## 2.2 Интерфейс веб приложения

При заходе в приложение пользователь может выбрать в меню одно из трех основных странниц, это добавление расчетных случаев, вывод расчетных случаев или в административной панели меню параметров.

При выборе меню параметров пользователь попадает на страницу, на которой может посмотреть все параметры, разбитые по категориям, находящиеся на данный момент в базе и форму добавления параметра в базу.

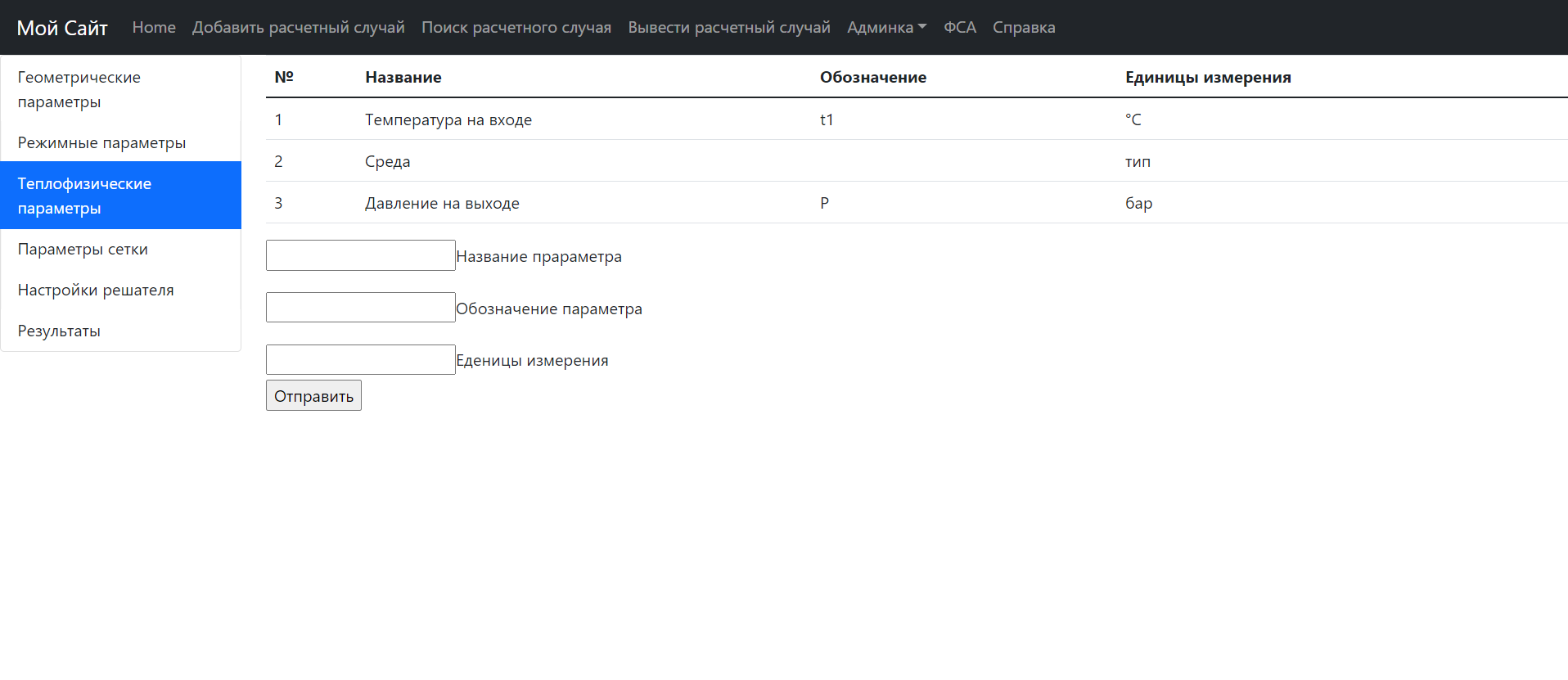


Рис.15.меню просмотра и добавление параметров

При добавлении параметров происходит проверка, есть ли такой параметр, если данный параметр уже есть, то появится сообщение об этом и добавление параметра будет отменено, иначе параметр будет добавлен в базу.

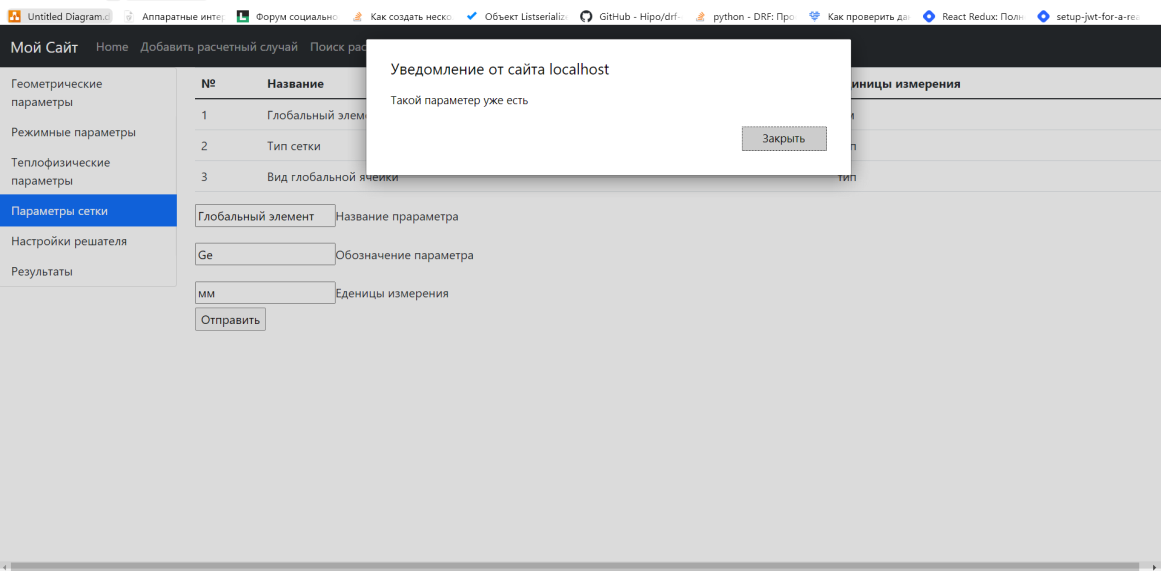


Рис.16.Сообщение если параметр такой уже есть

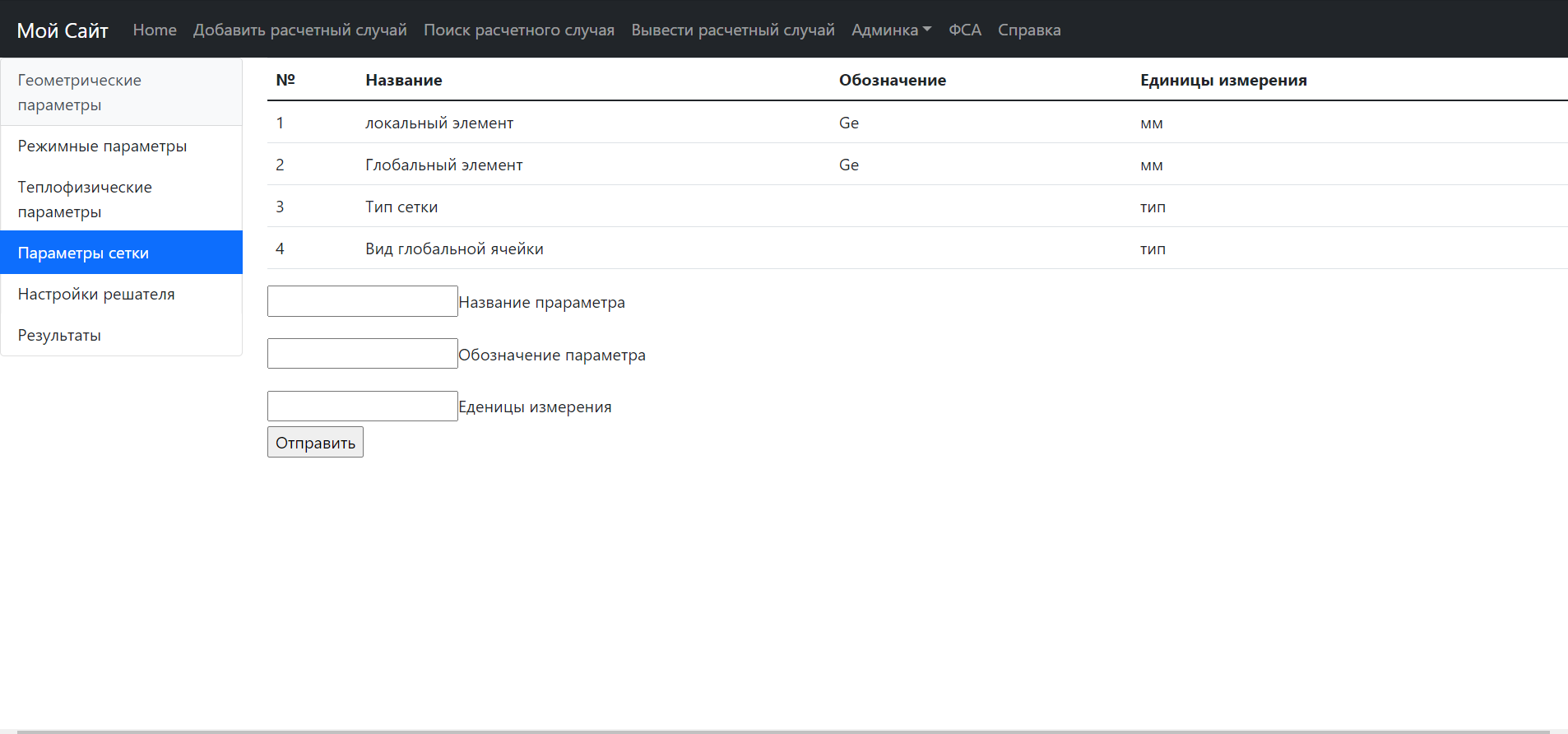


Рис.17.Добовление параметра в базу

При заходе в меню вывода ил добавление расчетного случая, появляется меню, в котором мы задаем основные параметры, в этом меню следящее поле зависит от предыдущих, сначала выбираются физический процесс и оборудование, после область расчетного случая,

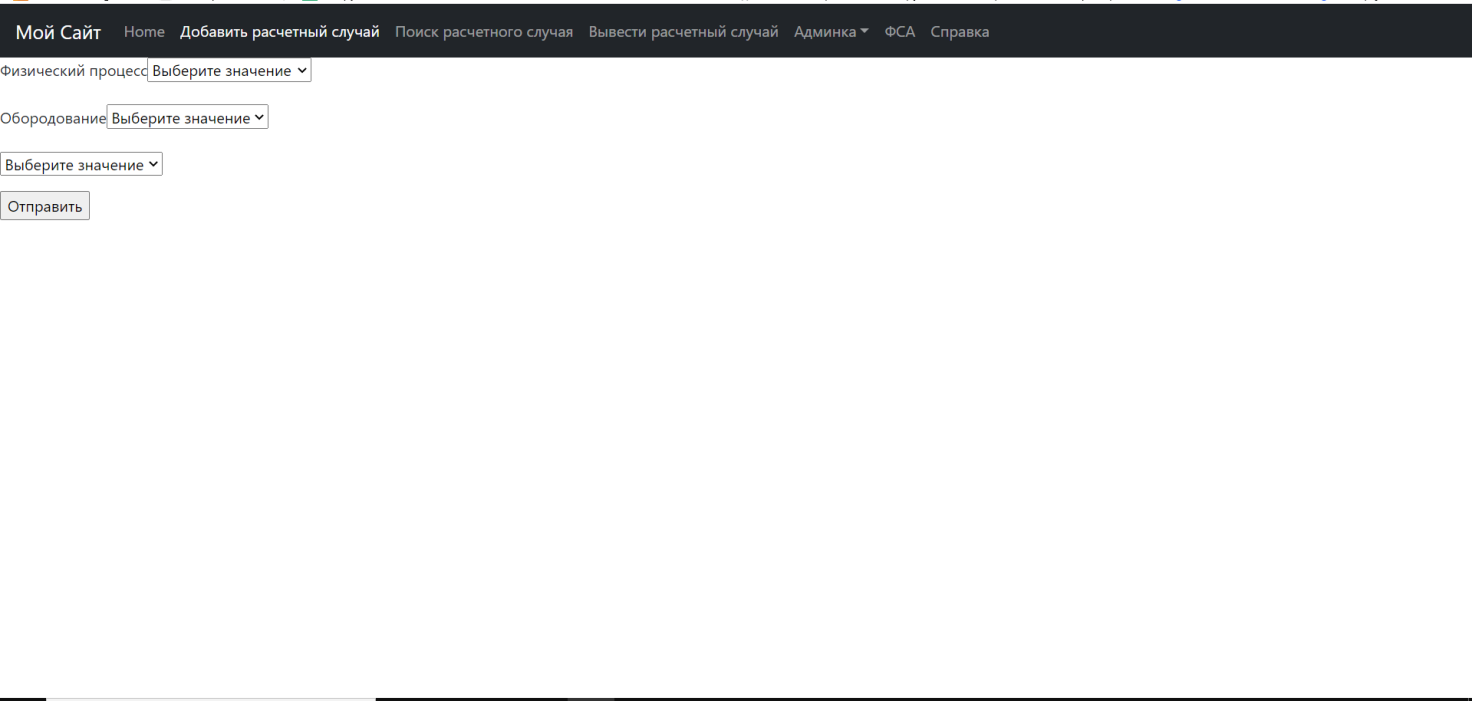


Рис.18.Изначальное меню выбора

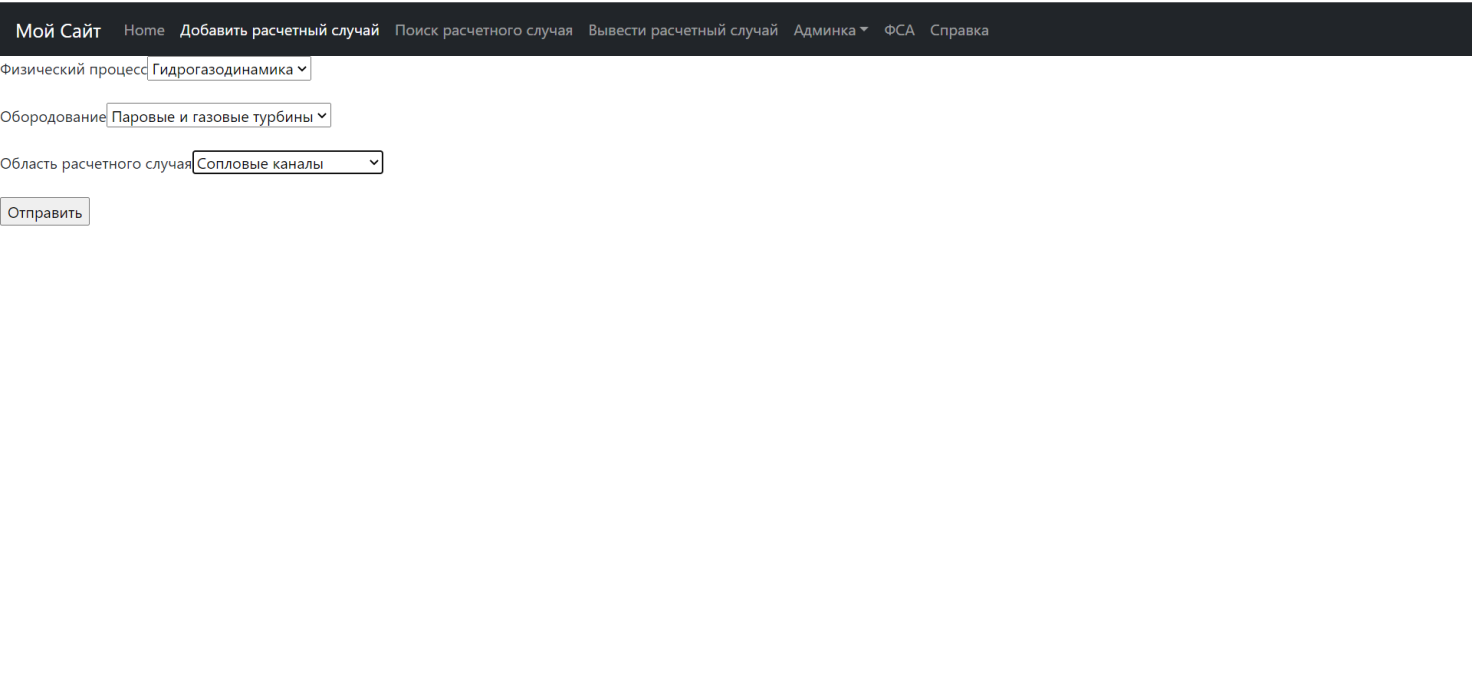
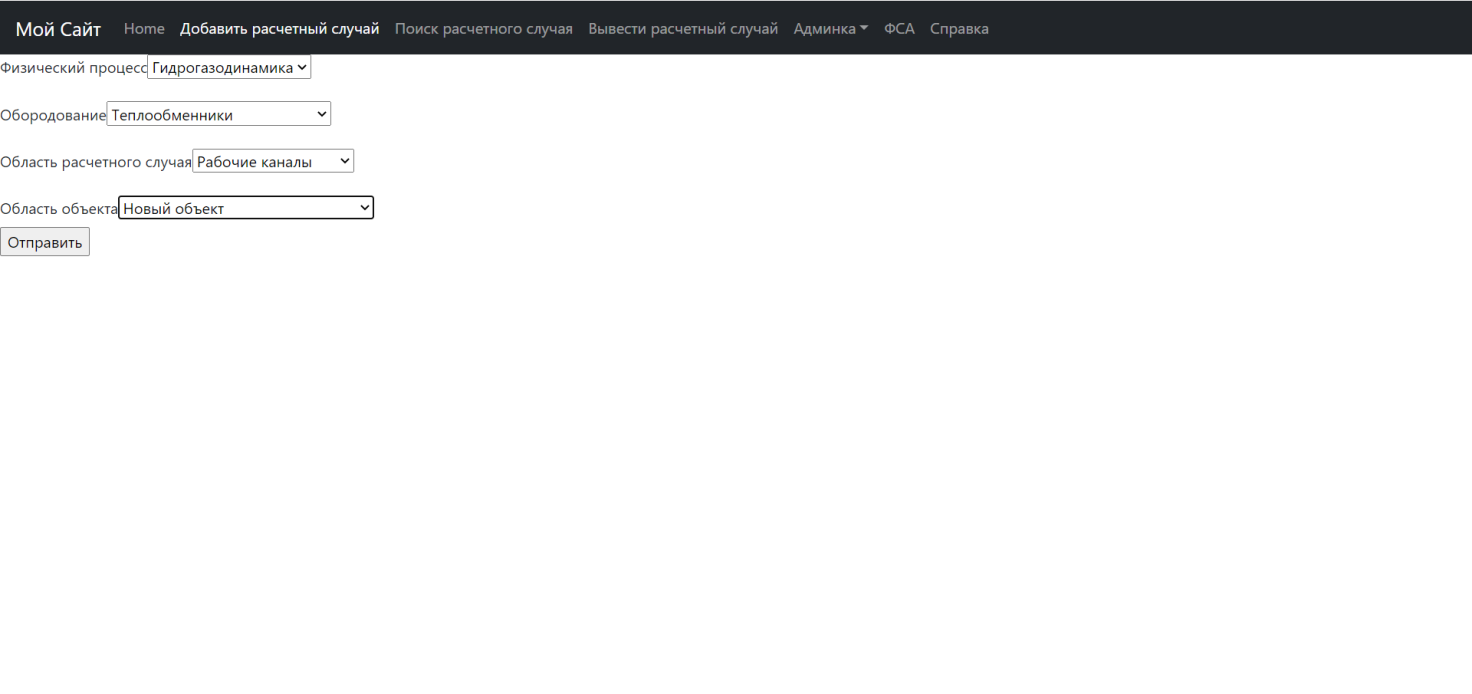


Рис.18.Меню после выбора

Так Сопловые каналы не являются родителем для подобластей, меню на нем и заканчивается, но, если у области расчетного случая есть наследники мы сможем углубиться в подобласть пример на рисунке 19.

 Рис.19.углубление в область объекта

Это проблема решается благодаря древовидной структуре, на стороне сервера мы можем получить наследников каждого уровня. Если они есть то он вернет их иначе или венет сообщение, что наследников нет.

Код возвращающий наследнеков

@api\_view(['GET'])

def ReturnAreaTree(request):

if request.method == 'GET':

area\_id = request.GET.get("area\_id", 0)

parent\_subarea\_id = request.GET.get("parent\_subarea\_id", 0)

phys\_proc\_id = request.GET.get("phys\_proc\_id", 1)

equpment\_type\_id = request.GET.get("equpment\_type\_id",1)

subareas = AreasTree.objects.filter(id\_area=area\_id, parent=parent\_subarea\_id, experimentclass\_\_id\_ph\_p\_t\_p\_e\_field\_\_id\_ph\_p\_field=phys\_proc\_id,experimentclass\_\_id\_ph\_p\_t\_p\_e\_field\_\_id\_t\_p\_e\_field=equpment\_type\_id)

area = Area.objects.get(pk=area\_id);

subareas\_list = []

for item in subareas:

subareas\_list.append({

'id':item.id\_subarea,

'name':item.name\_subarea

})

if(len(subareas\_list)!=0):

return response.Response({'area\_name':area.name\_area,'subareas':subareas\_list},status.HTTP\_200\_OK)

return response.Response({'status': "failed"}, status.HTTP\_404\_NOT\_FOUND)

На стороне пользователя мы обрабатываем данное дерево отправляем запрос, и если ответ положительный добавляем выпадающее меню:

const handleChangeArea = (area\_id, parentSubareId, index) => {

getArea(area\_id, parentSubareId, PhisicalProcess, PowerEqupment).then(res => {

const subarea = {

id: area\_id,

areaName: res.data.area\_name,

subareas: res.data.subareas

};

subareas.splice(index);

SetSubareas(subareas.concat(subarea));

}).catch(() => {

subareas.splice(index);

}

);SetArea(parentSubareId)

}

function SelectArea(){return subareas.map((subarea, index) =>

<div key={subarea.id}>

<label>{subarea.areaName}</label>

<select key={subarea.id} onChange={(event) => handleChangeArea(subarea.id+1,event.target.value,(index+1))}>

<option disabled hidden selected>

Не выбран

</option>

{subarea.subareas.map(elem=><option key={elem.id} value={elem.id}>{elem.name}</option>)}

</select></div>);}

После выбора пользователя отправляет на страницу вывода или добавления значения, в зависимости от выбранной ранее в меню страницы.

Таблица строится динамически на основе данных хранящихся в БД, и каждый класс может иметь свое число параметров и расчетных случаев.

Меню добавления если случай раньше был создан, то мы увидим форму с уже внесенными значениями которую мы можем изменять.

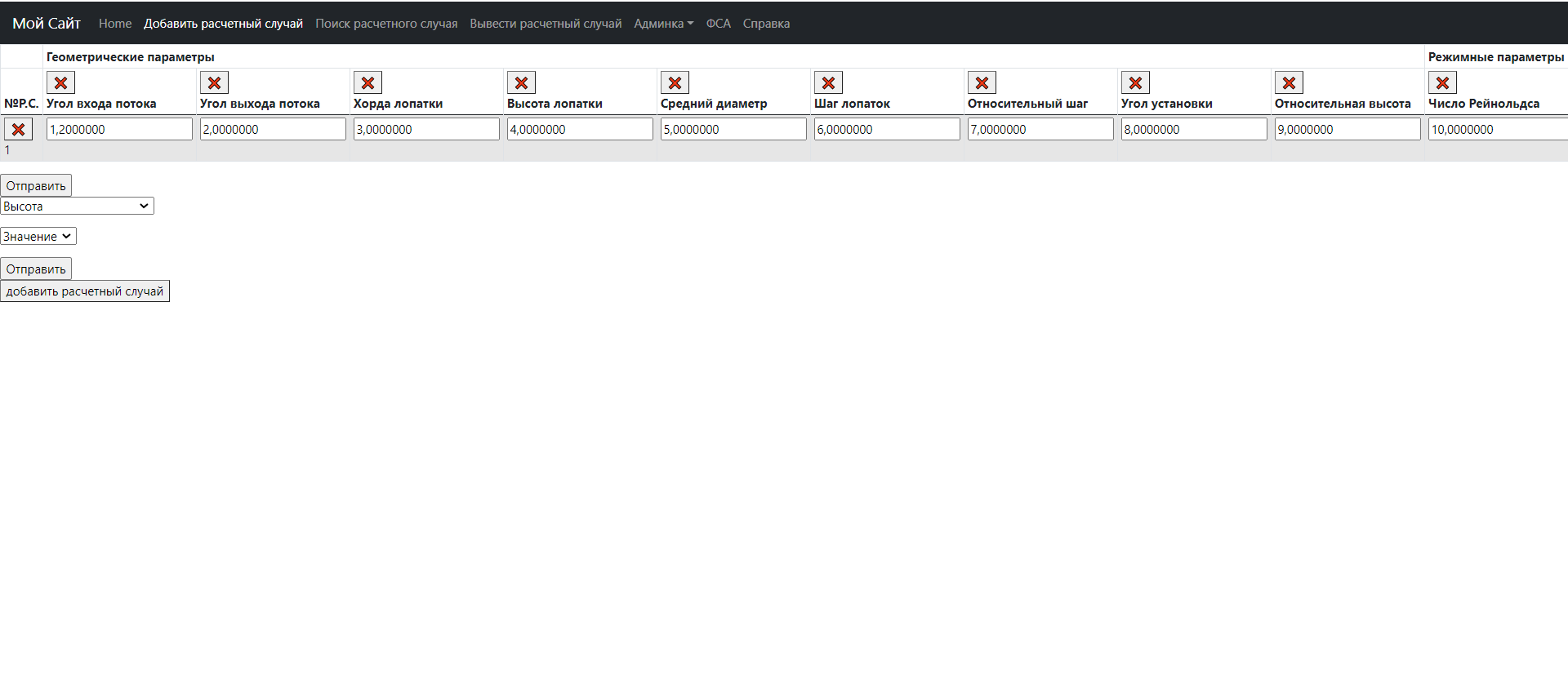


Рис.19.Меню добавление, расчетного случая, ранее созданного

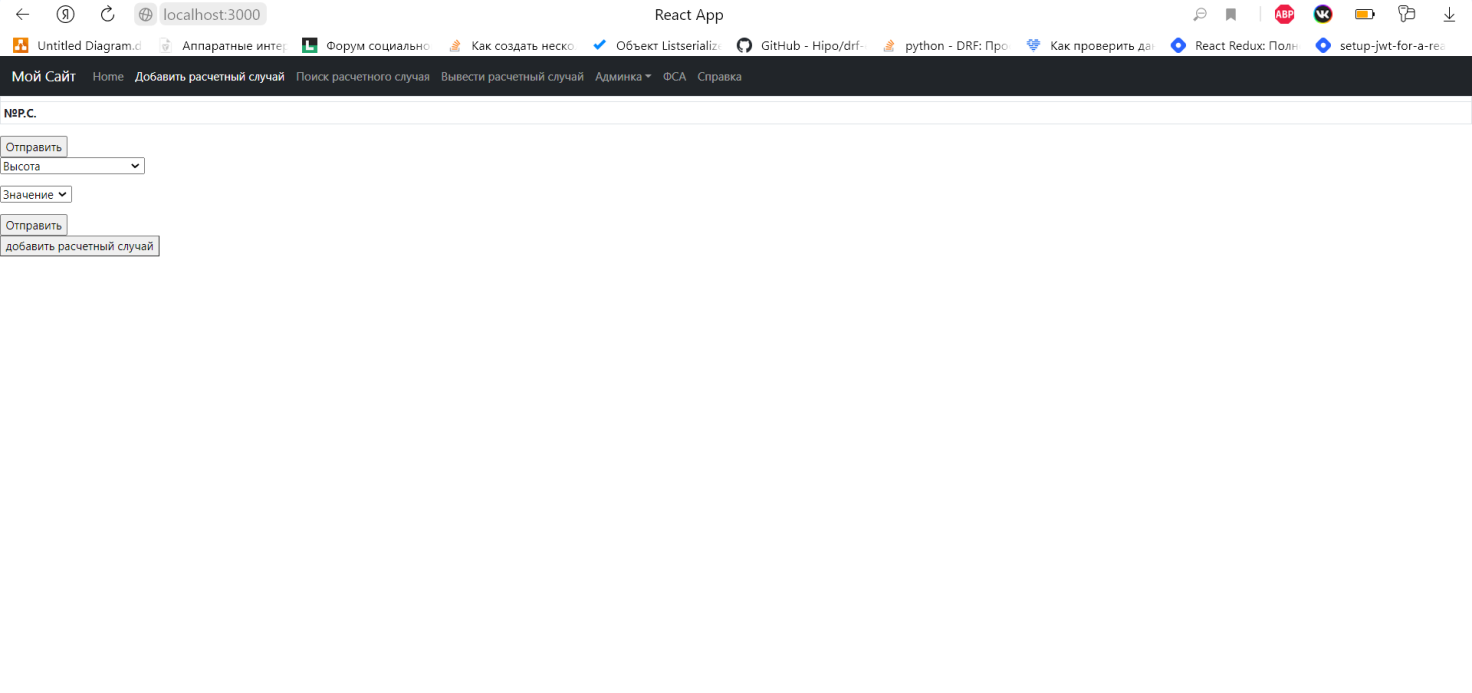


Рис.20.Меню добавление, расчетного случая, если значений не было

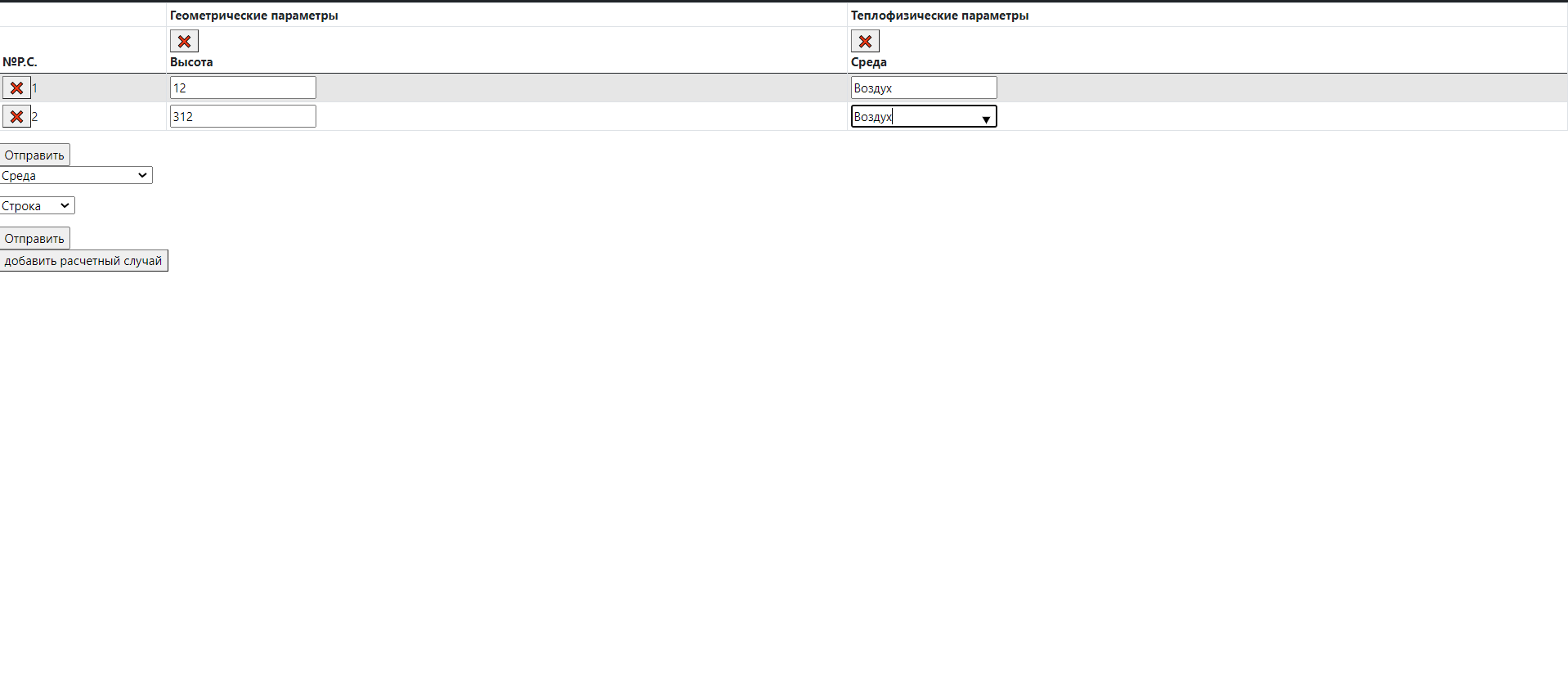


Рис.20. добавление данных

В случае отправки данных, и отсутствию связи сервера данные сохранятся в Local storage в браузере пользователя и позже можно будет их отправить.

if (deleteRow.length || deleteColumn.length) {

console.log(deleteRow.length || deleteColumn.length);

DeleteExpirement({

"experiment\_class": props.class,

"rows": deleteRow,

"columns": deleteColumn

}).catch(error => {

if (!error.response) {

console.log("Please check your internet connection.");

const deleteExpirement ={

"experiment\_class": props.class,

"rows": deleteRow,

"columns": deleteColumn

};

localStorage.setItem("Ispalit\_delete\_Expirement",JSON.stringify(JSON.stringify(deleteExpirement)));

}

});

}

if (updateValues.length)

UpdateExpirement(updateValues).catch(error => {

if (!error.response) {

console.log("Please check your internet connection.");

localStorage.setItem("Ispalit\_update\_Expirement",JSON.stringify(updateValues));

}

});

if (createValue.length)

CreateExpirement(createValue).catch(error => {

if (!error.response) {

console.log("Please check your internet connection.");

localStorage.setItem("Ispalit\_create\_Expirement",JSON.stringify(createValue));

}

После отправки проверяется ответ от сервера и если ответ, что соединение нарушено if (!error.response) с помощью команды localStorage.setItem() данные сохраняются.

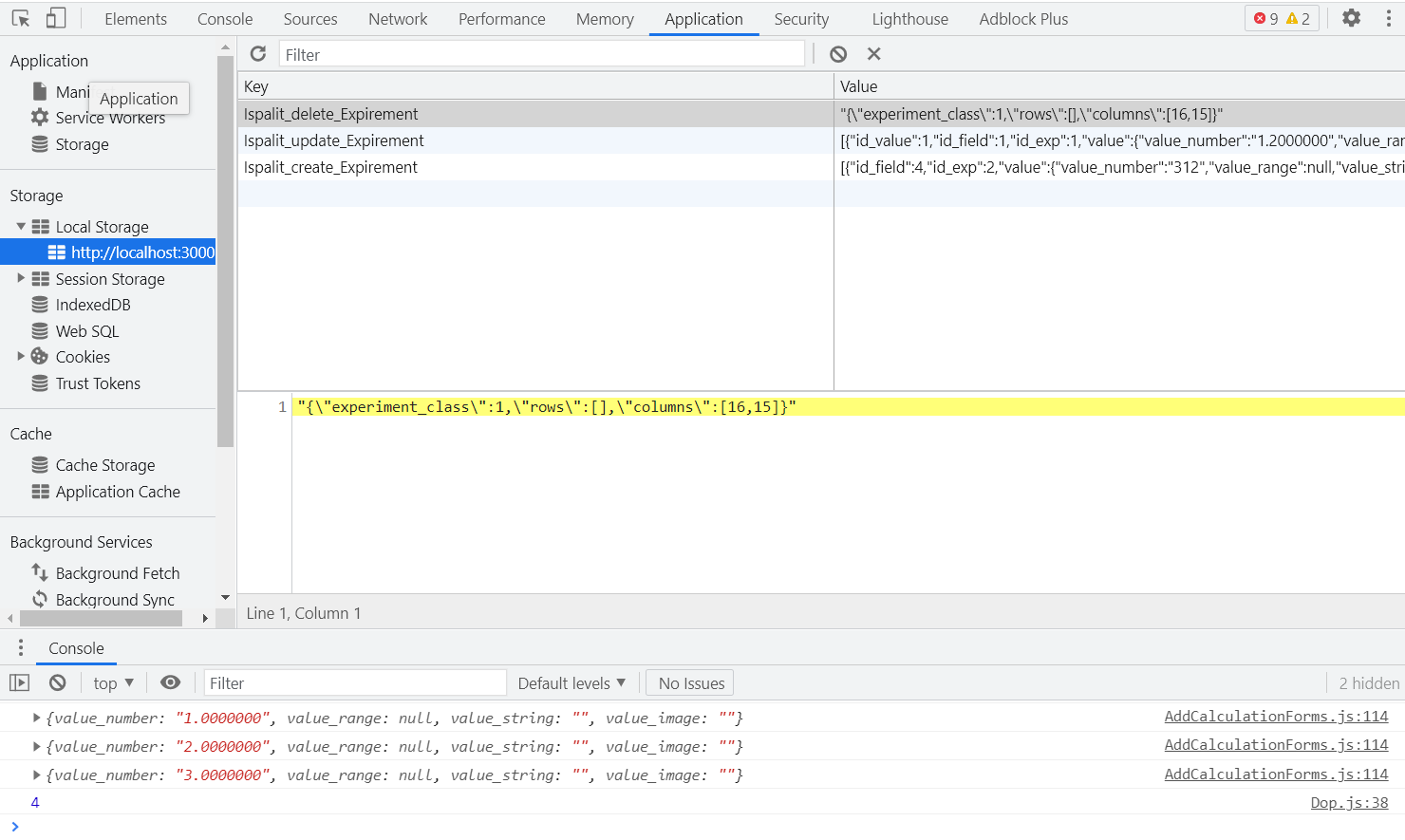


Рис.21. Сохранение данных в локальном хранилище

В случае выбора вывести расчетного случая и выбора всех будет показана таблица со всеми внесенными случаями.

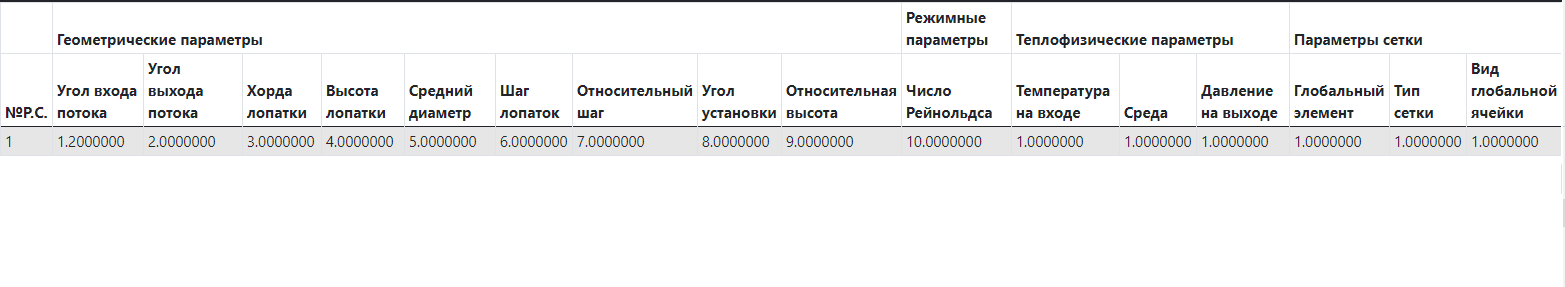


Рис.22Страница вывода данных

Также приложение имеет мобильную адаптацию

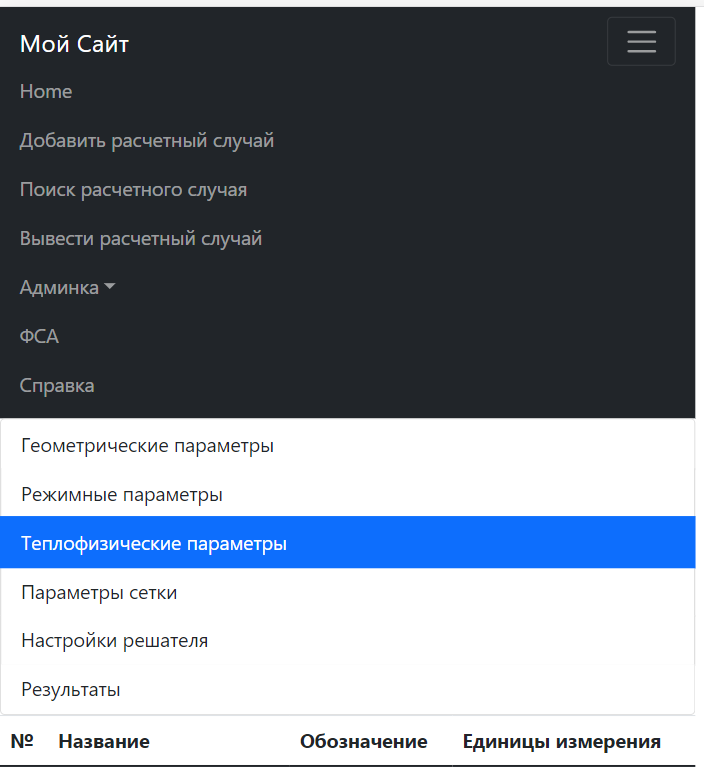


Рис.23. Мобильная адаптация

Благодаря мобильной адаптации и возможности сохранения данных, это приложение может использоваться и как мобильное в случае отсутствия полноценного мобильного приложения.

# ЗАКЛЮЧЕНИЕ

Таким образом, было разработано веб приложение для хранения, и отображения данных энергетических процессов. Данное ПО включает в себя базу данных и веб сервер и react приложение.

Функционал ПО позволяет формировать задачи любой сложности и с любым количеством параметров. Формирование задач любой сложности обеспечено за счет древовидной структуры базы и ее максимальной универсальности. Также данное приложение предоставляет возможность автономной работы пользователю при заполнении таблицы благодаря Localstorage и возможность использования его, как мобильного благодаря адаптивному дизайну.

# СПИСОК ЛИТЕРАТУРЫ

1. Одностраничные и многостраничные веб-приложения [Электронный ресурс]. Режим доступа: : <https://vc.ru/seo/108149-odnostranichnye-spa-i-mnogostranichnye-pwa-veb-prilozheniya> (дата обращения 22.06.2021).

2. React или Angular или Vue.js — что выбрать [Электронный ресурс]. Режим доступа: <https://habr.com/ru/post/476312/> (дата обращения 22.06.2021).

3. Local Storage [Электронный ресурс]. Режим доступа: <https://learn.javascript.ru/localstorage> (дата обращения 10.07.2021).

4.JavaScript.ru [Электронный ресурс]. Режим доступа: [https://learn.javascript.ru/](%20https://learn.javascript.ru/)

5. Документация Django rest framevork [Электронный ресурс]. Режим доступа: <https://www.django-rest-framework.org/> (дата обращения 10.07.2021).

6.Документация Django [Электронный ресурс]. Режим доступа: <https://www.djangoproject.com/>(дата обращения 10.07.2021).

7.Документация React [Электронный ресурс]. Режим доступа: <https://ru.reactjs.org/docs/getting-started.html>(дата обращения 10.07.2021).

8.Документация React-hook-form [Электронный ресурс]. Режим доступа: [https://react-hook-form.com/](https://react-hook-form.com/%20) (дата обращения 10.07.2021).

9.Документация React-router [Электронный ресурс]. Режим доступа: [https://reactrouter.com/](https://reactrouter.com/%20) (дата обращения 10.07.2021).