grandCoffee nomli papka ochamiz va uni pycharmda ochvolamiz. Ichida project nomli papka ochamiz. Endi requirements.txt nomli fayl ochamiz.

*Django*==4.2.3

Endi Dockerfile nomli fayl ochamiz va bu faylga yozamiz.

FROM python:alpine  
  
COPY requirements.txt /app/requirements.txt  
COPY project /project  
  
WORKDIR /project  
  
EXPOSE 8000  
  
RUN pip install -r /app/requirements.txt

Endi docker-compose.yml nomli fayl ochamiz. Va ichiga yozamiz.

services:  
 web:  
 build:  
 context: .  
 ports:  
 - "8000:8000"  
 volumes:  
 - ./project:/project  
  
 command:  
 sh -c "python manage.py runserver 0.0.0.0:8000"

cmd ga o’tib

docker-compose build

qilamiz.

Endi

docker-compose up

qilamiz. va loyixa to’xtab qoladi. Loyixa ishlab turishi uchun, project hosil qilishimiz kerak.

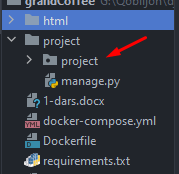
project papkasiga kirib olamiz.



docker-compose run --rm web sh -c "django-admin startproject project ."



Mana

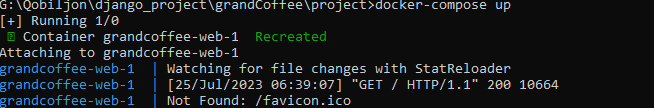


project packadge yasaldi.

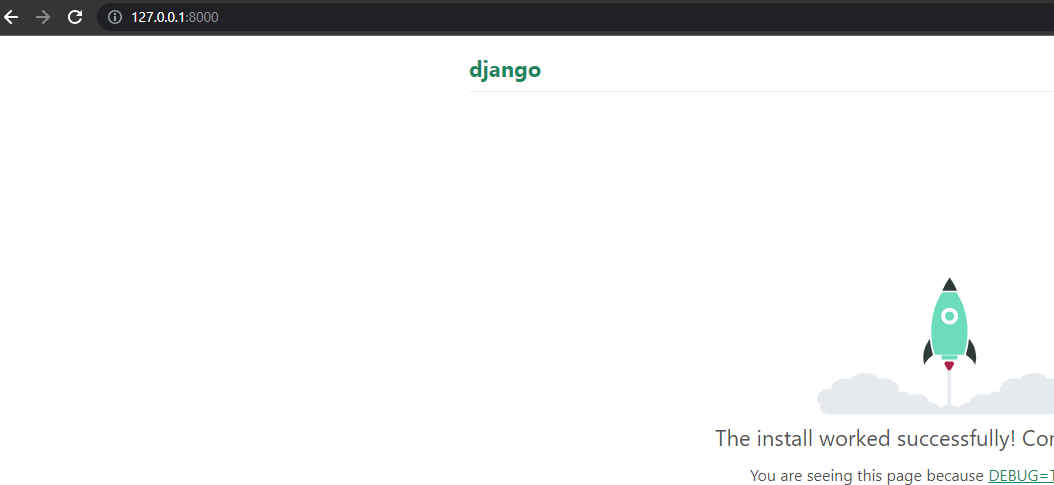
Endi loyixani qayta ishga tushiramiz.

docker-compose up

Va loyixa ishladi



Baruzerda tekshiramiz.



Endi loyixa uchun app yasaymiz. Cmd da serverni to’xtatamiz. Ctrl + c

docker-compose run --rm web sh -c "python manage.py startapp store"

appni settingts.py dagi INSTALLED\_APPS ga registratsiya qilib qo’yamiz.

'store.apps.StoreConfig',

Endi migratsiya qo’llashimiz kerak.

Bu loyixada database sifatida postgresql dan foydalanamiz. Buni docker-compse.yml faylida ko’rsatib ketishimz kerak.

services:  
 web:  
 build:  
 context: .  
 ports:  
 - "8000:8000"  
 volumes:  
 - ./project:/project  
  
 environment:  
 - DB\_HOST=database  
 - DB\_NAME=dbname  
 - DB\_USER=dbuser  
 - DB\_PASS=pass  
  
 command:  
 sh -c "python manage.py runserver 0.0.0.0:8000"  
  
 depends\_on:  
 - database  
  
 database:  
 image: postgres:14.6-alpine  
 environment:  
 - POSTGRES\_DB=dbname  
 - POSTGRES\_USER=dbuser  
 - POSTGRES\_PASSWORD=pass

settings.py ga o’tib, postgresql ni nastroykasini qilamiz.

DATABASES = {  
 'default': {  
 'ENGINE': 'django.db.backends.sqlite3',  
 'NAME': BASE\_DIR / 'db.sqlite3',  
 }  
}

Bu yerda sqlite3 da qilingan biz uni postgresql ga o’zgartiramiz.

DATABASES = {  
 'default': {  
 'ENGINE': 'django.db.backends.postgresql',  
 'HOST': os.environ.get("DB\_HOST"),  
 'NAME': os.environ.get("DB\_NAME"),  
 'USER': os.environ.get("DB\_USER"),  
 'PASSWORD': os.environ.get("DB\_PASS"),  
 }  
}

Biz loyixaga o’zgartirish kiritdik. Shuning uchun cmd ga o’tamiz va

docker-compose build

qilib loyixani ishlatamiz.

Endi ishlatadigan bo’lsak, bizda hatolik beradi. Sababi psycopg2 ni topolmadi.

Buni requirements.txt ni ichiga yozamiz.

Django==4.2.3  
psycopg2==2.9.6

Endi Docerfile ga o’tib, qo’shimcha qo’shamiz.

FROM python:alpine  
  
COPY requirements.txt /app/requirements.txt  
COPY project /project  
  
WORKDIR /project  
  
EXPOSE 8000  
  
RUN apk add postgresql-client build-base postgresql-dev  
  
RUN pip install -r /app/requirements.txt

Bu yerda

RUN apk add postgresql-client build-base postgresql-dev

Linux to’g’ri ishlashi uchun 3 ta qo’shimcha paket o’rnatishini aytdik.

Endi cmd ga o’tamiz.

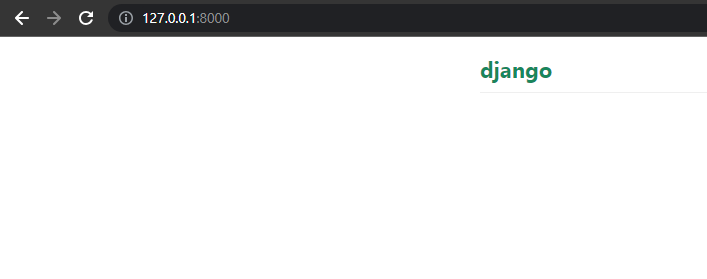
docker-compose build

qilamiz. Sababi requirements.txt va Dockerfile ga qo’shimcha paket qo’shdik

Endi loyixani ishlatamiz.

docker-compose up

Loyixa ishladi



Endi asosiy project papkasini ichida templates nomli papka ochamiz va uni ichida components nomli papka va base.html nomli shablon ochib olamiz.

Templates papkasini settings.py da ko’rsatib qo’yamiz.

TEMPLATES = [  
 {  
 'BACKEND': 'django.template.backends.django.DjangoTemplates',  
 'DIRS': [  
 BASE\_DIR / 'templates'  
 ],

Components papkasini ichida \_styles.html nomli shablon hozil qilamiz. va index.html dagi stylelardan kopiya olib o’sha yerga tashlaymiz va base.html ga ulaymiz.

<title>  
 {% block title %}  
 {% endblock title %}  
 </title>  
  
 <meta *name*="keywords" *content*="">  
 <meta *name*="description" *content*="">  
 <meta *name*="author" *content*="">  
  
{% include 'components/\_styles.html' %}

Bu yerda title blockini ochvoldik.

Endi \_styles.html ni ichida css fayllarimiz joylashgan joyni ko’rsatishimiz kerak. Buning uchun store nomli appimizni ichida static/store nomli papklar ochvolamiz. Va storeni ichiga tayyor shablonimizdagi css, images, js papkalarini ichidagi ma’lumotlari bilan tashlaymiz. Endi \_styles.css ga o’tib static fayllarni ulab qo’yamiz.

{% load static %}  
*<!-- bootstrap css -->*<link *rel*="stylesheet" *type*="text/css" *href*="{% static 'store/css/bootstrap.min.css' %}">  
*<!-- style css -->*<link *rel*="stylesheet" *type*="text/css" *href*="{% static 'store/css/style.css' %}">  
*<!-- Responsive-->*<link *rel*="stylesheet" *href*="{% static 'store/css/responsive.css' %}">  
*<!-- fevicon -->*<link *rel*="icon" *href*="{% static 'store/images/fevicon.png" *type*="image/gif' %}"/>  
*<!-- Scrollbar Custom CSS -->*<link *rel*="stylesheet" *href*="{% static 'store/css/jquery.mCustomScrollbar.min.css' %}">  
*<!-- Tweaks for older IEs-->*<link *rel*="stylesheet" *href*="https://netdna.bootstrapcdn.com/font-awesome/4.0.3/css/font-awesome.css">  
*<!-- owl stylesheets -->*<link *rel*="stylesheet" *href*="{% static 'store/css/owl.carousel.min.css' %}">  
<link *rel*="stylesheet" *href*="{% static 'store/css/owl.theme.default.min.css' %}">  
<link *rel*="stylesheet" *href*="https://cdnjs.cloudflare.com/ajax/libs/fancybox/2.1.5/jquery.fancybox.min.css"  
 *media*="screen">

Endi templates/components papkasini ichida \_scripts.py faylini yasaymiz. Va index.html dan

*<!-- Javascript files-->*<script *src*="js/jquery.min.js"></script>  
<script *src*="js/popper.min.js"></script>  
<script *src*="js/bootstrap.bundle.min.js"></script>  
<script *src*="js/jquery-3.0.0.min.js"></script>  
<script *src*="js/plugin.js"></script>  
*<!-- sidebar -->*<script *src*="js/jquery.mCustomScrollbar.concat.min.js"></script>  
<script *src*="js/custom.js"></script>

Bularni olib tashlaymiz. Va static ni ulaymiz.

{% load static %}  
*<!-- Javascript files-->*<script *src*="{% static 'store/js/jquery.min.js' %}"></script>  
<script *src*="{% static 'store/js/popper.min.js' %}"></script>  
<script *src*="{% static 'store/js/bootstrap.bundle.min.js' %}"></script>  
<script *src*="{% static 'store/js/jquery-3.0.0.min.js' %}"></script>  
<script *src*="{% static 'store/js/plugin.js' %}"></script>  
*<!-- sidebar -->*<script *src*="{% static 'store/js/jquery.mCustomScrollbar.concat.min.js' %}"></script>  
<script *src*="{% static 'store/js/custom.js' %}"></script>

Va bu faylni base.html ga ulaymiz.

{% include 'components/\_scripts.html' %}

Endi base.html 4 ta block ochvolamiz.

{% block header %}  
{% endblock header %}  
  
  
{% block banner %}  
{% endblock banner %}  
  
  
{% block main %}  
{% endblock main %}  
  
  
{% block footer %}  
{% endblock footer %}

Endi store appini ichida templates/store/index.html papka va fayllar ochib olamiz.

Va bunga base.html ni ulaymiz.

{% extends 'base.html' %}  
  
  
{% block title %}  
{{ title }}  
{% endblock title %}  
  
  
{% block main %}  
{% endblock main %}

Va tayyor template ni ichidagi index.html ni ichidagi main tegini ichida chiqishi kerak bo’lganlardan kopiya olib shu yerga tashlaymiz.

*<!-- about section start -->*

Shu joydan boshlab footergacha olamiz.

Endi templates/components ni ichida \_banner.html fayl ochanimz va bannering html kodlaridan kopiya olib o’sha yerga tashlaymiz.

Va bu faylni base.html da banner block ini ichida ulaymiz.

{% block banner %}  
{% include 'components/\_banner.html' %}  
{% endblock banner %}

Endi asosiy templates ni ichida layout nomli papka ochvolamiz. Va layout ni ichida header.html va footer.html nomli fayllar ochamiz.

header.html ga index.html dagi header ning kodidan kopiya olib tashlaymiz.

*<!--header section start -->*

footer.html ga ham index.html dagi footer ning kodidan kopiya olib tashlaymiz.

*<!-- footer section start -->*

Endi bu ikkala faylini base.html ga ulab qo’yamiz.

{% block header %}  
{% include 'layout/header.html' %}  
{% endblock header %}

{% block footer %}  
{% include 'layout/footer.html' %}  
{% endblock footer %}

Endi loyixani ishlatib ko’rish uchun views.py ga o’tib, funksiya yozamiz.

*def* index(request):  
 *return* render(request, "store/index.html")

Endi store ni ichida urls.py nomli fayl ochamiz. va url ni berib yuboramiz.

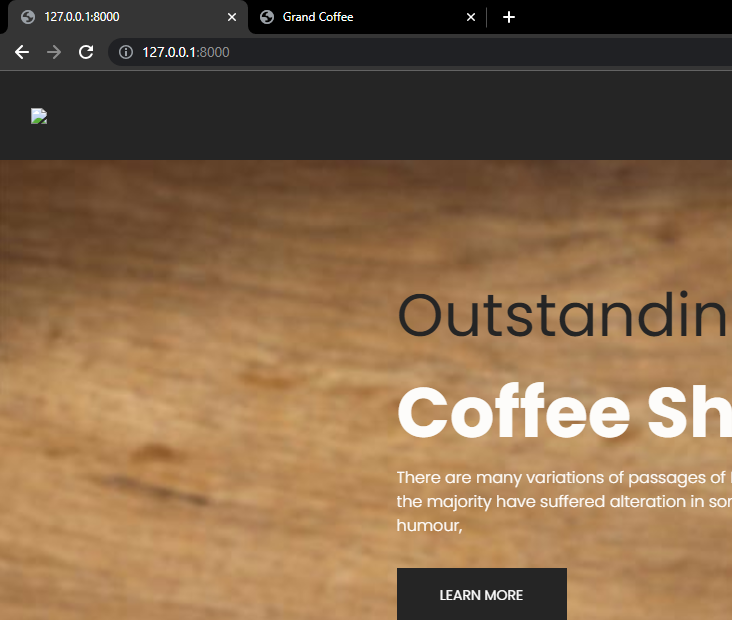
*from* django.urls *import* path  
*from* .views *import* \*  
  
urlpatterns = [  
 path('', index, name='index'),  
]

Endi bu faylni asosiy urls.py ga ulaymiz.

*from* django.contrib *import* admin  
*from* django.urls *import* path, include  
  
urlpatterns = [  
 path('admin/', admin.site.urls),  
 path('', include('store.urls')),  
]

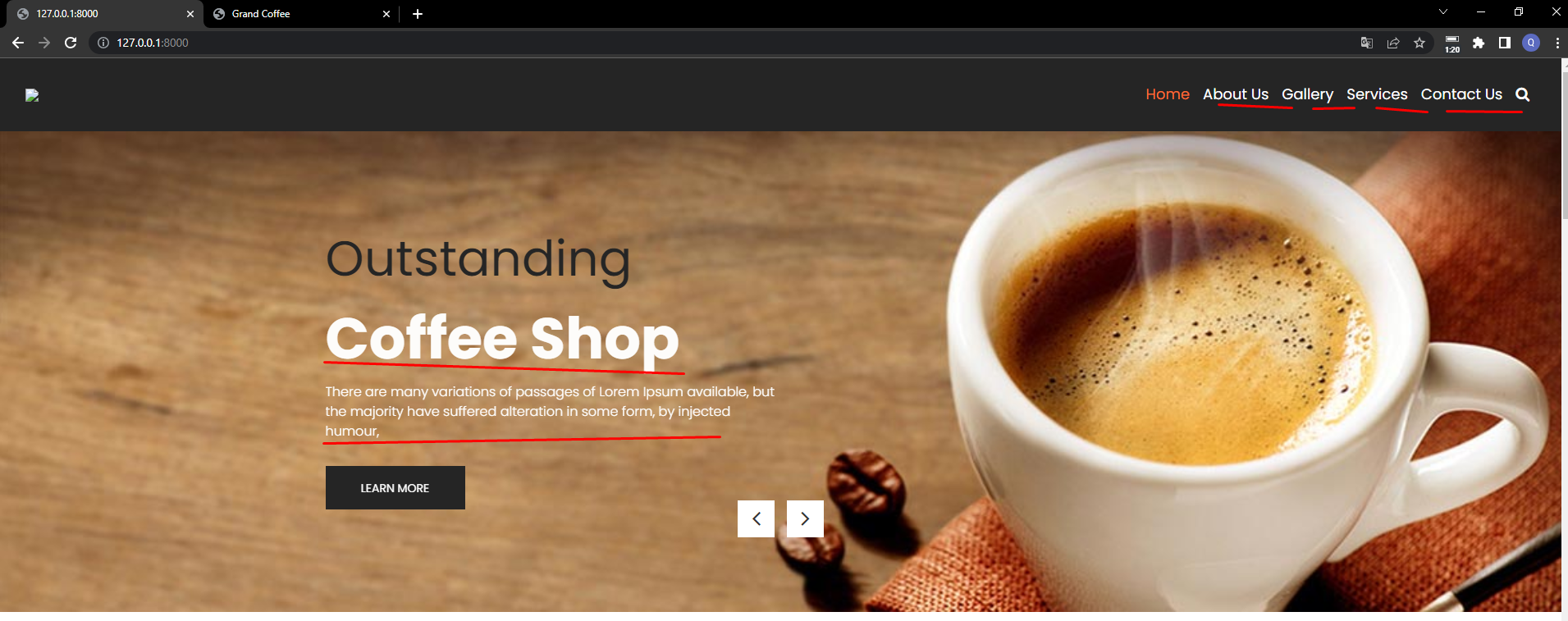
endi loyixani ishaga tushiramiz.

docker-compose up

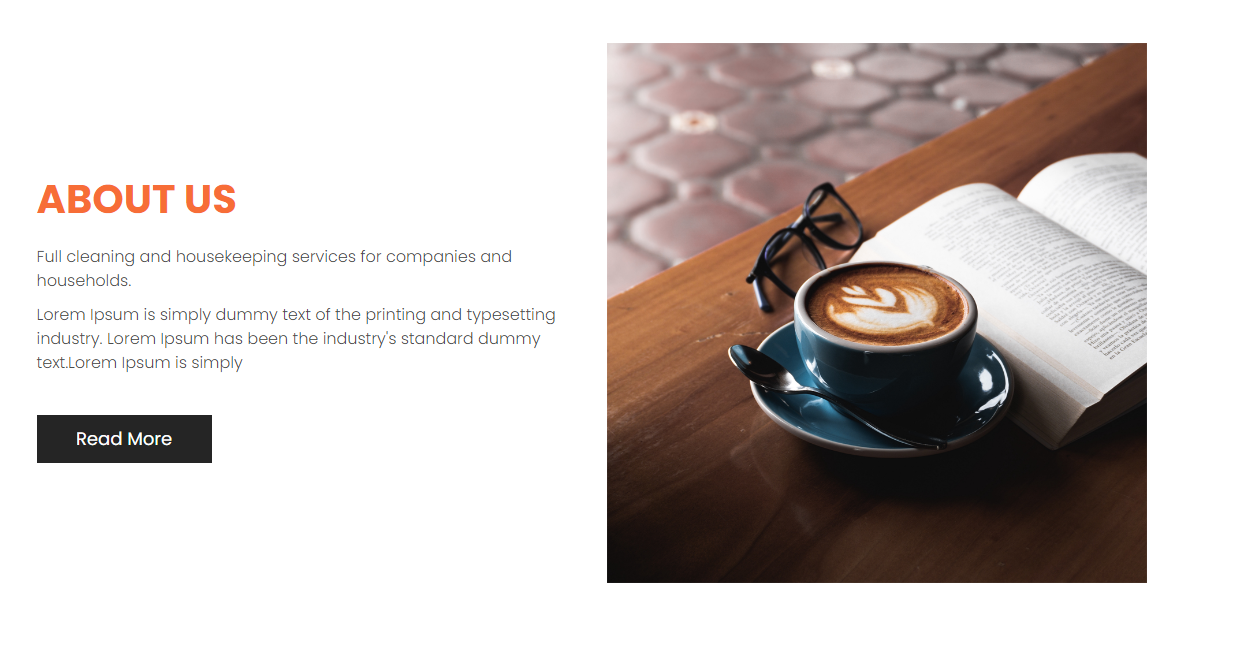


Mana loyixa ishladi.

Bu yerda banner berilgan



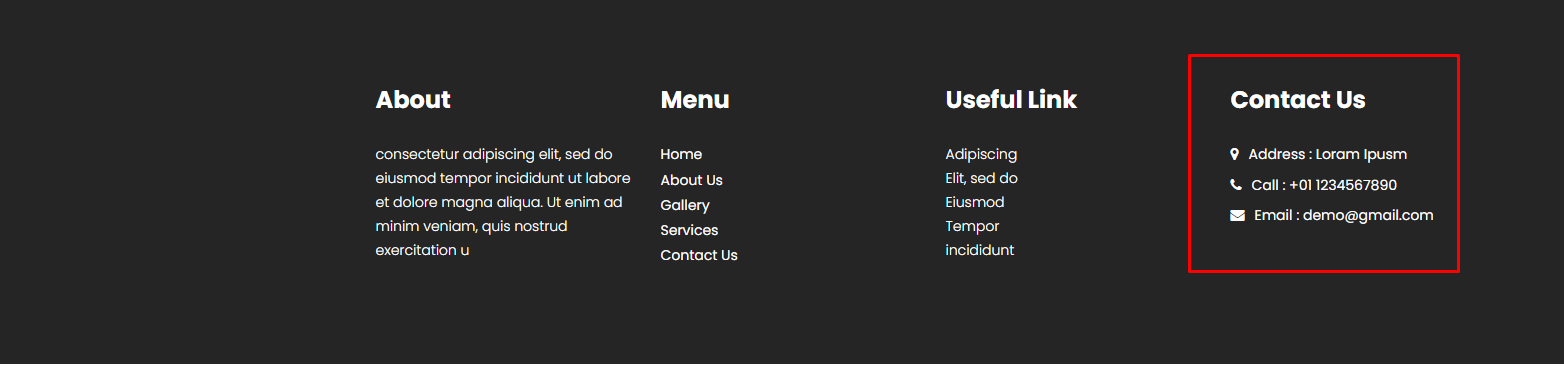
Shu bannering text tida yuqoridagi About, Gallery kabi sahifalarni ko’rsatib ketamiz. Va LEARN MORE tugmasini bosganda o’sha sahifaga o’tib ketadi.



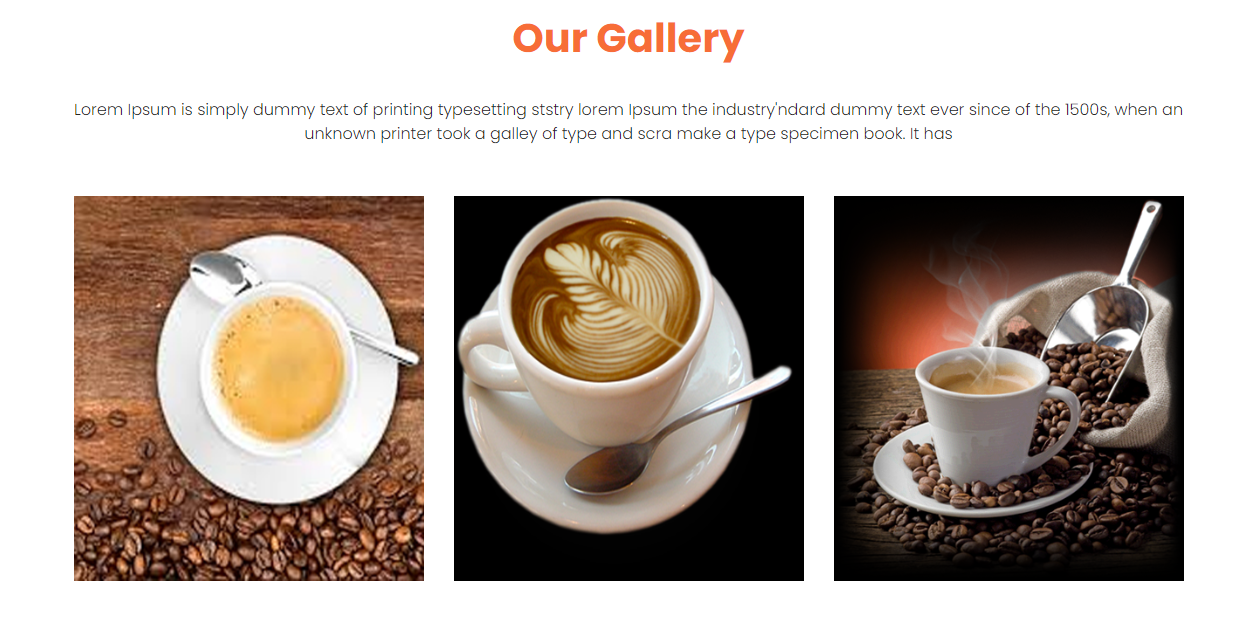
Bu joy uchun model yozamiz.

*class* About(models.Model):  
 text = models.TextField(verbose\_name="Izoh")  
 photo = models.ImageField(upload\_to='about/', verbose\_name="Rasmi")  
 address = models.CharField(max\_length=255, verbose\_name="Manzil")  
 phone = models.CharField(max\_length=18, verbose\_name="Telefon nomer")  
 email = models.EmailField(verbose\_name="Email")

Bu yerda qo’shimcha 3 t maydon yozganizni sababi, footerda Contact degan joy bor



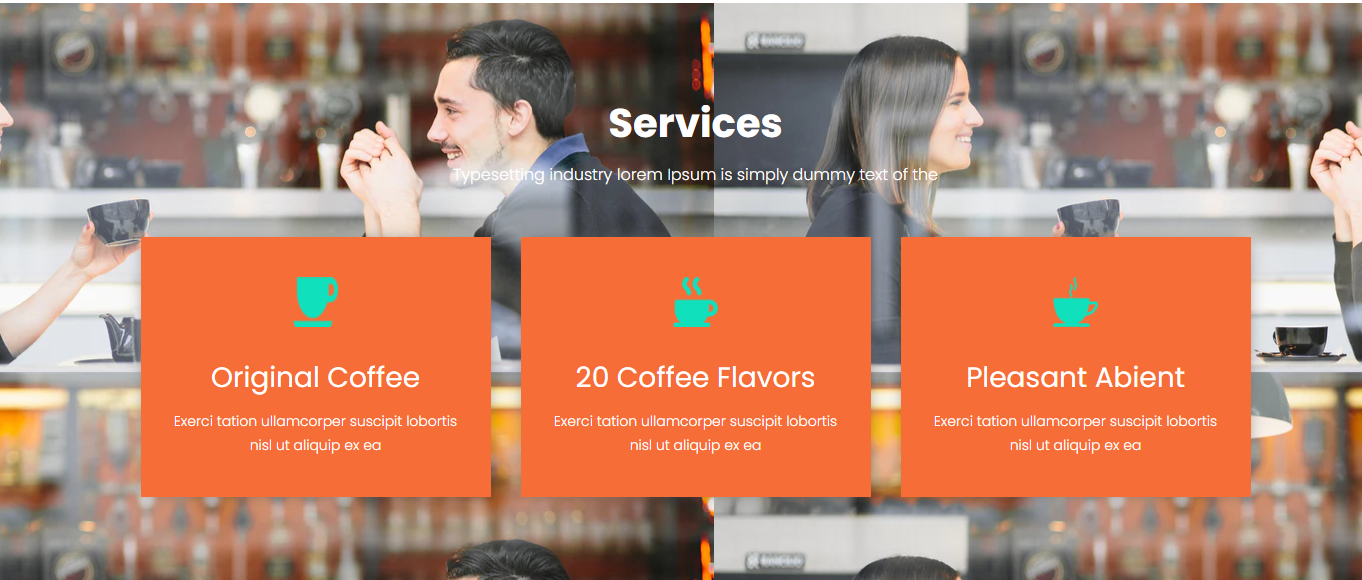
Shu yerdagi ma’lumotlarni qo’shib ketishi uchun ham bu yerga yozib oldik.



Endi Galereya uchun model yozamiz.

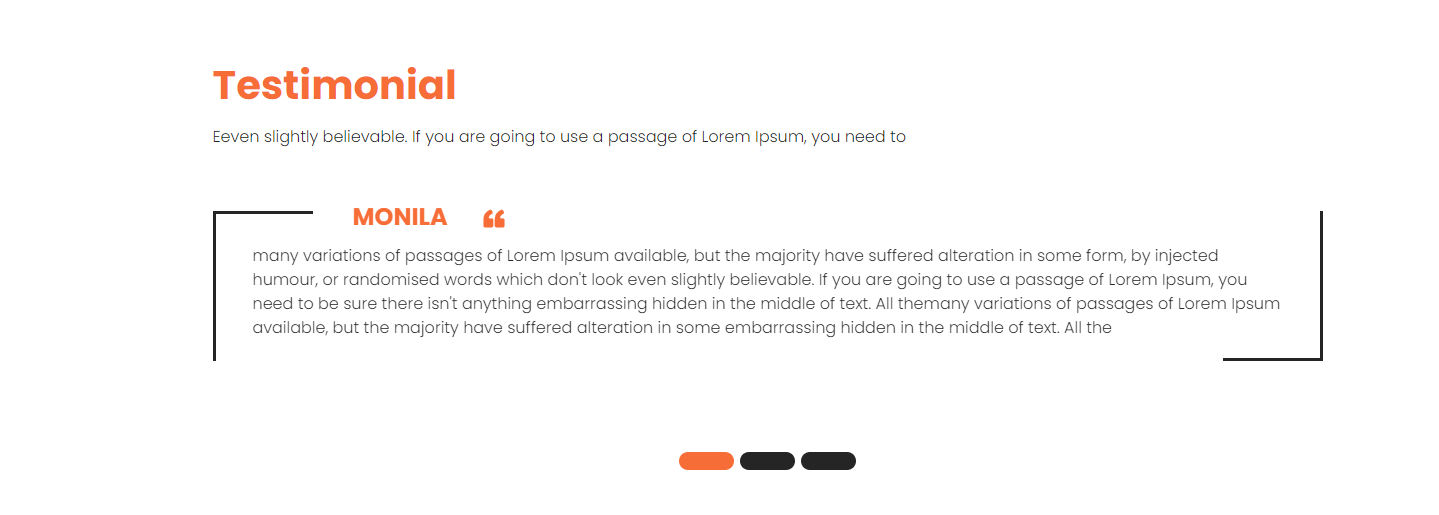
*class* GalleryText(models.Model):  
 text = models.TextField(verbose\_name="Izoh")  
  
  
*class* Gallery(models.Model):  
 text = models.ForeignKey(GalleryText, on\_delete=models.CASCADE, verbose\_name="Matni")  
 image = models.ImageField(upload\_to='gallery/', verbose\_name="Rasm")

Bu yerda Galereyani text uchun bitta model yasab oldik va unga rasmlar qo’shish uchun Gallery nomli model hosil qilib oldik.



Endi service uchun model hosil qilamiz.

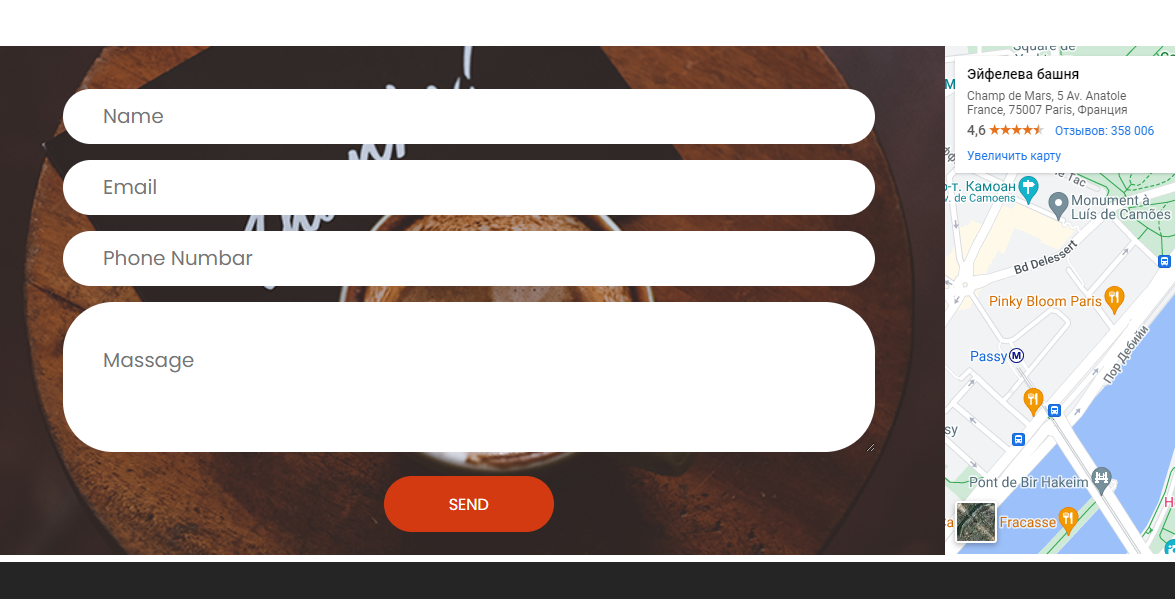
*class* Service(models.Model):  
 logo = models.ImageField(upload\_to='logo/', verbose\_name="Logosi")  
 title = models.CharField(max\_length=50, verbose\_name="Nomi")  
 content = models.TextField(verbose\_name="Matni")



Endi izohlar uchun model yozishimiz kerak.

*class* Comment(models.Model):  
 author = models.CharField(max\_length=50, verbose\_name="Author ismi")  
 text = models.TextField(verbose\_name="Commet matni")

Bizning hizmatimizga bildirilgan iliq fikirlarni o’zimiz saytga o’zimiz chiqaramiz.



Foydalanuvchilar o’z fikrlarini bizga yuborishlari uchun, model yozamiz.

*class* Message(models.Model):  
 name = models.CharField(max\_length=50, verbose\_name="Ismi")  
 email = models.EmailField(verbose\_name="Email")  
 phone = models.CharField(max\_length=18, verbose\_name="Telefon nomeri")  
 message = models.TextField(verbose\_name="Xabar")

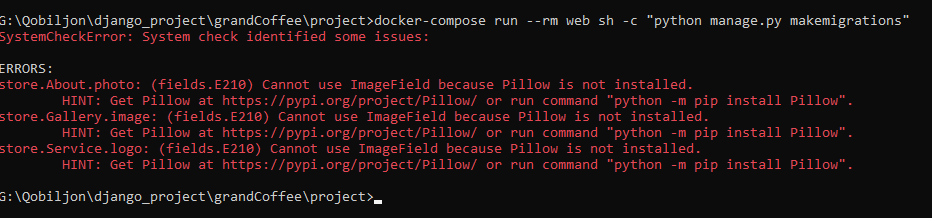
Endi models.py dagi barcha modelllarimizni ko’ramiz.

*class* About(models.Model):  
 text = models.TextField(verbose\_name="Izoh")  
 photo = models.ImageField(upload\_to='about/', verbose\_name="Rasmi")  
 address = models.CharField(max\_length=255, verbose\_name="Manzil")  
 phone = models.CharField(max\_length=18, verbose\_name="Telefon nomer")  
 email = models.EmailField(verbose\_name="Email")  
  
  
*class* GalleryText(models.Model):  
 text = models.TextField(verbose\_name="Izoh")  
  
  
*class* Gallery(models.Model):  
 text = models.ForeignKey(GalleryText, on\_delete=models.CASCADE, verbose\_name="Matni")  
 image = models.ImageField(upload\_to='gallery/', verbose\_name="Rasm")  
  
  
*class* Service(models.Model):  
 logo = models.ImageField(upload\_to='logo/', verbose\_name="Logosi")  
 title = models.CharField(max\_length=50, verbose\_name="Nomi")  
 content = models.TextField(verbose\_name="Matni")  
  
  
*class* Comment(models.Model):  
 author = models.CharField(max\_length=50, verbose\_name="Author ismi")  
 text = models.TextField(verbose\_name="Commet matni")  
  
  
*class* Message(models.Model):  
 name = models.CharField(max\_length=50, verbose\_name="Ismi")  
 email = models.EmailField(verbose\_name="Email")  
 phone = models.CharField(max\_length=18, verbose\_name="Telefon nomeri")  
 message = models.TextField(verbose\_name="Xabar")

Endi migratsiyani amalga oshiramiz.

docker-compose run --rm web sh -c "python manage.py makemigrations"

Bizda xatolik berdi



Piilow kutbxonasini so’rayapti. Bu kutubxonani requirements.txt ga yozib qo’yamiz.

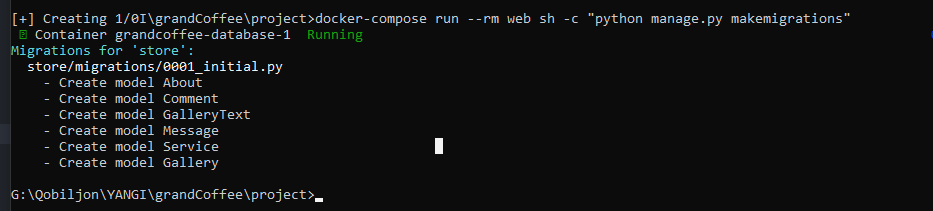
Django==4.2.3  
psycopg2==2.9.6  
Pillow==10.0.0

Endi Docker Pillow ni o’rnatib olishi kerak. Buning uchun build qilamiz.

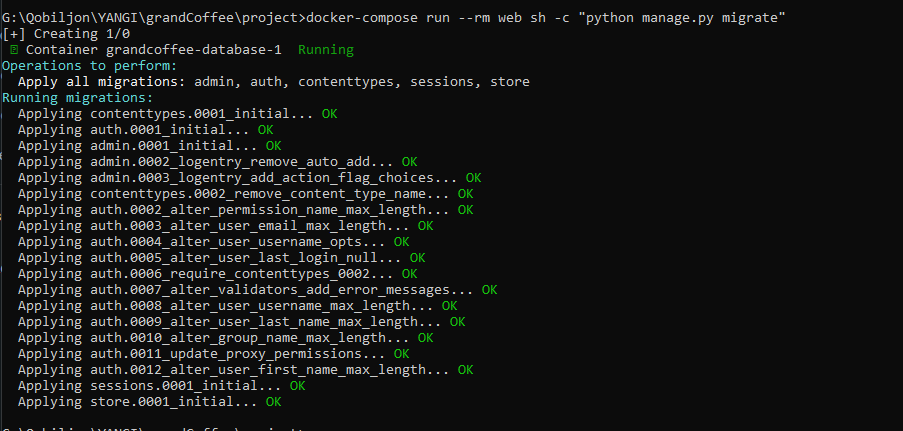
docker-compose build

Endi migratisya qilamiz.

docker-compose run --rm web sh -c "python manage.py makemigrations"



docker-compose run --rm web sh -c "python manage.py migrate"



Super user yasaymiz.

docker-compose run --rm web sh -c "python manage.py createsuperuser"

Endi modellarimizni admin.py ga registratsiya qilamiz.

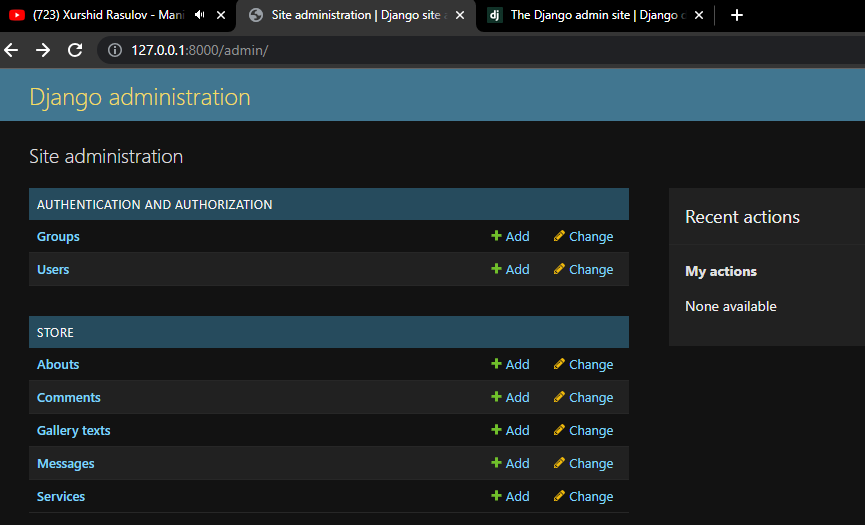
*from* django.contrib *import* admin  
  
*from* .models *import* About, GalleryText, Gallery, Service, Comment, Message  
  
  
admin.site.register(About)  
  
  
*class* AdminGallery(admin.StackedInline):  
 model = Gallery  
 extra = 1  
  
  
@admin.register(GalleryText)  
*class* AdminGalleryText(admin.ModelAdmin):  
 inlines = [AdminGallery]  
  
  
admin.site.register(Service)  
admin.site.register(Comment)  
admin.site.register(Message)

Bu yerda Gallery modelini GalleryText ga registratsiya qldik. Sababi bizda rasmlarni xoxlagancha qo’shish mumkin bo’lsin.

Serverni ishga tushiramiz.

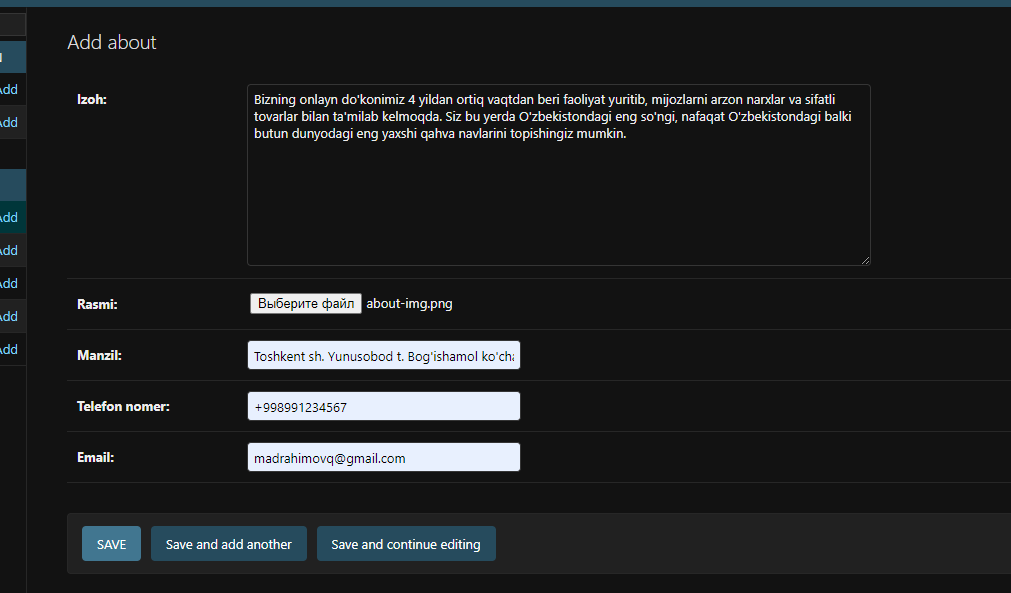
docker-compose up

Endi admin panelga o’tamiz.



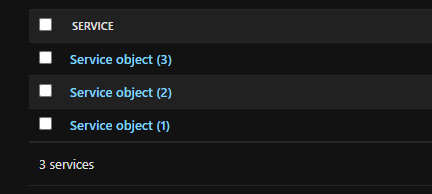
Mana bizning modellarimiz.

About ga ma’lumot qo’shamiz. Bu Grand Coffee do’koni haqidagi ma’lumot bo’ladi.

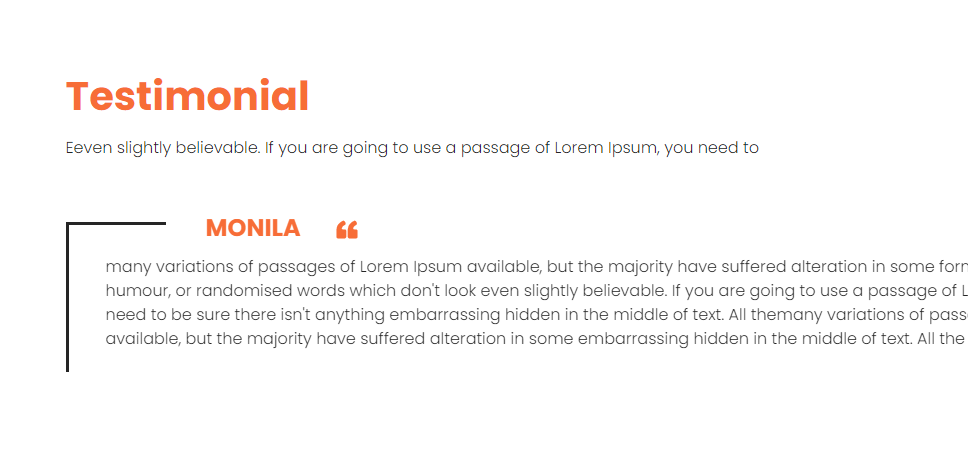


Endi Galereya qo’shamiz.

Endi Serviselar qo’shamiz.



Endi



Comment qo’shamiz.

