ПЕТРОЗАВОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра физики твердого тела

WEB-ориентированное программирование

Лабораторная работа (на стадии тестирования)

«Создание простого одностраничного сайта на Django»

Петрозаводск, 2020

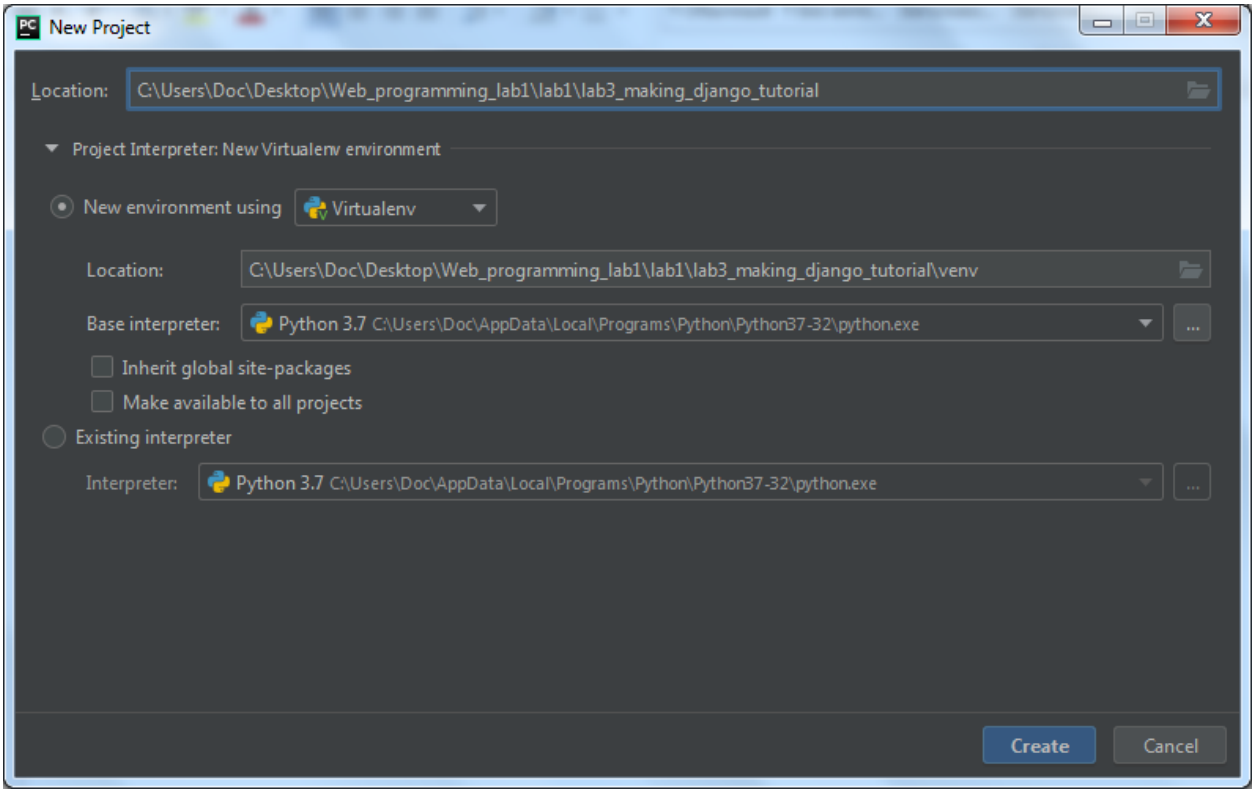
Оглавление

Петрозаводск, 2020

1. **Создание проекта и установка необходимых зависимостей**

Запустите среду разработки PyCharm, выберите пункт «Create new project», укажите папку проекта и установите следующие настройки:

- **Использование виртуального окружения virtualenv (venv).**
  Представьте, что у вас два проекта. Для одного требуется Django версии 3.0, а для другого – 1.10. Установленной можно иметь только одну версию пакета. Пришлось бы каждый раз при переключении между этими проектами удалять одну версию пакета и устанавливать другую.
  Виртуальное окружение позволяет устанавливать все пакеты для каждого проекта отдельно.
  Если по какой-то причине, вы не можете подключить виртуальное окружение, вы можете либо установить venv и заново создать проект, либо продолжить без него. В таком случае все пакеты будут установлены глобально.
- **Выбор интерпретатора.**
  Убедитесь, что у вас выбран интерпретатор для нового проекта. В графе «Base interpreter» должен быть прописан путь до Python 3.7



После того как проект будет создан, вам необходимо установить Django. Сделать это можно двумя способами:

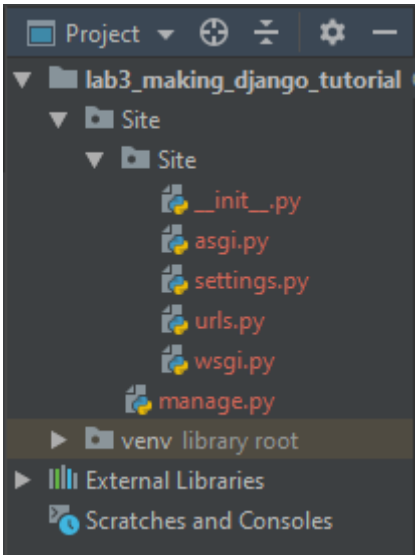
- Либо откройте терминал в PyCharm (с помощью сочетания клавиш Alt+F12, либо View→Tool Windows→Terminal) и введите команду pip install django
- Либо выберите элемент меню File→Settings→Project:<имя проекта>→Project Interpreter→нажмите на кнопку «+», введите в строке поиска Django и нажмите «Install Package»

В процессе установки также можно указать версию фреймворка (на данный момент, последняя стабильная версия – 3.0.3)

После того как всё будет установлено, все лишние окна можно закрыть.
Откройте терминал и введите следующую команду:

```
django-admin startproject Site
```

Данная команда создаст новый django проект с именем “Site” и добавит в директорию Site некоторые файлы. После выполнения этой команды, дерево проекта будет выглядеть следующим образом:



- **Внешний каталог Site/** – это просто контейнер для вашего проекта. Его название никак не используется Django, и вы можете переименовать его во что угодно.
- **manage.py**: Скрипт, который позволяет вам взаимодействовать с проектом Django.
- **Внутренний каталог Site/** - это пакет Python вашего проекта. Его название – это название пакета Python, которое вы будете использовать для импорта чего-либо из проекта (например, mysite.urls).
- **Site/__init__.py:** Пустой файл, который указывает Python, что текущий каталог является пакетом Python. (Читайте о пакетах в официальной документации Python, если вы новичок в Python.)
- **Site/settings.py:** Настройки/конфигурация проекта.
- **Site/urls.py:** Конфигурация URL-ов для вашего проекта Django. Это “содержание” всех Django-сайтов.
- **Site/wsgi.py:** Точка входа вашего проекта для WSGI-совместимых веб-серверов.

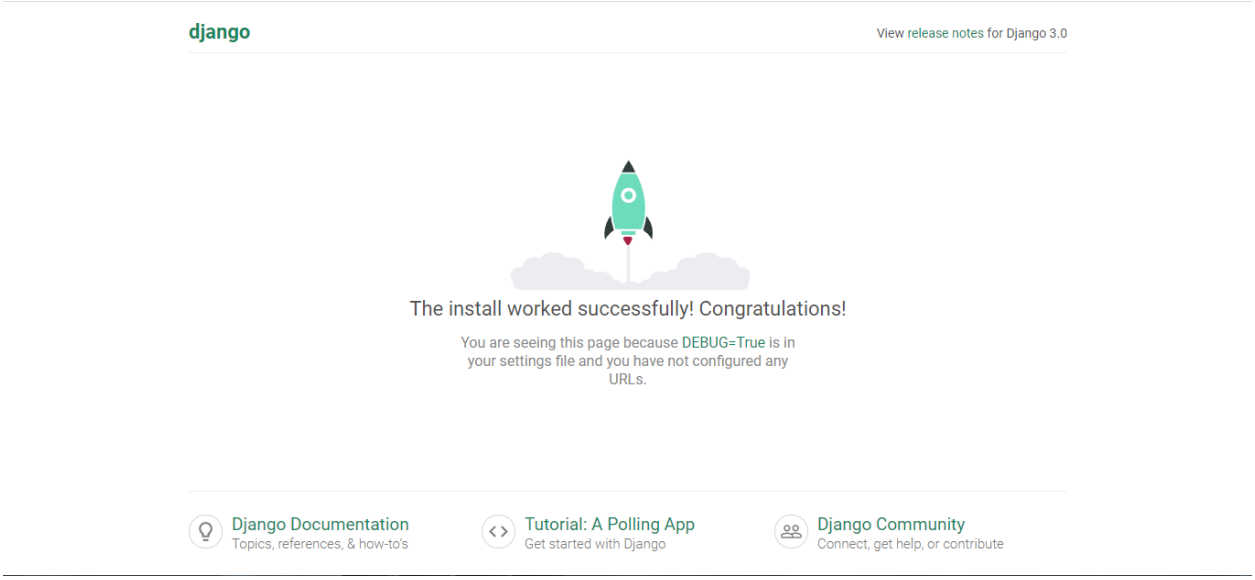
Теперь вы можете проверить работоспособность сервера. Перейдите в директорию Site с помощью терминала (команда cd). Вы должны оказаться в одной директории с файлом manage.py, после чего выполните команду:

```
python manage.py runserver
```

В дальнейшем, когда вы встретите фразу «выполните команду python manage.py <команда>», будет подразумеваться, что вы выполняете эту команду в

терминале, находясь в одной папке с файлом manage.py (содержимое текущей папки можно проверить с помощью команды dir). Обратите внимание, что если вы ранее создали виртуальное окружение, то оно должно быть активировано, чтобы все выполняемые команды работали корректно (если виртуальное окружение активировано, то в терминале перед текущим путем будет написано (venv))

После этого, откройте в браузере страницу с адресом http://127.0.0.1:8000/ и убедитесь, что всё работает:



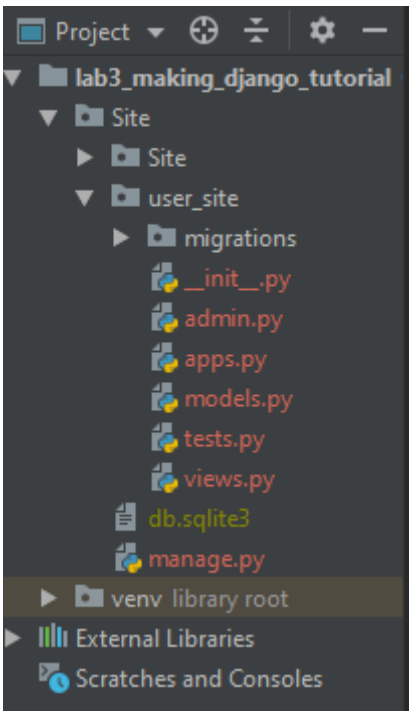
Теперь можно создать своё приложение Django.

Каждое приложение Django состоит из пакета Python, который следует некоторым соглашениям. Django содержит команду, которая создает структуру для нового приложения, что позволяет вам сосредоточиться на написании кода, а не на создании каталогов.
Какая разница между приложением и проектом? Приложение – это Web-приложение, которое предоставляет определенный функционал – например, Web-блог, хранилище каких-то записей или простое приложение для голосования. Проект – это совокупность приложений и конфигурации сайта. Проект может содержать несколько приложений. Приложение может использоваться несколькими проектами.

Выполните следующую команду, чтобы создать приложение user_site:

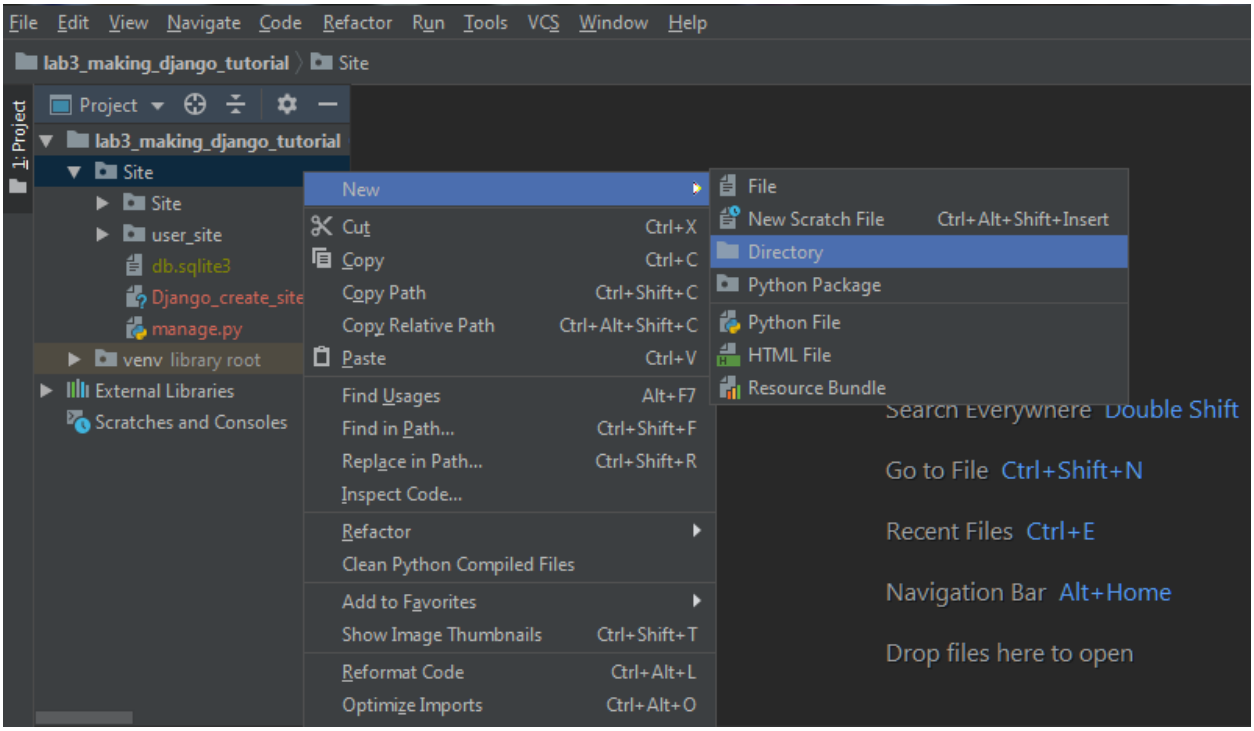
```
python manage.py startapp user_site
```

Данная команда создаст каталог user_site со следующим содержимым:

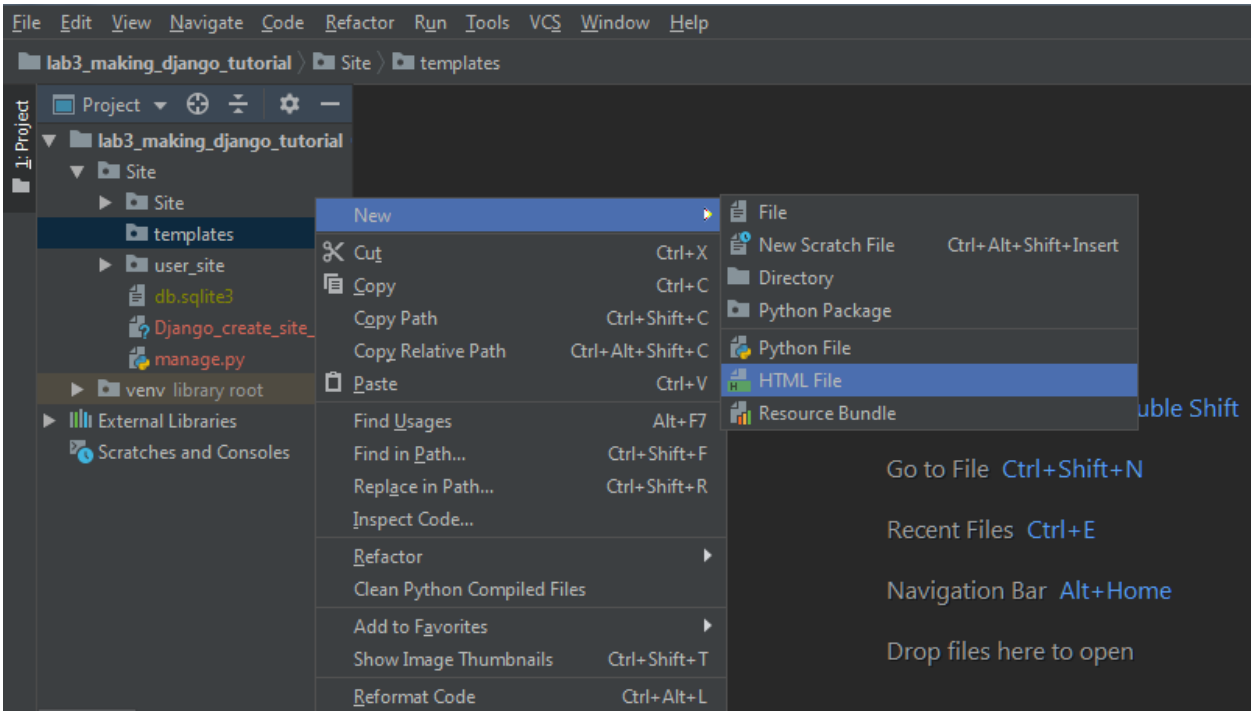


2. Создание web-страницы в приложении user_site

В одном каталоге с файлом manage.py (внешний каталог Site) создайте директорию шаблонов сайта “templates”:



Затем, в каталоге templates создайте файл index.html:



Далее, в этот файл добавьте следующий html-код:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Say hello</title>
</head>
<body>
    <h1>Hello, world!</h1>
</body>
</html>
```

После того как web-страница была создана, необходимо настроить её отображение. Для этого выполните следующие действия:

1) Откройте файл с настройками сайта “settings.py”. Этот файл расположен в каталоге “Site”, найдите там список конфигурации шаблонов «TEMPLATES» и добавьте к ключу “DIRS” следующее значение:

```
BASE_DIR, 'templates'
```

После этого, данный список будет выглядеть так:

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR, 'templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

Данная настройка указывает путь к каталогу с шаблонами сайта. Теперь в дальнейшем, при указании имени шаблона «index.html», который мы создали ранее, Django будет искать его в каталоге templates.

Эта настройка указывается только один раз. В дальнейшем ничего добавлять не придется.

2) Также, в файле настроек необходимо зарегистрировать наше приложение. Делается это путем добавления в список «INSTALLED_APPS» имени нашего приложения:

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'user_site',
]
```

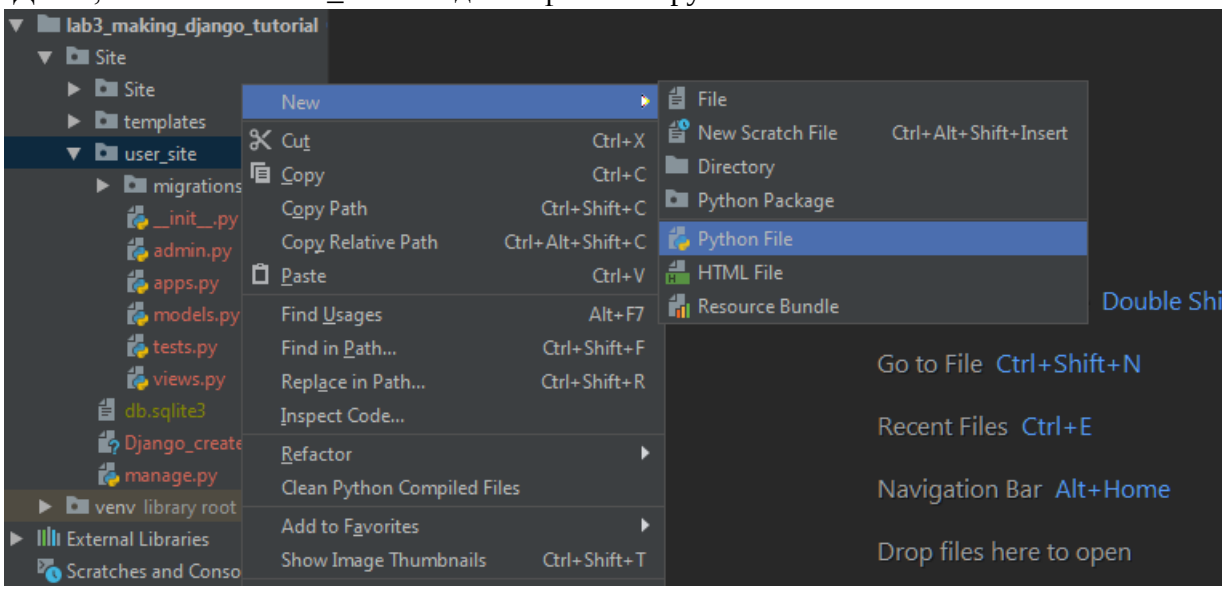
Эта настройка указывается только один раз. В дальнейшем ничего добавлять не придется.

3) Теперь необходимо подключить пути urls.py нашего приложения к путям urls.py проекта. Откройте файл “urls.py”, расположенный в каталоге Site. Добавьте в него строки, как показано на ниже:

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('user_site.urls')),
]
```

Эта настройка указывается только один раз. В дальнейшем ничего добавлять не придется.

4) Далее, в каталоге «user_site» создайте файл urls.py:



В этом файле будут прописаны пути ко всем представлениям (view) приложения user_site. Таким образом будет осуществляться маршрутизация различных страниц сайта.

В этот файл нужно добавить следующий код:

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
]
```

Таким образом, мы сопоставили путь «» (пустой путь, который соответствует имени хоста. Например, http://127.0.0.1:8000/) с функцией index, расположенной в файле views.py (пока что мы её не создали)

5) Теперь время создать представление (view). Откройте файл views.py приложения user_site и добавьте туда следующие строки:

```
from django.shortcuts import render


def index(request):
    return render(request, 'index.html')
```

Теперь вы можете запустить сервер (командой python manage.py runserver) и, открыть URL http://127.0.0.1:8000/ и убедиться, что выводится то, что вы написали в файле index.html.

Краткое описание работы фрейморка для тех, кто не понял, что он только что сделал:

1) Как только мы запускаем сервер (python manage.py runserver), Django считывает файл настроек проекта. Там, среди прочего, он видит наше зарегистрированное приложение, перечисленное в списке INSTALLED_APPS. Также, он считывает список TEMPLATES, в котором мы указали путь до директории с шаблонами проекта. Кроме этого, в данном файле прописывается корневая маршрутизация запросов:

```
ROOT_URLCONF = 'Site.urls'
```

Эта настройка означает, что все запросы, приходящие на сервер, будут проверяться на соответствие того, что указано в файле Site/urls.py.

2) Когда пользователь заходит на наш сайт, Django начинает проверять соответствие пути, по которому перешел пользователь. В нашем случае, путь был пустым (указан только адрес хоста и порт: http://127.0.0.1:8000/). Как уже было сказано выше, согласно настройке ROOT_URLCONF, Django ищет соответствия в файле Site/urls.py. Там он встречает следующую строчку:

```
path('', include('user_site.urls')),
```

Таким образом, если указанный путь пуст, Django подключит пути нашего приложения (user_site).

3) Далее, фреймфорк ищет соответствие уже в путях приложения user_site, то есть в файле user_site/urls.py. Встретив строку, соответствующую пустому пути, Django начнет выполнять код представления (view). В качестве этого представления мы указали функцию index:

```
path('', views.index, name='index'),
```

4) Так как общение между клиентом и сервером осуществляется посредством запросов, для того чтобы получить информацию от сервера, браузер передал самый простой GET-запрос (структуру запроса можно посмотреть в браузере, нажав клавишу F12). Сервер обработал его и создал объект request, который передал в функцию index. Этот объект содержит тело запроса (пустое, в нашем случае) и заголовки. Далее, всё, что нужно сделать – вызвать функцию render, в которую мы передадим имя шаблона и объект request.

5) Функция render, в соответствие с указанной директорией шаблона (TEMPLATES) и именем шаблона (index.html), создаст ответ (response), который передаст на клиентское устройство.

3. Создание простой формы регистрации заявок и страницы их просмотра

Если собрать код «по шагам» окажется для вас непосильной задачей, в конце есть готовый вариант страницы index.html.

Для создания формы с адаптивным дизайном воспользуемся готовыми стилями Bootstrap (https://getbootstrap.com/)

Давайте в словесной форме опишем архитектуру сайта в том виде, в котором он будет представлен пользователю:

1) Когда пользователь заходит на сайт, он видит список всех заявок на конференцию. Если заявок еще нет, на сайте будет представлена элемент-заглушка, приветствующая пользователя.
2) При нажатии кнопки «Подать заявку» открывается модальное окно с формой регистрации.
3) При нажатии на кнопку подтверждения, заявка отправляется на сервер, страница перезагружается, и пользователь видит свою заявку среди остальных.

Теперь приступим к разработке дизайна web-страницы. Сначала в файле index.html подключим bootstrap стили и скрипты:

```
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap
.min.css" integrity="sha384-
Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
crossorigin="anonymous">
<script src="https://code.jquery.com/jquery-3.4.1.slim.min.js"
integrity="sha384-
J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n"
crossorigin="anonymous"></script>
<script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min
.js" integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
crossorigin="anonymous"></script>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.m
in.js" integrity="sha384-
wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6"
crossorigin="anonymous"></script>
```

Данный код лучше вставить перед кодом заголовка страницы (head)

Затем, текущий код внутри тега body заменим кодом модального окна с формой регистрации заявки:

```
<div id="myModalBox" class="modal fade">
  <div class="modal-dialog">
    <div class="modal-content">
      <!-- Заголовок модального окна -->
      <div class="modal-header">
        <h4 class="modal-title">Подать заявку на конференцию</h4>
        <button type="button" class="close" data-dismiss="modal" aria-
hidden="true" style="left:10px;">×</button>
      </div>
      <!-- Основное содержимое модального окна -->
      <div class="modal-body">
        <form method="POST">
            {% csrf_token %}
            <div class="form-group">
                <label for="userName">ФИО участника</label>
                <input type="text" class="form-control" id="userName"
name="userName">
            </div>

            <div class="form-group">
                <label for="InputEmail">Email адрес</label>
                <input type="email" class="form-control"
```

```
id="InputEmail" name="InputEmail" aria-describedby="emailHelp">
            </div>

            <div class="form-group">
                <label for="phonenumber">Телефон</label>
                <input type="text" class="form-control"
id="phonenumber" name="phonenumber">
            </div>

            <div class="form-group">
                <label for="university">Полное наименование
ВУЗа</label>
                <input type="text" class="form-control"
id="university" name="university">
            </div>

            <div class="form-group">
                <label for="sciAdviser">ФИО научного
руководителя</label>
                <input type="text" class="form-control"
id="sciAdviser" name="sciAdviser">
            </div>

            <div class="form-group">
                <label for="article">Название статьи</label>
                <input type="text" class="form-control" id="article"
name="article">
            </div>
<div class="form-group">
    <label for="FormControlTextarea1">Описание статьи</label>
    <textarea class="form-control" id="FormControlTextarea1"
name="FormControlTextarea1" rows="3"></textarea>
</div>


            <button type="submit" class="btn btn-
primary">Отправить</button>
        </form>
      </div>
    </div>
  </div>
</div>
```

В коде, представленном выше, есть одна вещь, которая не относится к HTML, CSS или JS. Строка «{% csrf_token %}» - это код для шаблонизатора Django (к которому мы вернемся позже). Сейчас давайте разберемся, для чего нужен CSRF-токен.

«CSRF (англ. Cross Site Request Forgery — «межсайтовая подделка запроса», также известна как XSRF) — вид атак на посетителей веб-сайтов, использующий недостатки протокола HTTP. Если жертва заходит на сайт, созданный злоумышленником, от её лица тайно отправляется запрос на другой сервер (например, на сервер платёжной системы), осуществляющий некую вредоносную операцию» [ru.wikipedia.org/wiki/Межсайтовая_подделка_запроса].

«Эффективным и общепринятым на сегодня способом защиты от CSRF-Атаки является токен. Под токеном имеется в виду случайный набор байт, который сервер передает клиенту, а клиент возвращает серверу.

Защита сводится к проверке токена, который сгенерировал сервер, и токена, который прислал пользователь» [https://habr.com/ru/post/318748/].

Таким образом, при каждой отправке формы на сервер, нам нужно передавать серверу CSRF-токен. Django позволяет сильно упростить взаимодействие с этим механизмом, просто добавив одну строку в код html-страницы.

Теперь давайте создадим кнопку-ссылку, при нажатии на которую, будет появляться модальное окно:

```
<a class="btn btn-primary btn-lg" href="#" role="button"
onclick="$('#myModalBox').modal('show');">Подать заявку</a>
```

Создадим «заглушку» и добавим туда кнопку:

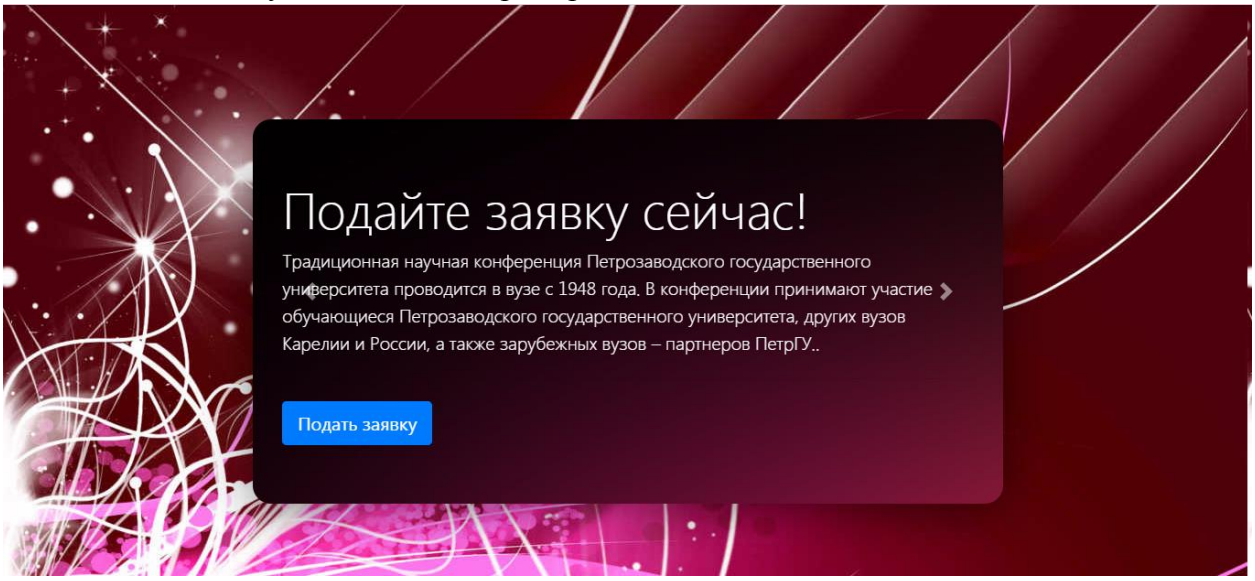
```
<div id="carouselExampleControls" class="carousel slide card1" data-
ride="carousel">
  <div class="carousel-inner">
    <div class="carousel-item active">
        <div class="jumbotron card2">
            <h1 class="display-4">Подайте заявку сейчас!</h1>
            <p class="lead">Традиционная научная конференция
Петрозаводского государственного университета проводится в вузе с 1948
года.
                В конференции принимают участие обучающиеся
                Петрозаводского государственного университета, других
вузов Карелии и России, а также зарубежных вузов – партнеров
ПетрГУ..</p>
            <hr class="my-4">
            <a class="btn btn-primary btn-lg" href="#" role="button"
onclick="$('#myModalBox').modal('show');">Подать заявку</a>
        </div>
    </div>
  </div>
  <a class="carousel-control-prev" href="#carouselExampleControls"
role="button" data-slide="prev">
    <span class="carousel-control-prev-icon" aria-
hidden="true"></span>
    <span class="sr-only">Previous</span>
  </a>
  <a class="carousel-control-next" href="#carouselExampleControls"
role="button" data-slide="next">
    <span class="carousel-control-next-icon" aria-
hidden="true"></span>
    <span class="sr-only">Next</span>
  </a>
</div>
```

Теперь добавим немного стилей, чтобы всё отображалось более-менее красиво:

```
<style>
body{
    background:
url("https://storge.pic2.me/c/1360x800/889/55532b845c7b6.jpg");
}
.card1{
  width:60%;
  height: 60%; /* если высота не будет задана явна, она будет равна
100% */
  position: absolute;
```

```
  top: 0;
  bottom: 0;
  left: 0;
  right: 0;
  margin: auto;
  border-radius: 20px;
  box-shadow: 0 50px 50px rgba(0,0,0,0.5); /* Параметры тени */
}
.card2{
    border-radius: 20px;
    background: url("https://w-
dog.ru/wallpapers/15/15/330935819042751/razmytie-fon-rozovyj-
oranzhevyj-svet.jpg");
    color: #FFF;
    width:100%;
    height:100%;
}
</style>
```

После этого, сайт будет выглядеть примерно так:



Если всё работает, можно переходить к следующему пункту.

4. Обработка данных на сервере, подключение базы данных, использование моделей.

Как уже говорилось ранее, данные, которые пришли на сервер, Django упаковывает в объект request и передает в нашу функцию представления (index). Теперь, обратившись к этому объекту, мы можем получить все данные формы. Давайте попробуем вывести значения полей в консоль.

```
def index(request):
    if request.method == 'POST':
        userName = request.POST['userName']
        InputEmail = request.POST['InputEmail']
        phonenumber = request.POST['phonenumber']
        university = request.POST['university']
        sciAdviser = request.POST['sciAdviser']
```

```
        article = request.POST['article']
        FormControlTextarea1 = request.POST['FormControlTextarea1']

        print(userName,InputEmail,phonenumber,
              university,sciAdviser,article,
              FormControlTextarea1)
    return render(request, 'index.html')
```

Сохраните изменения и перезапустите сервер.
Обратите внимание, что если в настройках проекта включен режим «Debug», то при сохранении изменений сервер перезапустится автоматически.

Теперь нажмите на кнопку «Подать заявку», введите данные в поля и убедитесь, что результат вывелся в консоль PyCharm.

Далее, нам нужно задуматься над тем, как хранить эти данные в базе данных. К счастью, при установке Django вы установили и базу данных SQLite3, поэтому никаких дополнительных настроек делать не требуется. Вам только нужно будет создать модель и сделать миграцию (применить изменения к базе данных).

«Веб-приложения Django получают доступ и управляют данными через объекты Python, называемые моделями. Модели определяют структуру хранимых данных, включая типы полей и, возможно, их максимальный размер, значения по умолчанию, параметры списка выбора, текст справки для документации, текст меток для форм и т. д. Определение модели не зависит от основной базы данных - вы можете выбрать один из нескольких компонентов вашей настройки проекта. После того, как вы выбрали какую базу данных хотите использовать, вам не нужно напрямую работать с ней - вы просто пишете свою структуру модели и код, а Django делает всю грязную работу, связанную с базой данных за вас» [https://developer.mozilla.org/ru/docs/Learn/Server-side/Django/Models].

Давайте создадим модель «Article» в файле user_site/models.py:

```
from django.db import models


# Create your models here.
class Article(models.Model):
    username = models.CharField(max_length=50)
    email = models.EmailField()
    phonenumber = models.CharField(max_length=12)
    university = models.CharField(max_length=100)
    adviser = models.CharField(max_length=50)
    article = models.CharField(max_length=50)
    about = models.TextField()
```

Теперь применим изменения. Для этого, остановите сервер, если он запущен (перейдите к окну терминала и нажмите Ctrl+C) и введите в терминал **поочередно** следующие команды:

```
python manage.py makemigrations
python manage.py migrate
```

Таким образом, вы создадите в базе данных таблицу, соответствующую вашей модели.

После этого, можно записать в базу данных переданные значения. Для этого, в начале файла views.py импортируйте созданную модель:

```
from .models import Article
```

Затем, в функции index, вместо вывода значений (print) добавьте следующие строки:

```
new_article = Article(username=userName, email=InputEmail,
                      phonenumber=phonenumber,
                      university=university,
                      adviser=sciAdviser, article=article,
                      about=FormControlTextarea1)
new_article.save()
```

Не забудьте правильно расставить отступы, иначе ничего не будет работать.

Снова запустите сервер, введите данные в форму и убедитесь, что ошибок нет.

5. Вывод значений из базы данных на форму. Шаблонизатор Django.

Для того, чтобы выбрать значений из базы данных, измените код функции index следующим образом:

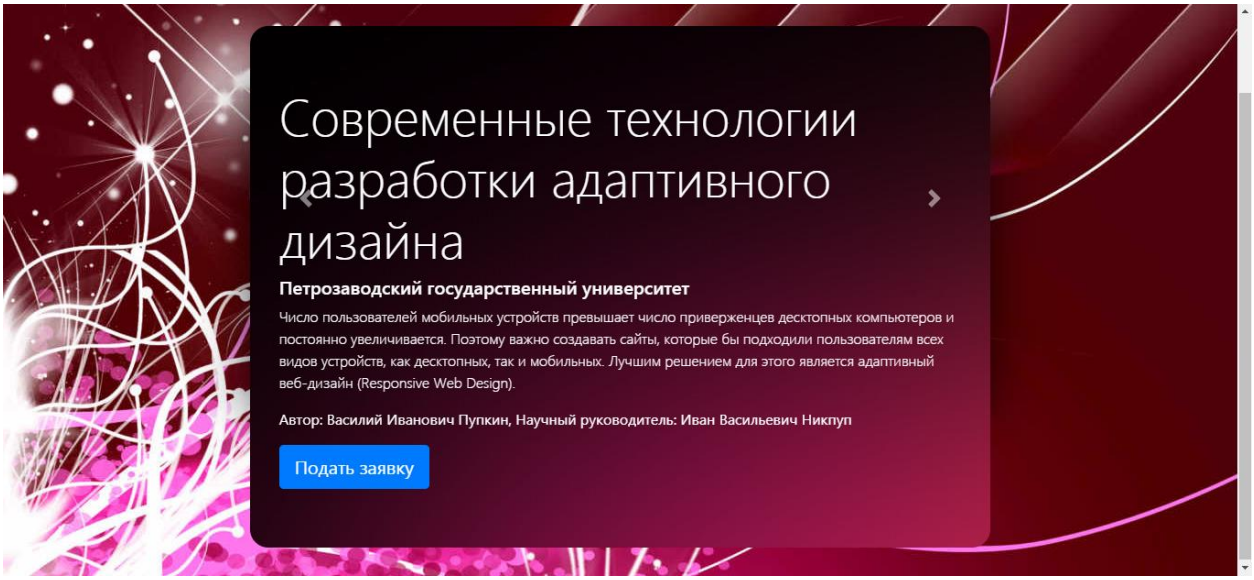
```
def index(request):
    if request.method == 'POST':
        userName = request.POST['userName']
        InputEmail = request.POST['InputEmail']
        phonenumber = request.POST['phonenumber']
        university = request.POST['university']
        sciAdviser = request.POST['sciAdviser']
        article = request.POST['article']
        FormControlTextarea1 = request.POST['FormControlTextarea1']

        new_article = Article(username=userName, email=InputEmail,
                              phonenumber=phonenumber,
                              university=university,
                              adviser=sciAdviser, article=article,
                              about=FormControlTextarea1)
        new_article.save()
    articles = Article.objects.all()
    active_article = articles.first()
    return render(request, 'index.html', {'articles': articles,
'active_article': active_article})
```

Теперь, в файле index.html нужно обработать переданный список статей. Для этого воспользуемся шаблонизатором. Модифицируем вывод «заглушки». Давайте будем её показывать только тогда, когда в базе данных нет заявок, а в противном случае просто выводить эти статьи:

```
<div id="carouselExampleControls" class="carousel slide card1" data-
ride="carousel">
  <div class="carousel-inner">
```

```
      {% if not articles %}
    <div class="carousel-item active">
        <div class="jumbotron card2">
            <h1 class="display-4">Подайте заявку сейчас!</h1>
            <p class="lead">Традиционная научная конференция
Петрозаводского государственного университета проводится в вузе с 1948
года.
                В конференции принимают участие обучающиеся
                Петрозаводского государственного университета, других
вузов Карелии и России, а также зарубежных вузов – партнеров
ПетрГУ..</p>
            <hr class="my-4">
            <p>Авторы</p>
            <a class="btn btn-primary btn-lg" href="#" role="button"
onclick="$('#myModalBox').modal('show');">Подать заявку</a>
        </div>
    </div>
      {% else %}
            {% for article in articles %}
                <div class="carousel-item {% if article ==
active_article %} active {% endif %}">
                    <div class="jumbotron card2">
                        <h1 class="display-4">{{article.article}}</h1>
                        <h5>{{article.university}}</h5>
                        <p class="lead" style="font-
size:1em;">{{article.about}}</p>
                        <p>Автор: {{article.username}}, Научный
руководитель: {{article.adviser}}</p>
                        <a class="btn btn-primary btn-lg" href="#"
role="button" onclick="$('#myModalBox').modal('show');">Подать
заявку</a>
                    </div>
                </div>
            {% endfor %}
      {% endif %}
  </div>
  <a class="carousel-control-prev" href="#carouselExampleControls"
role="button" data-slide="prev">
    <span class="carousel-control-prev-icon" aria-
hidden="true"></span>
    <span class="sr-only">Previous</span>
  </a>
  <a class="carousel-control-next" href="#carouselExampleControls"
role="button" data-slide="next">
    <span class="carousel-control-next-icon" aria-
hidden="true"></span>
    <span class="sr-only">Next</span>
  </a>
</div>
```

В итоге, у вас должно получиться что-то наподобие этого:



В качестве шаблонизатора вы также можете использовать Jinja2.

6. Подключение админ-панели.

Чтобы полноценно управлять сайтом, вы можете подключить «админку». В ней можно будет создавать, удалять или редактировать заявки. Чтобы это сделать, перейдите в файл user_site/admin.py и добавьте туда следующий код:

```
from django.contrib import admin
from .models import Article

admin.site.register(Article)
```

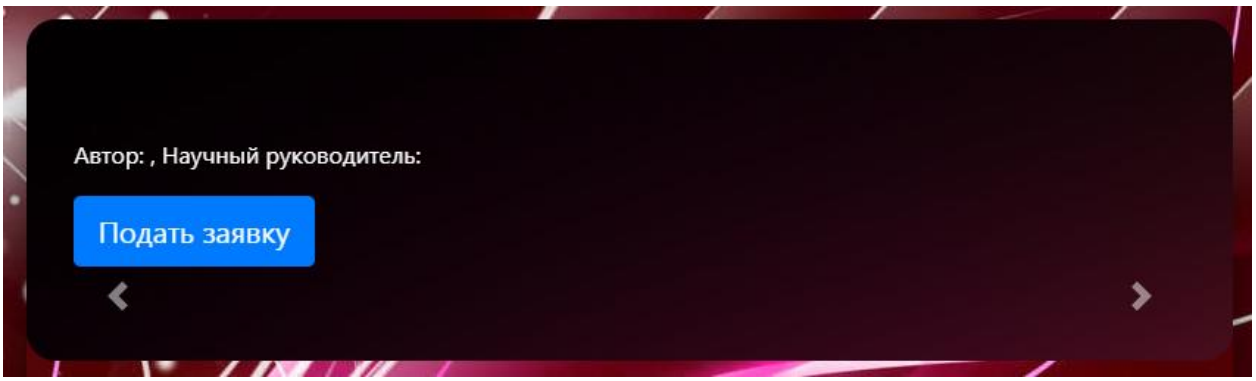
Затем, создайте учетную запись администратора. Для этого выполните команду

```
python manage.py createsuperuser
```

После чего, введите логин, email и пароль. Теперь вы можете снова запустить сервер и перейти по адресу http://127.0.0.1:8000/admin чтобы зайти в «админку».

7. Дополнительно.

В процессе создания сайта мы не учли некоторые детали. Например, если не вводить данные в форму и нажать кнопку «Отправить», то вы увидите следующую картину:



Вам предлагается самостоятельно исправить все «баги», согласно следующему списку:

- Фоновое изображение (тег body) нужно «растягивать» на весь экран, чтобы оно не повторялось, когда размеры просматриваемой области увеличиваются
- Данные формы нужно проверять перед отправкой на сервер. Сделайте все поля обязательными для заполнения и ограничьте максимальную длину каждого из них.
- Когда в базе несколько сотен заявок, предложенный вариант просмотра может оказаться неудобным. Добавьте функцию вывода всех заявок в таблицу на отдельной странице.

8. Используемые версии интерпретатора и библиотек.

Интерпретатор:

python 3.7

Зависимости:

asgiref==3.2.3 Django==3.0.3 pytz==2019.3 sqlparse==0.3.0

Полный код файла index.html

```
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
crossorigin="anonymous">
<script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-
J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n"
crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
crossorigin="anonymous"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"
integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6"
crossorigin="anonymous"></script>

<head>
    <meta charset="UTF-8">
    <title>Say hello</title>
</head>
<style>
body{
    background: url("https://storge.pic2.me/c/1360x800/889/55532b845c7b6.jpg");
}
.card1{
  width:60%;
  height: 60%; /* если высота не будет задана явна, она будет равна 100% */
  position: absolute;
  top: 0;
  bottom: 0;
  left: 0;
  right: 0;
  margin: auto;
  border-radius: 20px;
  box-shadow: 0 50px 50px rgba(0,0,0,0.7); /* Параметры тени */
}
.card2{
    border-radius: 20px;
    background: url("https://w-dog.ru/wallpapers/15/15/330935819042751/razmytie-fon-
rozovyj-oranzhevyj-svet.jpg");
    color: #FFF;
    width:100%;
    height:100%;
}
</style>
<body>

<div id="carouselExampleControls" class="carousel slide card1" data-ride="carousel">
  <div class="carousel-inner">
      {% if not articles %}
    <div class="carousel-item active">
        <div class="jumbotron card2">
            <h1 class="display-4">Подайте заявку сейчас!</h1>
            <p class="lead">Традиционная научная конференция Петрозаводского
государственного университета проводится в вузе с 1948 года.
                В конференции принимают участие обучающиеся
                Петрозаводского государственного университета, других вузов Карелии и
России, а также зарубежных вузов – партнеров ПетрГУ..</p>
            <hr class="my-4">
            <p>Авторы</p>
            <a class="btn btn-primary btn-lg" href="#" role="button"
onclick="$('#myModalBox').modal('show');">Подать заявку</a>
        </div>
    </div>
      {% else %}
            {% for article in articles %}
                <div class="carousel-item {% if article == active_article %} active {%
endif %}">
                    <div class="jumbotron card2">
                        <h1 class="display-4">{{article.article}}</h1>
                        <h5>{{article.university}}</h5>
                        <p class="lead" style="font-size:1em;">{{article.about}}</p>
                        <p>Автор: {{article.username}}, Научный руководитель:
{{article.adviser}}</p>
                        <a class="btn btn-primary btn-lg" href="#" role="button"
onclick="$('#myModalBox').modal('show');">Подать заявку</a>
                    </div>
```

```
                </div>
            {% endfor %}
        {% endif %}
    </div>
    <a class="carousel-control-prev" href="#carouselExampleControls" role="button" data-
slide="prev">
      <span class="carousel-control-prev-icon" aria-hidden="true"></span>
      <span class="sr-only">Previous</span>
    </a>
    <a class="carousel-control-next" href="#carouselExampleControls" role="button" data-
slide="next">
      <span class="carousel-control-next-icon" aria-hidden="true"></span>
      <span class="sr-only">Next</span>
    </a>
</div>

    <!-- HTML-код модального окна -->
<div id="myModalBox" class="modal fade">
    <div class="modal-dialog">
      <div class="modal-content">
        <!-- Заголовок модального окна -->
        <div class="modal-header">
          <h4 class="modal-title">Подать заявку на конференцию</h4>
          <button type="button" class="close" data-dismiss="modal" aria-hidden="true"
style="left:10px;">×</button>
        </div>
        <!-- Основное содержимое модального окна -->
        <div class="modal-body">
          <form method="POST">
              {% csrf_token %}
              <div class="form-group">
                  <label for="userName">ФИО участника</label>
                  <input type="text" class="form-control" id="userName" name="userName">
              </div>

              <div class="form-group">
                  <label for="InputEmail">Email адрес</label>
                  <input type="email" class="form-control" id="InputEmail"
name="InputEmail" aria-describedby="emailHelp">
              </div>

              <div class="form-group">
                  <label for="phonenumber">Телефон</label>
                  <input type="text" class="form-control" id="phonenumber"
name="phonenumber">
              </div>

              <div class="form-group">
                  <label for="university">Полное наименование ВУЗа</label>
                  <input type="text" class="form-control" id="university"
name="university">
              </div>

              <div class="form-group">
                  <label for="sciAdviser">ФИО научного руководителя</label>
                  <input type="text" class="form-control" id="sciAdviser"
name="sciAdviser">
              </div>

              <div class="form-group">
                  <label for="article">Название статьи</label>
                  <input type="text" class="form-control" id="article" name="article">
              </div>
              <div class="form-group">
                  <label for="FormControlTextarea1">Описание статьи</label>
                  <textarea class="form-control" id="FormControlTextarea1"
name="FormControlTextarea1" rows="3"></textarea>
              </div>

              <button type="submit" class="btn btn-primary">Отправить</button>
          </form>
        </div>
      </div>
    </div>
  </div>
</div>

</body>
</html>
```