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: «Проектирование и разработка сервиса заказа такси с использованием фреймворка Django»

**Выполнил:**

Студент группы 181-322

|  |  |
| --- | --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Поздняков Н. И. |
| подпись, дата | **Принял:**  Старший преподаватель |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Даньшина М.В. |
| ​подпись, дата |  |

Москва 2021

**Оглавление**

[**ВВЕДЕНИЕ** 2](#_Toc61647325)

[**Описание проблемы** 2](#_Toc61647326)

[**Цель проекта** 2](#_Toc61647327)

[**Аналоги** 2](#_Toc61647328)

[**ОСНОВНАЯ ЧАСТЬ** 3](#_Toc61647329)

[**Разработка инфологической и физической схем базы данных** 3](#_Toc61647330)

[**Создание Django-приложения** 6](#_Toc61647331)

[**Настройка административного интерфейса Django** 10](#_Toc61647332)

[**Кейсы использования приложения** 11](#_Toc61647333)

[**Реализация REST API** 12](#_Toc61647334)

[**Реализация импорта и экспорта данных в базу данных** 13](#_Toc61647335)

[**ЗАКЛЮЧЕНИЕ** 14](#_Toc61647336)

[**Результат** 14](#_Toc61647337)

[**Ссылки на итоговые материалы** 14](#_Toc61647338)

[**СПИСОК ИСТОЧНИКОВ** 15](#_Toc61647339)

# **ВВЕДЕНИЕ**

## **Описание проблемы**

Необходимо разработать базу данных и административный интерфейс для приложения, предметной областью которого является сервис заказа такси.

## **Цель проекта**

Целью данного проекта является проектирование и создание реляционной базы данных для веб-сервиса по заказу такси и дальнейшая настройка административной панели для управления данной базой данных с помощью фреймворка Django.

## **Аналоги**

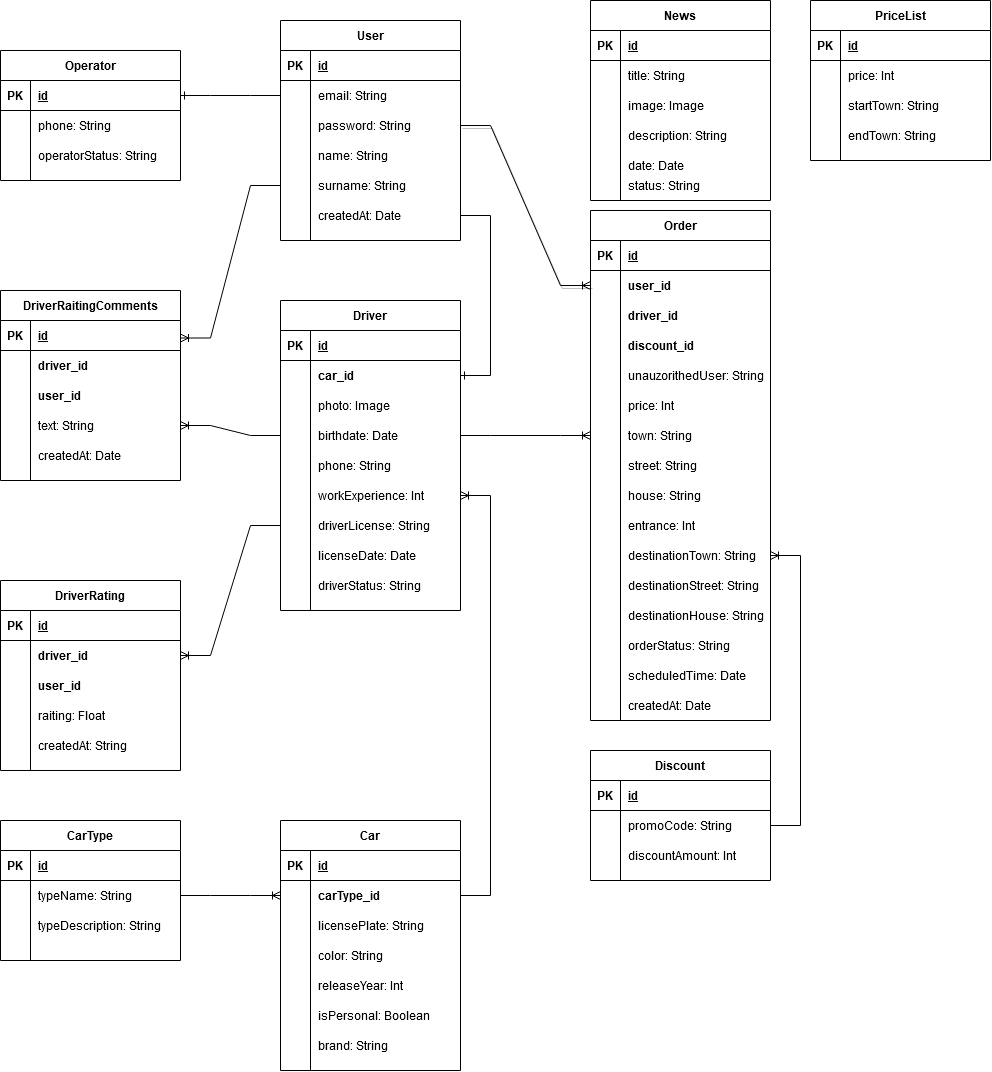
Аналогами данного проекта являются:

* Сервис “Яндекс Такси”, который позволяет клиентам заказывать машину с помощью мобильного приложения
* Различные сервисы заказа такси, где клиент оформляет заказ через оператора (например, такси “Соло” в г. Раменское)

# **ОСНОВНАЯ ЧАСТЬ**

## **Разработка инфологической и физической схем базы данных**

Данный сервис должен предоставлять быстрый и эффективный поиск такси, оформление заказа, с учетом специфичных пожеланий клиентов, с выводом стоимости доставки и дополнительной информации по такси для каждого критерия заказа. С учетом вышеописанных требований была разработана схема базы данных, представленная на рисунке далее. (Рисунок 1)



*Рисунок 1 - Инфологическая схема базы данных*

Данная схема обладает следующими сущностями:

* **User (Пользователь)**

Сущность имеет следующие атрибуты:

*Id* - уникальный идентификатор пользователя, является первичным ключом, также используется для связи данной таблицы с другими.

*Email* – уникальная электронная почта пользователя, необходимая при регистрации

*Password* – пароль

*Name* – имя пользователя

*Surname* – фамилия пользователя

*CreatedAt* – дата регистрации

* **Operator (Оператор)**

Сущность имеет следующие атрибуты:

*Id* - уникальный идентификатор оператора, является первичным ключом, также используется для связи данной таблицы с другими.

*Phone* – номер телефона

*OperatorStatus* – статус оператора (разговаривает ли он с клиентом, нет на рабочем месте и т д)

Данная сущность связана с Пользователем и наследует все ее свойства.

* **Driver (Водитель)**

Сущность имеет следующие атрибуты:

*Id* - уникальный идентификатор водителя, является первичным ключом, также используется для связи данной таблицы с другими.

*Car\_id* – id автомобиля

*Photo* – фотография водителя

*Phone* – номер телефона

*Birthdate* – дата рождения

*WorkExperience* – опыт работы (кол-во лет)

*DriverLicense* – номер водительских прав

*licenseDate* – дата получения прав

*DriverStatus* – статус водителя (ожидает клиента, ожидает заказ и т д)

Данная сущность связана с Пользователем и наследует все ее свойства.

* **DriverRaitingComments (Комментарии пользователей о водителях)**

Сущность имеет следующие атрибуты:

*Id* - уникальный идентификатор комментария, является первичным ключом, также используется для связи данной таблицы с другими.

*Driver\_id* – id водителя

*User\_id* – id пользователя

*Text* – текст комментария

*CreatedAt* – дата создания комментария

* **DriverRaiting (Оценки пользователей)**

Сущность имеет следующие атрибуты:

*Id* - уникальный идентификатор оценки, является первичным ключом, также используется для связи данной таблицы с другими.

*Driver\_id* – id водителя

*User\_id* – id пользователя

*Raiting* – оценка (целое число от 1 до 5)

*CreatedAt* – дата создания оценки

* **News (Новости)**

Сущность имеет следующие атрибуты:

*Id* - уникальный идентификатор новости, является первичным ключом, также используется для связи данной таблицы с другими.

*Title* – заголовок

*Image* – картинка

*Description* – основной текст статьи

*Date* – дата публикации

*Status* – статус новости (черновик или опубликована)

* **PriceList (Список цен)**

Сущность имеет следующие атрибуты:

*Id* - уникальный идентификатор цены, является первичным ключом, также используется для связи данной таблицы с другими.

*Price* – цена

*StartTown* – населенный пункт, откуда нужно забрать клиента

*EndTown* – населенный пункт, куда клиента нужно доставить

* **Order (Заказ)**

Сущность имеет следующие атрибуты:

*Id* - уникальный идентификатор заказа, является первичным ключом, также используется для связи данной таблицы с другими.

*Driver\_id* – id водителя

*User\_id* – id пользователя

*Discount\_id* – id скидки

*Price* – цена

*UnauthorizedUser* – имя клиента (используется в случае, если клиент решил вызвать машину через звонок оператору)

*Town* – населенный пункт, откуда нужно забрать клиента

*Street* – улица, с которой нужно забрать клиента

*House* – дом, из которого нужно забрать клиента

*Entrance* – подъезд дома клиента

*DestinationTown* – населенный пункт, куда нужно доставить клиента

*DestinationStreet* – улица, на которую нужно доставить клиента

*DestinationHouse* – дом, к которому нужно доставить клиента

*OrderStatus* – статус заказа (клиент доставлен, клиент ждет машину и т д)

*ScheduledTime* – время, когда клиента нужно забрать (если он его указал)

*CreatedAt* – время оформления заказа

* **Discount (Скидки)**

Сущность имеет следующие атрибуты:

*Id* - уникальный идентификатор скидки, является первичным ключом, также используется для связи данной таблицы с другими.

*PromoCode* – промокод, который пользователь должен использовать, чтобы получить скидку

*DiscountAmount* – размер скидки (в %)

* **Car (Автомобили)**

Сущность имеет следующие атрибуты:

*Id* - уникальный идентификатор автомобиля, является первичным ключом, также используется для связи данной таблицы с другими.

*licensePlate* – гос номер автомобиля

*Color* – цвет автомобиля

*ReleaseYear* – год выпуска

*isPersonal* – является ли личным транспортом водителя

*brand* – брэнд

*CarType\_id* – id типа автомобиля

* **CarType (Типы автомобилей)**

Сущность имеет следующие атрибуты:

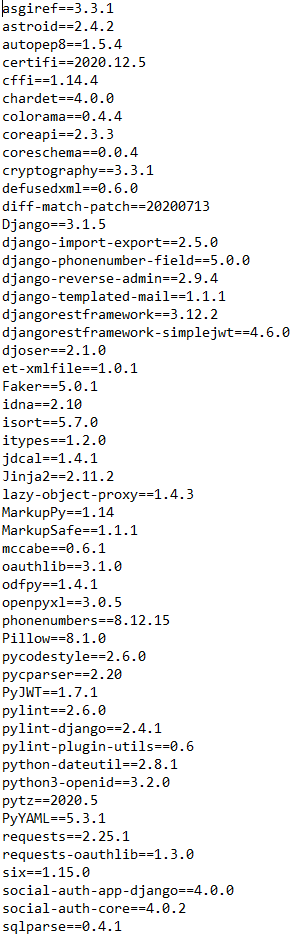
*Id* - уникальный идентификатор автомобиля, является первичным ключом, также используется для связи данной таблицы с другими.

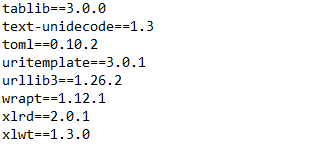
*TypeName* – название типа

*TypeDescription* – описание типа

## **Создание Django-приложения**

Для создания Django приложения использовался Python 3.8, а также некоторые дополнительные программные пакеты, которые можно посмотреть в файле requirements.txt, расположенном в корне проекта, либо на рисунке (Рисунок 2)



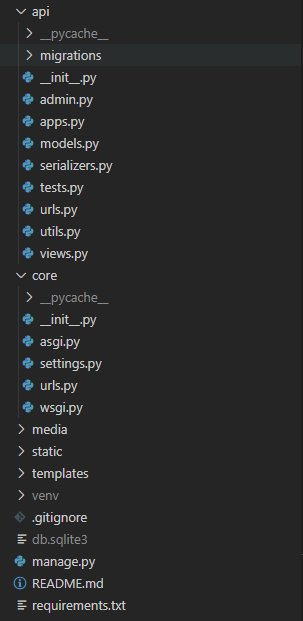


*Рисунок 2 - Дополнительные программные пакеты*

Важно отметить использование в данном приложении Django Rest Framework - библиотеки, которая работает со стандартными моделями Django для создания гибкого и мощного REST API.

Также в проекте применялась библиотека Djoser, которая является готовой REST-реализацией системы аутентификации Django и предоставляет набор представлений Django Rest Framework для обработки основных действий, таких как регистрация, вход в систему, выход из системы, сброс пароля и активация учетной записи. Djoser также работает с пользовательской моделью пользователя.

На следующем рисунке (Рисунок 3) можно рассмотреть структуру проекта.



*Рисунок 3 – Структура проекта*

В папке core расположены основные файлы проекта Django. Для описания отдельного REST api и моделей базы данных было создано отдельное приложение в папке api. В файле api/models.py были описаны все модели структуры БД, которые с помощью миграций были преобразованы в таблицы (использовались команды *python manage.py makemigrations* и *python manage.py migrate*). Данный проект на Django использует СУБД SQLite. Для модели пользователя пришлось переписать стандартную модель, предоставляемую Django по умолчанию. Новая модель, показанная на следующих рисунках (Рисунки 4 и 5), для регистрации пользователя требует только его почту, имя, фамилию и пароль.



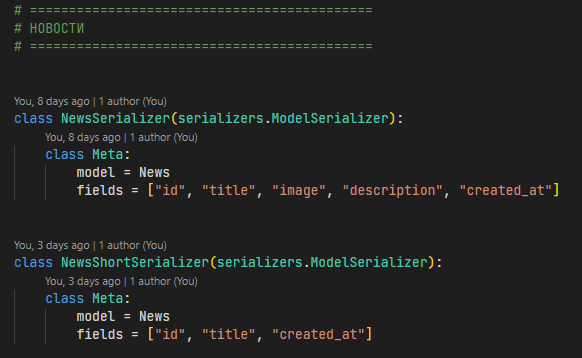
*Рисунок 4 – Новый User Manager*

Также для данной модели создано дополнительное поле *userType,* которое означает тип пользователя (администратор, обычный пользователь, оператор или водитель).



*Рисунок 5 – Новая модель пользователя*

В файле api/serializers.py описаны все сериализаторы. Сериализаторы в REST framework работают аналогично классам Django Form и ModelForm. Используемый класс Serializer дает мощный, общий способ управления вашими ответами, а также класс ModelSerializer - полезный и быстрый способ создания сериализаторов, которые имеют дело с экземплярами модели и querysets. Таким образом можно легко определять, какие типы данных можно отправлять для конкретного типа запроса. Например, в зависимости от запроса используются разные сериализаторы (Рисунок 6) модели News, и при получении списка новостей применяется NewsShortSerializer, а для получения данных о конкретной новости – NewsSerializer.



*Рисунок 6 – Сериализаторы модели News*

В файле api/views.py описаны все viewsets, которые будут использоваться в API запросах. Django REST framework позволяет объединить логику набора связанных views в один класс, называемый набором представлений. В других фреймворках можно найти концептуально похожие реализации, названные чем-то вроде "ресурсов" или "контроллеров". Класс ViewSet - это просто тип представления на основе классов, который не предоставляет никаких обработчиков методов, таких как get() или post (), а вместо этого предоставляет такие действия, как .list() и .create().

В файле api/urls.py указаны все маршруты REST API запросов и используемые в них Viewset’ы. Здесь используются готовые запросы библиотеки Djoser, позволяющие быстро настраивать авторизацию моделей пользователей, отличающихся от стандартной модели.

Далее описанные urlpatterns подключаются в основной файл urls.py в папке core.

## **Настройка административного интерфейса Django**

Настройка админ панели Django описана в файле api/admin.py. На данном рисунке (Рисунок 7) продемонстрировано описание выводимых данных модели News в админ панели.



*Рисунок 7 – Описание модели News в админ панели*

В классе NewsAdmin описывается, по каким полям будет происходить фильтрация строк (*list\_filter),* какие полябудут выводиться в списке (*list\_display*), дополнительный admin actions, которые можно применять для нескольких строк таблицы (actions), по каким полям будет происходить поиск нужной строки в таблице (*search\_fields),* какие полябудут выводиться для отдельного объекта (*fieldsets*). Для некоторых моделей указываются поля, которые нельзя редактировать в админ панели (*readonly\_fields*). Некоторые классы наследуются от класса *ImportExportActionModelAdmin*, который позволяет использовать import и export данных для конкретных таблиц. Вышеописанные атрибуты используются и в описании вывода в админ панель и других моделей БД.

## **Кейсы использования приложения**

1) Просмотр списка новостей, связанных с сервисом такси

2) Оформление заказа пользователем

3) Изменение статуса заказа водителем

4) Оформление заказа оператором (когда пользователь решил оформить заказ через звонок оператору)

5) Изменение статуса заказа оператором

6) Регистрация нового водителя администратором

7) Регистрация нового оператора администратором

8) Оценивание водителя пользователем

9) Отмена заказа пользователем

10) Оставление комментария о водителе пользователем

## **Реализация REST API**

Новости: <http://taxi-service.std-884.ist.mospolytech.ru/api/v1/news/>

GET запрос – выводит список всех новостей со статусом published

GET запрос /news/<int pk> - выводит данные конкретной новости

Водители: <http://taxi-service.std-884.ist.mospolytech.ru/api/v1/drivers/>

GET запрос – выводит список водителей (доступен только операторам и администраторам)

GET запрос /drivers/<int pk> - выводит данные конкретного водителя (доступен всем авторизованным пользователям)

PUT /drivers/<int pk> запрос с параметром driverStatus – изменение статуса водителя (доступен только водителям, операторам и администраторам)

Операторы: <http://taxi-service.std-884.ist.mospolytech.ru/api/v1/operators/>

GET запрос – выводит список операторов (доступен всем авторизованным пользователям)

GET запрос /operators/<int pk> - выводит данные конкретного водителя (доступен всем авторизованным пользователям)

PUT /operators/<int pk> запрос с параметром operatorStatus – изменение статуса водителя (доступен только операторам и администраторам)

Заказы: <http://taxi-service.std-884.ist.mospolytech.ru/api/v1/orders/>

GET запрос – выводит список заказов (доступен водителям, операторам, администраторам)

PUT /orders/<int pk> запрос с параметром orderStatus – изменение статуса заказа (доступен только водителям, операторам и администраторам)

POST запрос с параметрами "discount", "town", "street", "house", "entrance", "destinationTown", "destinationStreet", "destinationHouse", "scheduledTime" – добавление нового заказа (доступен всем авторизованным пользователям)

GET запрос /orders/<int pk> - выводит данные конкретного заказа (доступен всем авторизованным пользователям)

Оценки водителей: <http://taxi-service.std-884.ist.mospolytech.ru/api/v1/drivers-raiting/>

GET запрос – выводит список оценок

POST запрос с параметрами driver (id водителя) и raiting (оценка) – добавление новой оценки водителю (доступен всем авторизованным пользователям)

Комментарии пользователей: <http://taxi-service.std-884.ist.mospolytech.ru/api/v1/drivers-comments/>

GET запрос – выводит список комментариев

POST запрос с параметрами driver (id водителя) и text (текст комментария) – добавление нового комментария (доступен всем авторизованным пользователям)

Список цен: <http://taxi-service.std-884.ist.mospolytech.ru/api/v1/price-list/>

GET запрос – выводит список цен

<http://taxi-service.std-884.ist.mospolytech.ru/api/v1/auth/token/login/> - POST запрос для входа в учетную запись с параметрами email и password

<http://taxi-service.std-884.ist.mospolytech.ru/api/v1/auth/users/> - POST запрос для регистрации пользователя с параметрами name, surname, email, password, re\_password

<http://taxi-service.std-884.ist.mospolytech.ru/api/v1/auth/token/logout/> - POST запрос для выхода из учетной записи (передается только токен)

<http://taxi-service.std-884.ist.mospolytech.ru/api/v1/auth/users/set_password/> - POST запрос для смены пароля с параметрами current\_password, new\_password

В запросах, которые требуют проверки на авторизацию и тип пользователя вместе с параметрами отправляется заголовок Authorization с токеном пользователя.

## **Реализация импорта и экспорта данных в базу данных**

Для реализации импорта и экспорта данных была использована библиотека django-import-export. Для использования данной библиотеки при описании моделей БД в админ панели использовался класс ImportExportActionModelAdmin, а также класс ModelResource - подкласс ресурсов для обработки моделей Django.

# **ЗАКЛЮЧЕНИЕ**

## **Результат**

В ходе данного проекта была разработана структура базы данных, реализованы админ панель с использованием Django и Django Rest Framework, и база данных на СУБД SQLite. Для данного сервиса заказа такси была создана landing страница с описанием всех особенностей данного проекта и его видео демонстрацией.

## **Ссылки на итоговые материалы**

Лэндинг проекта

<http://taxi-service-landing.std-884.ist.mospolytech.ru/>

Сайт

<http://taxi-service.std-884.ist.mospolytech.ru/>

Админ панель

<http://taxi-service.std-884.ist.mospolytech.ru/admin/>

GitHub репозиторий

<https://github.com/nikita220800/taxi-service/>

Видео демонстрация

<https://www.youtube.com/watch?v=2T6aAD1Wugk&feature=emb_logo>

# **СПИСОК ИСТОЧНИКОВ**

Серия статей для изучения Postman

<https://medium.com/@bullet4myv13>

Статья по настройке Django админ панели

<https://pythonist.ru/kastomizacziya-admin-paneli-django/>

Официальная документация Django

<https://docs.djangoproject.com/en/3.0/>

Разделы для настройки административной панели

<https://docs.djangoproject.com/en/3.0/ref/contrib/admin/>

<https://docs.djangoproject.com/en/3.0/ref/contrib/admin/actions/>

Статья о том, как обрабатываются запросы в Django

<https://django.fun/tutorials/put-ot-request-do-response-v-django/>