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: «Сервис доставки продуктов»

**Выполнил:**

Студент группы 191-321

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Грибанов А. В.

подпись, дата

**Принял:**

Старший преподаватель

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

подпись, дата

Москва 2021

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc61604516)

[ОСНОВНАЯ ЧАСТЬ 4](#_Toc61604517)

[ЗАКЛЮЧЕНИЕ](#_Toc61604518) 17

[ПОЛЕЗНЫЕ ССЫЛКИ 18](#_Toc61604519)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 18](#_Toc61604519)

# **ВВЕДЕНИЕ**

Основная цель проекта «Сервис доставки продуктов» – создать сервис доставки продуктов, работающий в крупных городах страны и позволяющий людям заказывать продукты прямо из дома, ведь их всегда доставят вовремя наши курьеры.

На сегодняшний день люди все чаще и чаще готовы заказывать продукты прямо на дом, ведь далеко не всегда есть возможность отвлечься от каких-то более важных дел, ради похода в магазин. Наш сервис предоставляет пользователям великолепную возможность забыть о долгом выборе продуктов и стоянии в очередях, ведь им нужно всего лишь заказать еду в нашем сервисе и курьеры всё сделают за покупателя.

Наш проект отвечает современным запросам и идет в ногу со временем, чтобы у каждого пользователя, который посетил наш интернет-ресурс, всегда была возможность выбрать лучшие продукты из лучших магазинов своего города и доплатить небольшую сумму за доставку.

Актуальность разработки состоит в том, что у аналоговых ресурсов, исходя из проведенного нами анализа, был выявлен ряд, как недостатков так и достоинств, и вышло так, что далеко не все аналоги соответствуют требованиям современного потребителя. Учитывая все плюсы и минусы подобных ресурсов, опираясь на существующие в данный момент тенденции, мы создали свой ресурс, позволяющий заказывать еду из магазина прямо домой.

Аналогами «Сервисов доставки еды» являются такие сервисы как:

* proedu.ru – сервис, доставляющий продукты на дом, в офис и даже в больницу. Доступен большой выбор еды, а также продукция нескольких компаний-партнеров
* sbermarket.ru – сервис доставки еды от «Сбера». Доступен в 135 городах России, а также множество магазинов-партнеров.
* europa-market.ru – сайт по доставке еды от гипермаркета «Европа», доступна еда только из данного магазина.

# **ОСНОВНАЯ ЧАСТЬ**

1. **Анализ аналогов**

В наше время достаточно много различных сервисов для доставки еды.

Начнем наш анализ с Proedu. Данный сервис действует только на территории г. Брянска, из-за этого конкурентом он будет являться только в данном регионе. Но, это также и минус, так как у сервиса заключено партнерство практически со всеми поставщиками еды в данном городе, а потому он, скорее всего, пользуется наибольшей популярностью у покупателей.

Главная страница сразу дает возможность выбрать категорию продуктов для заказа. Помимо этого, указан номер телефона, через который также можно заказать продукты (не совсем ясно, почему этот номер создан не для технической поддержки покупателей, в случае неполадок, а именно для заказа продуктов). К минусам можно отнести отсутствие выбора магазинов, а только выбор изготовителя или поставщика, из этого можно сделать выбор, что данный сервис сам является магазином. Также присутствует поиск на странице и есть удобный список категорий, по которым можно достаточно быстро найти нужные продукты. В целом, сервис достаточно хорошо сделан, пускай и с некоторыми недочетами.

Более крупным и известным сервисом является доставка от Сбера. Данный сервис работает во многих городах России, также доступен выбор из множества магазинов, многие из которых даже не являются продуктовыми. Видимо, на данном сайте можно заказать не только продукты. Дизайн выполнен в ассоциируемых со Сбером зеленых тонах, также присутствует достаточно приятный баннер, который настраивает клиента на доверие к данному сервису. К минусам можно отнести достаточно долгую доставку, так как на сайте указано, что курьер доставит продукты только «через несколько часов». К-сожалению, если Вы с утра решите заказать продукты на завтрак, то до обеда Вам придется сидеть голодным. Не лучшее решение, ведь ничто так не способствует преодолению лени, как голод, а ведь лень является главной союзницей данных сервисов.

Что касается сервиса Europa-market, то на данном сервисе осуществляется доставка продуктов только из сети магазинов «Европа».

Тем не менее, доставка доступна практически во всех крупных городах России, что возможно даже больше, чем у Сбера. К-сожалению, нет выбора изготовителей и поставщиков, как в первом сервисе и магазинов, как во втором. На сайте зачем-то присутствует раздел новостей города Курск, хотя это привлекает лишнее внимание и даже настораживает, ведь новости никак не относятся не к магазину, не даже к продуктам. В целом, из всех сервисов, данный мне показался наименее привлекательным, хоть поначалу и подкупает большим выбором населенных пунктов, хотя также и настораживает, ведь магазин находится всего в 11 городах России, которые располагаются в центральной части страны. Не совсем ясно, как доставка будет осуществляться в Сибирь и сколько за нее заплатит покупатель.

1. **Проектирование инфологической модели**

*Инфологическая модель* — ориентированная на человека и не зависимая от типа СУБД модель предметной области, определяющая совокупности информационных объектов, их атрибутов и отношений между объектами, динамику изменений предметной области, а также характер информационных потребностей пользователей.

*Цели инфологического моделирования*

Обеспечение наиболее естественных для человека способов сбора и представления той информации, которую предполагается хранить в создаваемой базе данных. Поэтому инфологическую модель данных пытаются строить по аналогии с естественным языком. Основными конструктивными элементами инфологических моделей являются сущности, связи между ними и их свойства.

*Основные понятия*

* Сущность – любой различимый объект, информацию о котором необходимо хранить в базе данных.
* Атрибут – поименованная характеристика сущности.
* Ключ – минимальный набор атрибутов, по значениям которых можно однозначно найти требуемый экземпляр сущности.
* Связь – ассоциирование двух или более сущностей.

На рисунке 1 представлена инфологическая модель базы данных данного проекта

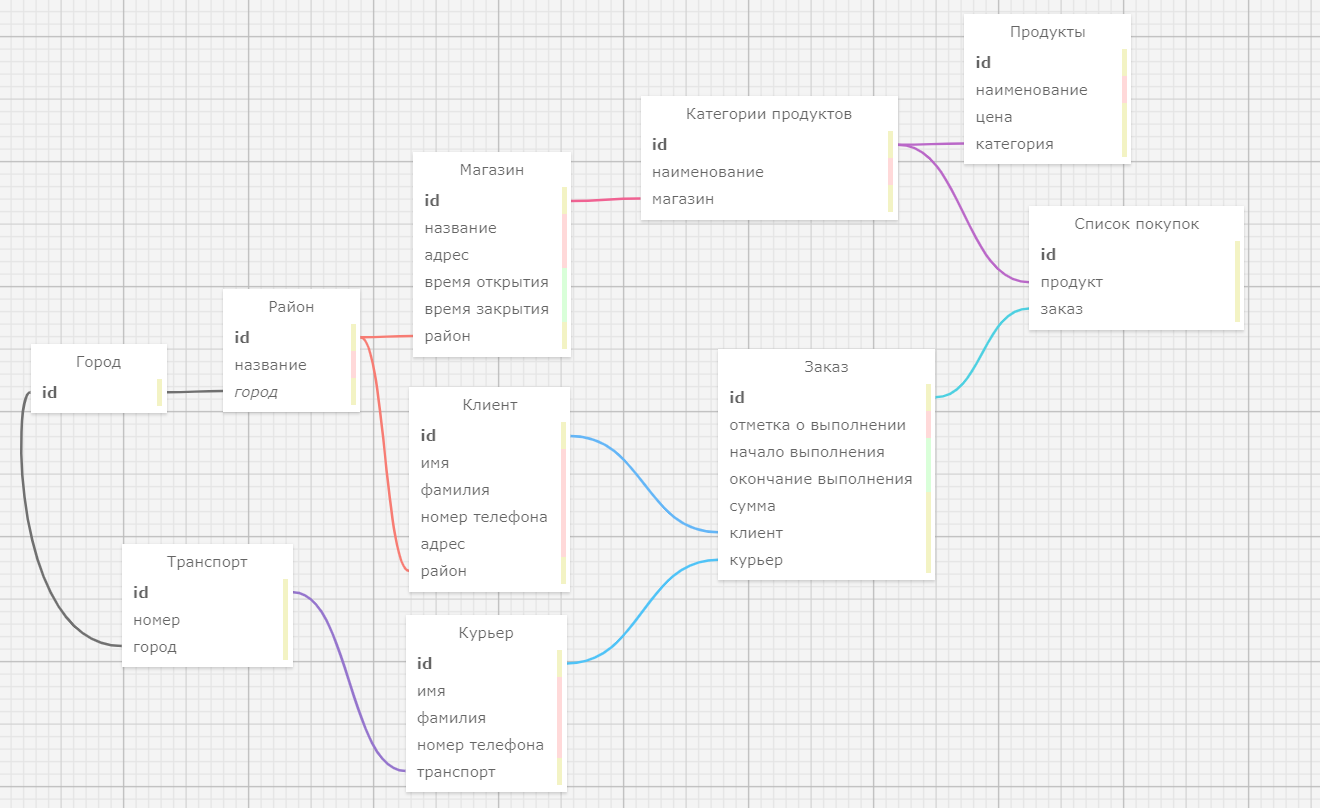


Рисунок 1 – инфологическая модель

1. **Проектирование физической структуры**

*Физическая модель* – это привязка логической модели к конкретной среде хранения и методам хранения данных.

*Целью* физического проектирования базы данных является описание способа физической реализации логического проекта базы данных.

В случае с реляционной моделью данных под физическим проектированием подразумевается следующее:

* создание набора реляционных таблиц и ограничений для них на основе информации, представленной в глобальной логической модели данных;
* определение конкретных структур хранения данных и методов доступа к ним, обеспечивающих оптимальную производительность СУБД;
* разработка средств защиты создаваемой системы.

На рисунке 2 представлена физическая модель базы данных данного проекта

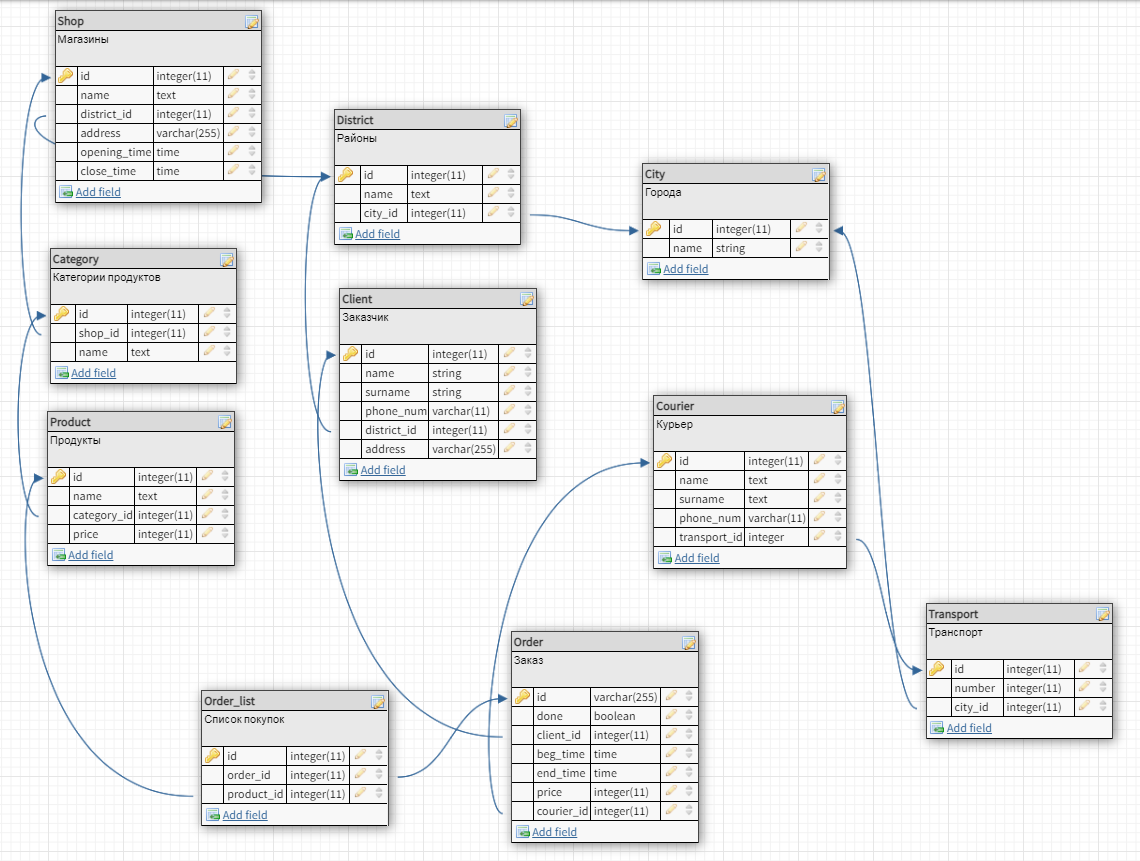


Рисунок 2 – физическая модель

1. **Создание Django-приложения**

Для разработки Django-приложения, необходимо установить Python. Для этого требуется скачать актуальную версию Python с официального сайта. Во время установки выбираем установку «по умолчанию», которая установит все необходимые компоненты Python в PATH (для доступа к языку Python в командной строке Windows по команде «python») (см. рисунок 3).

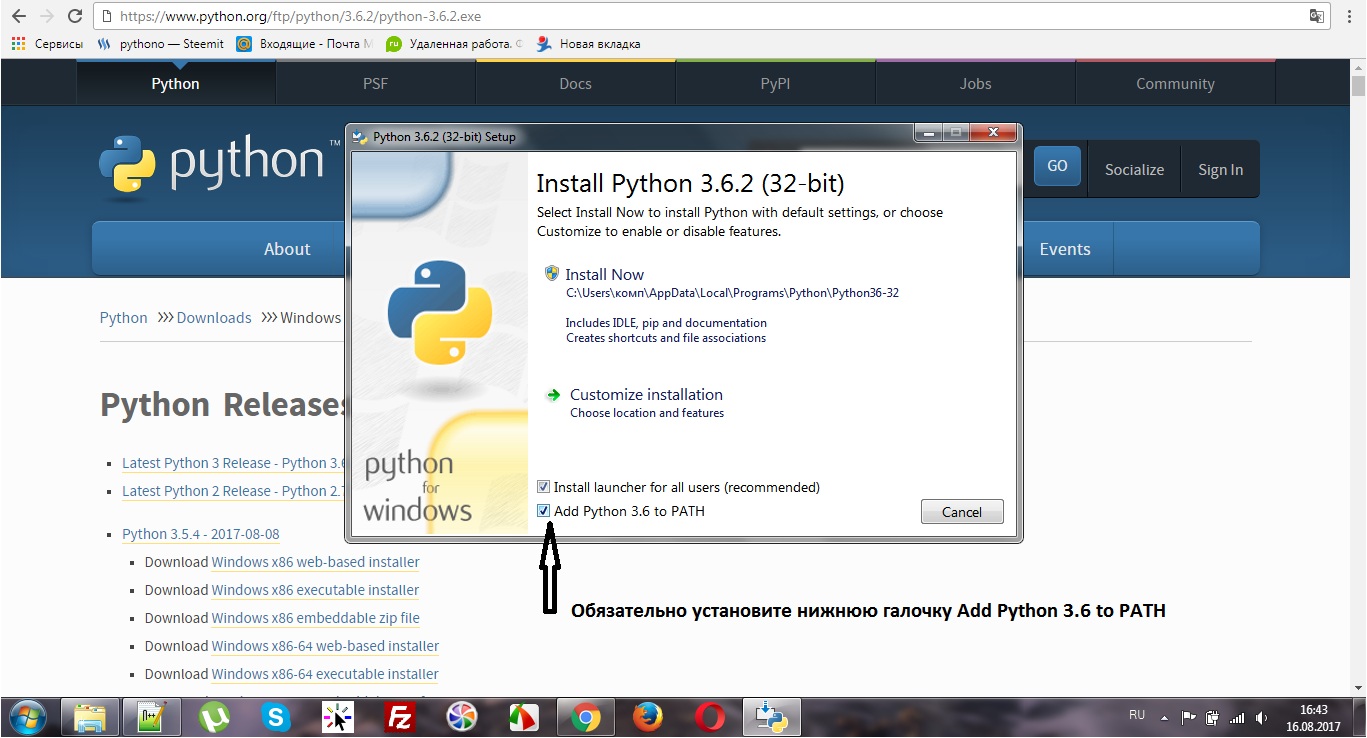


Рисунок 3 – окно установки python

Python установлен, теперь установим Django через pip.

После этого можно приступать к разработке приложения Django.

Создадим проект, для этого напишем в командную строку django-admin startproject EP.

После создания проекта, появятся следующие директории:

* EP /\_\_init\_\_.py — пустой файл, который говорит Python, что данная директория должна восприниматься в качестве пакета.
* EP /settings.py содержит конфигурацию нашего проекта.
* EP /urls.py — здесь объявляются URL.
* EP /wsgi.py — с помощью него приложение может работать с веб-сервером по протоколу WSGI.
* manage.py позволяет взаимодействовать с проектом.

Теперь произведем запуск приложения. Для этого в командной строке нужно написать python manage.py runserver. После этого в адресной строке браузера нужно написать: http://127.0.0.1:8000/.

После успешной проверки работоспособности сервера, можно создать модели в models.py. Некоторые классы модели представлены ниже на рисунке 4.



Рисунок 4 – пример создания классов

После создания классов необходимо сделать миграцию:

* python manage.py makemigrations
* python manage.py migrate

Теперь мы можем начать пользоваться панелью администратора. Но для этого нам нужен пользователь. Его создание будет описано в следующем пункте.

1. **Настройка административного интерфейса Django**

Для того, чтобы войти в админ-панель, необходимо создать учетную запись пользователя. Создадим учетную запись суперпользователя для просмотра и создания записей, учетная запись такого типа даст полный доступ к сайту и все необходимые разрешения, используя manage.py.

Для создания суперпользователя нужно использовать следующую команду:

* python manage.py createsuperuser

После заполнения всей информации о пользователе, он будет добавлен в базу данных. Теперь нужно перезапустить сервер, чтобы можно было проверить вход на сайт. Это можно сделать с помощью команды:

* python manage.py runserver.

Для входа в админ-панель нужно открыть http://127.0.0.1:8000/admin и ввести логин и пароль созданного суперпользователя.

В этой части сайта отображаются все созданные модели, сгруппированные по установленному приложению.

Далее мы можем кастомизировать нашу панель с помощью атрибута list\_display, которое мы пропишем нашему классу в admin.py. Данный артибут указывает, какие столбцы будут отображаться в change list’e. Это значение представляет собой кортеж атрибутов моделируемого объекта.

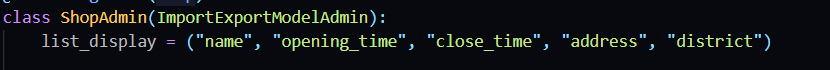
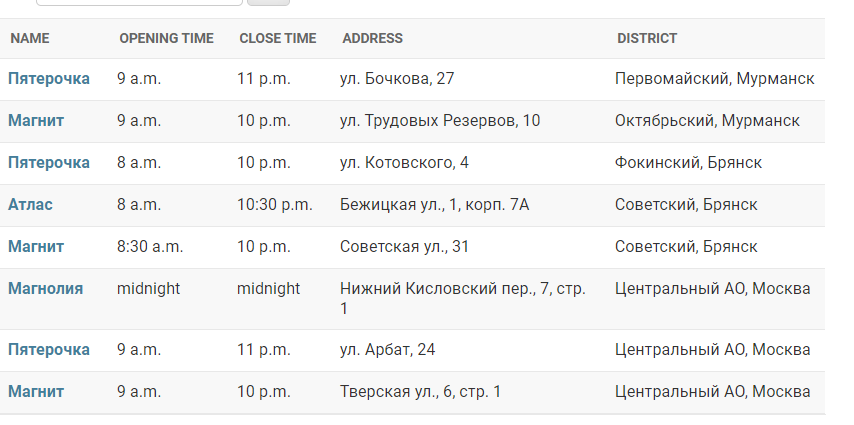
Далее на рис. 5 представлен пример.

Рис. 5 – кастомизация панели, с помощью list\_desplay



**Добавление фильтрации**

Для добавления фильтрации можно использовать встроенный виджет. Для примера рассмотрим добавление атрибута «list\_filter» к классу ShopAdmin в admin.py. (см. рисунок 6)

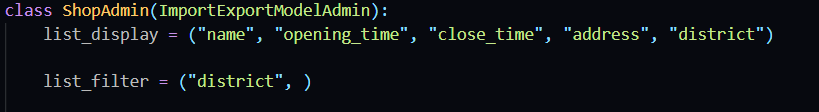
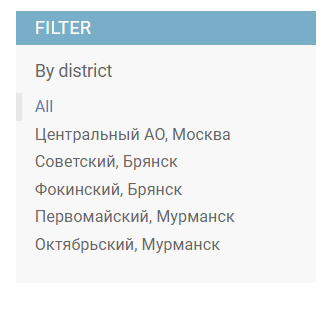


Рисунок 6 – добавление фильтров



**Добавление поиска**

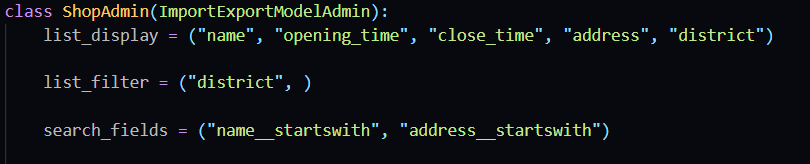
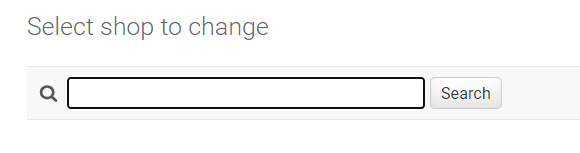
Также был реализован поиск на сайте с помощью опции search\_fields, которая добавляет на экран поле поиска. Он устанавливается с помощью кортежа, содержащего имена полей, которые будут использоваться для построения поискового запроса к базе данных. (см. рисунок 7)

Рисунок 7 – добавление поиска



1. **Наполнение БД**

Идет в ручном формате, так как на интернет-ресурсе, на котором проектировалась БД, нет возможности перенести сгенерированные таблицы в SQLite.

Так же были продуманны варианты использования разработанного веб-ресурса. Они представлены в таблице 1.

Таблица 1 – варианты использования админ-панели

|  |  |  |
| --- | --- | --- |
| Пользователь | Администратор | Курьер |
| Пользователь из города Брянск решил заказать себе несколько продуктов из ближайшего магазина. Заказ поступил курьеру, но по пути произошла авария. | Администратор получил сообщение от курьера, что тот попал в аварию, но заказ пользователя так и остался висеть. Администратор вручную заменил курьера на другого. Затем тот продолжил выполнение заказа. | Курьер так торопился выполнить заказ, что не заметил выскочившую на дорогу кошку, к-счастью, наши курьеры любят животных, и он пожертвовал машиной, ради спасения кошки. В итоге заказ поступил другому курьеру. |
| Пользователь из города Москва переехал в другой район, но забыл его изменить и вызвал сервис доставки еды на старый. Затем, он незамедлительно связался с администрацией нашего сервиса, чтобы сообщить новый адрес. | Администратор принял сообщение пользователя и поспешил изменить адрес на сайте. | Курьер заметил новый адрес и доставил продукты в другой район вовремя. |
| Пользователь из города Мурманск сделал заказ в сильнейший снегопад у сервиса-конкурента и те так и не доставили продукты. Он отказался от заказа и заказал у нас. | Администратор заранее предупредил курьеров о сильнейших морозах, дабы не возникло неприятных ситуаций. | Крьер подготовился к снегопадам и надел свой самый теплый пуховик. Помимо этого, он доставлял заказ на удобной машине с зимней резиной, так что снегопад не смог остановить нашего курьера. |
|  | Администратор постоянно контролирует и обновляет списки магазинов, категорий и продуктов на нашем сервисе. |  |

**7. Реализация импорта и экспорта данных в БД**

Чтобы интегрировать *django-import-export* с моделью проекта, мы создадим класс ShopResource в admin.py (см. рисунок 21), которой будет описывать, как этот ресурс может быть импортирован или экспортирован:

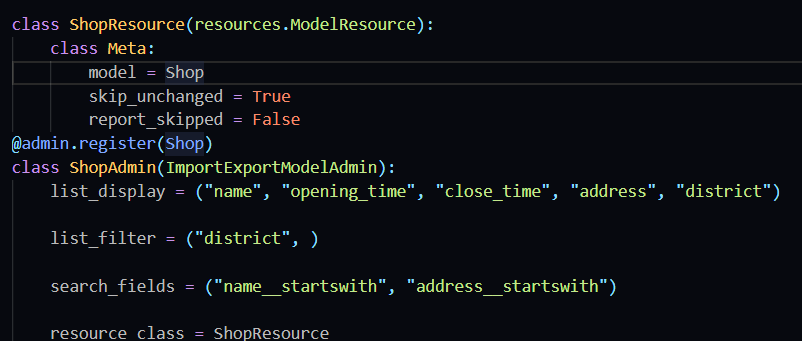


Рисунок 21 – реализация импорта и экспорта

Установим «skip\_unchanged», чтобы не импортировать каждый раз все записи, а только измененные.

Установим report\_skipped, этот параметр определяет, будут ли отображаться пропущенные записи в Result объект импорта.

Не забудем ниже указать ссылку на наш класс, прописав resource class = ShopRecource.

**8. Реализация REST API**

Для работы с REST API необходимо установить специальный фреймворк для Django делается это с помощью команды: pip install djangorestframework (см. рисунок 10)

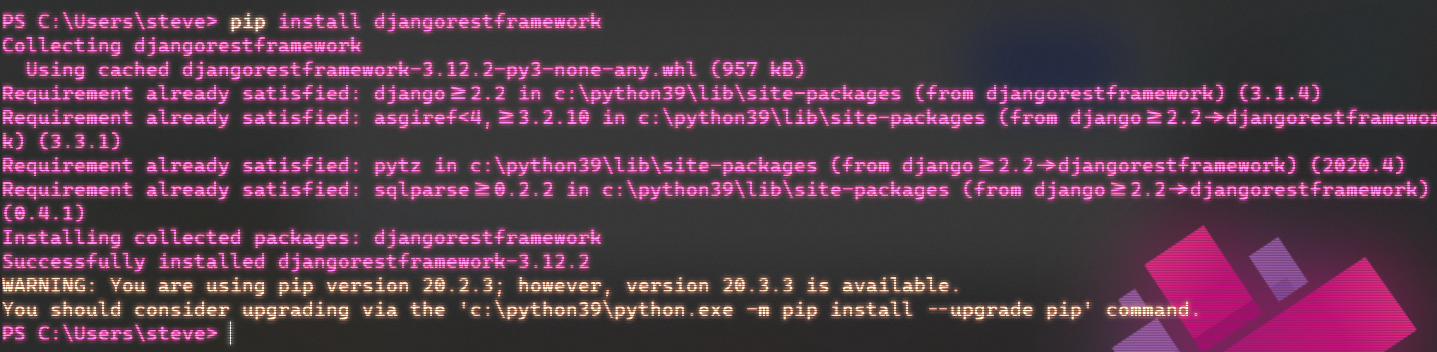


Рисунок 10 – установка Django REST API

Далее добавляем в INSTALLED\_APPS наш фреймворк как показано ниже. (см. рисунок 11)

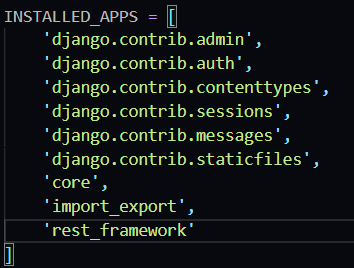


Рисунок 11 – добавление фреймворка в список

Затем создадим serializers.py, который будет использоваться для представления данных.

После создаем сериализаторы, которые показаны ниже на рисунке 12

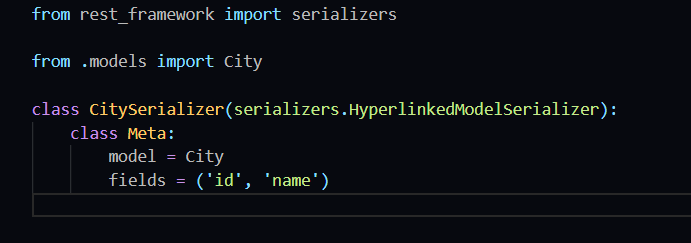


Рисунок 12 – serializers.py

Затем во views.py были созданы класс, который показан ниже на рисунке 13

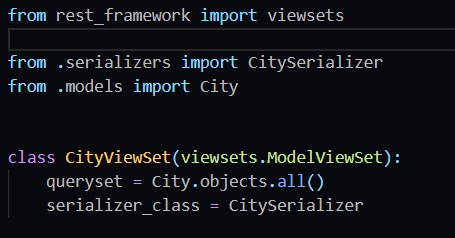


Рисунок 13 – views.py

Сначала подключаем и прописываем пути к нашему приложению в основной urls.py (см. рисунок 14)

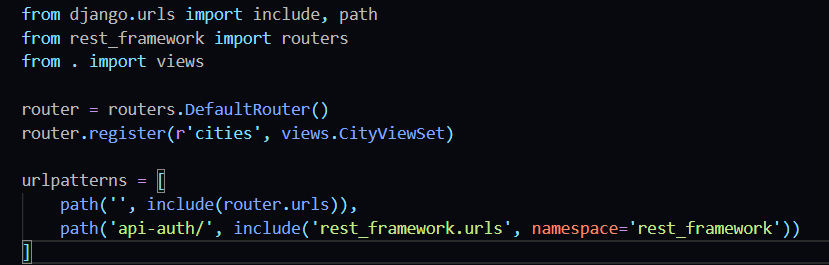


Рисунок 14 – подключение основных приложений

Теперь следует подключить дополнительные страницы, учитывая правила REST API, в main/urls.py и настроить как показано ниже на рисунке 15

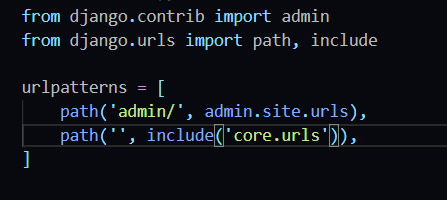


Рисунок 15 – подключение дополнительных приложений

На этом создание и настройка REST API завершена.

Для проекта был использован Postman для облегчения работы с REST API и для его более легкого тестирования. На рисунках ниже (см. рисунки 16 – 18) представлены все запросы, созданные для данного проекта.

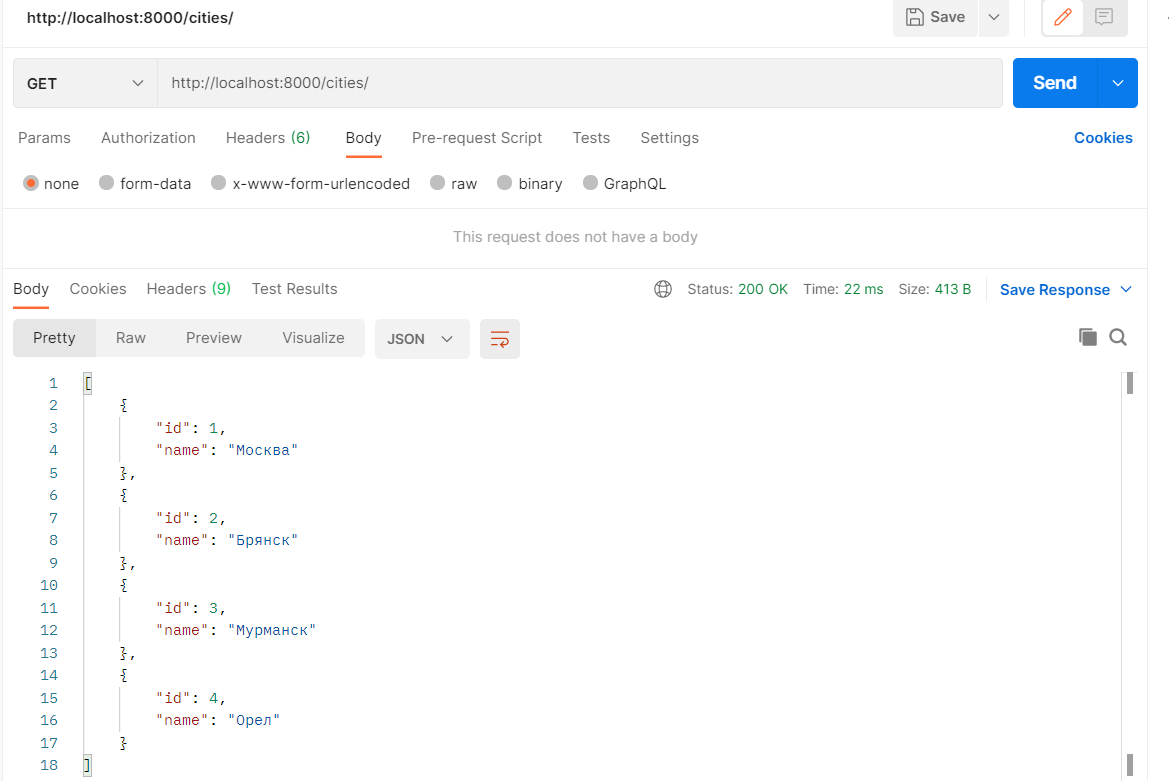


Рисунок 16 – get-запрос по всем городам

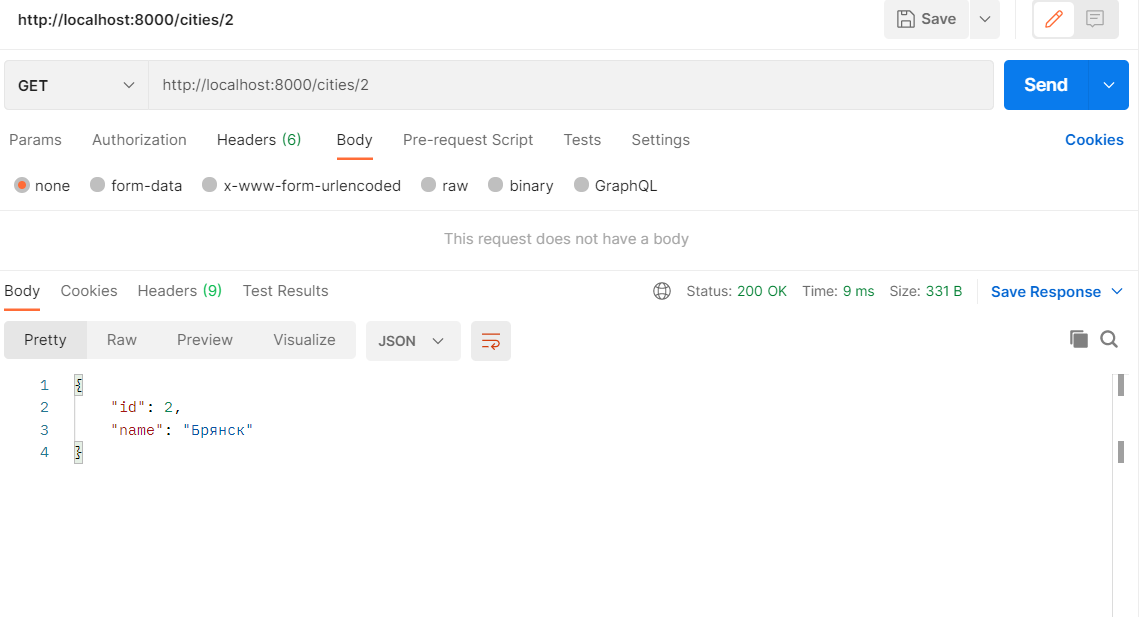


Рисунок 17 – get-запрос по конкретному городу

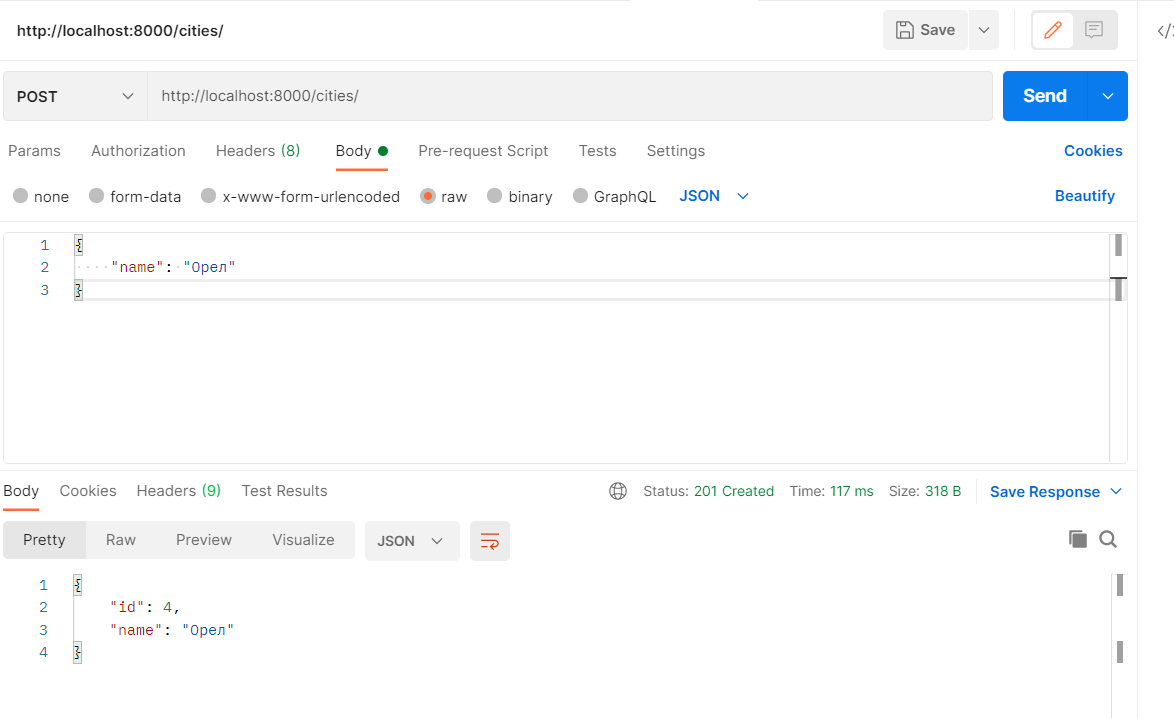


Рисунок 18 – создание нового города

# **ЗАКЛЮЧЕНИЕ**

Результат выполненной работы может применяться для создания нового сервиса доставки продуктов.

Выполненный инженерный проект позволит создать новый бизнес-проект, развитие которого будет полностью завесить от его владельца.

В данном проекте был проведен качественный анализ аналогов. В ходе разработки проекта были изучены основные понятия и принципы построения инфологической и физической модели базы данных, также данные модели были спроектированы для нашего проекта. Была изучена необходимая документация языка программирования Python и фреймворка Django. После происходил процесс настройки админ-интерфейса, который был создан при помощи Django. В него входили такие инструменты как: фильтрация, поиск, фильтр по дате, прочее, вывод информации из других таблиц. Далее была внедрена функция импорта и экспорта данных в базу данных, как сказано выше при помощи библиотеки для Django. В процессе реализации БД в Django был реализован REST API и произведено тестирование API через сервис Postman..

# 

# ПОЛЕЗНЫЕ ССЫЛКИ

<https://github.com/ArtazarXXX/EP> - ссылка на гитхаб репозиторий проекта

<https://drive.google.com/drive/folders/1mmLSZfJAGJcDPtMr32kyIOyBzcY_rK54?usp=sharing> – ссылка на папку на гугл диске с документацией роекта, а также с видеопрезентацией.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Дронов В. А. Django 2.1 Практика создания веб-сайтов на Python.: BHV, 2019. 672с.
2. <https://docs.djangoproject.com/en/3.0/> Официальная документация
3. <https://medium.com/@bullet4myv13> Статья для изучения Postman
4. <https://pythonist.ru/kastomizacziya-admin-paneli-django/> Статья по настройке Django админ-панели
5. <https://pythonist.ru/django-optimizacziya-raboty-s-bazoj-dannyh/> Оптимизация работы с базой данных
6. <https://stackoverflow.com> Система вопросов и ответов о программировании
7. <https://coderoad.ru> Система вопросов и ответов о программировании
8. <https://django.fun/tutorials/put-ot-request-do-response-v-django/>  Статья о том, как обрабатываются запросы в Django