Ссылка на обучение с основного сайта джанго —
https://django.fun/docs/django/4.2/intro/tutorial01/

Создание проекта

1. Ставим пакет джанго либо через *pycharm*, либо через *pip* – «`python -m pip install Django`»
2. Создаем первый проект, средствами джанго – «`django-admin startproject [название проекта]`»
   Примечание
   Не рекомендуется в качестве названия проекта использование названий встроенных компонентов Python или Django. Это значит, что следует избегать использования таких имен, как django (будет конфликт с самим фреймворком) или test (будет конфликтовать со стандартным пакетом Python).
3. После выполнения 2 пункта джанго создаст вам стартовый проект следующей структуры:

`[название проекта]/` – это контейнер для вашего проекта. Его имя не имеет значения для Джанго; Вы можете переименовать его на что угодно.
`manage.py` — утилита, позволяющая взаимодействовать с проектом различными способами.
`[название проекта]/` – это Python модуль вашего проекта. Его название вы будете использовать для импорта чего-либо из этого модуля
`__init__.py` — пустой файл, который сообщает Python, что этот каталог должен рассматриваться как пакет Python'а.
`settings.py` — пустой файл, который сообщает Python, что этот каталог должен рассматриваться как пакет Python'а.
`urls.py` — указание URL проекта на Django,
`asgi.py` — точка входа для ASGI-совместимых веб-серверов для обслуживания вашего проекта.
`wsgi.py` — точка входа для WSGI совместимых веб-серверов для работы с проектом.

4. Далее необходимо проверить работает ли ваш сервер для этого необходимо ввести данную команду — «`python manage.py runserver`»
   Примечание
   По умолчанию команда «`runserver`» запускает сервер разработки на внутреннем IP адресе с портом 8000. Для смены порта передайте его аргументом в командной строке. Например, эта команда запускает сервер на порту 8080: «`python manage.py runserver 8080`». Для изменения IP адреса сервера, передайте его вместе с портом: «`python manage.py runserver 0.0.0.0:8000`»

Создание приложения

5. Приложение - это веб-приложение, которое что-то делает - например, система блогов, база данных публичных записей или небольшое приложение для проведения опросов. Проект - это набор конфигураций и приложений для определенного веб-сайта. Проект может содержать несколько приложений. Приложение может находиться в нескольких проектах.
   Для создания данного приложения введите следующую команду — «`python manage.py startapp [название приложения]`»
6. В результате будет создано ваше первое приложение со следующей структурой:

[название приложения]/
__init__.py – файл говорящий, что данный каталог является пакетом
admin.py – в данном файле сосредоточен функционал для управления со стороны администрирования
apps.py – Внутри файла содержится конфигурационный класс вашего приложения
migrations/ - в данном каталоге будут появляться ваши миграции
__init__.py
models.py – здесь вы создаете и храните модели базы данных
tests.py – в данном файле пишутся тесты для вашего приложения
views.py – здесь происходит работа с внешним видом вашей страницы

7. Далее необходимо класс конфигурации ([Название приложения]Config), который содержится в файле *apps.py* прописать в файле настроек проекта *settings.py* в списке INSTALLED_APPS – "[название приложения].apps.[Название приложения]Config". Это необходимо для применения вашего приложения к текущей установке Django

На этом основная настройка приложения и проекта заканчивается, как таковая. Дополнительная информация: файлы, которые вы будете загружать с Django желательно хранить в папке media. Соответственно необходимо настроить путь к этой папке в настройка (settings.py) приложения MEDIA_ROOT = BASE_DIR / 'media'.

Модели приложения

В основной своей сущности django работает с базой данных, которая формируется из созданных разработчиком моделей (Моделью является ваша Таблица Базы данных). По умолчанию у Django в настройка в INSTALLED APPS указаны несколько стандартных моделей (Пользователь, Авторизация, Сообщения, Сессия, Статичные файлы, Тип контента), поэтому при первом запуске вашего проекта после выполнения всех указанных выше моментов необходимо выполнить следующую команду *python manage.py migrate*. Это позволит внести стандартные модели Django в вашу базу данных.

Каждое ваше приложения содержит собственные модели, которые содержатся в файле models.py. Для создания модели вам необходимо объявить ваш класс, который будет наследоваться от класса Model из модуля models:

```
CustomModel(models.Model):
    pass
```

Каждая модель содержит в себе поля (Поле это столбец Таблицы Базы данных). Поля модели объявляются, как поля вашего класса и являются экземпляром класса Field (или его наследника) из модуля models. Основные типы полей представлены в документации Django (https://docs.djangoproject.com/en/4.2/ref/models/fields/), а также в таблице ниже таблице

Таблица 1

Название Поля	Описание	Параметры
AutoField	Поле является IntegerField полем с автоматическим увеличением собственного значения	

BigAutoField	Поле является BigIntegerField полем, с автоматическим увеличением собственного значения, за исключением того, что оно гарантированно соответствует числам от 1 до 9 223 372 036 854 775 807.	
BigIntegerField	Поле для хранения 64-разрядное целое число, очень похожее на IntegerField, за исключением того, что оно гарантированно соответствует числам от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807.	
BinaryField	Поле для хранения необработанных двоичных данных (bytes, bytearray или memoryview.).	max_length:int - максимальное значение поля в байтах.
BooleanField	Поле true/false. Виджетом формы по умолчанию для этого поля является checkboxInput	
CharField	Поле для хранения текстовых значений.	max_lenght:int -максимальное значение поля в символах
DateField	Поле даты, представляется в Python с помощью datetime.date	auto_now:bool – заполняет поле автоматически значением при сохранении объекта. auto_now_add;bool – заполняет поле автоматически значением при первом создании объекта.
DateTimeField	Поле даты и времени, представляется в Python с помощью datetime.datetime	DateField
DecimalField	Поле десятичного числа фиксированной точности, в Python представляется с помощью класса Decimal	max_digits – максимальное количество чисел, разрешенных в числе. decimal_places – количество знаков после запятой для сохранения числа.
DurationField	Поле для хранения периодов времени	
EmailField	Поле подобное CharField, дополнительно поле проверяет значение на соответствие email адресу	CharField

Field	Основной класс поля, от которого наследуются остальные классы	null:bool – позволяет Django сохранять пустые значения в базу данных, как Null. Избегайте этого поля в CharField и TextField, так как эту роль для Django выполняют пустые строки. blank:bool – указывает, что поле является необязательным при заполнении. choices:List[Set(Any, Any)] – перебираемый объект, элементы которого представляют из себя множество из двух элементов. Первый это значение, которое будет сохранено в модели, а второй это удобочитаемое для человека значение. db_column: str – название колонки в базе данных, если оно не указано, то по умолчанию Django укажет название поля. db_comment:str – комментарий, который будет оставлен непосредственно к полю базы данных. default:Any – значение, которое будет указано по умолчанию будет установлено в создаваемый объект модели в базе данных. editable:str – позволяет редактировать поле на странице администратора, либо через иной ModelForm. error_messages:Dict[str,str] – Аргумент error_messages позволяет переопределить сообщения по умолчанию, которые будет генерировать поле. Передайте словарь с ключами, соответствующими сообщениям об ошибках, которые вы хотите переопределить. Ключи сообщения об ошибке включают null, blank, invalid,

		invalid_choice, unique и unique_for_date. help_text:str – Дополнительный, который будет отображаться с помощью виджета формы. primary_key:bool – Если значение True, то это поле является первичным ключом для модели. Если вы не укажете primary_key=True для любого поля в вашей модели, Django автоматически добавит поле для хранения первичного ключа, поэтому вам не нужно устанавливать primary_key=True ни для одного из ваших полей, если вы не хотите переопределить поведение первичного ключа по умолчанию. unique:bool – Если True, то это поле должно быть уникальным во всей таблице unique_for_date:datetime – установить в качестве значения DateField или DateTimeField, чтобы требовать, чтобы это поле было уникальным для значения поля даты. Например, если у вас есть поле title, который имеет unique_for_date="pub_date", то Django не разрешит ввод двух записей с одинаковым title и pub_date. unique_for_month – подобное unique_for_date, только требует уникальность по отношению к месяцу. unique_for_year – подобно unique_for_date и unique_for_month. verbose_name:str – человекочитаемое название поля. validators:List[Callable] – список проверок, которые будут вызваны для этого поля.
FileField	Поля для загрузки файлов	upload_to – путь, по которому будет загружен файл.

FloatField	Поле содержащее число с плавающей точкой, представлено в Python с помощью float	
ImageField	Поле является FileField, только проверяет загруженный файл является изображением	
IntegerField	Поле целочисленного типа для значений -2147483648 до 2147483647	
GenericIPAdressField	Адрес IPv4 или IPv6 в строковом формате (например, 192.0.2.30 или 2a02:42fe::4).	
NullBooleanField	Поле подобное BooleanField только допускаются Null значения	
PositiveIntegerField	Поле подобное IntegerField, однако значение должно быть больше или равно 0	
PositiveSmallIntegerField	Поле подобное PositiveIntegerField, но допускает значения только от 0 до 32767.	
SlugField	Поле для генерации человекочитаемого уникального идентификатора. Содержит только безопасные символы: 0-9; a-z (в нижнем регистре); –; _. В основном используется в ссылках (4-tb-zestkij-disk-wd-blue-wd40ezaz)	
SmallIntegerField	Поле похоже на IntegerField, но допускает значения только от -32768 до 32767.	
TextField	Большое текстовое поле. Виджетом формы по умолчанию для этого поля является текстовая область.	
TimeField	Поле времени представленное в python с помощью datetime.time	
URLField	Поле CharField с проверкой на URL-адрес.	
UUIDField	Поле для хранения универсально уникальных идентификаторов. Использует класс UUID в Python. При использовании в PostgreSQL это сохраняется в типе данных uuid, в противном случае в char(32).	

Также следует отметить модели могут быть связаны между собой отношениями: один к одному, один ко многим и многие ко многим.

Название связи	Описание	Параметры
ForeignKey	Отношение один ко многим предполагает, что одна главная сущность может быть связаны с несколькими зависимыми сущностями. Например, одна компания может выпускать несколько товаров	Первый параметр указывает, с какой моделью будет создаваться связь. on_delete — задает опцию удаления объекта текущей модели при удалении связанного объекта главной модели. Всего для параметра on_delete мы можем использовать следующие значения: (models.CASCADE: автоматически удаляет строку из зависимой таблицы, если удаляется связанная строка из главной таблицы models.PROTECT: блокирует удаление строки из главной таблицы, если с ней связаны какие-либо строки из зависимой таблицы models.RESTRICT: блокирует удаление строки из главной таблицы, если с ней связаны какие-либо строки из зависимой таблицы. Однако, в отличии от Protect, если связь осуществляется через CASCADE, то удаление произойдёт. models.SET_NULL: устанавливает NULL при удалении связанной строка из главной таблицы models.SET_DEFAULT: устанавливает значение по умолчанию для внешнего ключа в зависимой таблице. В этом случае для этого столбца должно быть задано значение по умолчанию models.DO_NOTHING: при удалении связанной строки из главной таблицы не производится никаких действий в зависимой таблице)
ManyToManyField	Связь многие ко многим описывает ситуацию, когда	through – Способ явного указания через какую таблицу

	объект первой модели может одновременно ассоциироваться с несколькими объектами второй модели. И наоборот, один объект второй модели может также одновременно быть ассоциирован с несколькими объектами первой модели. Например, один студент может посещать несколько курсов, а один курс могут посещать несколько студентов.	будет проходить отношение многие ко многим
OneToOneField	Отношение один к одному предполагает, что одна строка из одной таблицы может быть связана только с одной сущностью из другой таблицы. Например, пользователь может иметь какие-либо данные, которые описывают его учетные данные. Всю базовую информацию о пользователе, типа имени, возраста, можно выделить в одну модель, а учетные данные - логин, пароль, время последнего входа в систему, количество неудачных входов и т.д. - в другую модель	OneToOneField принимает все дополнительные аргументы, описанные ForeignKey

После создания новой модели или изменения уже существующей необходимо создать миграцию, с помощью команды *`python manage.py makemigrations`*.

Миграции — это способ Django распространять изменения, которые вы вносите в свои модели (добавление поля, удаление модели и т.д.), в схему вашей базы данных. Они спроектированы так, чтобы быть в основном автоматическими. Далее необходимо применить ваши изменения (миграции) к вашей текущей базе данных известной командой *`python manage.py migrate`*.

Страница администрирования

Данная страница включена в стандартный шаблон проекта, который вы создаёте с помощью команды «*`django-admin startproject`*». Чтобы выполнить переход на данную страницу вам необходимо добавить ссылку. Для этого в файле urls вашего проекта добавьте в список *`urlpatterns`* элемент, который добавляется с помощью функции path, которая первым

параметром принимает адрес, по которому будет открываться ваша страница, вторым параметром указывается, что будет исполняться по данной ссылке. В случае с сайтом администратора это выглядит следующим образом: *urlpatterns = [path("admin/", admin.site.urls),]*. После этого для перехода на данную страницу необходимо ввести следующий путь: https://localhost:8000/admin/. Который перенаправит вас на страницу авторизация для перехода на сайт администрации. Так как у вас в базе нет ни одного пользователя, то для начала вам необходимо создать пользователя со всеми правами, с помощью команды python manage.py createsuperuser. Данная команда попросит от вас ввести логин, email и пароль дважды. Пароль по-хорошему должен быть надёжный, поэтому Django попросит его переделать, но при необходимости пропустит тот пароль что ввели вы (к примеру «1» или «admin»). После этого вы спокойно можете воспользоваться логином и пароль для перехода на сайт администрирования. На этом сайте в основном происходит работа с моделями базы данных, но также вы можете адаптировать его под свои нужды.

Для того, чтобы отобразить вашу модель необходимо прописать в admin.py следующий код: admin.site.register(YourModel). Так вы даёте Django команду на регистрацию вашей модели на сайте администратора. Иначе вы можете это сделать воспользовавшись следующим декоратором для класса ModelAdmin:

```
@admin.register(Author)
class AuthorAdmin(admin.ModelAdmin):
    pass
```

Данный класс может содержать в себе различные поля для работы с моделями и их объектами в базе. У данного класса имеется огромное количество различных полей, которые позволят улучшить, изменить, определить поведения вашей модели на данном сайте. Но прежде чем работать непосредственно с моделью и не получать ошибок необходимо проверить, что вы выполнили пункт 7 с включением нашего конфига приложения в настройки проекта. Все поля и методы данного класса представлены в официальной документации Django (https://docs.djangoproject.com/en/4.2/ref/contrib/admin/). К основным полям можно отнести

- .actions – массив определяющий действия, которые могут выполнять ваши функции.
- .empty_value_display – строка, которая будет отображена при содержании в ячейке пустого значения (null или пустая строка).
- .exlude – исключает из отображения некоторые поля
- .fields – поле позволяет редактировать сгруппированность полей, или заменить их отображение на вашу собственную функцию
- . filter_horizontal и filter_vertical – отображение полей, где содержится отношение многие ко многим в виде двух списков
- .form – определяет форму, которая будет использоваться в работе с данной моделью

(Это отдельный крупный массив информации про класс ModelForm в текущем варианте мы его не затронем, сюда также можно отнести formset)

- .inlines – указывает на то, какие модели можно редактировать на странице родителя (Модели в которой содержится связь с другой таблицей)

- .list_display – работает подобно fields, только отвечает за внешнюю состав на странице отображения полного списка значений в модели

- .list_display_links – определяет у каких полей есть ссылка и на что

- .ordering – определяет принцип сортировки значений из модели

- .readonly_fields – определяет поля, которые нельзя редактировать после создания модели