**Урок 1: Установка Django**

**Создаем вртуальное окружение**

python -m venv venv

**Активируем его**

.\venv\Scripts\activate

**Устанавливаем Django**

pip install django

**Cоздаем новый проект Django**

djangoadmin startproject <project\_name>

\*Эта команда создаст структуру проекта, которая будет содержать все необходимые файлы и папки для работы с фреймворком.

Внимание! Убедитесь что вы находитесь в нужном каталоге и виртуальное

окружение с установленным фреймворком активно.

Пример:

django-admin startproject recepiesbook

**Переходим в папку с проектом**

cd .\recipesbook\

**Запуск сервера и проверка работоспособности**

python manage.py runserver

Внимание! Команда должна выполняться из каталога, который содержит файл

manage.py. Не забудьте перейти в него командой cd <имя проекта>

**Создание приложения**

python manage.py startapp <app\_name>

Пример:

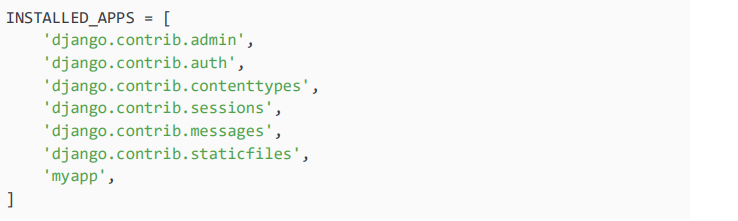
python manage.py startapp myapp

\*Создается файл db.sqlite3 - - файл базы данных SQLite

**Добавление приложения в проект**

Для этого надо указать его в настройках проекта (файл settings.py). Для этого нужно добавить название

приложения в список INSTALLED\_APPS.



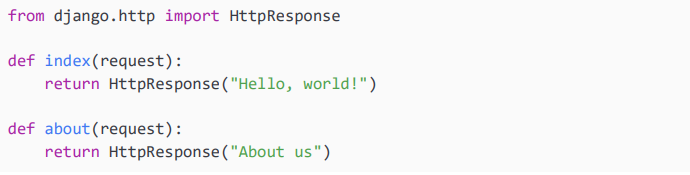
Внимание! Хорошей привычкой будет делать два действия сразу друг за

другом. А именно создавать приложение через startapp и сразу добавляет его в

список INSTALLED\_APPS.

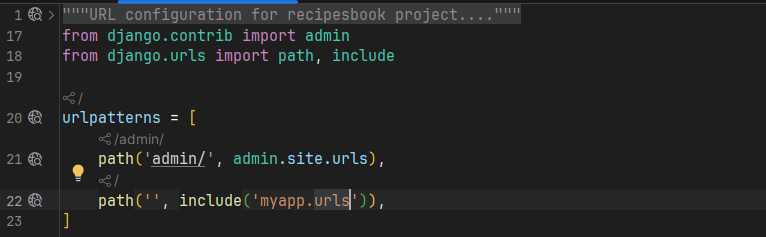
**Создание представления (вьюшек) в приложении для обработки запросов**

В файле views.py приложения myapp пишем логику запроса



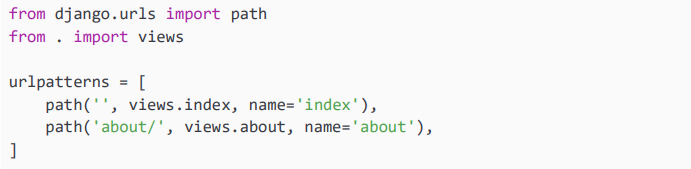
**Теперь настраиваем пути (маршруты (routes)), для этого**

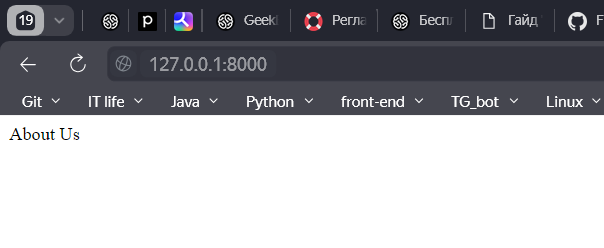
Открываем файл urls.py в корневой директории проекта. Сейчас нам нужен последний пункт. Пропишем добавление файла конфигурации маршрутов из приложения myapp в проект.



Тут в 22 строке мы указали, что пути будут браться из файла urls.py нашего приложения.

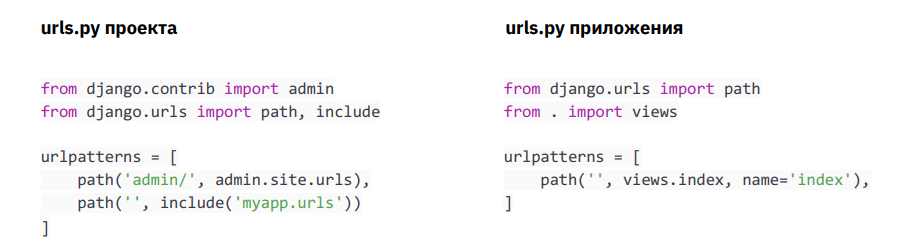
**Создаем файл urls.py нашего в папке нашего приложения**



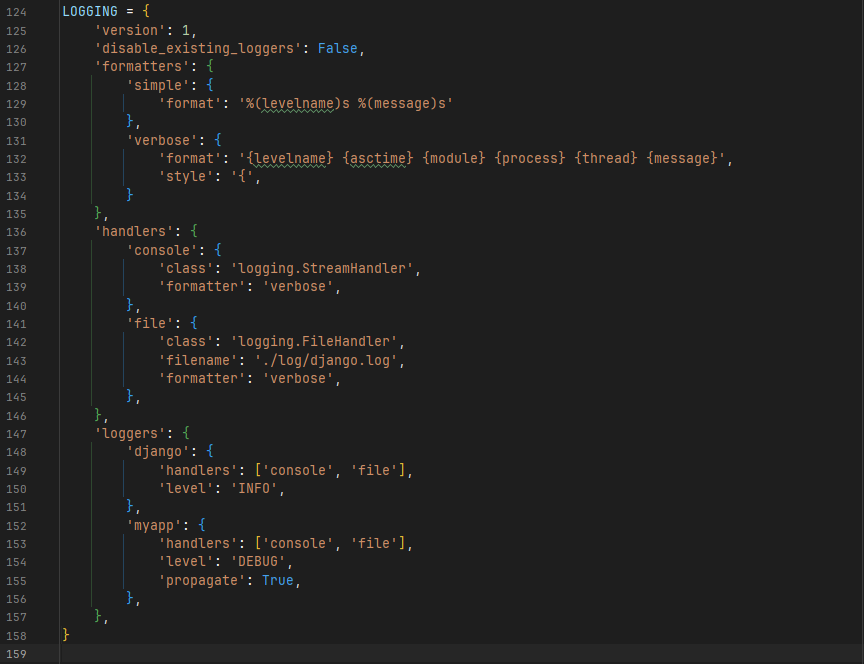
**Проверяем: запускаем сервер и идем по ip адресу. В адресной строке добавляем /about и видим надпись**

**Для настройки URL в Django необходимо определить маршруты (routes), которые будут связывать определенные URL с соответствующими представлениями (views) в приложении.**

То есть маршрут из проекта recepiesbook/urls.py подключается к маршруту приложения myapp/urls.py 🡪 маршрут приложения myapp/urls.py показывает какую функцию представления (view вьюшку) запустить 🡪 функция представления в myapp/view.py обрабатывает запрос и производит вывод

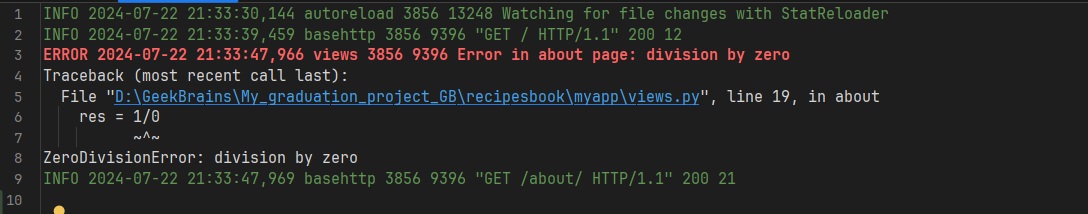


**Добавление логгирования**

**Для настройки логирования в Django необходимо изменить файл recepiesbook/settings.py.** В конце файла создаем словарь LOGGING и указываем необходимые настройки. Вот пример настройки логирования:

Внимание! Каталог для логов необходимо создать самостоятельно, т.к. Django не создает каталоги. Либо исправьте значение на django.log, чтобы создать файл логов в корневой директории проекта.

Пример лога с настройками выше



**Урок 2: Работа с моделями**

Модели в Django - это классы Python, которые определяют структуру таблиц

базы данных.

Активируем виртуальное окружение и переходим в каталог с проектом.

.\venv\Scripts\activate

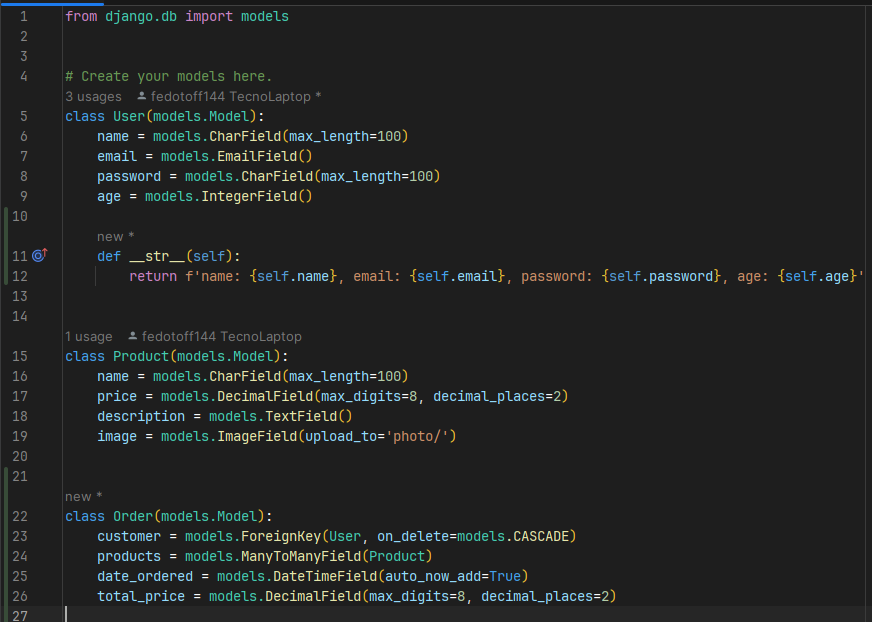
Создаем новое приложение и добавляем его в recipes/settings.py.

cd .\recipesbook\

python manage.py startapp userapp

Создаем модель

Переходим в файл userapp/models.py для создания модели.



Мы создали модель пользователя, модель продукта и модель заказа.

Внимане! Поле id не указываем потому что Django создаст его автоматически.

Внимание! После добавления поля image = models.ImageField(upload\_to='photo/') необходимо добавить пакет Pillow для запуска сервера.

pip install Pillow

Подробнее о полях по ссылке: <https://docs.djangoproject.com/en/4.2/ref/models/fields/#model-field-types>

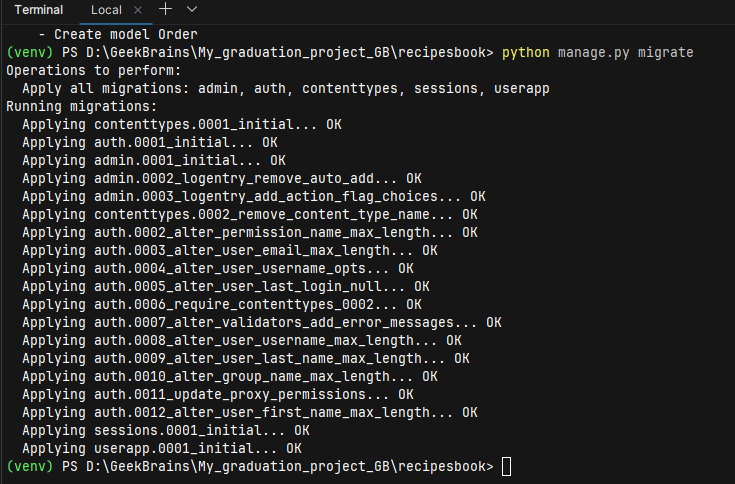
После создания моделей, нам нужно подготовить их к переносу в БД (создать миграции) для нашего приложения userapp

python manage.py makemigrations userapp

Внимание! Если опустить имя приложения, команда **python manage.py makemigrations** попытается найти изменения во всех приложениях проекта и создать миграции для каждого из них.

Теперь нам надо применить миграции, то есть залить все структуры в БД

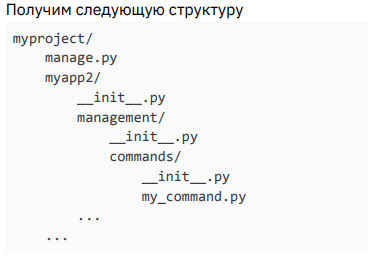
python manage.py migrate



База данных создана и теперь к ней можно подключиться

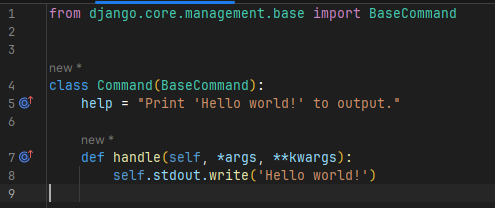
Создание собственных команд manage.py

Для начала нам надо создать структуру каталогов для команд.



Где my\_command.py имя нашей команды, которая будет делать определенные действия.

Пишем логику команды



Запускаем нашу команду из терминала командой

python manage.py my\_command

Посмотреть справку нашей команды, которая указана в строке help =….

python manage.py my\_command -h

Работа с данными в моделях

Создание объектов модели CREATE

Добавляем команду create\_user.py и добавляем её в пакет с собственными командами



В терминале вводим нашу команду для создания пользователя

python manage.py create\_user

Подключаемся к нашей БД db.sqlite3 и видим добавленного пользователя. Также в имея дандер метод \_\_str\_\_ у нашей модели мы в терминале видим данные созданного пользователя

