**Серверная часть**

* 1. настройка проекта

Первым делом нужно установить framework Django для написания моделей, представлений сайта. Для установки Django нужно прописать команду:

* pip install Django.

После успешной установки необходимо создать и настроить проект, выполнив следующие команды:

* Django-admin startproject KausticApi;
* cd KausticApi;
* py manage.py startapp articles.

Для проверки работы приложения создадим простое представление на главной страничке приложения, которая показана на рисунках 1 и 2.

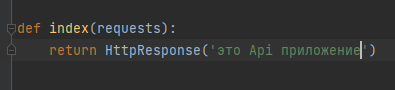
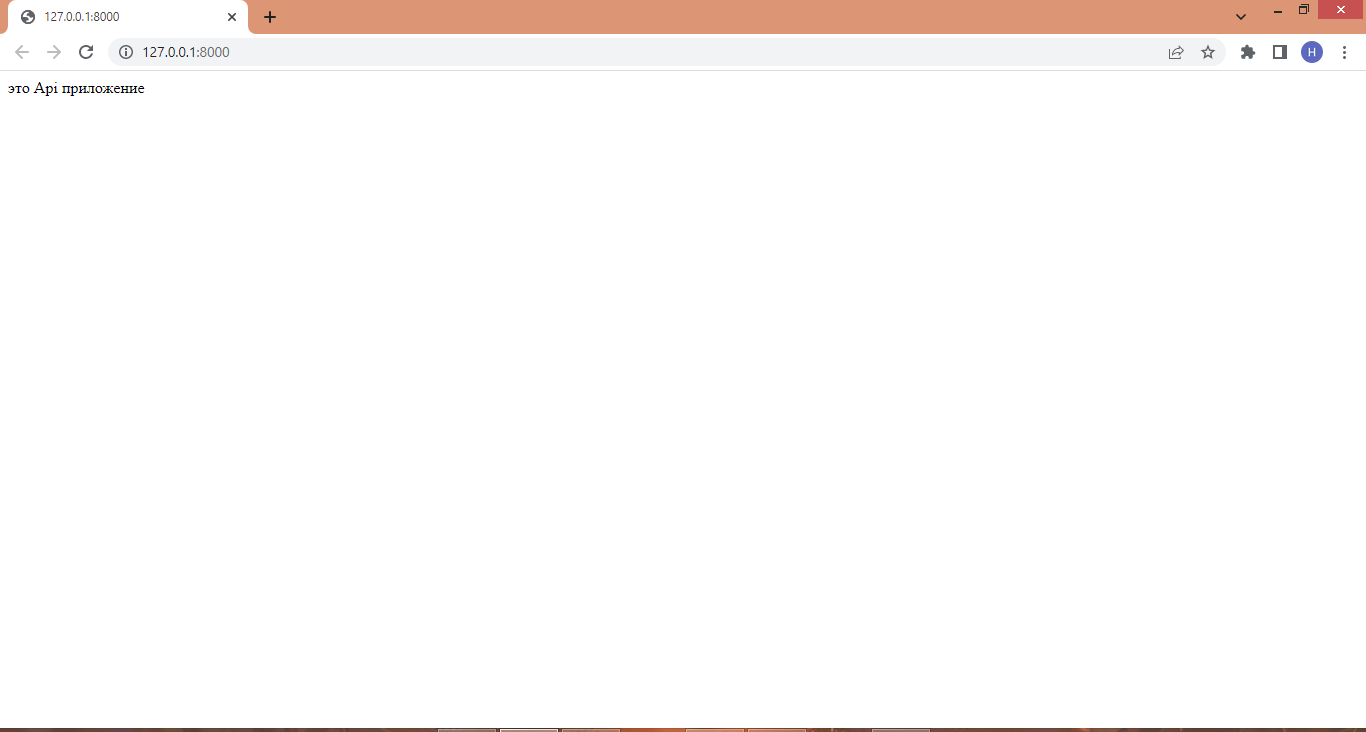


Рисунок 1 функция представления

Рисунок 2 вывод отображения в браузер

(сделать обрезание)

Проект работает. Следующим шагом нужно установить Django rest framework – это фреймворк для написания Rest Api приложений.

**API** (Application Programming Interface или интерфейс программирования приложений) — это совокупность инструментов и функций в виде интерфейса для создания новых приложений, благодаря которому одна программа будет взаимодействовать с другой. Данная API будет прослойкой между Back-end частью и Front-End частью сайта.

Для установки потребуется применить команду:

* Pip install Django rest framework.

Загрузить фрейморк в проект.

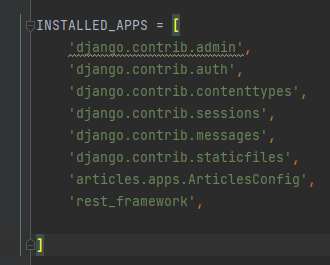


Рисунок 3 добавление drf в настройки проекта

Базовая первичная настройка проекта готова. Теперь можно приступить к написанию логики сайта.

* 1. Создание моделей проекта

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. Для выполнения задач проекта было создано 3 модели: Вакансии, Документы и статьи. Пока что модели- это просто представления данных, в них нельзя хранить информацию.

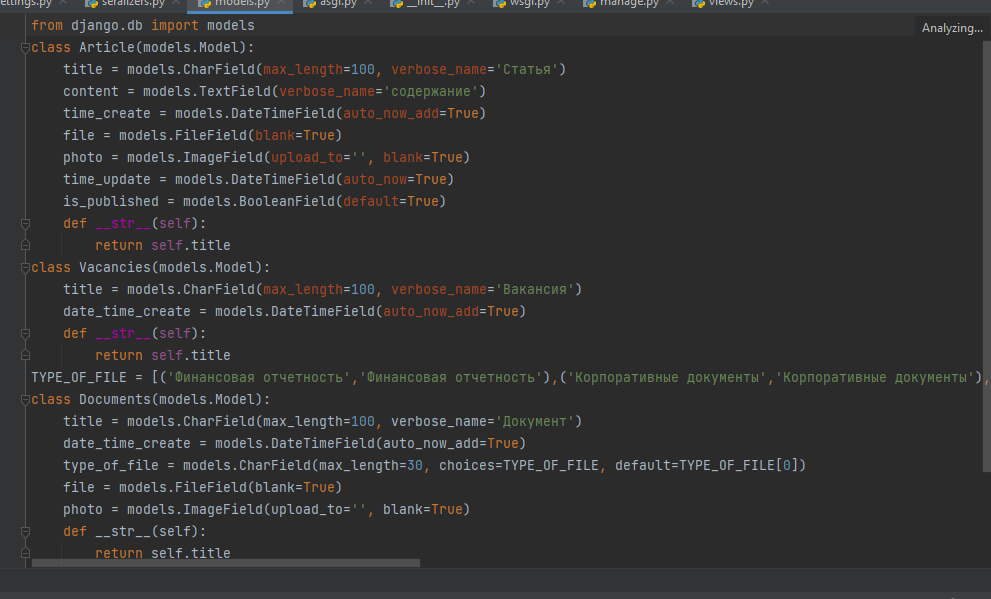


Рисунок 4 Модели проекта

(сделать таблицой)

Чтобы это исправить нужно прописать команды:

* “py manage.py makemigrations” ,
* “py manage.py migrate”.

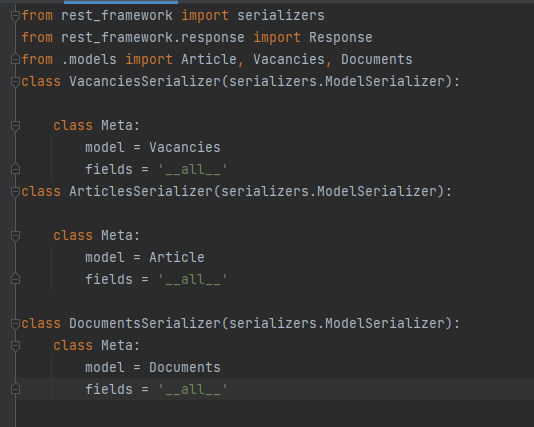
Это создаст таблицы в базе данных.

* 1. Написание сериализаторов

Модели Django интуитивно представляют данные, хранящиеся в базе, но API должен передавать информацию в менее сложной структуре. Хотя данные будут представлены как экземпляры классов Model, их необходимо перевести в формат JSON для передачи через API.

Сериализатор DRF производит это преобразование. Когда пользователь передает информацию (например, создание нового экземпляра) через API, сериализатор берет данные, проверяет их и преобразует в нечто, что Django может сложить в экземпляр модели. Аналогичным образом, когда пользователь обращается к информации через API, соответствующие экземпляры передаются в сериализатор, который преобразовывает их в формат, который может быть легко передан пользователю как JSON.

Папке приложения articles создаем файл seralizers.py. Классы сериализаторов представлен на рисунке 5.



(сделать обрезание)Рисунок 5 классы сериализаторов

Данным сериализаторам передается модель и нужные поля.

* 1. Написание представлений .

На рисунке 6 предоставленные классы представлений будущих страниц для работы с API.

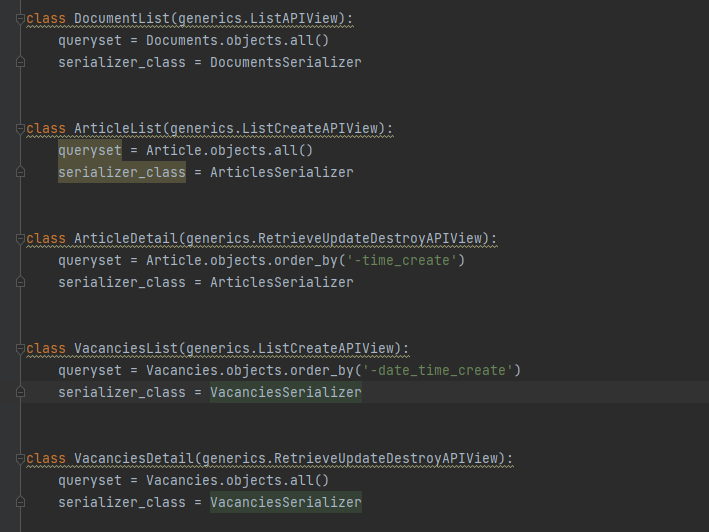
(заменить скрин)

Рисунок 6 представления API.

Распишем все классы по отдельности. DocumentList наследуется от класса ListAPIView. Благодаря принципам OOP класс DocumentList перенял весь функционал ListAPIView, что облегчает разработку. Данный класс отвечает за предоставление записей из БД. В queryset запрашиваем все Документы и указываем сериалиатор, который был создан ранее. Классы ArticleList и Vacancies работают по тому же принципу, но также имеют возможность создания записей. Остальные классы ArticleDetail и VacanciesDetail созданы для работы с конкретной, отдельной записью. Их queryset запрашивает все объекты отсортированные по дате добавления, чтобы на сайте с начала выводились актуальные записи.

* 1. Написание маршутов.

Для работы API на фронтовой части нужна ссылка, которая будет взаимодействовать с front-end частью сайта. В конфигурационном файле проекта есть файл urls.py, там и будем прописывать маршуты. Маршутизация API на рисунке 7.



(артиклисит заменить)

Рисунок 7 маршутизация API.

* 1. Django-cors-headers.

Наша API и сайт будут лежать на разных портах. Для корректной работы необходимо установить модуль Django-cors-headers, следующей командой:

* pip install django-cors-headers

Добавить его в Проект, как это показано на рисунке 8



Рисунок 8 добавление модуля в проект.

Так же следует добавить следующие настройки в settings.py

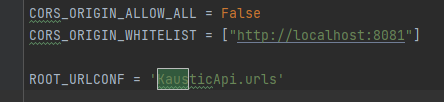


Рисунок 9 настройка модуля в проекте (обрезание)

CORS\_ORIGIN\_ALLOW\_ALL = False запрещает использовать API на других портах. CORS\_ORIGIN\_WHITELIST задает белый список, где можно применять наш Api.

* 1. Тестирование

Для примера перейдем по ссылке “http://127.0.0.1:8000/api/v1/vacancii/4”/.

На данной странице детальное отображение вакансии с id равное 4.

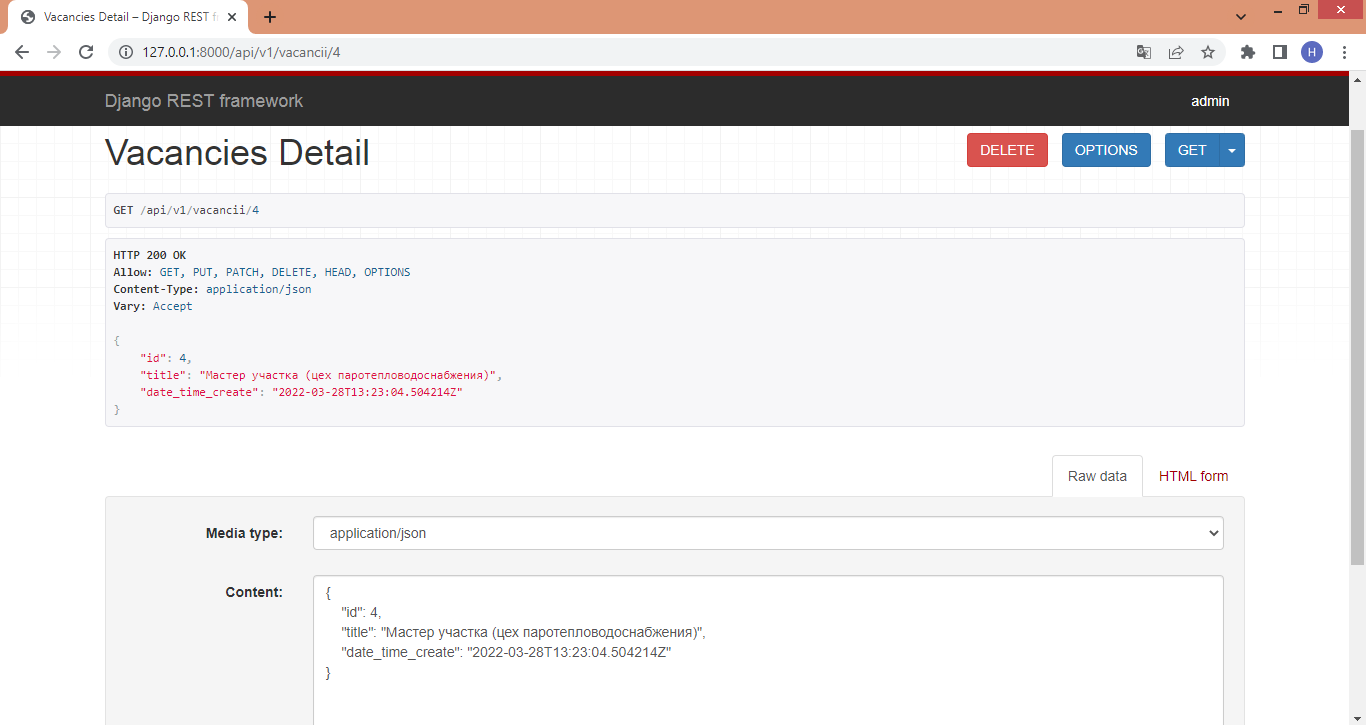


Рисунок 10 вывод отдельной записи.

Для проверки работоспособности заменим название вакансии на Мастер цеха.

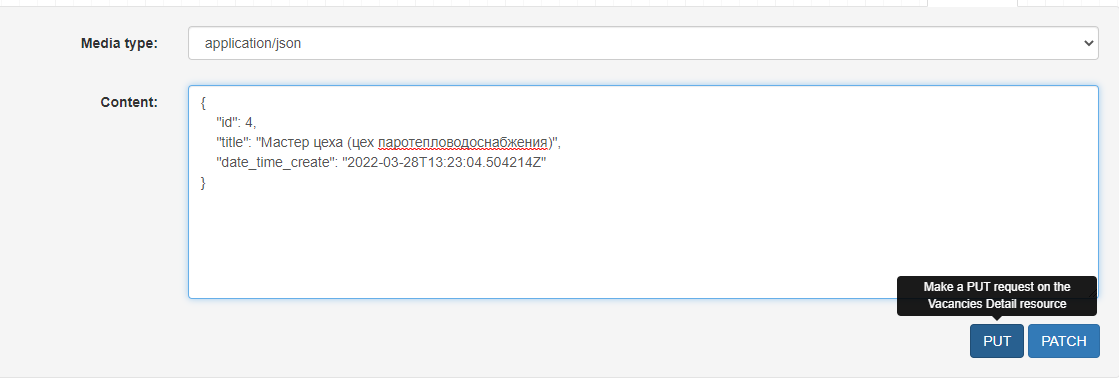


Рисунок 11 изменение записи.

Тестирование прошло успешно. Запись изменилась.

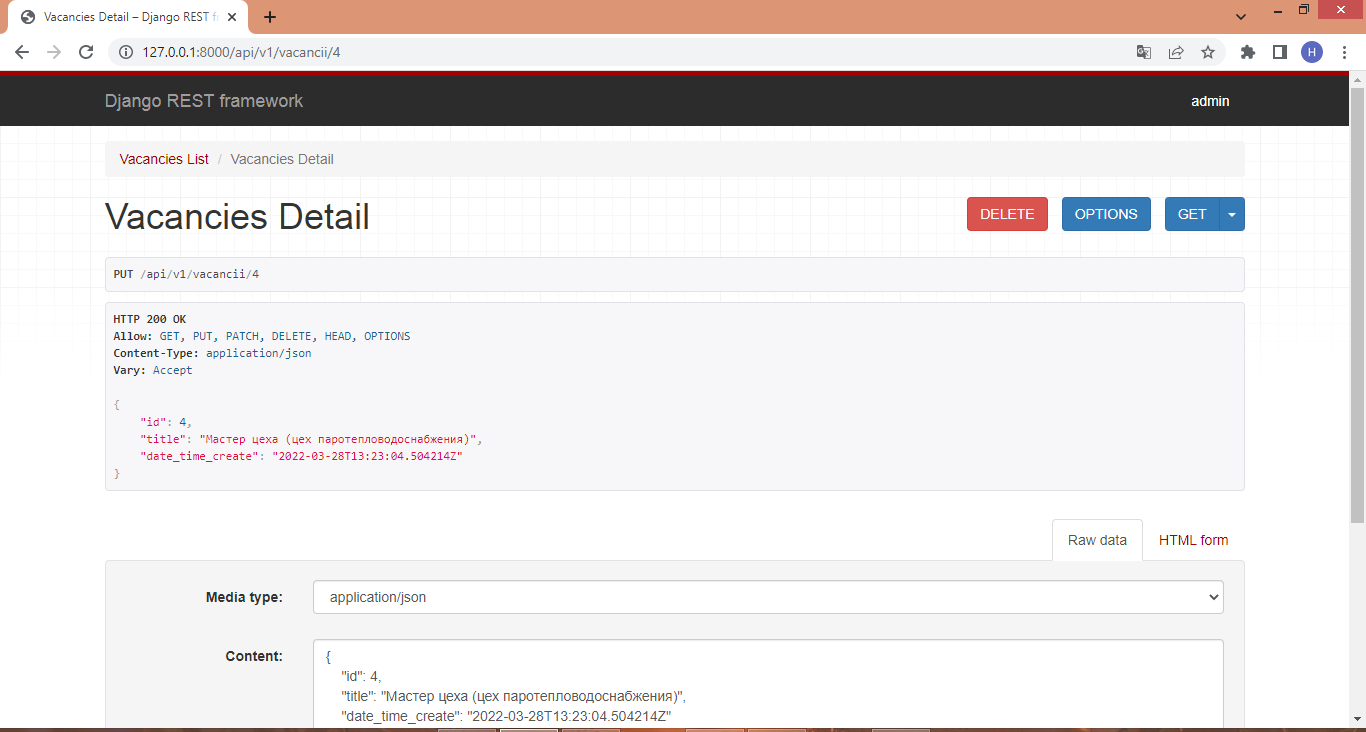


Рисунок 12 детальное отображение отредактированного элемента. (нормально поставить)