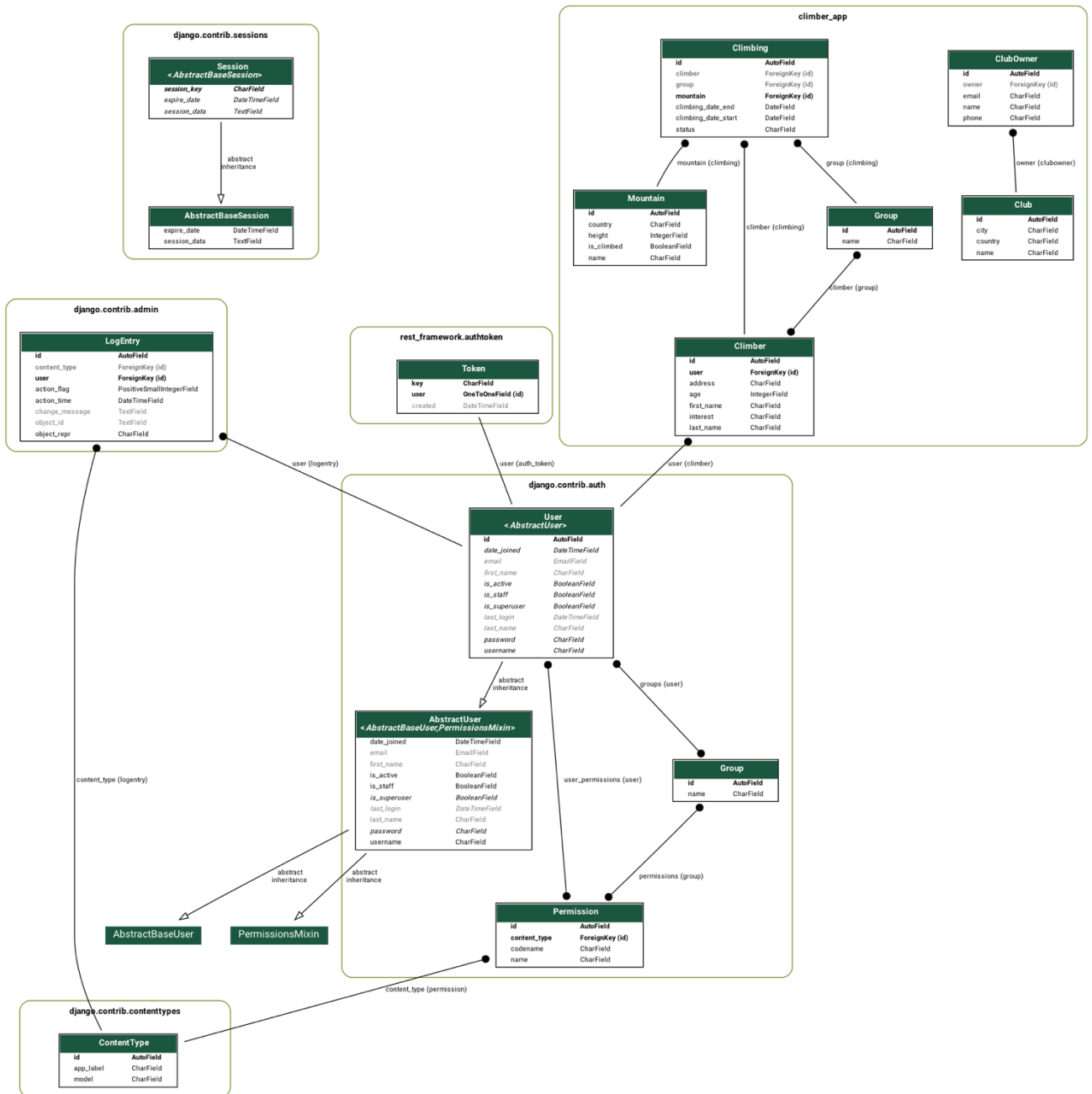


Рисунок 2 - Диаграмма базы данных web-сервиса

## 2. РЕАЛИЗАЦИЯ WEB-СЕРВИСА

### 2.1. ОБЩАЯ ИНФОРМАЦИЯ

Проект разделен на следующие главные интерфейсы: Восхождения, Альпинисты, Группы, Вершины.

Основная навигация в проекте, впрочем, как и дизайн реализован с помощью плагина Vue.js Vuetify. Существует Navigation-drawer, с помощью которого администратор может перемещаться по web-сервису(Рисунок 3).

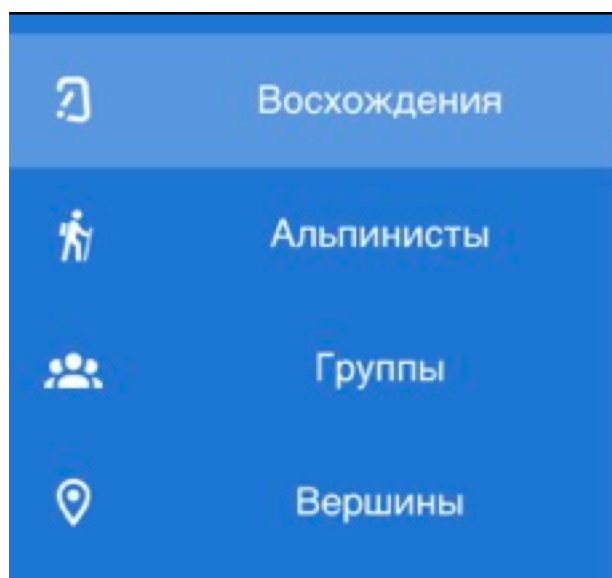


Рисунок 3 - Navigation-drawer web-сервиса.

## 2.2. РЕАЛИЗАЦИЯ ИНТЕРФЕЙСА “ВОСХОЖДЕНИЯ”

Данный интерфейс отображает общую информацию о восхождениях, соответствуя поставленным задачам. На странице с восхождениями реализованы три запроса:

- 1) Фильтр по статусу восхождения (запланировано, в процессе, завершено) (рис. 4)
- 2) Фильтр по типу восхождения (групповое, одиночное) (рис. 5)
- 3) Фильтр по дате восхождения (начало, конец) (рис. 6)

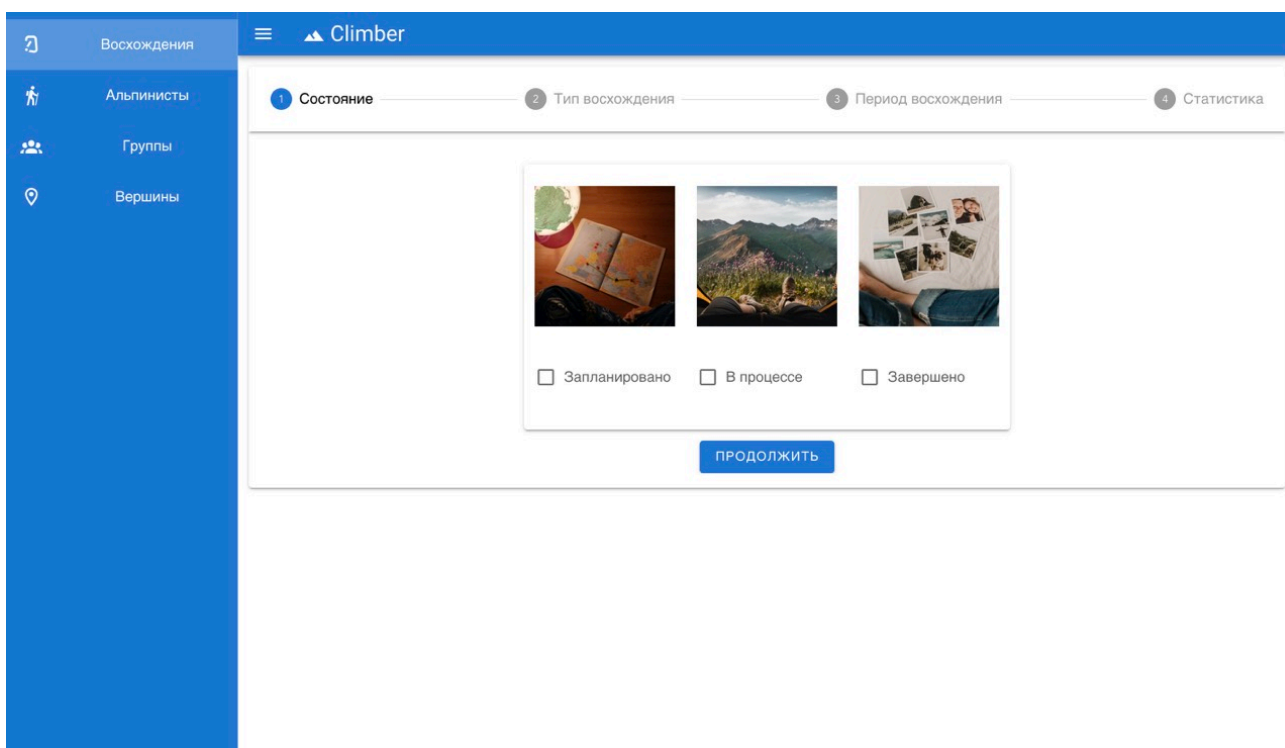


Рисунок 4 – интерфейс фильтрации по статусу

Рисунок 5 - интерфейс фильтрации по типу восхождения

Рисунок 6 - интерфейс фильтрации по периоду восхождения

Задача отображения статистики реализована с помощью модуля django-filters. Был создан фильтр- ClientFilter, который сортирует клиентов по возрасту. Далее с помощью сериалайзера, а также библиотеки generics.ListAPIView выводится список отсортированных



клиентов. Сортировка происходит в самом запросе путем добавления к ссылке различных и приравнивания аргумента к нужному значению.

Итоговая

ссылка:

"http://localhost:8000/api/climbing/list?is\_planned=&is\_in\_progress=&is\_finished=&is\_group=&is\_solo=&date\_start=&date\_end=", реализованный вывод списка восхождений (рис. 7)

Состояние Тип восхождения Период восхождения Статистика

Стрельбище, Россия, г.Приозерск, 50 м	Запланировано, 2020-06-24 - 2020-07-24	Тип: Групповое
Скала по дороге в Кузнечное, Россия, 80 м	Завершено, 2020-07-09 - 2020-07-09	Тип: Групповое

ВЕРНУТЬСЯ В НАЧАЛО НАЗАД

Рисунок 7 - интерфейс фильтрации по периоду восхождения



### 2.3. РЕАЛИЗАЦИЯ ИНТЕРФЕЙСА “Альпинисты”

Реализация данного интерфейса производилась с помощью библиотеки Django Rest Framework - `generics.CreateAPIView`. На представленной странице пользователю предоставляется возможность создать заказ - создать новую запись в базе данных в таблице “Заказ”. Для этого были созданы (аналогично `generics.ListAPIView`) `serializer`, а также `django view`

Для демонстрации взаимодействия с объектом альпиниста приведены примеры выполнения различных методов http запроса при помощи Django Admin:

- 1) Инициализация объекта альпиниста при помощи метода POST (рис. 8) и результат выполнения запроса (рис. 9)
- 2) Обновление полей объекта при помощи метода PUT (рис. 10) и результат выполнения запроса (рис. 11)
- 3) Пример удаления объекта альпиниста при помощи метода DELETE (рис. 12) и результат выполнения запроса (рис. 13)

Add climber

First name:	<input type="text" value="Анна"/>
Last name:	<input type="text" value="Эгамова"/>
Age:	<input type="text" value="22"/>
Address:	<input type="text" value="МСТ"/>
User:	<input type="text" value="anetchka"/>  
Interest:	<input type="text" value="Skiing"/>

[Save and add another](#) [Save and continue editing](#) [SAVE](#)

Рисунок 8 - Пример добавления альпиниста / Метод POST

✔ The climber "Анна, 22" was added successfully.

Select climber to change

ADD CLIMBER +

Action:  Go 0 of 1 selected

☐ CLIMBER

☐ Анна, 22

1 climber

Рисунок 9 - Пример выполнения post запроса

Change climber

HISTORY

First name:

Анна

Last name:

Эгимова

Age:

21

Address:

МГР

User:

anetchka

Interest:

Skiing

Delete

Save and add another

Save and continue editing

SAVE

Рисунок 10 - Пример редактирования информации об альпинисте / Метод put

✔ The climber "Анна, 21" was changed successfully.

Select climber to change

ADD CLIMBER +

Action:  Go 0 of 1 selected

☐ CLIMBER

☐ Анна, 21

1 climber

Рисунок 11 - Пример выполнения put запроса

Are you sure?

Are you sure you want to delete the climber "Анна, 21"? All of the following related items will be deleted:

Summary

▪ Climbers: 1

Objects

▪ Climber: Анна, 21

Yes, I'm sure

No, take me back

Рисунок 12 - Пример удаления альпиниста / Метод delete



Рисунок 13 - Пример выполнения delete запроса

Реализованы интерфейсы компонента Альпинисты:

- 1) Просмотр альпинистов (рис. 14)
- 2) Форма добавления альпиниста (рис. 15)
- 3) Изменить информацию об альпинисте (рис. 16)

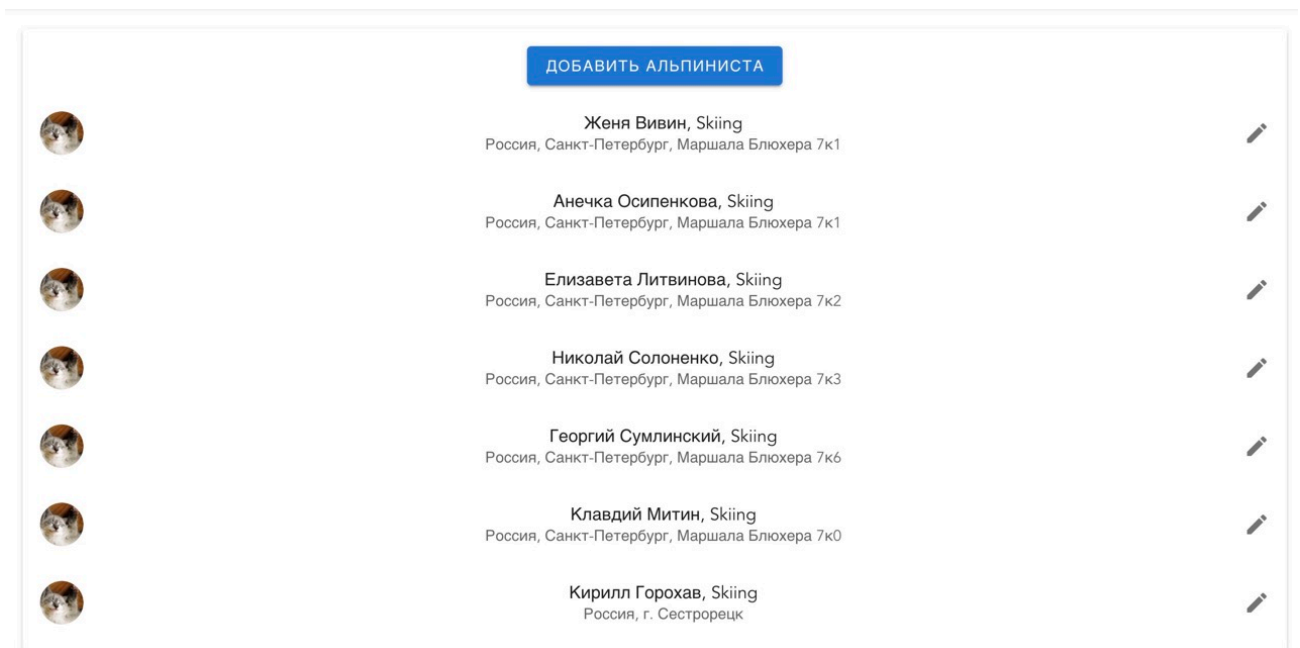


Рисунок 14 – Интерфейс просмотра альпинистов

ДОБАВИТЬ АЛЬПИНИСТА

### Добавить альпиниста

Имя\*       Фамилия\*

Имя пользователя\*       Адрес\*

Email\*       Пароль\*

Возраст\*       Интересы\*

\* Обязательные для заполнения поля

ЗАКРЫТЬ   ДОБАВИТЬ

Рисунок 15 – Форма добавления альпиниста

ДОБАВИТЬ АЛЬПИНИСТА

Женя Вивин, Skiing  
Россия, Санкт-Петербург, Маршала Блюхера 7к1

### Изменить информацию по альпинисту

Legal first name\*       Legal last name\*   
example of persistent helper text

Address\*

Age\*       Interests

\* indicates required field

CLOSE   SAVE

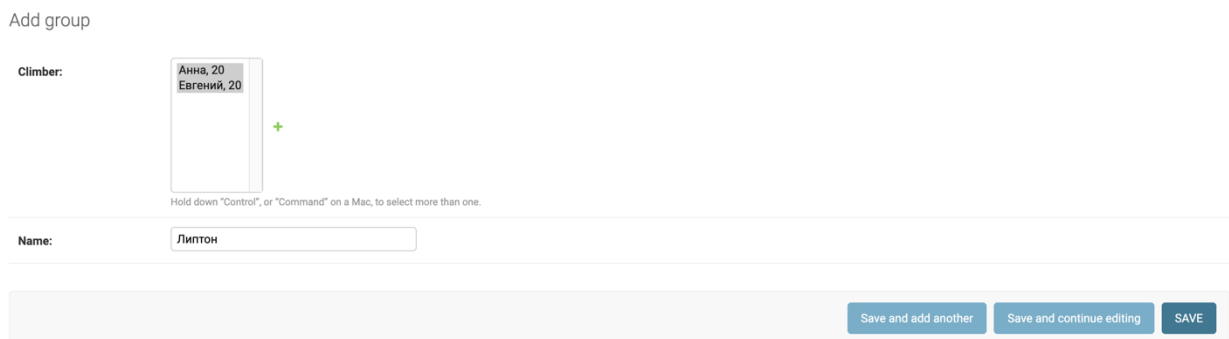
Рисунок 16 – Интерфейс изменения информации об альпинисте

## 2.4. РЕАЛИЗАЦИЯ ИНТЕРФЕЙСА “Группы”

Реализация данного интерфейса производилась с помощью библиотеки Django Rest Framework - `generics.CreateAPIView`. На представленной странице пользователю предоставляется возможность создать заказ - создать новую запись в базе данных в таблице “Заказ”. Для этого были созданы (аналогично `generics.ListAPIView`) `serializer`, а также `django view`

Для демонстрации взаимодействия с объектом альпиниста приведены примеры выполнения различных методов http запроса при помощи Django Admin:

- 1) Инициализация объекта группы при помощи метода POST (рис. 17) и результат выполнения запроса (рис. 18)
- 2) Обновление полей объекта при помощи метода PUT (рис. 19) и результат выполнения запроса (рис. 20)
- 3) Пример удаления объекта группы при помощи метода DELETE (рис. 21) и результат выполнения запроса (рис. 22)



Add group

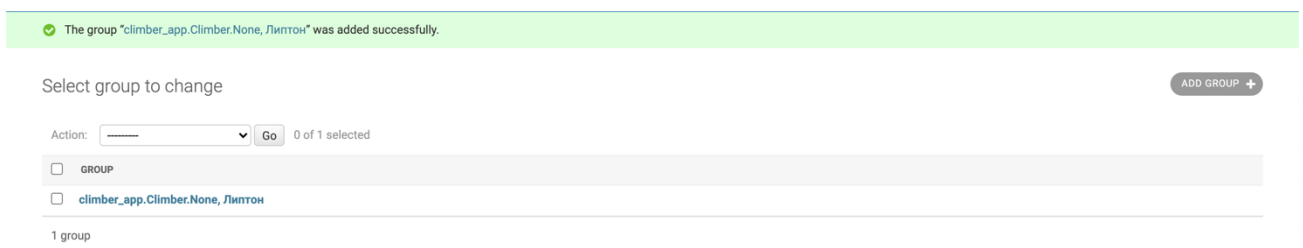
Climber: Анна, 20  
Евгений, 20 +

Hold down "Control", or "Command" on a Mac, to select more than one.

Name:

Save and add another Save and continue editing SAVE

Рисунок 17 – Пример добавления группы / Метод POST



✔ The group "climber\_app.Climber.None, Липтон" was added successfully.

Select group to change ADD GROUP +

Action: ----- Go 0 of 1 selected

<input type="checkbox"/>	GROUP
<input type="checkbox"/>	climber_app.Climber.None, Липтон

1 group

Рисунок 18 – результат выполнения POST запроса

Change group HISTORY

Climber:

Анна, 20  
Евгений, 20

+

Hold down "Control", or "Command" on a Mac, to select more than one.

Name:

Delete

Save and add another

Save and continue editing

SAVE

Рисунок 19 - Обновление полей / Метод PUT

✔ The group "climber\_app.Climber.None, Липтоны" was changed successfully.

Select group to change ADD GROUP +

Action:  Go 0 of 1 selected

<input type="checkbox"/>	GROUP
<input type="checkbox"/>	climber_app.Climber.None, Липтоны

1 group

Рисунок 20 - Результат выполнения запроса

Are you sure?

Are you sure you want to delete the group "climber\_app.Climber.None, Липтоны"? All of the following related items will be deleted:

**Summary**

- Groups: 1
- Group-climber relationships: 2

**Objects**

- Group: climber\_app.Climber.None, Липтоны
  - Group-climber relationship: Group\_climber object (1)
  - Group-climber relationship: Group\_climber object (2)

Yes, I'm sure

No, take me back

Рисунок 21 – Пример удаления объекта группы / Метод DELETE

✔ The group "climber\_app.Climber.None, Липтоны" was deleted successfully.

Select group to change ADD GROUP +

0 groups

Рисунок 22 – Результат выполнения запроса

Реализованы интерфейсы компонента Группы:

- 1) Просмотр групп (рис. 23)
- 2) Форма добавления группы (рис. 24)
- 3) Изменить информацию о группе (состав, название) (рис. 25)

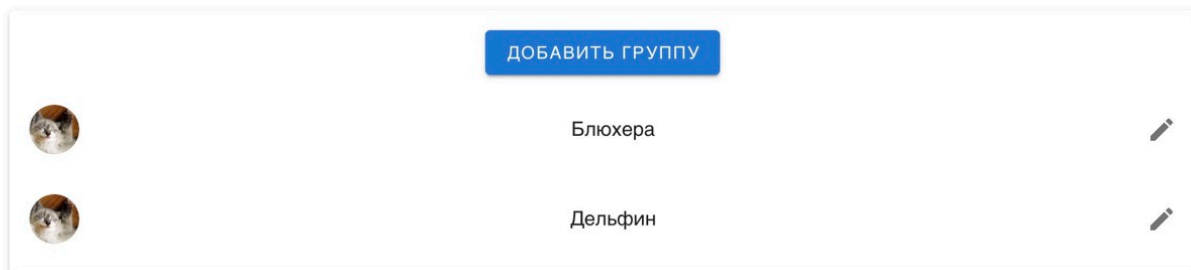


Рисунок 23 – Интерфейс просмотра групп

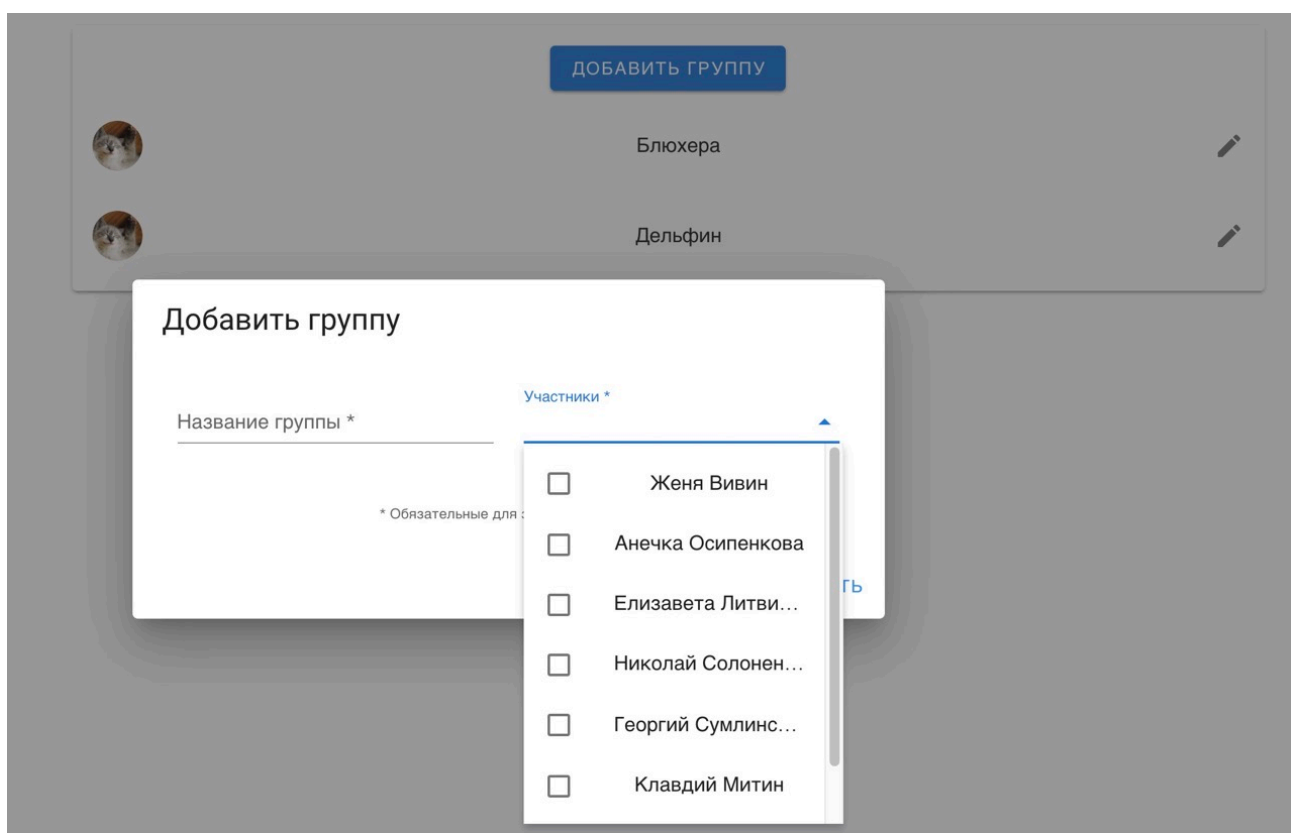


Рисунок 24 – Форма добавления группы



## Изменить группу

Название группы \*  
Блюхера

Участники \*  
Женя Вивин,  
Анечка Осипенкова

\* Обязательные для :

- ☒ Женя Вивин
- ☒ Анечка Осипенкова
- ☐ Елизавета Литви...
- ☐ Николай Солонен...
- ☐ Георгий Сумлинс...
- ☐ Клавдий Митин

гь

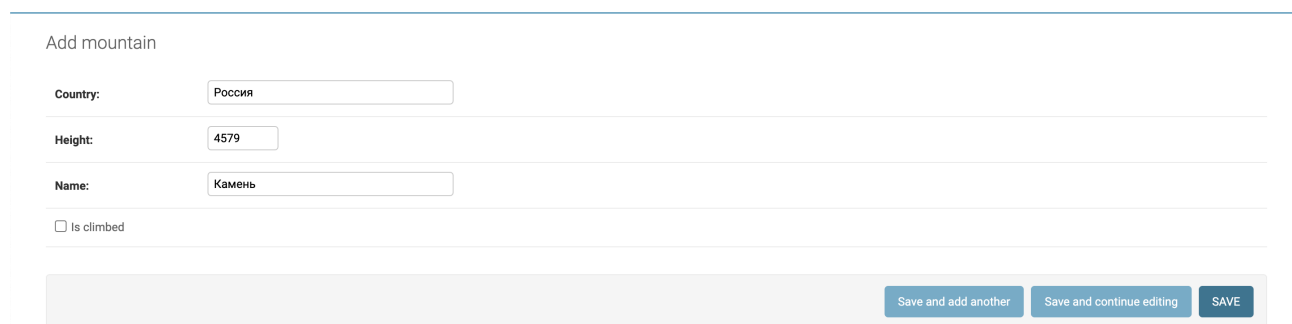
Рисунок 25 – Форма изменения информации о группе

## 2.5. РЕАЛИЗАЦИЯ ИНТЕРФЕЙСА “ВЕРШИНЫ”

Реализация данного интерфейса производилась с помощью библиотеки Django Rest Framework - `generics.CreateAPIView`. На представленной странице пользователю предоставляется возможность создать заказ - создать новую запись в базе данных в таблице “Заказ”. Для этого были созданы (аналогично `generics.ListAPIView`) `serializer`, а также `django view`

Для демонстрации взаимодействия с объектом альпиниста приведены примеры выполнения различных методов http запроса при помощи Django Admin:

- 1) Инициализация объекта вершины при помощи метода POST (рис. 26) и результат выполнения запроса (рис. 27)
- 2) Обновление полей объекта при помощи метода PUT (рис. 28) и результат выполнения запроса (рис. 29)
- 3) Пример удаления объекта вершины при помощи метода DELETE (рис. 30) и результат выполнения запроса (рис. 31)



Add mountain

Country:

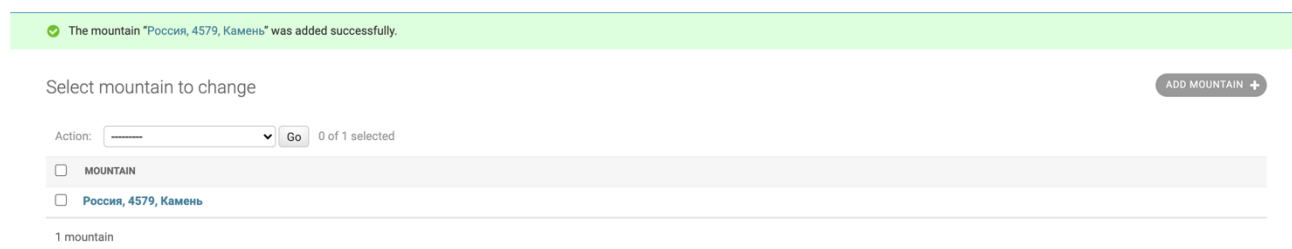
Height:

Name:

☐ Is climbed

[Save and add another](#) [Save and continue editing](#) [SAVE](#)

Рисунок 26 – Пример добавления вершины / Метод POST



✓ The mountain "Россия, 4579, Камень" was added successfully.

Select mountain to change [ADD MOUNTAIN +](#)

Action:   0 of 1 selected

<input type="checkbox"/>	MOUNTAIN
<input type="checkbox"/>	Россия, 4579, Камень

1 mountain

Рисунок 27 – Пример выполнения запроса

Change mountain

HISTORY

Country:

Россия

Height:

4579

Name:

Стратовулкан Камень

☐ Is climbed

Delete

Save and add another

Save and continue editing

SAVE

Рисунок 28 – Обновление полей объекта при помощи метода PUT

✓ The mountain "Россия, 4579, Стратовулкан Камень" was changed successfully.

Select mountain to change

ADD MOUNTAIN +

Action:

Go

0 of 1 selected

☐ MOUNTAIN

☒ Россия, 4579, Стратовулкан Камень

1 mountain

Рисунок 29 – Пример выполнения запроса

Are you sure?

Are you sure you want to delete the mountain "Россия, 4579, Стратовулкан Камень"? All of the following related items will be deleted:

**Summary**

- Mountains: 1

**Objects**

- Mountain: Россия, 4579, Стратовулкан Камень

Yes, I'm sure

No, take me back

Рисунок 30 – Пример удаления объекта вершины при помощи метода DELETE

✓ The mountain "Россия, 4579, Стратовулкан Камень" was deleted successfully.

Select mountain to change

ADD MOUNTAIN +

0 mountains

Рисунок 31 – Пример выполнения запроса

Реализованы интерфейсы компонента Вершины:

- 1) Просмотр альпинистов (рис. 32)
- 2) Форма добавления альпиниста (рис. 33)
- 3) Изменить информацию об альпинисте (рис. 34)

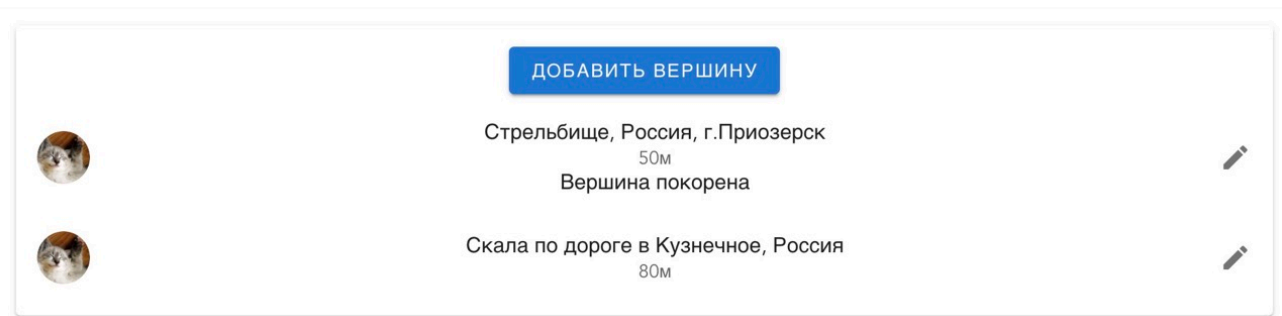


Рисунок 32 – Интерфейс просмотра вершин

## Добавить вершину

Название вершины *	Страна *
Стрельбище	Россия, г.Приозерск
Высота *	<input checked="" type="checkbox"/> Вершина была покорена *
50	

\* Обязательные для заполнения поля

ЗАКРЫТЬ   СОХРАНИТЬ

Рисунок 33 – Интерфейс добавления вершины

## Изменить вершину

Название вершины \*

Стрельбище

Страна \*

Россия, г.Приозерск

Высота \*

50



Вершина была покорена \*

\* Обязательные для заполнения поля

ЗАКРЫТЬ

СОХРАНИТЬ

Рисунок 34 – Интерфейс добавления вершины



## 2.5 РЕАЛИЗАЦИЯ ИНТЕРФЕЙСА “ Восхождения/Активность ”

В дополнение к основным интерфейсам был реализован дополнительный «Восхождения/Активность», который позволяет отразить самые активные группы по периоду восхождения, также учитывается количество покоренных вершин.

Изначально отображается общий рейтинг (рис. 35) по группам за весь период, затем после фильтрации отображается рейтинг групп за конкретный период с перечислением покоренных вершин (рис. 36).

Восхождения/активность

С  
ДД.ММ.ГГГГ

По  
ДД.ММ.ГГГГ

ВЫВЕСТИ ДАННЫЕ СБРОСИТЬ

Восхождения

Стрельбище, Россия, г.Приозерск, 50 м  
Запланировано, 2020-06-24 - 2020-07-24  
Тип: Групповое

Скала по дороге в Кузнечное, Россия, 80 м  
Завершено, 2020-07-09 - 2020-07-09  
Тип: Групповое

Лучшие группы

Блюхера  
Количество восхождений 1

Рисунок 35 – Интерфейс без фильтров

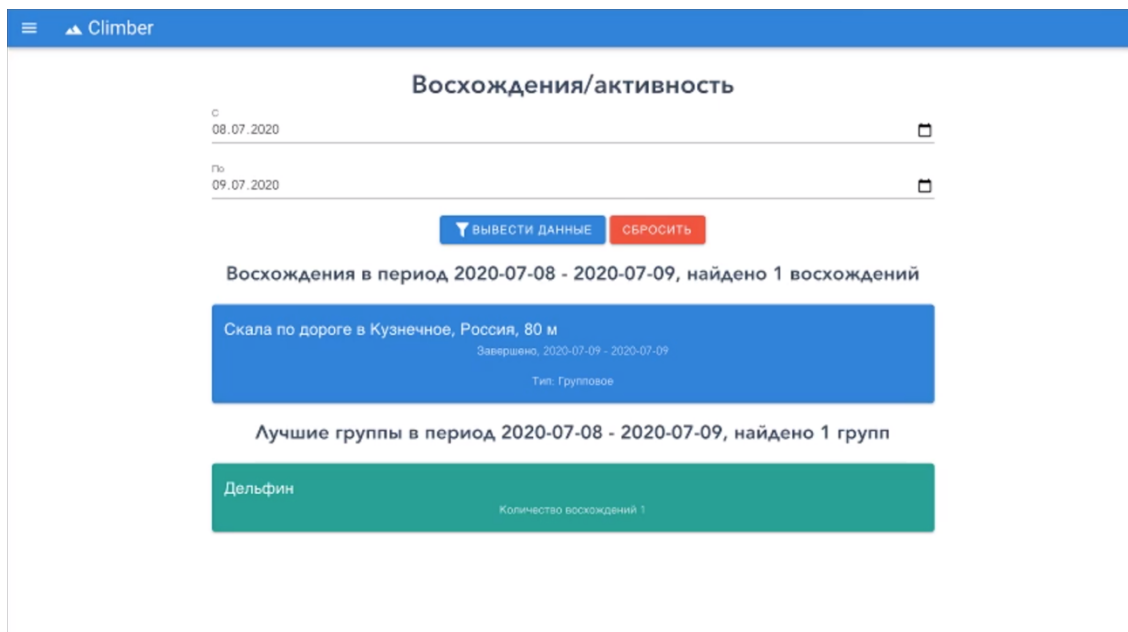


Рисунок 36 – Интерфейс после применения фильтров



## ЗАКЛЮЧЕНИЕ

Цель курсовой работы заключалась в реализации web-сервиса Climber, которое могла бы использовать схожая компания для организации и мониторинга восхождений.

Для решения данной задачи был выбран следующий стек технологий: Django Rest Framework, Vue.js, PostgreSQL, что позволило достаточно быстро и удобно реализовать задуманный проект. В результате выполнения курсовой удалось выполнить все вышеставленные задачи, которые соответствуют запросам технического задания. Цель курсовой работы достигнута в полном размере.

В качестве дальнейшего развития проекта можно усовершенствовать дизайн вэб сайта, добавить интерфейсы для учета инвентаря, контрольных точек восхождения.

## СПИСОК ЛИТЕРАТУРЫ

*Django Rest Framework*. Документация Django Rest Framework [Электронный ресурс]. URL: <https://www.django-rest-framework.org> (дата обращения: 29.06.2020).

*WebDevBlog*. Создание Django API используя Django Rest Framework [Электронный ресурс]. URL: <https://webdevblog.ru/sozdanie-django-api-ispolzuya-django-rest-framework-apiview/> (дата обращения: 29.06.2020).

*Evantotuts+*. JWT Аутентификация в Django [Электронный ресурс] URL: <https://code.tutsplus.com/ru/tutorials/how-to-authenticate-with-jwt-in-django--cms-30460> (дата обращения: 29.06.2020).

*Vue.js*. Документация Vue.js [Электронный ресурс]. URL: <https://vuejs.org> (дата обращения: 29.06.2020).

*Vuetify*. Документация Vuetify [Электронный ресурс]. URL: <https://vuetifyjs.com/ru/> (дата обращения: 29.06.2020).

*Stack OverFlow*. Set initial vuetify vue-select value [Электронный ресурс]. URL: <https://stackoverflow.com/questions/51392719/set-initial-vuetify-v-select-value> (дата обращения: 29.06.2020).

*Asyncee*. Продвинутые запросы в Django: сортировка по дате [Электронный ресурс]. URL: <https://asyncee.github.io/posts/advanced-django-querying-sorting-events-by-date/> (дата обращения: 29.06.2020).