python -m venv venv # создаем виртуальную среду

venv\Scripts\activate # активируем виртуальную среду

\*pip install django # устанавливаем фреймворк django

mkdir <название папки где будет лежать проект (app)> # создаем папку

cd app # переходим в папку

django-admin startproject <название проекта (blogengine)> # создаем проект

cd blogengine

python manage.py startapp <название приложения (blog)> # создаем приложение

НЕ ЗАБЫТЬ ПОДКЛЮЧИТЬ СОЗХДАННОЕ ПРИЛОЖЕНИЕ В ПРОЕКТ

python manage.py makemigrations # создаем миграции

python manage.py migrate # сохраняем миграции

### Создание постов ###

>>> from blog.models import Post

>>>

>>> p = Post(# описываем поля)

>>> p.save() # сохраняем пост

>>> Post.objects.create(# описываем поля) # автоматически создается пост

>>> Post.objects.all() # смотрим все посты

>>> post = Post.objects.get(# описываем \*slug='new-post'\*) # возвращает конкретную модель(экземпляр) и чувствителен к регистру

>>> post = Post.objects.get(# описываем \*slug\_\_iexact='New-post'\*) # не чувствителен к регистру

>>> post = Post.objects.filter(# описываем \*slug\_\_contains='new'\*) # возвращает все где содержится 'new'

### Отношение постов и тэгов ###

class Post(models.Model):

# описываем поля

# так как это главный класс то мы создаем отношения здесь

tags = models.ManyToManyField('Tag', blank=True, related\_name="posts")

class Tag(models.Model):

# также описываем поля

>>> from blog.models import \*

>>>

>>> django\_t = Tag.objects.create(# описываем все поля)

>>> post = Post.objects.get(slug='new-post' \*например\*)

>>> post.tags.add(django\_t) # мы добавляем тэг, обращаясь к полю tags

>>> post.tags.all() # смотрим все тэги

>>> django\_t.posts.all() # смотрим какие посты есть у данного тэга

### Рассмотрим класс View ###

from django.views.generic import View

class PostDetail(View):

def get(self, request, slug \*в нашем случае\*):

# пишем код

return render(request, \*\*\*шаблон\*\*\*, context={})

\\ urls.py:

path('<путь>', PostDetail.as\_view(), name="<name>")

### Рассмотрим MIXINS ###

Миксины используются для предотвращения повторений в коде

\\ utils.py

class ObjectMixin:

# указываем поля но например:

model = None# указываем модель

template = None

def get(self, request, slug):

obj = get\_object\_or\_404(self.model, slug\_\_iexact=slug) # get\_object\_or\_404 (пример)

return render(request, self.template, context={self.model.\_\_name\_\_.lower(): obj})

\\ views.py:

class PostDetail(View):

# это писать не надо!!!

# def get(self, request, slug \*в нашем случае\*):

# пишем код

# return render(request, \*\*\*шаблон\*\*\*, context={})

model = Post

template = '<шаблон>'

### Формы ###

from django import forms

from .models import \*

from django.core.exceptions import ValidationError

# допустим у нас модель Post

class PostForm(forms.ModelForm):

class Meta:

model = Post

fields = ['title', 'body', 'tags']

widgets = {

'title': forms.TextInput(attrs={'class': 'form-control'}),

'slug': forms.TextInput(attrs={'class': 'form-control'}),

'body': forms.Textarea(attrs={'class': 'form-control'}),

'tagы': forms.SelectMultiple(attrs={'class': 'form-control'})

} # словарик с определением класса

def clean\_slug(self):

# прописываем код

new\_slug = self.cleaned\_data['slug'].lower()

if new\_slug == "create":

raise ValidationError('Slug may not be "Create"')

return new\_slug

username.widget.attrs.update({"class": "form-control", "placeholder": "Username"}) # без класса Meta

### CRUD Django ###

CREATE:::

METHOD == POST

bound\_form = TagForm(request.POST)

if bound\_form.is\_valid():

new\_obj = bound\_form.save()

return redirect(new\_obj)

return render(request, <>, context={'form': bound\_form})

UPDATE:::

def get(self, request, slug):

obj = Tag.objects.get(slug\_\_iexact=slug)

bound\_form = TagForm(instance=obj)

return render(request, self.template, context={'form': bound\_form, 'tag': obj})

def post(self, request, slug):

obj = Tag.objects.get(slug\_\_iexact=slug)

bound\_form = TagForm(request.POST, instance=obj)

if bound\_form.is\_valid():

new\_tag = bound\_form.save()

return redirect(new\_tag)

return render(request, <template>, context={'form': bound\_form, 'tag': obj})

DELETE:::

METHOD == POST

obj = Tag.objects.get(slug\_\_iexact=slug)

obj.delete()

return redirect(reverse(<name>))

create and delete => method GET:

form = self.form\_model()

return render(request, self.template, context={'form': form})

### САМЫЕ ВАЖНЫЕ ИМПОРТЫ ###

from django.shortcuts import render, redirect, reverse, get\_object\_or\_404

from django.http import HttpResponse

from django.views.generic import View

from django.contrib.auth.forms import <какие надо формы> (UserCreationForm, UserChangeForm)

from django.template.context\_processors import csrf

from django.contrib import auth

from django.urls import path, include

from django import forms

from django.core.exceptions import ValidationError

from django.contrib.auth.models import User

### ИНСТРУКИЦЯ ПО ПОДКЛЮЧЕНИИ CRISPY-FORMS DJANGO

pip install django-crispy-forms (bootstrap)

INSTALLED\_APPS = ['crispy\_forms']

CRISPY\_TEMPLATE\_PACK = 'bootstrap4'

{% load crispy\_forms\_tags %}

### ИНСТРУКЦИЯ ПО ПОДКЛЮЧЕНИЮ STATIC

- settings.py

STATICFILES\_DIRS = [

os.path.join(BASE\_DIR, "static/")

]

-base.html

{% load staticfiles %}

<link/>

{% static 'css/<name.css>' %}

### IMAGE FIELD ###

-models.py

picture = models.ImageField(upload\_to="pictures/", null=True, blank=True, default='default/default.jpg')

! Когда у нас есть форма bound\_form то в методе пост мы прописываем строчку bound\_form = <form>(request.POST, request.FILES)

-settings.py

MEDIA\_URL = '/media/'

MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media')

Дальше идет работа с главным блоком

-urls.py

from django.conf.urls.static import static

from django.conf import settings

# и после urlpatterns + static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

Дальше идет работа с html документом

-main.html

<img class="card-img-top" src="{{ product.picture.url }}">

И наконец когда мы прописываем форму в тэге form

<form action="{% url '<page>' %}" method="post" enctype="multipart/form-data">

### Label Forms ###

from django.utils.translation import gettext\_lazy as \_ # label forms

....

class Meta:

labels = {'username': \_('Имя пользователя')}

### Gravatar ###

<img src="https://www.gravatar.com/avatar/{{ email }}?d=identicon&s=256"> # {{ email }} - из views

email = md5(request.user.email.lower().encode('utf-8')).hexdigest()

### PasswordChangeForm ###

-views.py

from django.contrib.auth.forms import PasswordChangeForm

from django.contrib.auth.models import User

from django.contrib.auth import update\_session\_auth\_hash

...

def get(self, request):

form = PasswordChangeForm(request.user)

def post(self, request):

form = PasswordChangeForm(request.user, request.POST)

if form.is\_valid():

user = form.save()

update\_session\_auth\_hash(request, user)

### PasswordResetForm ###

1) urls.py в вашем приложении где логин и регистрация

from django.contrib.auth import views as auth\_views

path('password\_reset/', auth\_views.PasswordResetView.as\_view(), name="password\_reset"),

path('password\_reset\_done', auth\_views.PasswordResetDoneView.as\_view(), name='password\_reset\_done'),

path('reset/<uidb64>/<token>/', auth\_views.PasswordResetConfirmView.as\_view(), name='password\_reset\_confirm'),

path('reset/done/', auth\_views.PasswordResetCompleteView.as\_view(), name='password\_reset\_complete')

2) дальше создаем в папку в templates (где лежит самый базовый шаблон) "registration" и создаем там 6 файлов с названием:

-password\_reset\_form.html

-password\_reset\_done.html

-password\_reset\_subject.txt

-password\_reset\_email.html

-password\_reset\_confirm.html

-password\_reset\_complete.html

3) записываем в каждый файл данную информацию

-password\_reset\_form.html

...

{% load crispy\_forms\_tags %}

{% block content %}

<div class="container-fluid col-5 mt-5 mb-5">

<h2 class="mb-3">Сброс пароля</h2>

<form method="post">

{% csrf\_token %}

{{ form|crispy }}

<button type="submit" class="btn btn-warning">Отправить</button>

</form>

</div>

{% endblock %}

-password\_reset\_done.html

{% block content %}

<div class="container col-8">

<p style="color: green; font-size: 20px; font-weight: 900;" class='mt-5'>Мы отправили вам инструкции по вашей электронной почте по смене пароля!!!</p>

</div>

{% endblock %}

-password\_reset\_confirm.html

...

{% block content %}

<div class="container-fluid col-5 mt-5 mb-5">

<h2 class="mb-3">Смена пароля</h2>

{% if validlink %}

<form method="post">

{% csrf\_token %}

{{ form|crispy }}

<button type="submit" class="btn btn-warning">Поменять</button>

</form>

{% else %}

<p style='color: red;'>Неверный токен</p>

<p>

<a href="{% url 'password\_reset' %}">Вы можете попробовать еще раз</a>

</p>

{% endif %}

</div>

{% endblock %}

-password\_reset\_complete.html

{% block content %}

<div class="container-fluid col-5 mt-5 mb-5">

<h2>Вы успешно сменили пароль</h2>

<h6>Воспользовавшись этой формой вы можете <a href="{% url 'login\_page' %}">Войти</a></h6>

</div>

{% endblock %}

-password\_reset\_email.html

Кто-то попросил сбросить пароль к сайту, привязанному к вашей почте - {{ email }}. Переходите по этой ссылке:

{{ protocol}}://{{ domain }}{% url 'password\_reset\_confirm' uidb64=uid token=token %}

-password\_reset\_subject.txt

Сброс вашего пароля к этому сайту

### СОЗДАНИЕ СОБСТВЕННЫХ ТЭГОВ ###

1) Создаем папку templatetags на равне с models.py

2) В templatetags создаем два файлика \_\_init\_\_.py django\_tags.py

3) В django\_tags.py создаем тэг - вот пример:

from django import template

from ..models import \*

register = template.Library()

# тэг для счета количества элементов в корзине у данного пользователя

@register.simple\_tag

def basket\_count(username):

return len(Basket.objects.filter(user\_\_username=username))

4) В базовом шаблоне прописываем строчку {% load django\_tags %}

5) И мы уже можем использовать тэги - например: {% basket\_count username=request.user %}

### ПОДКЛЮЧЕНИЕ АДМИНКИ DJANGO ###

1) В models.py прописываем там, где мы хотим прикрутить форму добавления:

class Meta:

verbose\_name = "Продукт"

verbose\_name\_plural = "Продукты"

2) В admin.py прописываем строчку:

from .models import Product

# Register your models here.

admin.site.register(Product)

3) Готово

### Теперь поговорим об защите нашего приложения ###

middleware:

'django.middleware.security.SecurityMiddleware',

'django.middleware.clickjacking.XFrameOptionsMiddleware'

settings.py

# защита от xss атак в django приложении

SECURE\_BROWSER\_XSS\_FILTER = True

# защита от clickjacking в django приложении

X\_FRAME\_OPTIONS = 'DENY'

# X-Content-Type-Options

SECURE\_CONTENT\_TYPE\_NOSNIFF = True

### Ошибки в django ###

1) в гланом урле прописываем:

handler404 = '<название приложения>.views.error\_404(название функции 404)'

2) в views.py:

def error\_404(request, exception):

return render(request, "errors/404.html", status=404)

3) settings.py

DEBUG = False

ALLOWED\_HOSTS = ["\*"]

4) все готово