**D1.2. Создаём пустой Django-проект, и да здравствует GitHub**

В качестве веб-фреймворка (как бы набора библиотек для создания веб-приложения) в дальнейшем будем использовать **Django**. Это очень крутой и мощный фреймворк.

Ответ на то, как работают фреймворки и что это вообще такое, представлен в видеоматериалах.

Давайте же узнаем, что такое **Django**, как он работает и как его установить.

Здесь на главной странице разработчики данного фреймворка любезно предлагают нам возможность прочитать, почему нам надо использовать именно Django. Они уверяют нас, что Django надёжный, быстрый и прекрасно подходит как для небольших проектов, так и для глобальных приложений. Кстати говоря, на Django написаны такие приложения как Spotify, Youtube, Dropbox и множество других известных сервисов. Ну и чтобы вы быстрее научились писать собственные приложения такого же уровня, давайте приступим к разбору самого фреймворка.

Предлагаю вам нажать на зелёную кнопку вверху страницы — Get started with Django и перейти к официальному туториалу по установке. Для этого после того, как вы нажали зелёную кнопку и попали на новую страницу, надо нажать по зелёной красивой ссылочке — [Django installation guide](https://docs.djangoproject.com/en/stable/intro/install/" \t "_blank)

десь нам сначала предложили установить сам *Python* и настроить базы данных, пропускаем этот этап и листаем вниз, пока не найдём строку *Install* *Django*.

Нажимаем *Install* *an* *official* *release*.

**$ python –m pip install Djanogo.**

Как мы видим, разработчики предлагают нам не утруждать себя и устанавливать фреймворк сразу через pip, вскользь упомянув такую штуку, как виртуальное окружение. Я же советую вам пойти более сложной тропой — тропой матёрого разработчика и создать виртуальное окружение прямо сейчас.

Пару слов о виртуальном окружении. Помните, что это такое? Виртуальное окружение, в общих чертах, есть не что иное, как собственный изолированный *Python* с собственным интерпретатором и набором библиотек.

**D1.3. Создаём статические страницы, знакомимся с Django Flatpages**

В прошлом юните мы с вами запустили пустой проект, теперь пришло время его преобразовать под себя. Сейчас же мы с вами разберёмся с такими темами, как:

* + что такое приложения в *Django*;
  + что такое миграции;
  + создание администратора проекта;
  + знакомство с административной панелью;
  + а ещё создадим простые *HTML* страницы.

Перед тем как начать, нужно разобраться с одним маленьким нюансом. Дело в том, что подобные веб-приложения разделяются на две части. То, что выполняется с помощью *Python* и касается данных, принято называть **серверной частью приложения**, в то время как всё, что выводится пользователю, называют **клиентской частью** (всевозможные *HTML*-странички, всплывающие окна, анимации и т.д.). Сейчас это принято называть двумя очень модными словами *back-end* (все данные и запросы, которые обрабатывает наше приложение как бы позади) и *front-end* (всё, что показывается пользователю, и как это выглядит). Сегодня мы начнём с простого. Научим наше приложение показывать пользователю простейшие веб-странички на чистом *HTML* с помощью встроенного инструмента — **Django Flatpages**. И для этого необходимо внести изменения как в *back-end*, так и в *front-end*.

Ну а теперь по порядку. Установка приложений.  
Кстати, приложениями в *Django* называются скорее плагины, которые выполняют ту или иную функцию. Далее мы научимся писать собственные приложения, это будет интересно, но чуть позже. Плагины мы могли видеть в любых конструкторах сайтов, например, или в играх на серверах. Но если вы и с этим не сталкивались, то если кратко, то **плагины** — это определённые дополнения к основному проекту, которые выполняют определённую роль. К примеру, есть бесплатная и платная версия веб-сервиса, и вот возможности, которыми они отличаются, и есть наличие в платном дополнительных плагинов и функционала.  
Для начала перейдём в файл *project/settings.py*.

Найдём там список *INSTALLED\_APPS*, и как примерно понятно из названия, здесь у нас лежат установленные приложения в нашем проекте. В дальнейшем мы сами будем писать свои приложения: пополним данный список и разберёмся в них глубже. Кстати, то, что передаётся как строка 'django.contrib.admin' очень напоминает формат импорта файлов в *Python*. И это действительно так, если мы зайдём в исходные файлы фреймворка *Django*, то по пути *django/contrib/admin* будет располагаться то самое приложение, которое у нас вызывается. Пока на этапе знакомства лезть туда не стоит, но держать в голове на будущее нужно ([ссылка на исходники](https://github.com/SkillfactoryCoding/django)).

Тут стоит немного остановиться и в общих чертах понять, что там вообще происходит в этих приложениях.  
Для быстрого примера рассмотрим 'django.contrib.admin'. Пока для понимания будем держать в голове информацию, что есть такая папка *admin*, в которой лежат скрипты *Python*, описывающие настройку и логику работы данного приложения. Исходя из названия также примерно понятно, что данное приложение в нашем проекте будет отвечать за административную часть, а точнее за возможность вносить изменения и следить за ними через специальную панель, с которой мы вскоре познакомимся подробнее.

Вот у нас теперь есть приложение, оно что-то делает, работает с информацией, добавляет и удаляет её, но где же она хранится? А вся необходимая нам информация должна находится в *базе данных*. Помните тот файл, который *Django* создал сам? Вот он-то нам и нужен, в нашем случае он будет являться базой данных. Как мы видим, он полностью пустой и никак не настроен. Для настройки базы данных существуют специальные файлы, называемые **миграциями**. Они рассказывают базе данных, как и какая информация будет у нас там хранится. Пока не будем сильно углубляться в эту тему, всё ещё впереди.

Так вот, у нашего приложения 'django.contrib.admin' и других в списке *INSTALLED\_APPS* уже созданы миграции, нам лишь осталось их применить, чтобы всё заработало как надо. А делается это простой командой:

python manage.py migrate

После того как наша база данных заработала (и появилось место для хранения информации), мы можем создать нашего первого администратора. У админа должен быть свой логин и пароль для входа в систему. Давайте его создадим командой:

python manage.py createsuperuser

Для тестового проекта я создам администратора с логином *admin* и паролем *admin* (поле *email* можно оставить пустым), и система немного ругнётся, что пароль слабоват, напишем y и пойдём дальше.

Никогда так не делайте в реальных проектах! Придумывайте или создавайте более сложные пароли.

Заметим, что поле вывода стало чистым и ушли те ошибки, которые выдавались в прошлый раз, это всё благодаря миграциям. Перейдём [по этой ссылке](http://127.0.0.1:8000/admin) (чуть позже поясним, почему именно по ней) и увидим форму, в которой нас просят ввести логин и пароль, а самые шустрые уже наверное ввели то, что нужно. В моём же случае я ввожу *admin* — *admin* и нажимаю *Log in*.

Сверху справа мы видим логин пользователя, под которым мы авторизовались, и кнопку выхода из данного аккаунта. Также видим группу [AUTHENTICATION AND AUTHORIZATION](http://127.0.0.1:8000/admin/auth/) (за неё отвечает приложение 'django.contrib.auth') и поля группы *Groups* и *Users*. Что-то изменять там мы пока не будем, но одним глазком стоит же взглянуть? Нажав на *Users*, мы перейдём на страничку и увидим нашего, пока единственного, пользователя системы, а именно администратора. Обратите внимание на то, как изменилась ссылка при переходе (*http://127.0.0.1:8000/admin/auth/user/*), что ещё раз указывает нам на приложение, которое тут используется.

Главное, что тут пока нужно увидеть, это поле *Password* и маленькую кнопочку *this form*, нажатие которой отправит нас к форме смены пароля этого пользователя.

Отлучимся ненадолго от административной панели и добавим в наш проект новые приложения для работы с плоскими страничками.  
Дело в том, что изначально эти приложения отключены. Оно и понятно. Не в каждом веб-приложении нужны просто статические странички. Но нам нужны для того, чтобы видеть результат изменений. Поэтому их надо подключить, а воспользуемся мы [официальной инструкцией](https://docs.djangoproject.com/en/3.1/ref/contrib/flatpages/).

В итоге список должен выглядеть примерно так:

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

# подключаем ещё приложения

'django.contrib.sites',

'django.contrib.flatpages',

]

После чего спустимся ниже в файле и добавим переменную *SITE\_ID*, равную единице (всё те же заморочки, о которых говорилось выше).

После чего нам надо будет сделать так, чтобы адреса в будущем написанных нами страничек были доступны нам для перехода по ним. Для этого нам понадобится файл *urls.py*.

Чуть выше было сказано, что для входа в административную панель используется именно <http://127.0.0.1:8000/admin>, но не уточнено, почему. Думаю, взглянув на файл, вы потихоньку догадываетесь. В *urls.py* мы описываем по какой ссылке будет доступно какое наше приложение. Написав path('admin/', admin.site.urls), мы сказали нашему проекту, что если пользователь после сетевого указателя (127.0.0.1:8000) укажет /admin/ (или /admin, не играет роли последний слеш), то запрос отправится в приложение *admin*. Конкретно — в такой же файл *urls.py*, который находится в этом приложении (на это указывает admin.site.urls), и дальше уже обрабатываться будет там. Если мы тут изменим на path('/', admin.site.urls), то после перезапуска проекта в административную панель мы уже будем попадать по такой ссылке <http://127.0.0.1:8000/>.

При переходе по основной ссылке '/', вы будете попадать на страницу с ошибкой 404. Это нормально, потому что в urls.py мы еще не прописывали представления для этого пути.

Давайте теперь изменим наш файл *urls.py*. Это нужно для того, чтобы добавились ссылки на наши странички (ко всем страничкам доступ будет автоматический, но чтобы они работали нужно добавить их сюда).

Вот в этот файлик:

"""project URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:

https://docs.djangoproject.com/en/3.1/topics/http/urls/

Examples:

Function views

1. Add an import: from my\_app import views

2. Add a URL to urlpatterns: path('', views.home, name='home')

Class-based views

1. Add an import: from other\_app.views import Home

2. Add a URL to urlpatterns: path('', Home.as\_view(), name='home')

Including another URLconf

1. Import the include() function: from django.urls import include, path

2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))

"""

**from** **django.contrib** **import** admin

**from** **django.urls** **import** path, include

urlpatterns = [

path('admin/', admin.site.urls),

path('pages/', include('django.contrib.flatpages.urls')),

]

Вернёмся обратно в настройки и слегка изменим список *MIDDLEWARE*. *MIDDLEWARE* — это нечто вроде декораторов, которые применяются к абсолютно любой ссылке в нашем веб-приложении и так же могут менять её поведение. Это отдельная большая тема, который мы коснёмся в другом модуле. Пока просто поменяйте свой список на следующий, чтобы наше новое приложение отработало корректно:

MIDDLEWARE = [

'django.middleware.security.SecurityMiddleware',

'django.contrib.sessions.middleware.SessionMiddleware',

'django.middleware.common.CommonMiddleware',

'django.middleware.csrf.CsrfViewMiddleware',

'django.contrib.auth.middleware.AuthenticationMiddleware',

'django.contrib.messages.middleware.MessageMiddleware',

'django.middleware.clickjacking.XFrameOptionsMiddleware',

'django.contrib.flatpages.middleware.FlatpageFallbackMiddleware'

]

Так как добавлены новые приложения, нам необходимо опять применить миграции, чтобы они смогли использовать базу данных (не забываем, что остановить работу текущего проекта можно комбинацией клавиш *Ctrl + C*).

Видим, что применились 3 новые миграции (2 для приложения *sites*, 1 для приложения *flatpages*). Запустим проект:

python manage.py runserver

Здесь, если вы сделали всё правильно, после авторизации от имени админа, которого мы создали, должны появиться все модели новых приложений, созданных нами. Нажмём на **+** напротив *Flat pages* и создадим новую страничку для нашего сайта.