# Стажировка

## Задание

Написать бэкэнд для моделирования работы автосалонов со своими покупателями и поставщиками.

## Принцип работы

1. Автосалоны

При создании автосалона указывается его имя, местоположение, характеристики тех автомобилей, которые автосалон предпочитает продавать в будущем, после создания по этим характеристикам находятся те модели автомобилей, которые наиболее подходят автосалону. Автосалон также должен иметь список автомобилей и количество каждой отдельной модели автомобиля. Покупка автомобилей происходит у поставщиков, которые будут подобраны по тем моделям, которые наиболее подходят (после процесса, который был описан выше). В итоге автосалон должен хранить свои характеристики (имя, месторасположение и т.д.), а также характеристики автомобилей, которые будут продаваться, по которым будут найдены соответствующие модели автомобилей и записаны в какое - то поле модели автосалона, затем по моделям автомобилей должны быть найдены соответствующие поставщики по самым выгодным ценам, т.к. поставщики могут продавать одни и те же модели автомобили, но цены могут отличаться - выбираем самые выгодные (может быть несколько поставщиков и к каждому поставщику будет указаны те модели, которые будут у них покупаться), также поле в модели автосалона. У автосалона также должен быть баланс. Автосалон также должен иметь историю продаж автомобилей. Автосалон должен хранить своих уникальных покупателей, которые будут записываться скриптом с соответствующем количеством покупок

автомобилей совершенных именно в данном автосалоне. Создать систему акций, при создании акций нужно указывать период проведения акции и на какие автомобили она распространяется и на какой автосалон он распространяется (также должны включаться информационные поля, такие как название, описание и т.п). Процент скидок индивидуален для каждого автосалона и соответственно может отличаться.

Celery запускает скрипт раз в 10 минут, который покупает автомобили у поставщиков (предположим, что у поставщиков автомобили бесконечны для облегчения), исходя из тех характеристик, которые были указаны выше и индивидуальны для каждого автосалона. Конечно же, так же должен проверяться баланс автосалона. Сначала должны покупаться автомобили, на которые имеется наибольший спрос (спрос вычисляется исходя из истории продаж автомобилей). При этом, при запуске скрипта, он должен проверять наличие каких - либо акций у поставщиков на автомобили, которые закупает какой - либо автосалон, если с учетом акции цена автомобиля будет меньше, чем у постоянного поставщика с учетом скидки постоянного клиента (у которого автосалон по стандарту покупает автомобиль), то закупить данную модель автомобиля нужно у поставщика, у которого происходит данная акция.

Celery запускает скрипт раз в час, который проверяет выгодность сотрудничества с поставщиками. Так как цена на автомобили у поставщиков может постоянно меняться или могут появляться новые поставщики, то нужно периодически проверять целесообразность сотрудничества с тем или иным поставщиком. Скрипт должен проверять цены на автомобили, которые покупает автосалон, у всех поставщиков (учитывая скидки постоянных клиентов) и обновлять список поставщиков, хранящийся в модели автосалона при нахождении более выгодной цены, чем у предыдущего поставщика. Скрипт также должен учитывать проценты и нужное количество покупок для скидок постоянных клиентов и выбирать наиболее выгодные (может быть ситуация, в данный момент цена более выгодная у одного поставщика, но при этом скидки лучше у другого и в будущей перспективе лучше выбирать другого поставщика и именно такая ситуация должна

проверяться, также нужно учитывать количество покупок, чтобы достичь выгодного процента скидок, так как если условно нужно совершить 5000 покупок для того, чтобы цена стала более выгодная, то в таком случае сотрудничество не совсем выгодно из - за огромного количества покупок).

2. Покупатель

У покупателя должен быть баланс, показывающий какое количество денег у него есть в данный момент. Также покупатель должен иметь историю своих совершенных покупок автомобилей. Покупка автомобиля происходит следующим образом: покупатель создает Offer, указывая максимальную цену автомобиля и указывая автомобиль, которым он интересуется. Затем Celery запускает скрипт, который по Offer пользователей по всем автосалонам ищет подходящие автомобили по наиболее выгодным ценам.

Требования к сделке: автомобиль должен быть тем, который был указан пользователем при создании Offer. Цена автомобиля должна быть меньше или равна той цене, которая была указана в Offer пользователем. Автомобиль действительно должен быть в наличии в автосалоне (количество данных автомобилей в автосалоне должно быть больше 0 - указано в разделе “Автосалоны”). У пользователя должно быть достаточно денег на балансе для осуществления покупки (Должно проверяться на стадии создания Offer).

У покупателя также должен быть профиль, в котором будет какая - то информация о пользователе, какую - то информацию пользователь должен вводить сам, а какая - то по типу истории покупок будет генерироваться автоматически скриптом.

3. Поставщик

Поставщик должен хранить информацию о себе (название, год основания, количество покупателей (дальше можете придумывать сами)), а также список автомобилей, которые продает данный поставщик и соответствующие цены к ним. Аналогично автосалону, у поставщиков должны быть реализованы системы акций и скидок

для постоянных покупателей. Поставщик должен иметь историю продаж автомобилей автосалонам.

**Примечание:** для облегчения работы будем считать, что все сделки происходят в USD. Выше описаны основные модели, которые будут задействованы в приложении, но это не означает, что это все модели, которые будут использованы. Ваша задача также из текста выделить еще какие - то сущности, которые было бы целесообразно вынести в отдельные модели.

# Требования

## Задание

1) Написать структуру базы данных, продумав все связи между моделями. Модели должны хранить boolean поле is_active, которое будет отвечать за то удален ли инстанс или нет, вместо фактического удаления из базы данных. Должны также храниться время последнего обновления и время создания инстанса. Используем PostgreSQL в качестве бд для приложения. Для локаций использовать django_countries.
2) Сразу на первых этапах подключить Docker и Docker Compose
3) Написать простейшее API для работы с моделями, причем продумать какие действие могут быть совершены с моделями, а какие нет. Например, вряд ли можно будет вручную обновлять баланс пользователю (возможно только если администратору), так как все должен обновлять скрипт, запущенный Celery по итогам совершенных сделок.
4) Реализовать JWT авторизацию.
5) Подключить Swagger для документирования эндпоинтов, подключить Django Debug Toolbar для того, чтобы отслеживать на сколько быстро идут запросы и дополнительную инфу.
6) Использовать django_filters для создания фильтрации по полям модели (наиболее целесообразные поля для фильтрации выбирайте сами). Также создать сортировку и поиск по полям.

7) Написать скрипты, которые будут производить покупки Покупатель -> Автосалон и Автосалон -> Поставщик (Запуск с помощью Celery + Celery Beat как планировщика)
8) Написать полноценную регистрацию с подтверждением на почту, с возможностью изменить пароль и восстановить пароль. Также возможность изменять свой логин / почту. Все должно происходить с подтверждением на почту. Причем пользователь сможет создавать какие - то Offer только после подтверждения своей почты.
9) Создать статистику по всем автосалонам / покупателям / поставщикам (как пример, количество проданных автомобилей автосалоном, количество прибыли, количество уникальных покупателей для автосалонов, для покупателей - количество потраченных денег, купленные автомобили. Примеры не исчерпывающие - статистика может быть какой угодно, которую вы сможете придумать самостоятельно)
10) Покрывать своей приложение тестами (желательно при добавлении какого - либо нового функционала, а не в конце разработки)
11) Подключить Gunicorn и Nginx к своему приложению.

**Примечание:** данные задания стоит выполнять в порядке их следования в данном списке, причем задания можно разделять на подзадания и скидывать отдельными Pull Request на ревью менторам.

## Технологии

1) Стек: Django REST Framework, PostgreSQL, Celery, Redis, Docker, Docker-compose, JWT authorization
2) PEP8 (https://pythonworld.ru/osnovy/pep-8-rukovodstvo-po-napisaniyu-koda-na-python.html)
3) Используйте flake8, mypy, pylint, black
4) Для тестов используйте PyTest
5) Class-based view
6) Использовать Mixins (ListModelMixin, RetrieveModelMixin, CreateModelMixin, GenericViewSet)
7) urls делать через routers (https://www.django-rest-framework.org/api-guide/routers/)
8) Git Flow (https://bitworks.software/2019-03-12-gitflow-workflow.html)

9) Не писать бизнес логику во view, выносить в отдельные сервисы
10) Сделать Docker, Docker-compose в первых этапах разработки и масштабировать его по мере необходимости
11) Для виртуального окружения использовать Pipenv
12) Использовать Django Toolbar
13) Использовать Swagger для документации своих эндпоинтов
14) Хорошая документация своего кода, используя docstrings к написанной логике.

## Процесс разработки

1) Создаем ветку develop
2) Для каждой задачи / фикса создаем отдельную ветку
3) По окончанию работы над задачей пушим на Github и создаем Pull Request в develop
4) Скидываем ссылку на Pull Request в чат с ментором и пингуем его и переходим к другой задаче
5) Исправляем недочеты, ждем апрува реквеста и мержим