Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский политехнический университет»

Кафедра «Инфокогнитивные технологии» Образовательная программа «Веб-технологии»

Отчет по курсовому проекту

по дисциплине «Инженерное проектирование»

Тема: «Изучение фреймворка Django и дополнительных веб-технологий»

Выполнил:

Студент группы 191-321

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Громова И.П.

подпись, дата

Принял:

Старший преподаватель

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Даньшина М.В.

подпись, дата

​

Москва 2020

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc76739217)

[ОСНОВНАЯ ЧАСТЬ 4](#_Toc76739218)

[Анализ аналогов 4](#_Toc76739219)

[Инфологическая модель базы данных 5](#_Toc76739220)

[Физическая модель базы данных 6](#_Toc76739221)

[Кейсы использования приложения пользователем, администратором или оператором 7](#_Toc76739222)

[Описание реализации всех этапов проекта 8](#_Toc76739223)

[Проектирование 8](#_Toc76739224)

[Подготовка к разработке 8](#_Toc76739225)

[Разработка проекта 8](#_Toc76739226)

[Тестирование 9](#_Toc76739227)

[Документация 10](#_Toc76739228)

[Типовые запросы к базе данных 11](#_Toc76739229)

[Сложные запросы 11](#_Toc76739230)

[ЗАКЛЮЧЕНИЕ 13](#_Toc76739231)

[СПИСОК ИСТОЧНИКОВ 14](#_Toc76739232)

# ВВЕДЕНИЕ

Цель данного проекта – изучение фреймворка Django, а также ознакомление с инструментом Postman и создание RESTfull API с использованием формата данных json. Тема проекта – продажа недвижимости. Необходимо спроектировать и составить базу данных на 6 таблиц, после чего настроить административную панель для управления базой данных с помощью фреймворка Django. Также необходимо создать три роли для пользователей, дашборд и CRUD.

# ОСНОВНАЯ ЧАСТЬ

## Анализ аналогов

<https://www.cian.ru/>

Циан – база данных недвижимости в городах России. Сайт является одним из первых, выдаваемых в органическом выводе поисковых систем. Для обычных пользователей есть возможности просматривать варианты недвижимости, рассчитывать примерную стоимость вместе с ипотекой, фильтровать и сортировать варианты различными способами, смотреть изменения цены в течение времени, также есть возможность связаться непосредственно с владельцем или застройщиками. Так же интересной функцией является просмотр других объявлений в просматриваемом доме. Для хозяина недвижимости есть функции добавления своих вариантов, редактирования и удаления их. Кроме того, на самом сайте есть различные информационные разделы, такие как статьи, новости, популярные объявления. Дизайн сайта минималистичен и удовлетворяет современным тенденциям. Сайт стабильно работает и удобен в использовании.

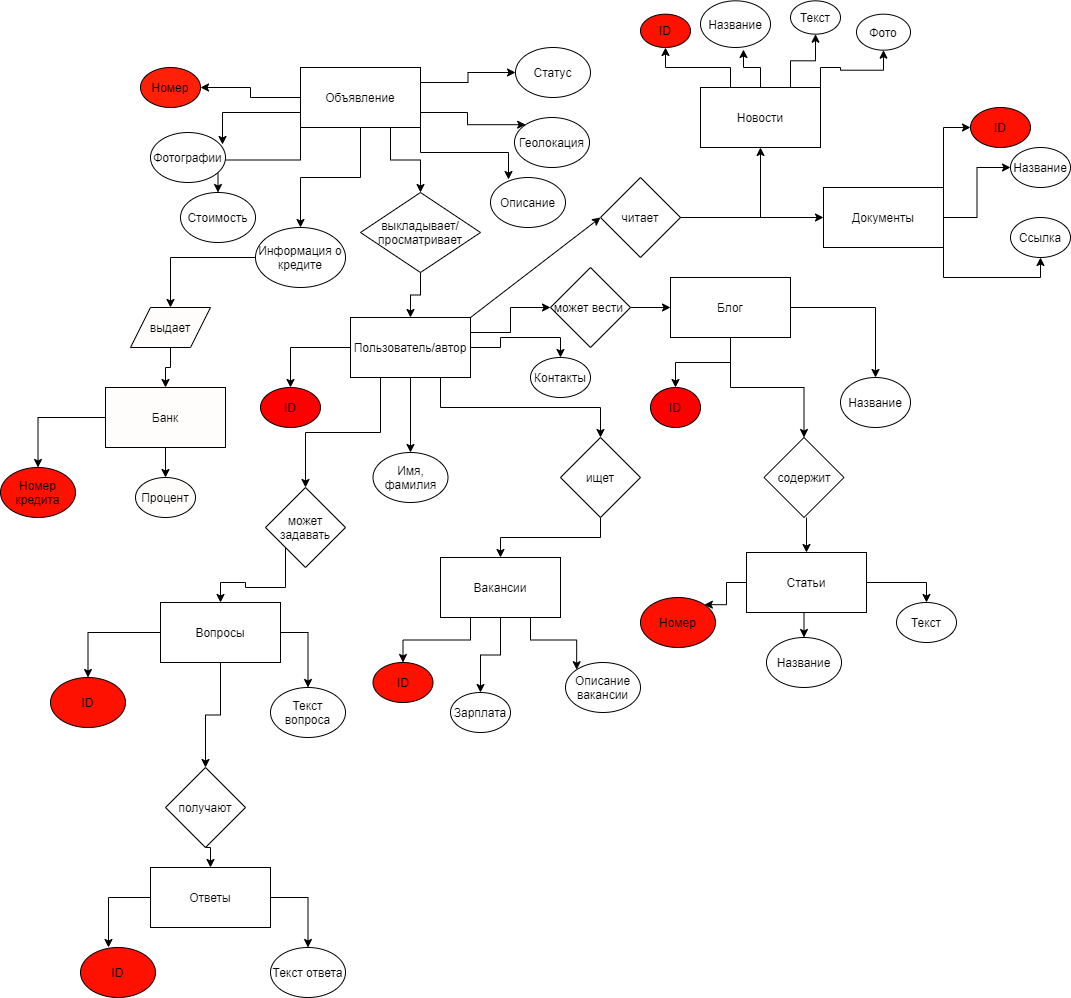
<https://www.domofond.ru/>

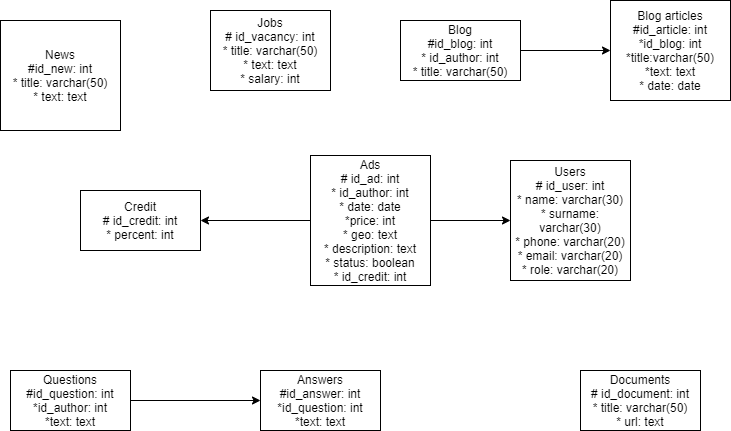
Домофонд – сайт с объявлениями о продаже и аренде жилья в различных городах России. Пользователь может просматривать объявления, смотреть объекты на карте и оценивать, что находится рядом, может рассчитать стоимость вместе с ипотекой, получить контакты риэлторов или других владельцев, а также фильтровать и сортировать варианты различными способами. Есть функция отображения рейтинга района, в котором расположено объявление, от местных жителей. Владелец может добавлять свои объявления, редактировать их и удалять. На главной странице присутствуют новости, блок новых объявлений, а также различные полезные функции, такие как расчет стоимости собственного жилья или ссылка на образцы документов, заполняемых при оформлении договора. Дизайн сайта минималистичен, но, на мой взгляд, немного устарел. Тем не менее, сайт работает стабильно, и им удобно пользоваться.

<https://move.ru/>

Move.ru – портал о недвижимости в Москве и Московской области, а также других регионов страны. Пользователь может просматривать объявления и оценивать их, смотреть список объектов рядом с домом, просматривать график изменения цен, связаться с владельцем и делать запросы более точно с помощью фильтров и сортировки. Владелец может добавлять свои объявления, редактировать и удалять их. На главной странице есть блок новостей, фильтры с категориями жилья, блок с специалистами и агентствами недвижимости и обзоры на новостройки. Дизайн сайта в целом отвечает современным тенденциям, сайтом удобно пользоваться и он стабильно работает.

## Инфологическая модель базы данных



Физическая модель базы данных

## Кейсы использования приложения пользователем, администратором или оператором

1. Незарегистрированный пользователь может зайти на сайт и зарегистрироваться.
2. Неавторизованный пользователь может зайти на сайт и авторизоваться.
3. Пользователь заходит на сайт с приложением и может просматривать объявления о продаже и аренде недвижимости.
4. Пользователь, просматривая объявление, может нажать кнопку «Получить контакты владельца», что приведет к выводу информации с контактами владельца.
5. Пользователь может создать свой блог и оставлять в нем записи.
6. Пользователь может задавать вопросы и отвечать на чужие.
7. Пользователь может просматривать новости о недвижимости.
8. Пользователь, просматривая объявление, может нажать кнопку «Рассчитать ипотеку», в следствие чего система выдаст пользователю результат цены недвижимости с стоимостью ипотеки.
9. Администратор может модерировать выложенные объявления, редактировать их и удалять.
10. Пользователь-владелец недвижимости может добавлять объявления, редактировать их и удалять.

## Описание реализации всех этапов проекта

### Проектирование

Была создана инфологическая и физическая модели базы данных. База данных состоит из 10 таблиц: Объявления (Ads), Пользователи (Users), Данные об ипотеках (Credits), Вопросы (Questions), Ответы (Answers), Новости (News), Вакансии (Jobs), Документы (Documents), Блог (Blog), Статьи блога (Blog articles). Каждая таблица содержит первичный ключ – ID, некоторые таблицы (например Questions и Answers) связаны с помощью Foreign Key.

На этапе проектирования также была изучена предметная область и проведен анализ аналогов.

### Подготовка к разработке

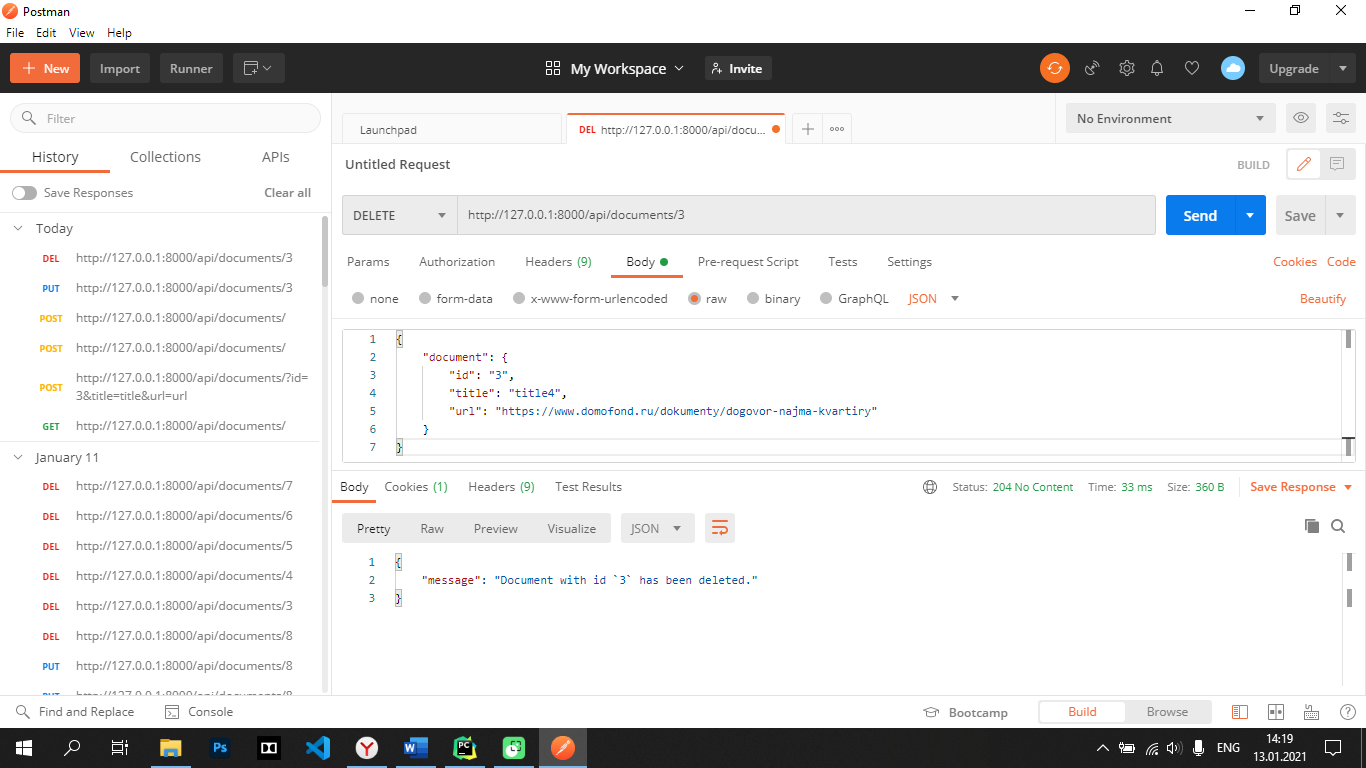
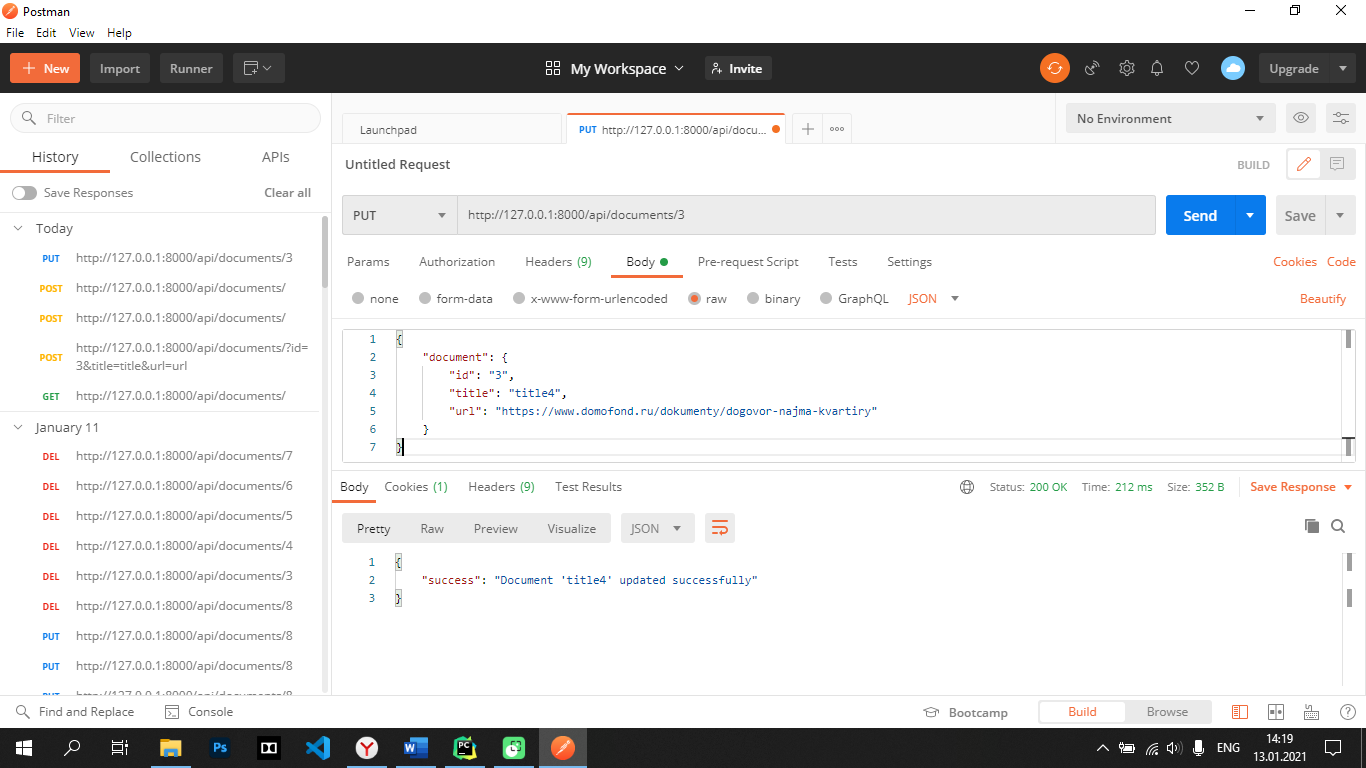
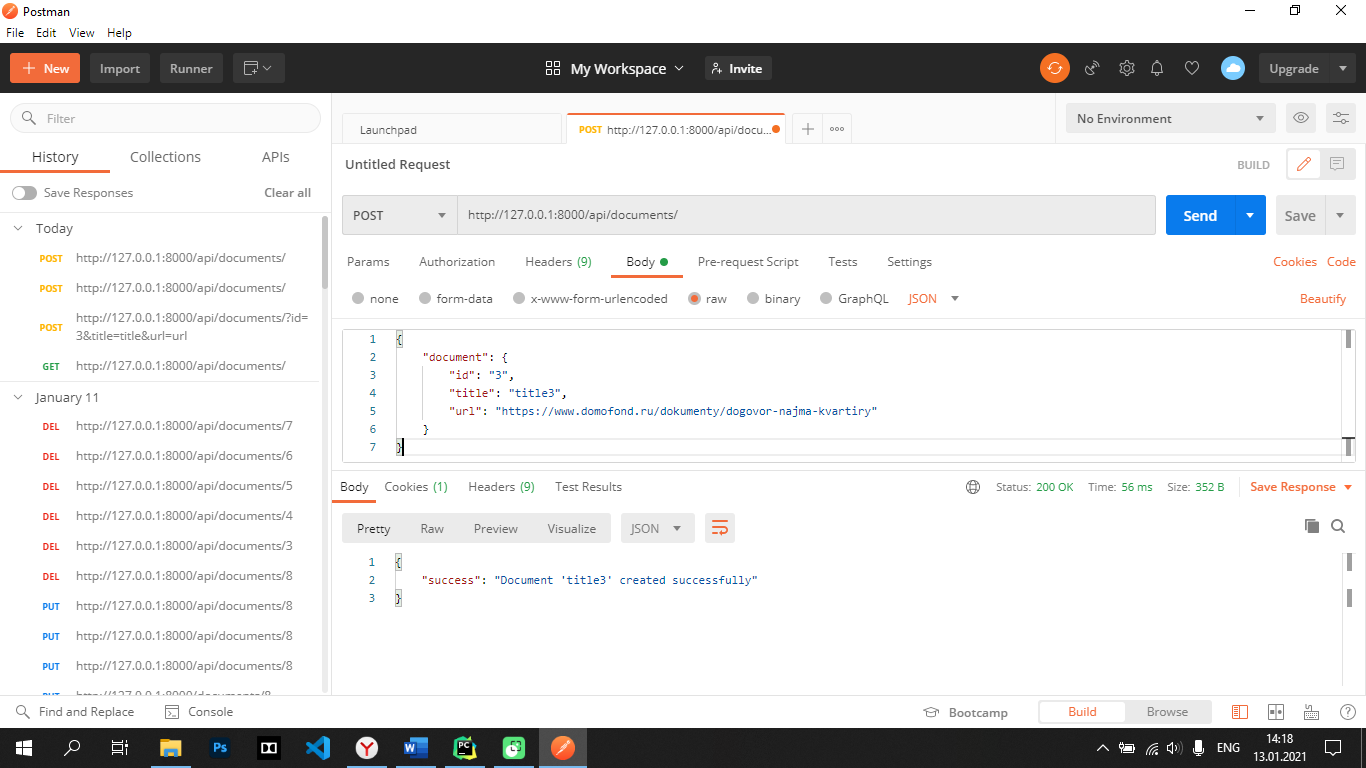
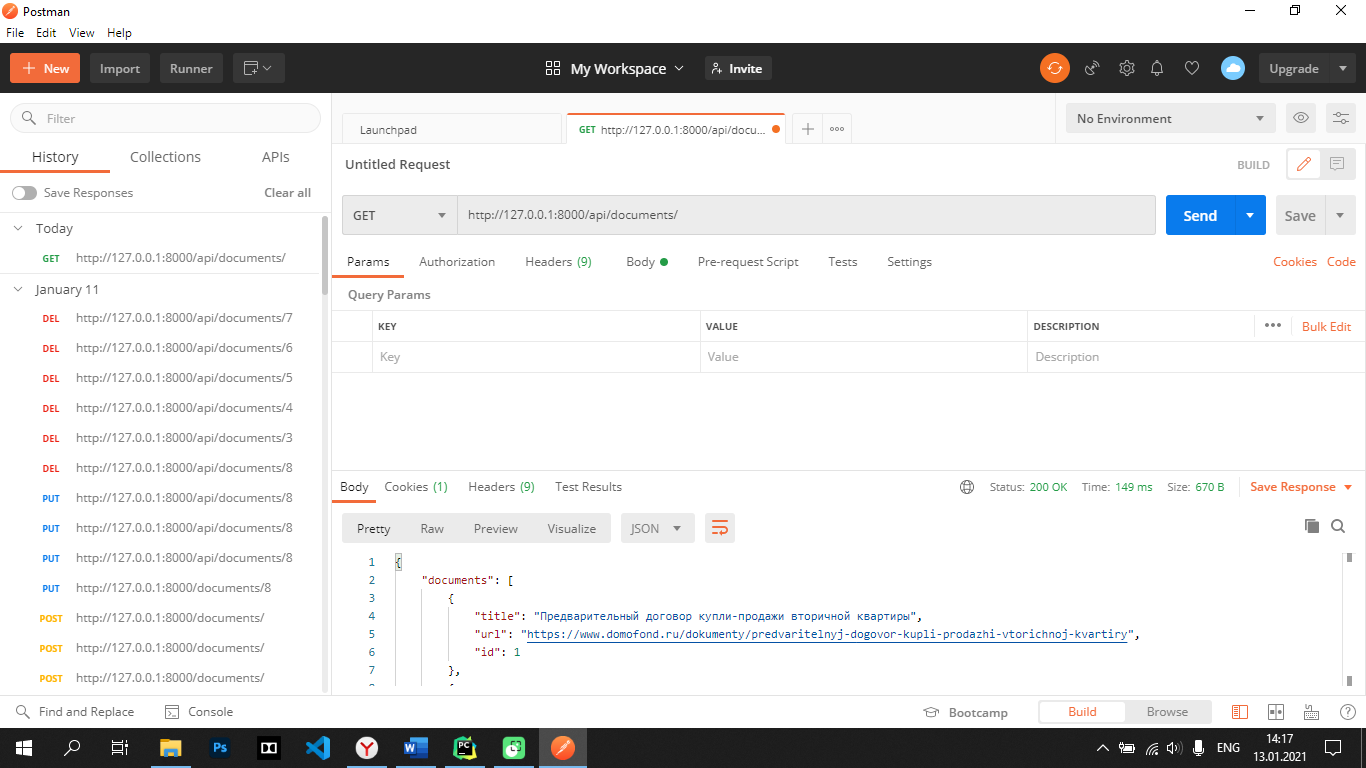
Был изучен фреймворк Django, а также дополнительные библиотеки для него, например django-import-export, необходимая для импорта и экспорта данных в панели администратора. Была просмотрена лекция от сотрудников Яндекс о Django, благодаря чему удалось лучше понять модель MTV и применить ее на практике. (ссылка - <https://www.youtube.com/watch?v=qMmI8qigObc>)

### Разработка проекта

Разработка проекта началась с создания репозитория проекта на github, основного приложения main и прописания в нем маршрутов url. Разработка велась с помощью веток git. После этого были созданы первые модели таблиц баз данных (в файле models.py) и написаны шаблоны с версткой для них (папка templates). Не забываем регистрировать модели в файле admin.py, чтобы к ним был доступ через панель администратора. Для панели администратора был создан “superuser” с логином и паролем admin/admin. Панель администратора доступна по адресу /admin. Чтобы связать шаблон с моделями страниц, в файле views.py создаем функции, в которых создаем объект с помощью запроса к модели таблицы базы данных и возвращаем функцию render(), в которую передаем путь к шаблону страницы и объект с данными. Чтобы страница была доступна по указанному в urls.py адресу, вторым параметром в path передаем функцию, которую написали в views.py (например views.index). Теперь наша страница доступна по адресу, указанным первым параметром в path. Перед любым использованием модели, метода или файла в языке python и его фреймворке Django необходимо в начале файла импортировать эти данные. После настройки приложения main по такому же принципу было создано приложение ad, отвечающее за вывод объявлений о продаже недвижимости. Схема базы данных находится в файле models.py. После добавления всех таблиц был реализован REST API для таблицы Documents. Теперь данные в ней можно добавлять, просматривать, редактировать и удалять с помощью запросов GET, POST, PUT и DELETE. Также для этой таблицы были добавлены функции сортировки и поиска в панели администратора. Для таблицы News был реализован импорт и экспорт данных с помощью библиотеки django-import-export. Теперь в панели администратора есть возможность импортировать и экспортировать данные для этой таблицы. На сайте можно зарегистрироваться и авторизоваться, пи этом обязательно проводится валидация введенных пользователем данных и вывод подсказок. Был добавлен CRUD для объявлений, теперь авторизованные пользователи могут добавлять объявления, удалять их и редактировать, при этом удалять и редактировать они могут только объявления, созданные ими. Модератор может удалять и редактировать все объявления пользователей. Администратор может назначать модераторов в панели администратора и управлять остальными таблицами. Был добавлен дашборд для таблицы вакансий, с сравнениями зарплат специалистов. Добавлен поиск по объявлениям (чувствительный к регистру).

### Тестирование

Для тестирования проекта было выбрано приложение Postman. Были проверены запросы GET, POST, PUT и DELETE для таблицы Documents.



### Документация

Была написана документация по проекту, опираясь на требования к ней.

## Типовые запросы к базе данных

Чтобы создать объект, создаем его экземпляр с помощью аргументов ключевого слова для класса модели, а затем вызываем [save()](https://django.fun/docs/django/ru/3.0/ref/models/instances/" \l "django.db.models.Model.save" \o "django.db.models.Model.save), чтобы сохранить его в базе данных. (То же, что INSERT в SQL).

Чтобы сохранить изменения в объекте, который уже находится в базе данных, используем [save()](https://django.fun/docs/django/ru/3.0/ref/models/instances/" \l "django.db.models.Model.save" \o "django.db.models.Model.save). (То же, что UPDATE)

Для получения всех объектов модели воспользуемся методом all().

Чтобы получить определенные данные, воспользуемся фильтрами:

filter(id=value) - возвращает новый [QuerySet](https://django.fun/docs/django/ru/3.0/ref/models/querysets/" \l "django.db.models.query.QuerySet" \o "django.db.models.query.QuerySet), содержащий объекты, которые соответствуют заданным параметрам поиска.

exclude(id=value) - возвращает новый [QuerySet](https://django.fun/docs/django/ru/3.0/ref/models/querysets/" \l "django.db.models.query.QuerySet" \o "django.db.models.query.QuerySet), содержащий объекты, которые не соответствуют указанным параметрам поиска.

Если вы знаете, что только один объект соответствует вашему запросу, вы можете использовать метод [get()](https://django.fun/docs/django/ru/3.0/ref/models/querysets/" \l "django.db.models.query.QuerySet.get" \o "django.db.models.query.QuerySet.get), который возвращает этот объект напрямую.

Эквиваленты выражений SQL LIMIT и OFFSET -

- этот код возвращает первые 5 объектов (LIMIT 5):

>>> Entry.objects.all()[:5]

- этот возвращает объекты с шестого по десятый (OFFSET 5 LIMIT 5):

>>> Entry.objects.all()[5:10]

## Сложные запросы

 Если вам нужны более сложные запросы (например, запросы с оператором OR), вы можете использовать объекты Q.

Объект Q – объект, используемый для инкапсуляции множества именованных аргументов для фильтрации. Аргументы определяются так же, как и в примерах выше.

Например, этот объект Q определяет запрос LIKE:

from django.db.models import Q

Q(question\_\_starts with='What')

Объекты Q могут быть объединены операторами & и |, при этом будет создан новый объект Q.

Например, это определение представляет объект Q, который представляет операцию “OR” двух фильтров с "question\_\_startswith":

Q(question\_\_startswith='Who') | Q(question\_\_startswith='What')

Этот фильтр равнозначен такому оператору SQL WHERE:

WHERE question LIKE 'Who%' OR question LIKE 'What%'

Каждый метод для фильтрации, который принимает именованные аргументы (например, filter(), exclude(), get()) также может принимать объекты Q. Если вы передадите несколько объектов Q как аргументы, они будут объединены оператором “AND”. Например:

Poll.objects.get(

Q(question\_\_startswith='Who'),

Q(pub\_date=date(2005, 5, 2)) | Q(pub\_date=date(2005, 5, 6))

)

Вы можете использовать одновременно объекты Q и именованные аргументы. Все аргументы (будь то именованные аргументы или объекты Q) объединяются оператором “AND”. Однако, если присутствует объект Q, он должен следовать перед именованными аргументами. Например:

Poll.objects.get(

Q(pub\_date=date(2005, 5, 2)) | Q(pub\_date=date(2005, 5, 6)),

question\_\_startswith='Who')

# ЗАКЛЮЧЕНИЕ

В результате проекта получилось приложение на Django, позволяющее взаимодействовать с базой данных (просматривать, добавлять, редактировать и удалять данные). Были изучены библиотеки для Django, а также изучен инструмент Postman для тестирования запросов.

http://project4sem.std-933.ist.mospolytech.ru/ - ссылка на проект

<https://github.com/kidmustrun/proekt3sem> - ссылка на репозиторий проекта

# СПИСОК ИСТОЧНИКОВ

1. <https://django.fun/>
2. <https://habr.com/ru/>
3. <https://stackoverflow.com/>
4. <https://www.djangoproject.com/>
5. <https://django-import-export.readthedocs.io/en/latest/>
6. <https://www.youtube.com/>