###### Роль в проекте.

**Юзов Евгений Борисович**, студент Geekbrain, являюсь backend разработчиком на Django. Frontend создан на React, backend – на Django REST Framework. Также мной создана небольшая база данных на PostgreSQL для хранения промежуточных данных. Ссылка на backend проекта: <https://github.com/Uzov/djangoreactproject>

###### ***Цель проекта.***

Создать собственное приложение по поиску отелей для отдыха по России по примеру booking.com, ostrovok.ru, multitour.ru и др.

###### ***Описание проекта.***

Изначально команда разработчиков предложила найти уже готовый API для поиска гостиниц и их предложений в России. Мною был найден готовый API оператора Мультитур (<https://www.multitour.ru/api_documentation/>). Оператор любезно предоставил мне токен для доступа как студенту Geekbrain. Справочники городов, отелей и предложения отелей берутся из данного API. Преимущества изначального выбора стороннего API:

* cтала понятна структура и алгоритм работы приложения;
* frontend разработчики смогли сразу включиться в разработку, не дожидаясь создания от backend создания БД со справочниками городов, отелей, предложений отелей и пр.

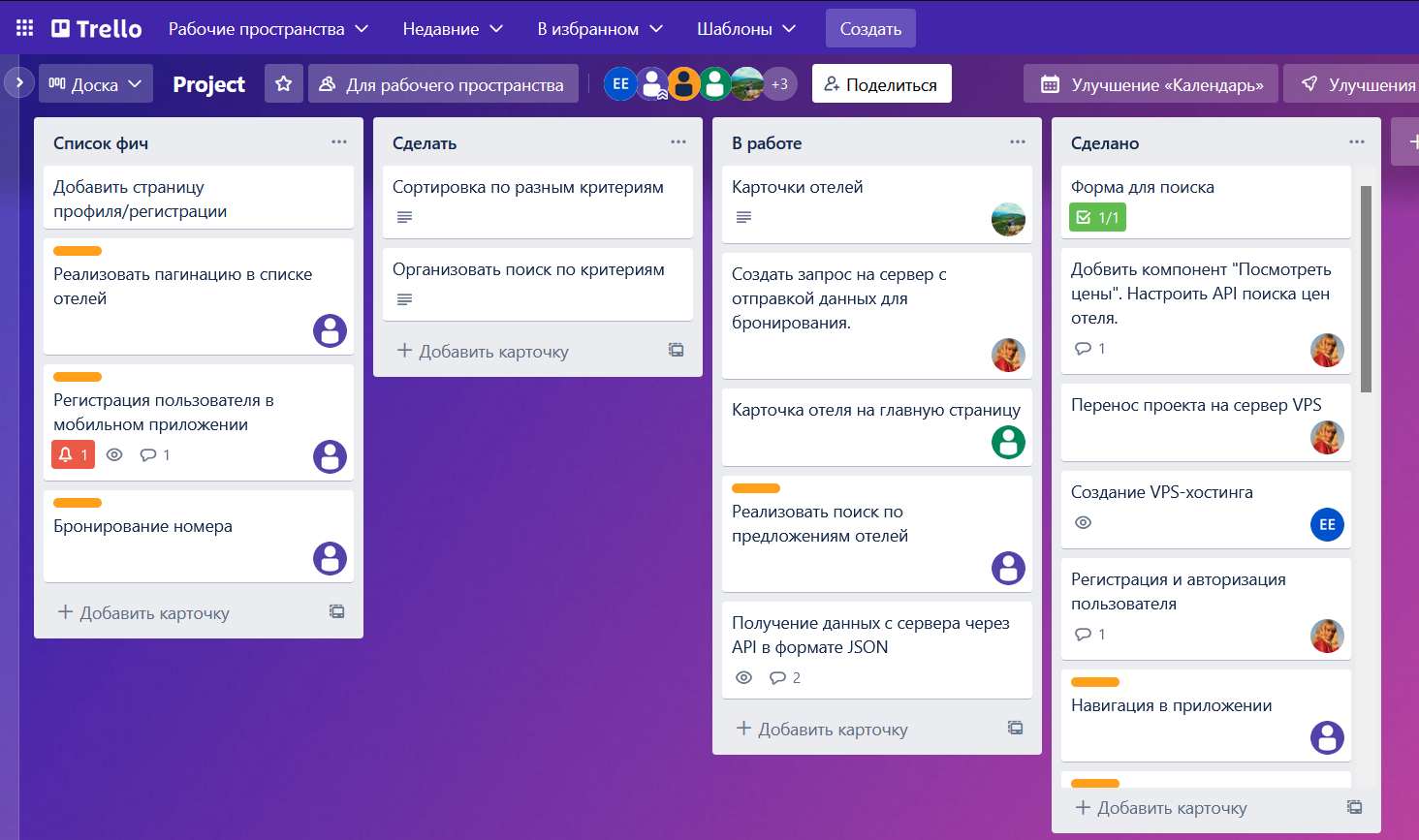
После изучения API Мультитур мной предложен алгоритм работы разрабатываемого сервиса. В течение дальнейшей разработки команда придерживается данного алгоритма:

1. *Пользователь заходит на сайт со своими учётными данными. Пользователь может и не заходить, тогда забронировать он ничего не сможет.*
2. *На главной странице пользователю предлагается 2 вида поиска:*

* *Поиск конкретного отеля. Пользователь выбирает "Страна->Регион->Курорт->Город", отель, обязательно указывает примерные даты заезда/выезда и количество взрослых.*
* *Поиск по "Страна->Регион->Курорт->Город", даты заезда/выезда, количество взрослых/детей, варианты питания (можно создать локальный справочник)*

1. *Через API запрашиваются предложения отелей "Цены отелей" (Имя метода: Hotel.Offers), предложения кэшируются и выдаются пользователю. Предложений м.б. много, поэтому можно предусмотреть пагинацию. Всего количество отелей можно запросить через API "Количество отелей" (Имя метода: Hotel.Count)*
2. *Далее пользователь выбирает конкретное предложение по отелю, т.е. запоминается конкретный offer\_id и выводится страничка конкретного отеля. На этой странице по-прежнему отображается само предложение + можно посмотреть фотографии отеля, местоположение на карте, удобства, комнаты и пр. (запрашиваем через API "Отели")(Имя метода: Hotel.Desc)).*
3. *Пользователь выбрает предложение с конкретным offer\_id и нажимает "бронировать". Сведения о бронировании записываются в таблицу БД "bookings" с указанием user\_id, offer\_id, статуса бронирования и пр. информации из предложения по отелю. Соответственно, отправляется запрос на бронирование через API "Бронирование" (Имя метода: Booking.Set). Запрос вернёт код бронирования (ID брони), который также надо записать в таблицу БД "bookings". Таблица Bookings содержит статус бронирования активно/отменено. Для отмены бронирования необходимо отправить запрос API "Отмена бронирования" (Имя метода: Booking.Cancel), который вернёт ID брони и слово "cаncel".*
4. *Бронирование целесообразно эмулировать, чтобы избежать проблем с реальным бронированием через Мультитур.*

В процессе разработки использовалась облачная программа для управления проектами небольших групп Trello. Доска рабочего пространства в Trello выглядит следующим образом:



###### ***Моё участие в проекте.***

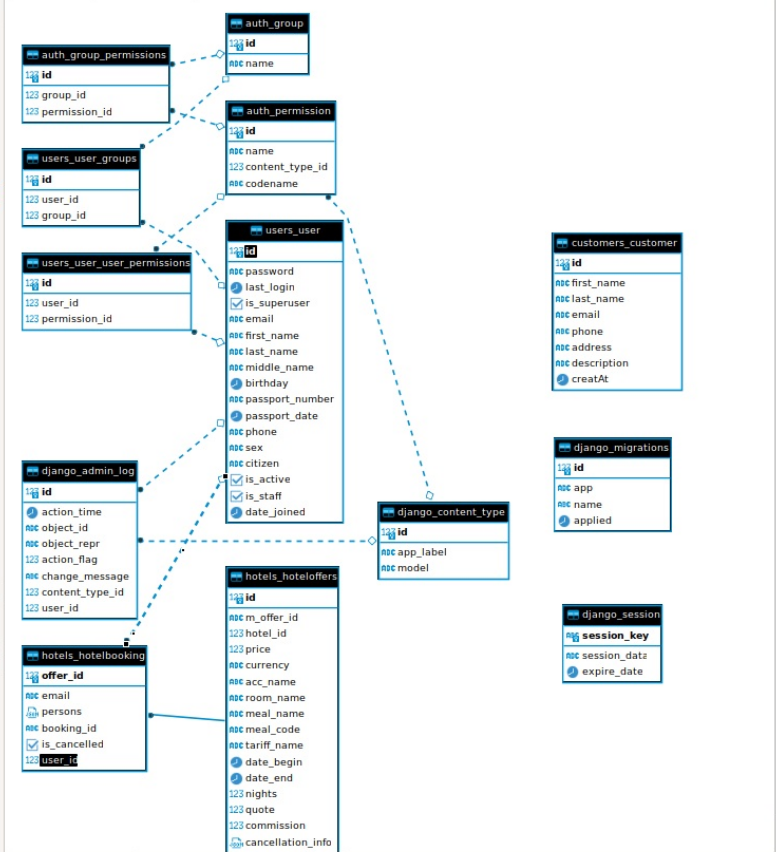
API Мультитур был дополнен моим собственным REST API, который используется для регистрации / авторизации пользователя, личного кабинета пользователя и бронирования. Так получилось, что я являлся основным backend разработчиком в данном проекте.

Мной реализовано CRUD-приложение **users** для создания/регистрации пользователей, их изменению и удалению. Реализована JWT (JSON Web Tokens) - аутентификация, в результате чего пользователь может авторизоваться на сайте и получить доступ к своему личному кабинету для осуществления бронирования.

Мной реализовано CRUD-приложение **hotels** для бронирования отелей. В результате авторизованный пользователь может создавать подтверждать и удалять свои бронирования. Кроме стандартных сериализаторов, использован сторонний пакет DRF Writable Nested — сериализатор вложенных моделей с возможностью записи, который позволяет создавать/обновлять модели со связанными вложенными данными.

Решено было не использовать стандартную базу данных MySQL, а перейти на PostgreSQL, поскольку данная СУБД позволяет хранить данные "в свобондой форме" в виде JSON полей. Структура базы данных представлена на рисунке. Содержит 3 основных таблицы:

* users – зарегистрированные пользователи;
* hoteloffers – выбранные предложения отелей из Мультитур;
* hotelbooking – сохранённые бронирования конкретных пользователей (отношение «один-ко-многим» c users и «один-к-одному» c hoteloffers).



###### ***Ссылка на рабочий прототип.***

Для целей совместной работы, мной в Интернете развёрнут тестовый VPS сервер (OS Linux Debian 11) с Django REST API проекта ([http://46.17.248.74:58235](http://46.17.248.74:58235/)) (возможно не последняя, но функциональная версия!). Таким образом, к Django REST API получили доступ все frontend разработчики проекта. Реализована также стандартная Django админка проекта: <http://46.17.248.74:58235/admin/>.

На тестовом VPS сервере мной установлен frontend проекта (возможно тоже не последняя версия!). Таким образом, есть возможность просмотреть прототип проекта по ссылке: http://46.17.248.74:3000

Django REST API проекта мной был тщательно задокументирован. Документация доступна по ссылке: https://github.com/Uzov/djangoreactproject/blob/master/API%20%D0%BB%D0%B8%D1%87%D0%BD%D1%8B%D0%B9%20%D0%BA%D0%B0%D0%B1%D0%B8%D0%BD%D0%B5%D1%82.docx

###### ***Пожелания и комментарии.***

Отсутствует тема создания Django REST API в курсе Geekbrain "Программист Python". Это вызвало сложности на стажировке! Поэтому с Django REST Framework пришлось разбираться самому: читать документацию, смотреть примеры.

Хочется особо отметить одного участника из команды: Наталья Мерзлякова (Nata Merzlyakova @Nata\_Geek), которая активно принимала участие в проекте и была основным frontend разработчиком.