# **Федеральное государственное автономное образовательное учреждение высшего образования**

# **«Национальный исследовательский университет «Высшая школа экономики» Факультет компьютерных наук**

# 

# 

# 

# 

# **О Т Ч Е Т**

# **по дисциплине**

# **«Теория Баз Данных»** **Perevozka**

# 

# 

# 

# 

# **Выполнили студенты 184 гр**

# **Юдин Виктор**

# **Зюляев Илья**

# 

# **(ФИО)**

# **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# 

# ***(подпись)***

# 

# 

# 

# 

# **Проверил:**

# 

# *(должность, ФИО руководителя практики)*

# 

# 

# *17.12.2020*

# *(дата)*

# 

# 

# 

# 

# **2020 год**

## Содержание

[1. Описание предметной области](#_rizvf83fj4qk)

[2. Описание процесса построения инфологической модели](#_w99e5fqx8im9)

[ER-диаграмма (нотация Чена)](#_2raixz456qtk)

[3. Реляционная модель](#_yb2527dwdp5t)

[Процесс перехода к реляционной модели](#_lbhxthotjca8)

[TR-диаграмма (нотация Чена)](#_7gvkogy8nijh)

[Словарь данных](#_vau20oomufon)

[4. Развертывание БД](#_50tpzz44chlk)

[DDL код создания](#_yarpwia9n9f2)

[Пример DML операторов вставки](#_b8gfc9egubfz)

[Пример триггера](#_uekc5dx8uwzp)

[5. Разработка клиентского приложения](#_i00fspfmunah)

[Архитектура](#_dqhzn0y4z4f9) (коротко про Django)

[Сценарии использования](#_ufns6eff5rul)

[Организация доступа к данным и Интерфейс с пользователем](#_2rlal5ojolkd)

[Примеры запросов](#_pj7ggffxvhc1)

Ограничение целостности

[6. Заключение](#_bykw0jboud7s)

[Объемные характеристики разработки](#_2lkq7ch1ycg1)

[Авторский вклад и комментарии по выполнению проекта](#_dy3fxzuj48kd)

# 1. Описание предметной области

В качестве предметной области мы рассмотрели такой кейс: Крупная компания, занимающаяся грузовыми перевозками. Специализация – перевозка сборных грузов по всей России. Услуги компании пользуются спросом благодаря минимальным срокам доставки груза, широкой филиальной сети, высокому уровню сервиса и оптимальной ценовой политике. Доставка осуществляется посредством автотранспорта. У вас есть собственные автопарки. Для перевозки принимаются как небольшие, так и крупногабаритные грузы весом до 20 тонн. Компания позволяет следить за перевозкой груза т.к. груз зачастую перевозится через промежуточные пункты сортировки.

В качестве итогового визуального решения (интерактивного) реализовали полноценныйы вебсервис с помощью Angular + Django который в онлайне позволяет не только менять базу данных, а и имитировать работу компании (перевозку грузов) и следить за путем перевозки.

# 2. Описание процесса построения инфологической модели

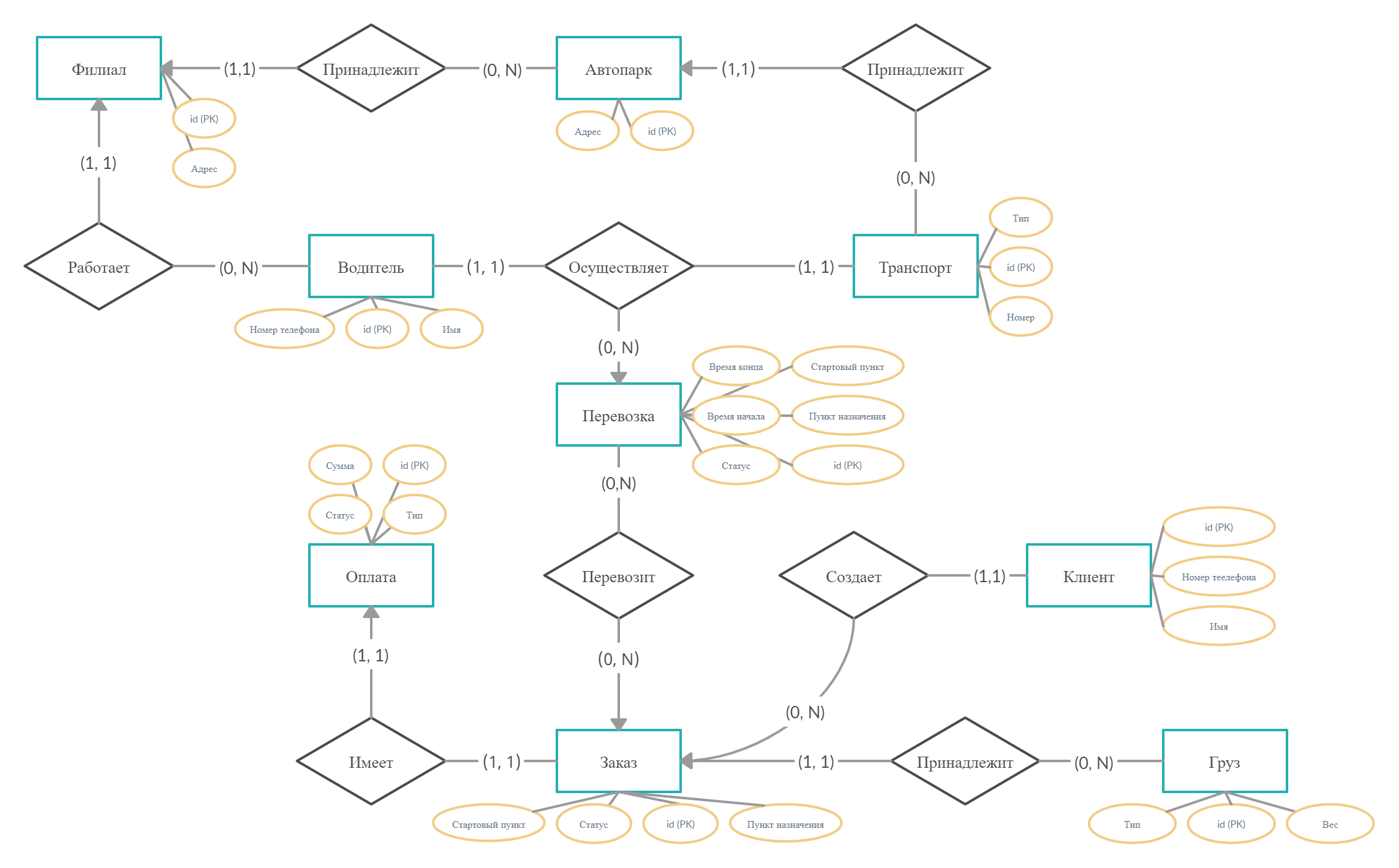
В процессе создания инфологической модели было выделено 9 сущностей. Для каждой из них заведем свой уникальный идентификатор. Уникальность подразумевается в рамках одной сущности, между объектами двух разных сущностей ключи могут совпадать. Поле “ключ” в описании сущностей опустим для простоты. А ссылки на другие таблицы будут видны из связей в дальнейшем. Давайте посмотрим на сущности и их атрибуты которые у нас получились:

* Сущность Клиент описывает клиента компании и включает в себя поля: имя клиента, номер телефона.
* Сущность Заказ описывает заказ который оставил клиент и содержит сущности: статус заказа, ключ клиента (который оставил заказ), пункт отправления и пункт назначения.
* Сущность Груз описывает груз который будет доставлен в рамках одного из заказа. В качестве атрибутов содержит: название груза, вес, тип.
* Сущность Оплата содержит информацию о оплате заказа. Содержит атрибуты: сумма, статус, тип.
* Сущность Перевозка - самая интересная сущность нашего проекта. Она содержит информацию о перевозке заказов между двумя пунктами. Т.е. перевозка осуществляется между двумя пунктами и некоторое кол-во заказов перемещается между этими пунктами (т.е. общий процесс доставки разбивается на такие мелкие перевозки между двумя соседними пунктами). Данная сущность содержит атрибуты: время отправки, время прибытия, стартовый пункт, пункт назначения, статус.
* Сущность Водитель описывает водителя, который занимается перевозкой заказов и включает в себя поля: имя, номер телефона.
* Сущность Филиал описывает филиал нашей компании и содержит поле адрес.
* Сущность Автопарк содержит информацию об автопарках и имеет единственное поле адрес.
* Сущность Транспорт содержит информацию о транспорте с помощью которого осуществляются перевозки. Имеющиеся атрибуты: серийный номер, тип.

Также в нашей базе данных представлены три типа вида связей: один-ко-многим, многие-ко-многим, один-к-одному (Одна из связей при этом тернарная). Рассмотрим их подробнее:

* Связь Создает (один ко многим) между Клиентом и Заказом означает, что клиент мог создать много заказов, а у заказа всегд есть определенный заказчик.
* Связь Принадлежит (один ко многим) между Заказом и Грузом показывает, что в заказе может быть много грузов, а конкретный груз содержится в конкретном заказе.
* Связь Имеет (один к одному) между Оплатой и Заказом показывает, что у заказа есть единственная информация об оплате и конкретная оплата происходит по определенному заказу.
* Связь Перевозит (многие ко многим) между Заказом и Перевозкой означает, что у заказа может быть несколько перевозок т.е. заказ побывает в промежуточных пунктах перед тем как прибыть на место назначения, при этом в одной перевозке может быть несколько заказов ведь за раз водитель перевозит сразу много заказов между двумя пунктами.
* Тернарная связь Осуществляется (один к одному ко многим) между Транспортом, Водителем и Перевозкой показывает, что перевозка осуществляется конкретным водителем на конкретном транспорте, однако в общем какой-то водитель и какой-то транспорт в разное время могут осуществлять разные перевозки.
* Связь Работает (один ко многим) между Филиалом и Водителем означает, что в филиале работает много водителей, а водитель числится за конкретным филиалом.
* Связь Принадлежит (один ко многим) между Филиалом и Автопарком означает, что у филиала может быть много автопарков, а определенный автопарк управляется определенным филиалом.
* Связь Принадлежит (один ко многим) между Автопарком и Транспортом показывает, что у автопарка много различного транспорта, а конкретный транспорт числится за конкретным автопарком.

## ER-диаграмма (нотация Чена)



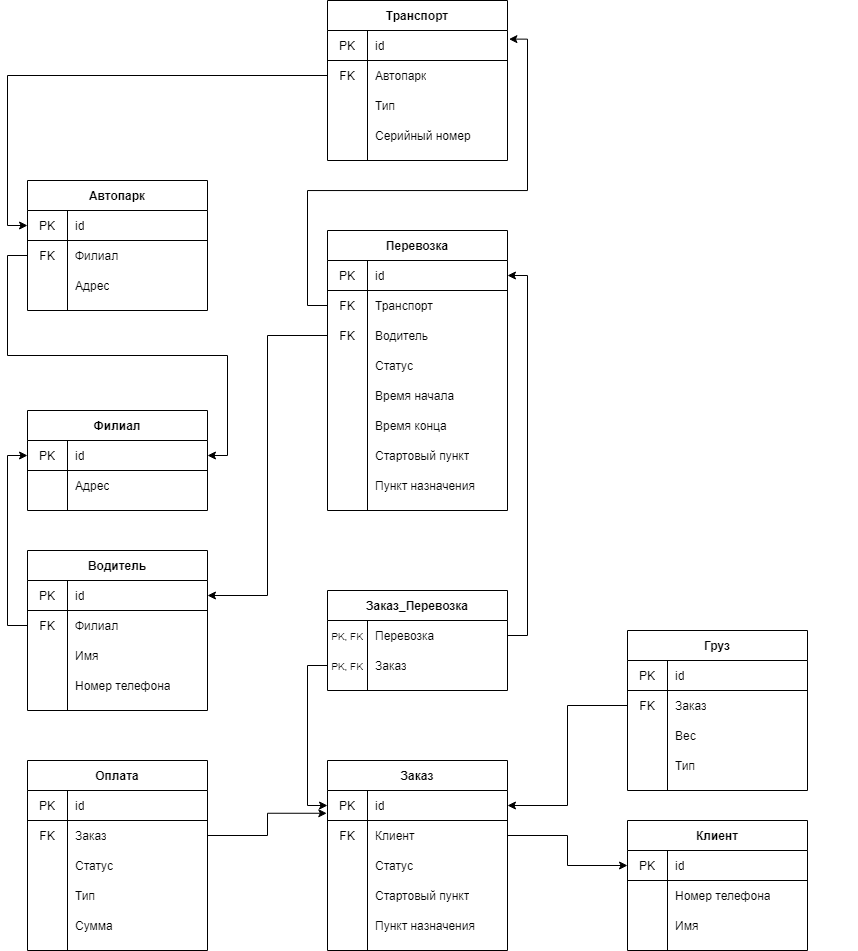
Ссылка на ER-диаграмму на GitHub: [ссылка](https://github.com/yudinvictor/DBMS/blob/master/ER.jpg)

# 3. Реляционная модель

## Процесс перехода к реляционной модели

При переходе к реляционной модели мы установили ограничения на связи. Для каждой связи в таблицах были установлены внешние ключи и ограничения множественности. Были добавлены таблицы пар ключей для реализации вида многие-ко многим. При связи один-к-одному внешний ключ помещался только в одну из таблиц (чтобы не было избыточности).

## TR-диаграмма (нотация Чена)



## Словарь данных

Филиал, Автопарк, Водитель, Транспорт, Оплата, Заказ, Груз, Клиент, Перевозка

# 4. Развертывание БД

## DDL код создания

CREATE TABLE branch (

id INT NOT NULL PRIMARY KEY,

address text NOT NULL

);

CREATE TABLE car\_park (

id INT NOT NULL PRIMARY KEY,

address text NOT NULL,

branch\_id INT NOT NULL REFERENCES branch (id)

);

CREATE TABLE client (

id INT NOT NULL PRIMARY KEY,

name VARCHAR(255) NOT NULL,

phone\_number VARCHAR(255) NOT NULL

);

CREATE TABLE driver (

id INT NOT NULL PRIMARY KEY,

name VARCHAR(255) NOT NULL,

phone\_number VARCHAR(255) NOT NULL,

branch\_id INT NOT NULL REFERENCES branch (id)

);

CREATE TABLE order (

id INT NOT NULL PRIMARY KEY,

status VARCHAR(255) NOT NULL,

departure\_address text NOT NULL,

destination\_address text NOT NULL,

client\_id INT NOT NULL REFERENCES client (id)

);

CREATE TABLE transport (

id INT NOT NULL PRIMARY KEY,

type VARCHAR(255) NOT NULL,

number VARCHAR(255) NOT NULL,

car\_park\_id INT NOT NULL REFERENCES car\_park (id)

);

CREATE TABLE shipping (

id INT NOT NULL PRIMARY KEY,

start\_time timestamp NOT NULL,

stop\_time timestamp NULL,

departure\_address text NOT NULL,

destination\_address text NOT NULL,

status VARCHAR(255) NOT NULL,

driver\_id INT NOT NULL REFERENCES driver (id) ,

transport\_id INT NOT NULL REFERENCES transport (id)

);

CREATE TABLE shipping\_orders (

shipping\_id INT NOT NULL REFERENCES shipping (id) ,

order\_id INT NOT NULL REFERENCES "order" (id) ,

PRIMARY KEY (shipping\_id, order\_id)

);

CREATE TABLE payment (

id INT NOT NULL PRIMARY KEY,

amount FLOAT NOT NULL,

type VARCHAR(255) NOT NULL,

status VARCHAR(255) NOT NULL,

order\_id INT NOT NULL UNIQUE REFERENCES "order" (id)

);

CREATE TABLE cargo (

id INT NOT NULL PRIMARY KEY,

weight FLOAT NOT NULL,

type VARCHAR(255) NOT NULL,

order\_id INT NOT NULL REFERENCES "order" (id)

);

## Пример DML операторов вставки

INSERT INTO Order

(status, departure\_address, destination\_address, client)

VALUES ('Создан', 'Москва', 'Сочи',

(SELECT id FROM Client WHERE name = 'Вася' LIMIT 1))

INSERT INTO Payment(amount, type, status, order)

VALUES (1000, 'Apple Pay', 'Ожидание оплаты', 1)

## Пример триггера

CREATE TRIGGER tr\_order ON Order

AFTER INSERT

DECLARE

BEGIN

INSERT INTO Payment(amount, type, status, order)

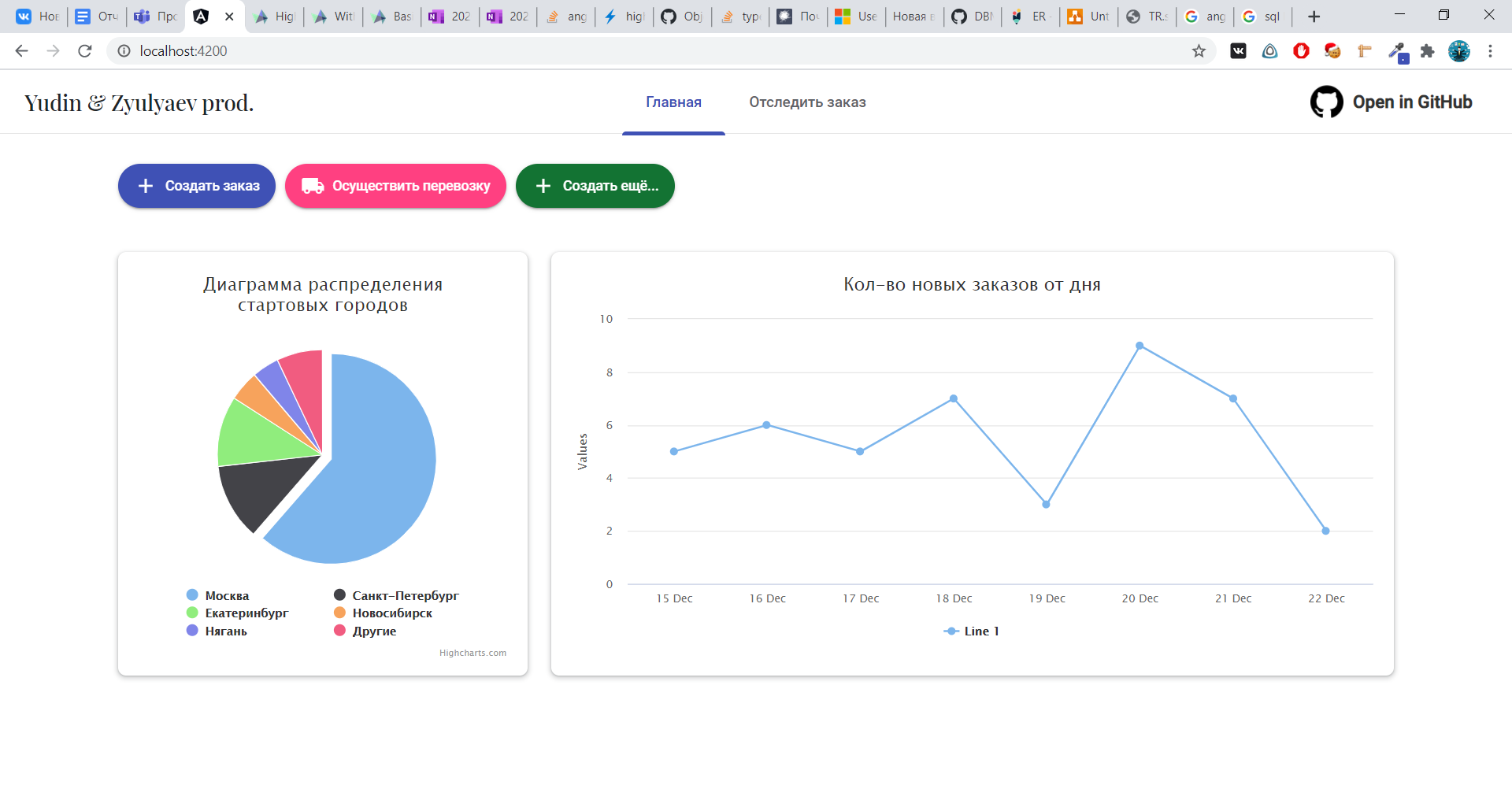
VALUES (1000, 'Тип не выбран', 'Ожидание оплаты', (SELECT id FROM inserted))

END

# 5. Разработка клиентского приложения

Скорее всего ссылка на рабочую версию на гите в README. Если ссылки нет/по ней не открывается – свяжитесь с нами.

Главная страница:

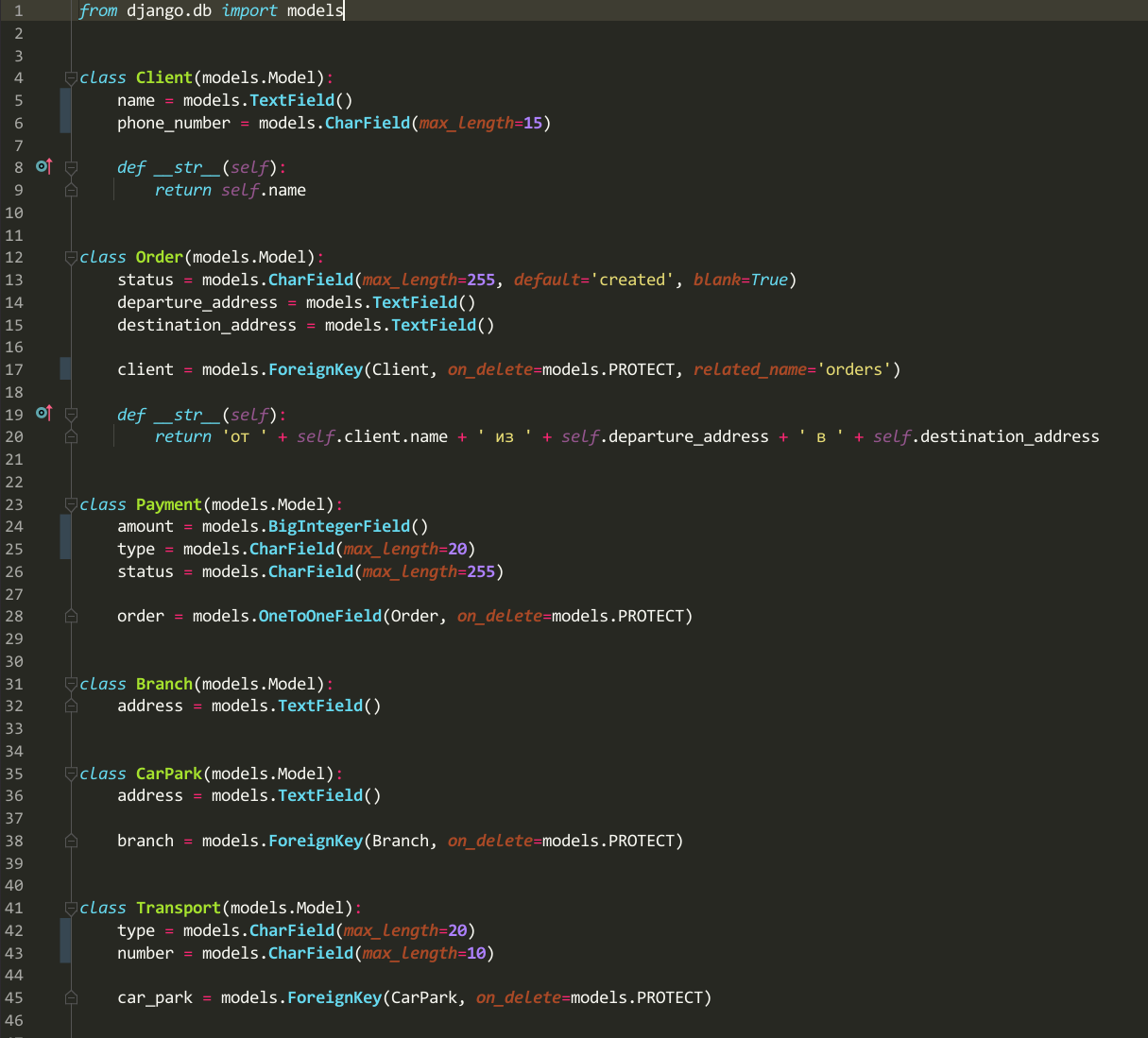


## Архитектура (коротко про Django)

В основе нашего вебсайта лежит Angular 10 + Django. При разработке в качестве БД использовалась SQLite, при развертывании PostgreSQL.

Фреймворк Django позволяет упростить и автоматизировать работу с БД. При использовании Django можно сильно снизить количество SQL кода в проекте, и использовать его только в тех местах, где на ваш взгляд Django делает не самые оптимальные запросы.

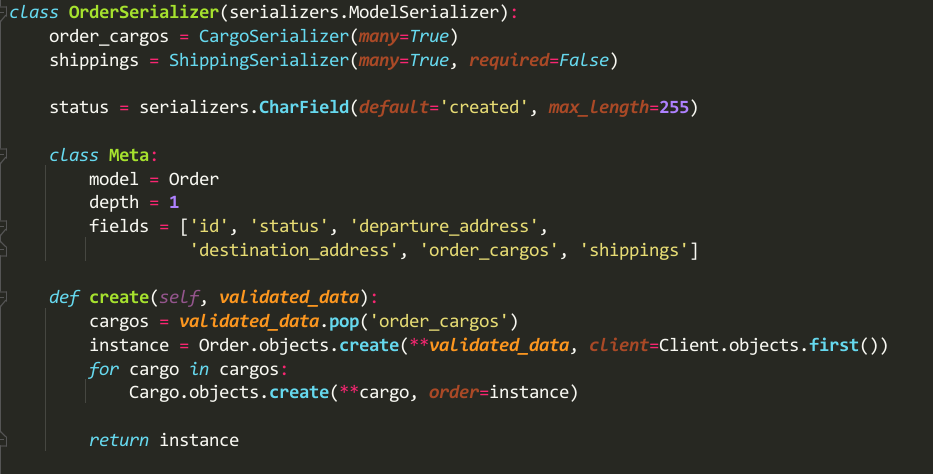
Также помощью Django можно создать необходимые таблицы в базе данных используя только Python код (все отношения и ограничения целостности также задаются через Python):

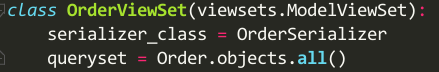




Кроме того библиотека rest\_framework помогает в создании REST API и реализации концепции CRUD.

К примеру можно достаточно легко реализовать создание нового заказа с новыми грузами, которые будут приходить от фронтенда в json формате:







И теперь при GET запросе по адресу /orders/ будет возвращаться JSON со всеми заказами, а при GET запросе по адресу /orders/id/ будет возвращаться информация конкретно по заказу с данным id.

Кроме того можно посылать POST запросы на создание нескольких грузов и нового заказа:

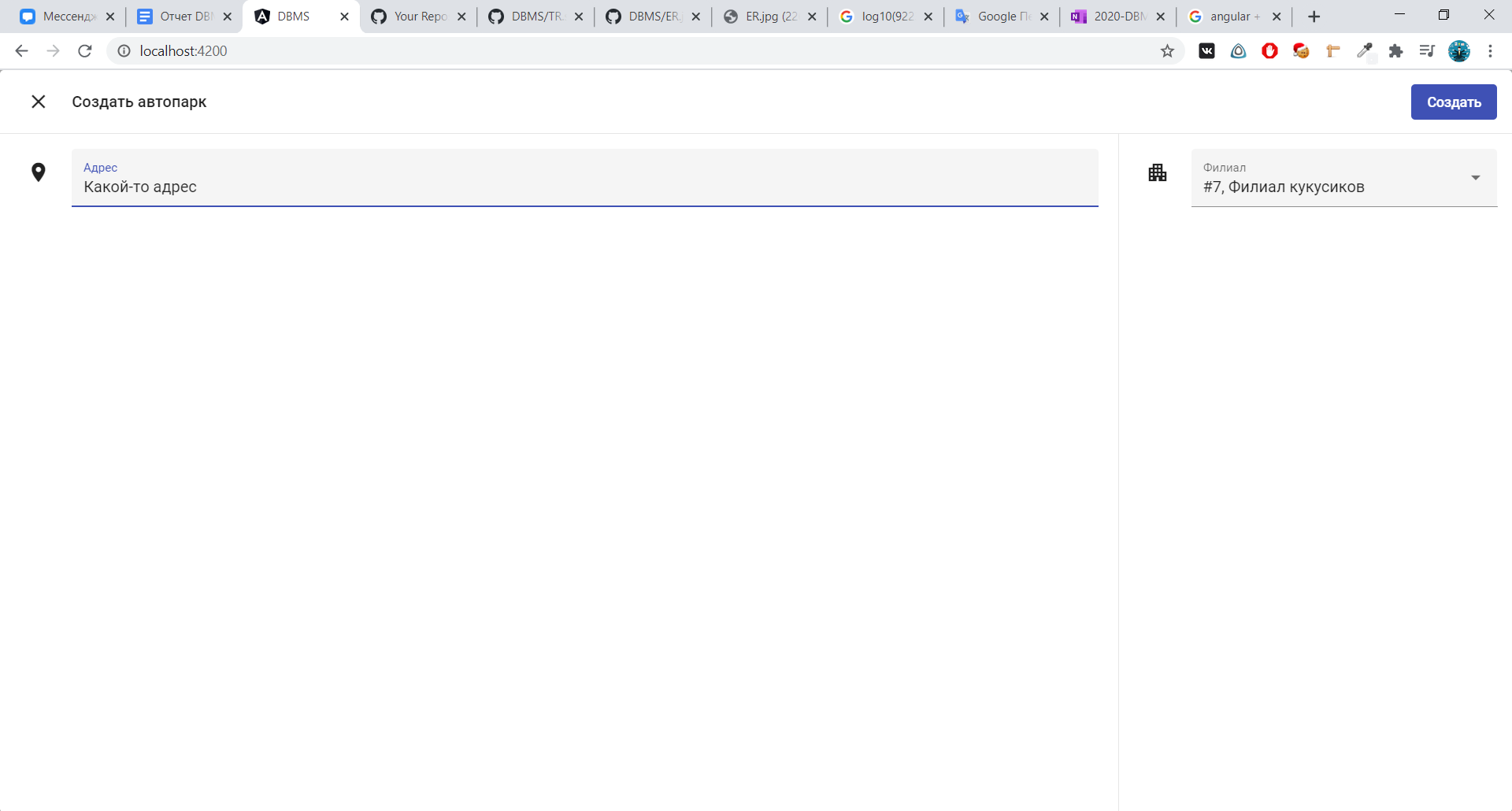
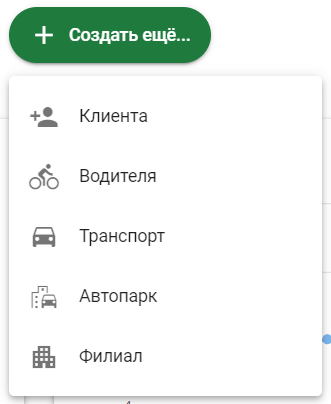


## 

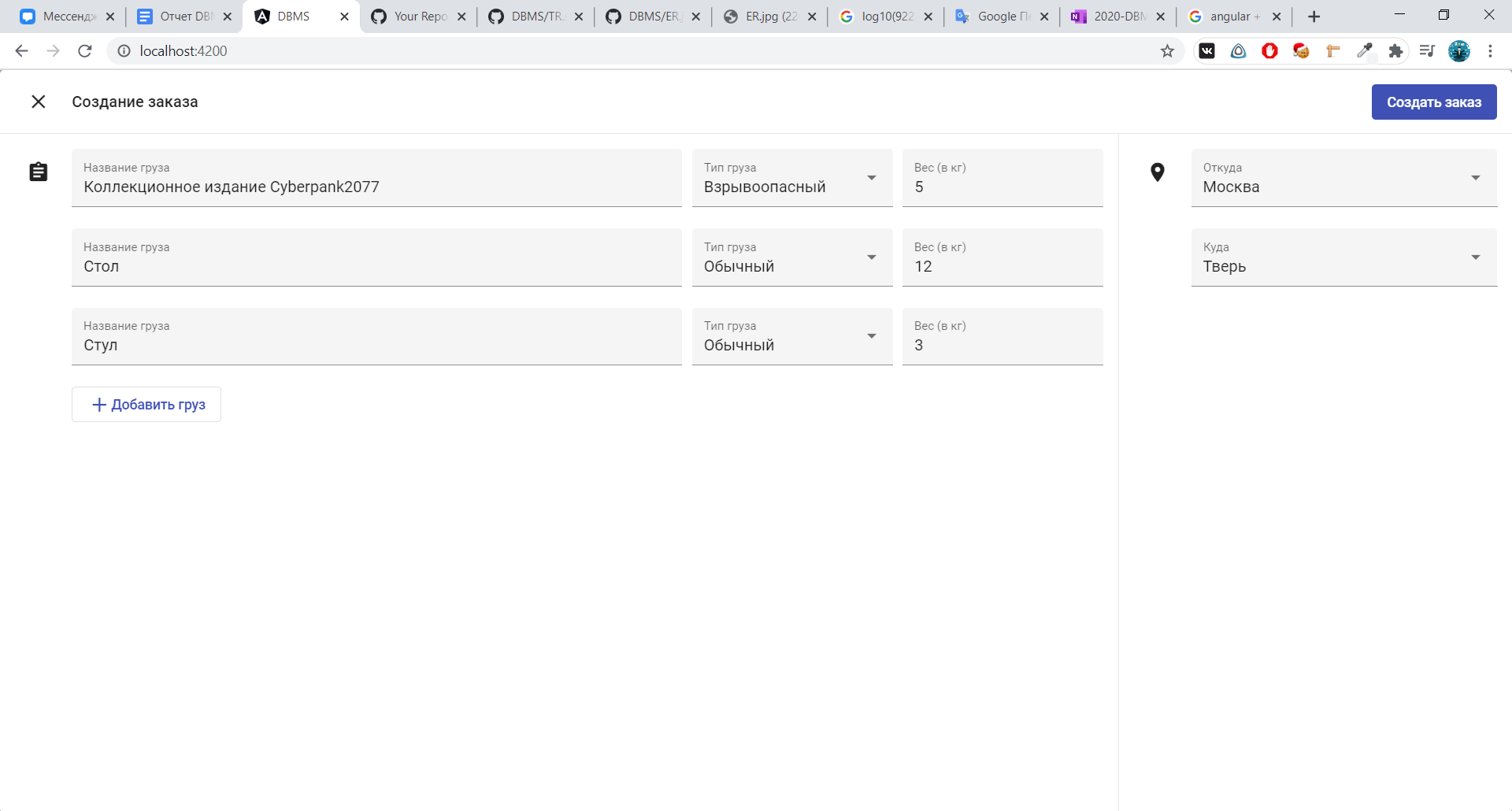
## Сценарии использования

Мы создали основу для полноценного SaaS, которая позволяет регистрировать новых пользователей, транспорт, водителей, создавать заказы, осуществлять перевозки заказов между городами и отслеживать их путь. Важный момент: мы не прокладываем сам маршрут доставки, мы лишь отслеживаем маршрут и т.д. по предоставленным данным.

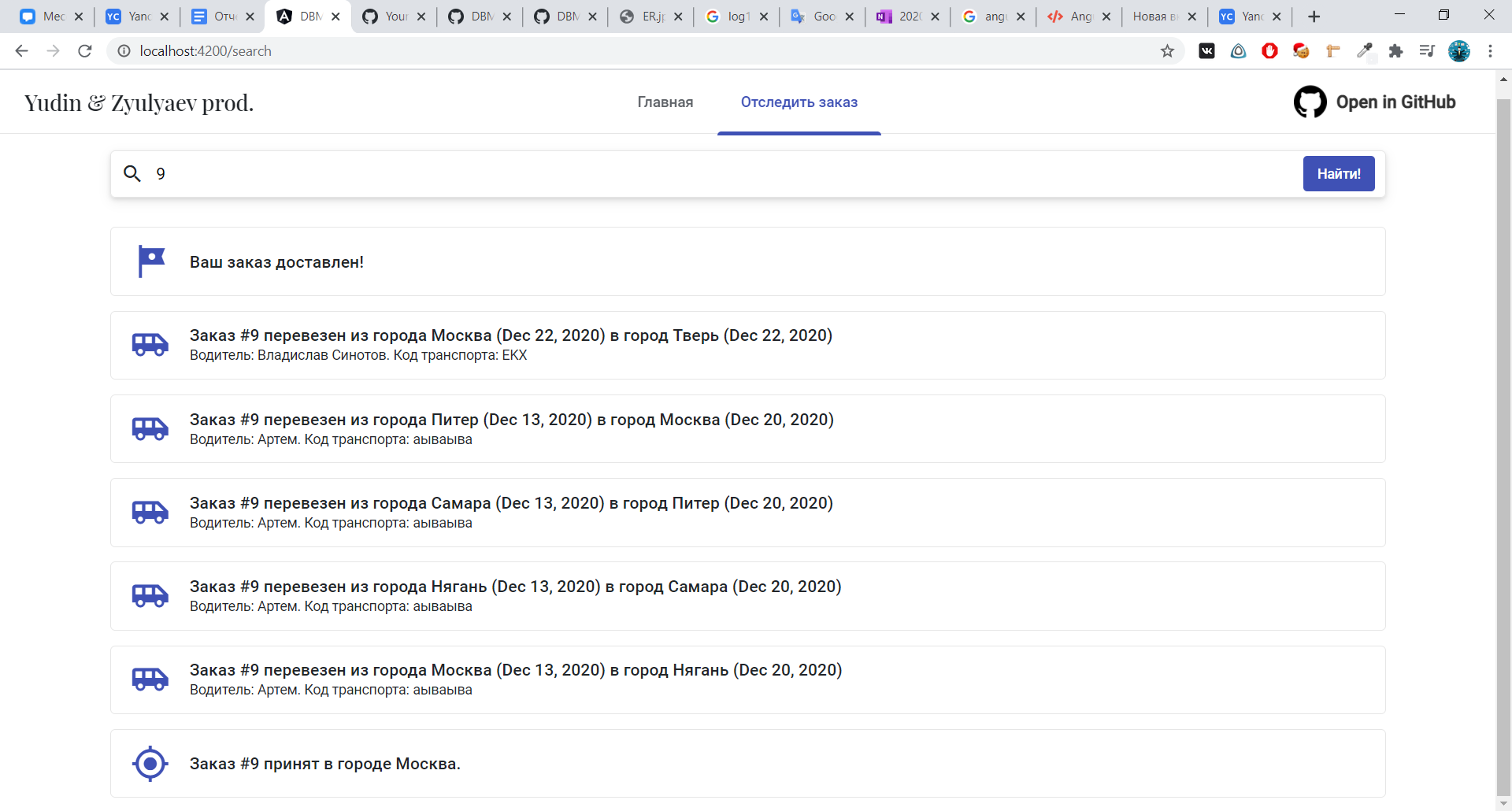
1. Сначала компания вносит необходимые сущности: филиалы, автопарки, водителей и т.д. в соответствующих разделах создания. Заполняет все необходимые поля (например для автопарка это адрес и филиал(из уже созданных)) и нажимает кнопку “Создать”. Если все сделано правильно будет показано сообщение об успехе.



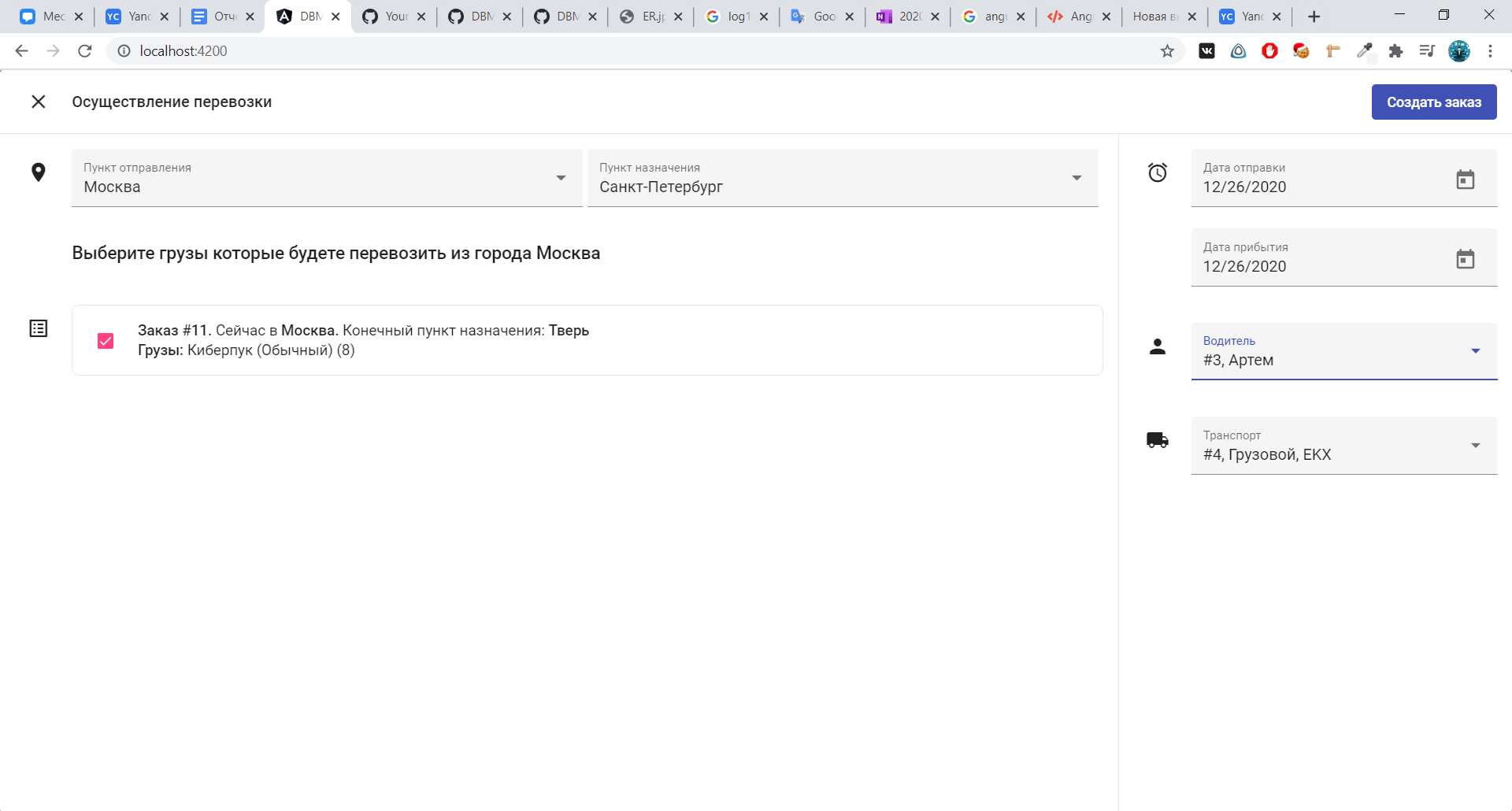
1. Теперь клиент/отдел продаж может создать новый заказ, нажав “Создать заказ”. Необходимо добавить грузы, а также откуда и куда нужна транспортировка. После нажатия кнопки Создать заказ добавится в базу данных.



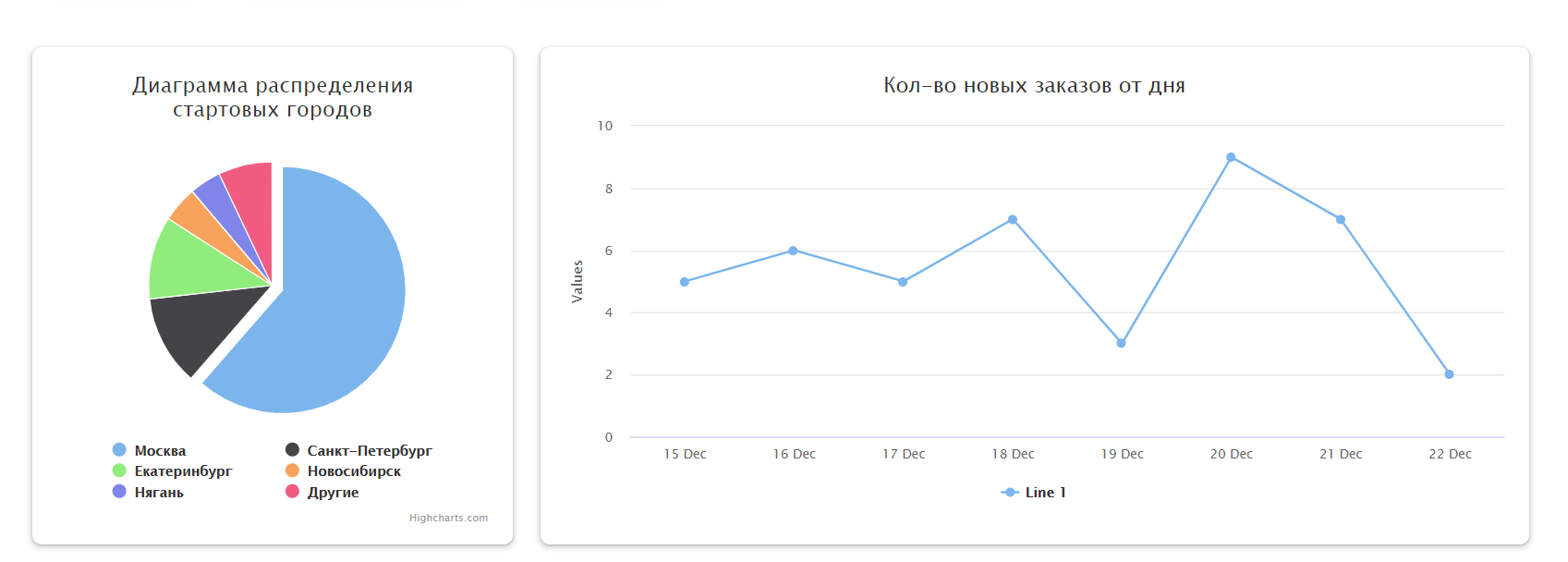
1. Любой (например сам клиент) может отследить путь своего заказа во вкладке Отследить. Достаточно ввести номер заказа и мы покажем все перевозки связанные с заказом (а также кто и на каком транспорте осуществлял ту или иную перевозку) и текущий статус.



1. Водитель/компания может осуществить перевозку нажав Осуществить перевозку. Для начала стоит выбрать водителя и транспорт. Затем выбрав начальный город мы покажем список доступных для транспортировки из него грузов. Необходимо выбрать 1 или несколько заказов, указать пункт назначения и время прибытия. После всего указанного, нажав кнопку Перевести мы зарегистрируем эту перевозку, обновим местоположения и статусы перевезенных заказов.



Также мы можем отображать любую необходимую статистику на красивых графиках и диаграммах.



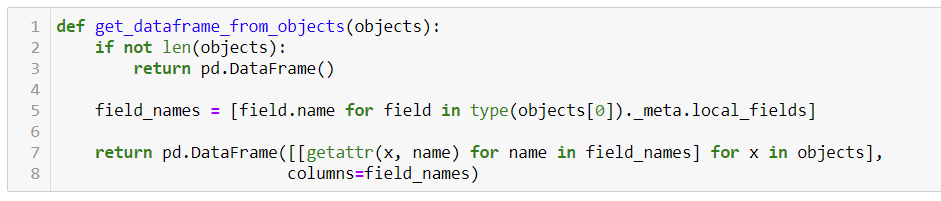
## Организация доступа к данным и Интерфейс с пользователем

Сейчас доступ к данным осуществляется либо через интерфейс сайта, либо есть возможность предоставить открытое api для запросов (Однако для получения доступа необходимо заранее с нами связаться чтобы мы могли настроить порты и cors).

Реализован симпатичный UI Material интерфейс (Скриншоты были в пунктах выше).

## Примеры запросов

Для визуализации результатов запросов напишем простую функцию, которая принимает массив объектов из моделей django и возвращает pandas.DataFrame со всеми их полям:



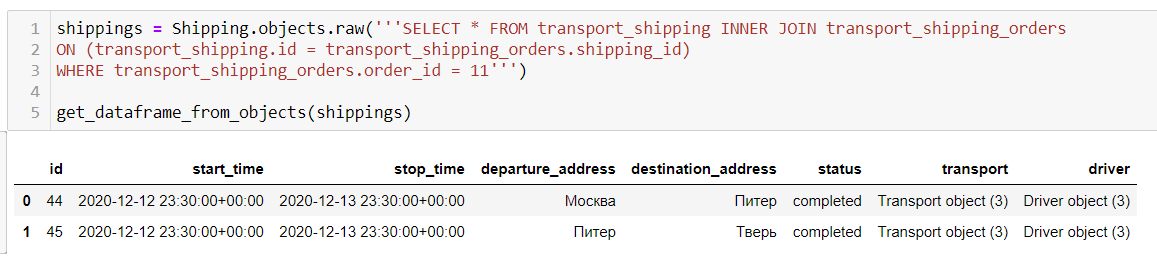
По заказу (ORDER\_ID) находим все перевозки в которых он участвовал

SELECT \* FROM shipping INNER JOIN shipping\_orders ON

(shipping.id = shipping\_orders.shipping\_id)

WHERE shipping\_orders.order\_id = ORDER\_ID

Результат запроса (здесь и далее перед названием таблиц используется префикс **transport\_** ):



По клиенту (CLIENT\_ID) находим все заказы, которые когда либо были в Москве

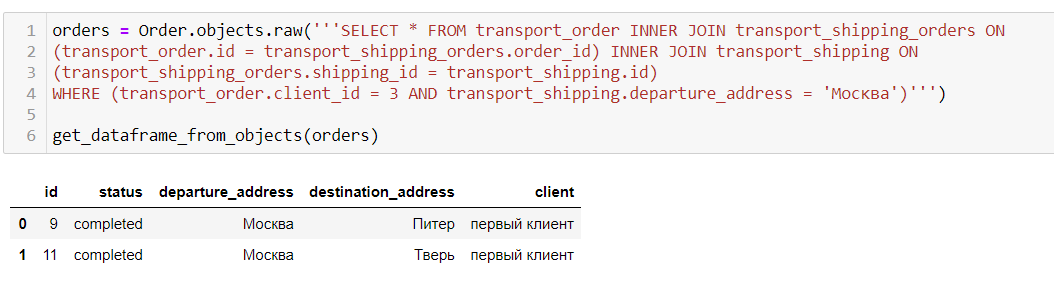
SELECT \* FROM order INNER JOIN shipping\_orders ON

(order.id = shipping\_orders.order\_id) INNER JOIN shipping ON

(shipping\_orders.shipping\_id = shipping.id)

WHERE (order.client\_id = CLIENT\_ID AND shipping.departure\_address = 'Москва')

Результат запроса:



По водителю (DRIVER\_ID) находим все грузы, больше 100 кг, которые он перевозил

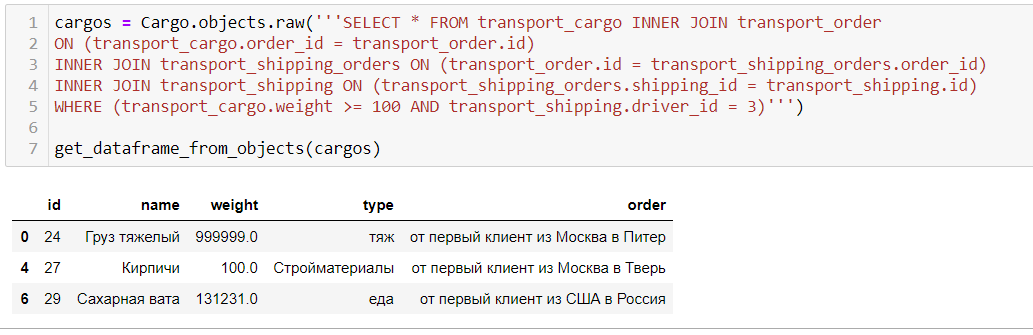
SELECT \* FROM cargo INNER JOIN order ON (cargo.order\_id = order.id)

INNER JOIN shipping\_orders ON (order.id = shipping\_orders.order\_id)

INNER JOIN shipping ON (shipping\_orders.shipping\_id = shipping.id)

WHERE (cargo.weight >= 100 AND shipping.driver\_id = DRIVER\_ID)

Результат запроса:

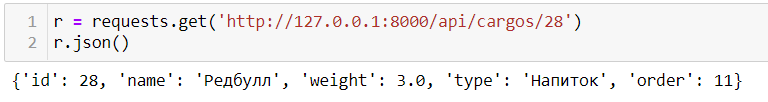
Также приведем два примера как django позволяет упростить запросы к БД:

Кидаем запрос к нашем api на получение всех грузов:



И получаем json с которым потом удобно работать.

Также можем получить информацию по конкретному грузу по id:



Кроме того, с помощью rest\_framework (библиотеки для django) можно достаточно легко получать более сложную информацию, например информацию про перевозку по её id, в том числе и подробную информацию о вложенных полях (о заказах, водителе, транспорте):



## Ограничения целостности

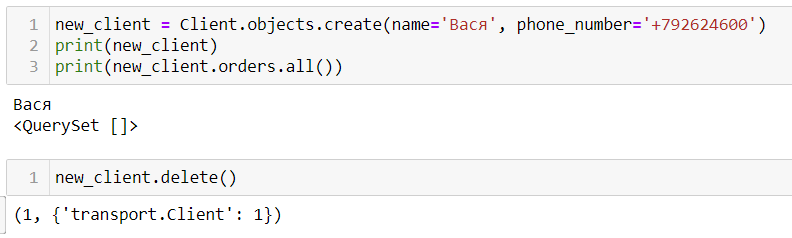
1) Для всех связей нам подошло значение параметра **ON DELETE** по умолчанию, а именно **NO ACTION**.

Связано это с тем, что к примеру при удалении клиента мы не хотим удалять его заказы, а также информацию об оплате этих заказов, так как эта информация достаточно важная, если мы говорим о реальном приложении. Кроме того не ставим в заказах поле клиента в null, так как опять же при реальном использовании, мы бы хотели анализировать какие заказы заказывают клиенты с определенными параметрами, поэтому мы не позволяем удалять объекты, на которые кто-то ссылается.

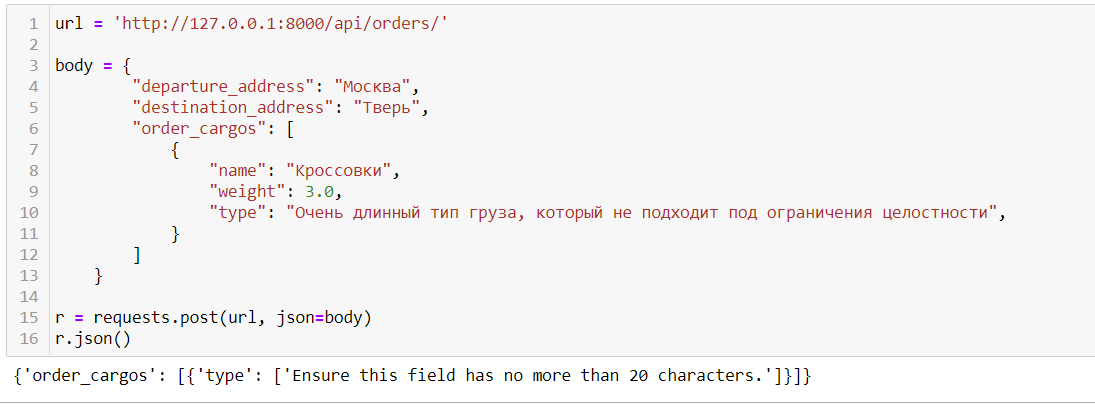
Пример попытки удалить клиента, у которого есть заказы:



И конечно клиент без заказов удаляется без проблем:



2) К примеру тип груза не может быть длиннее, чем 20 символов (VARCHAR(20)) символов. Попробуем создать заказ с грузом с неподходящим типом:



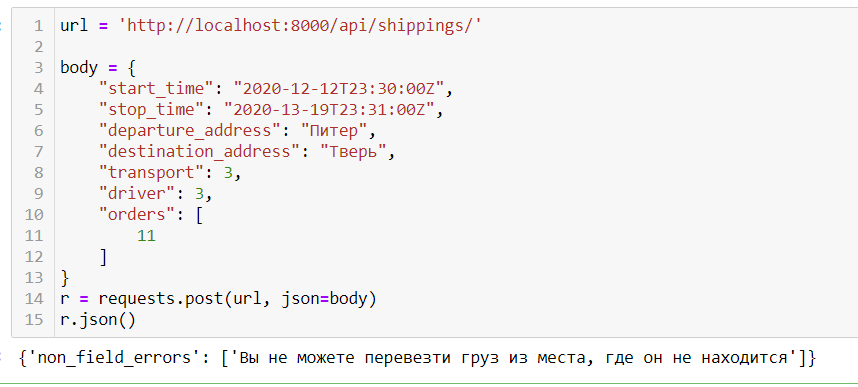
Как видим получаем ошибку с уведомлением об этом.

3) Проверка перевозок

Создадим заказ с несколькими грузами, с помощью запроса к реализованному на django API, который нужно перевезти из Москвы в Тверь



Теперь попробуем перевезти заказ из Питера в Тверь (хотя он находится в Москве):

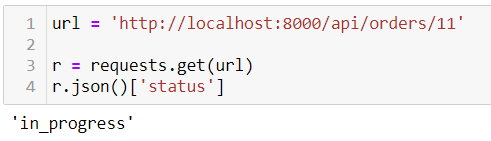


Как видим получаем ошибку, с уведомлением о том, что заказ в данный момент находится не там.

А вот из Москвы в Питер перевозка осуществляется без проблем:



Проверим статус нашего заказа:

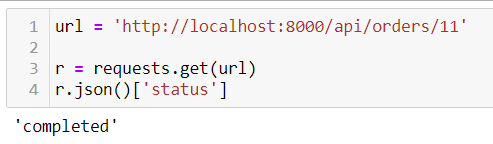


Как видим он **in\_progress**

Теперь перевезем заказ в пункт назначения (Тверь):



И еще раз проверим статус заказа:



Как видим он стал **completed**

# 6. Заключение

В итоге мы сделали основу для будущего SaaS с возможностью создания различных объектов, отслеживания грузов и наглядного просмотра статистики на основе Angular + Material + Highcharts + Django + PostgreSQL.

## Объемные характеристики разработки

Как только вы зайдете и пощупаете сайт - вы поймете масштаб работы) Говорить о кол-ве строк глупо потому что наш проект весит 500Мб без учета базы данных). Считаем что непомерный вклад в работу внесли Юдин Виктор и Зюляев Илья. Индусы на аутсорсе почти ничего не внесли.

## Авторский вклад и комментарии по выполнению проекта

Юдин Виктор и Зюляев Илья:

* Почти все: от создания моделей и проектирования БД до написания фронта и бэка.

Индусы на аутсорсе:

* Все остальное