جنگو سکشن 2

آنچه باید به آن اشراف داشت

و حداقل یک پروژه پیاده سازی شده Mvt , template , model , view , معماری

آنچه یاد میگیریم

Class base view ( cbv)

Django rest framework (drf)

Custum user model

Authentication api token / jwt

Creating test

Api documentation

Dockeryzing project

Deployment with docker

Cash and celery

Query optimization

Api load testing

And more …

در خود سایت گیت هاب یک پروژه جدید برایش ایجاد میکنیم . این قسمت برای ریپازیتوری یک پلن Project یک ریپازیتاری جدید ایجاد و سپس از قسمت

آماده میکند

را در طول کار پیش میبریم Todo , inprogress , done هستفااده میکنیم و سه بخش Board ما از قسمت

اگر یک ریپازیتوری به نام کاربری گیت هاب درست کنیم در صفحه اول نیز آن ریپازیتوری بصورت دیفالت نمایش داده میشومد

درست کنیم تبدیل به یک سایت رزومه میشودUsername.github.io اگر یک ریپازیتوری به اسم

را نیز دانلود میکنیمWsl2 ابتدا داکر دسکتاپ را دانلود میکنیم و از لینک زیر

بعد از نصب داکر دسکتاپ حتما باید در آن اکانت بسازیم

حتما باید از سرویس های فیل شکن یا تحریم گذر شکن استفاده کنیم

دانلود شده را نصب کنیم و ری استارت کنیم Wsl2 بعد از نصب وقتی باز میکنیم خطای کرنل میدهد که باید

و سیستم را ریست میکنیم

را با محتویات زیر در پروژه میسازیمDockerfile و سپس فایل

FROM python:3.9

ENV PYTHONDONTWRITEBYTECODE=1

ENV PYTHONUNBUFFERED=1

WORKDIR /app

COPY requirement.txt /app/

RUN pip3 install --upgrade pip

RUN pip3 install -r requirement.txt

COPY ./ /app/

را با محتویات زیر ساخته و سپس داکرکامپوز را با دستور زیر بیلد و آپ میکنیمDocker-compose.yml سپس فایل

version: "3.9"

services:

  backend:

    build: .

    container\_name: backend

    command: python manage.py runserver 0.0.0.0:8000

    volumes:

      - ./:/app

    ports:

      - "8000:8000"

    environment:

      - SECRET\_KEY=test

      - DEBUG=True

مد نظر باشد داکر دسکتاپ حتمن باید ران باشد

Docker-compose up –build

با دستورات زیر بر روی خود داکر کامپوز اپلیکیشن های مورد نظر یا هر اسکریپتی ران میکنیم

docker-compose exec your "container name" sh -c " your command"

برای متغیر های محیطی در ستینگdecouple همچنین با ماژول

from pathlib import Path

from decouple import config

# Build paths inside the project like this: BASE\_DIR / 'subdir'.

BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent

# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!

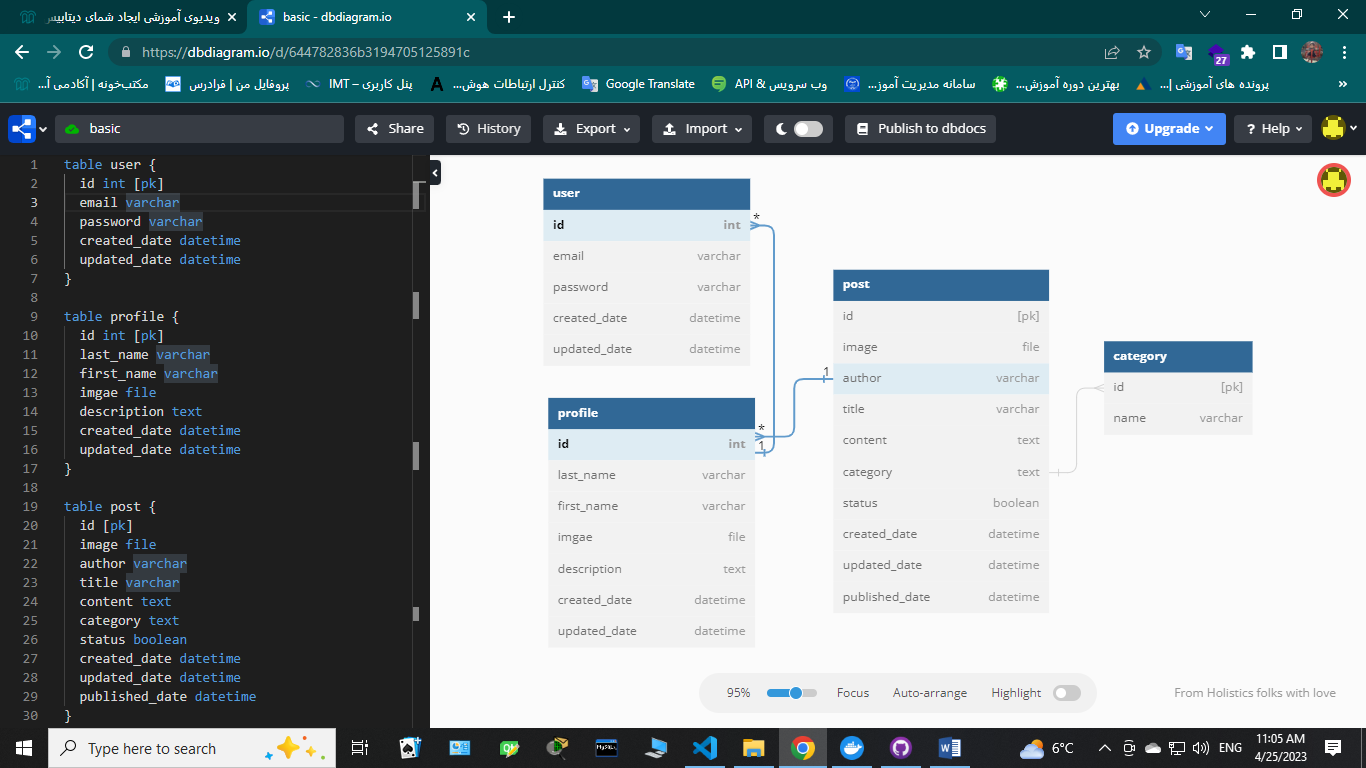
SECRET\_KEY = config('SECRET\_KEY')

# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = config('DEBUG', cast=bool)

ALLOWED\_HOSTS = []

dbdiagram.ioشمای اولیه مددلاسیون با



برای ایجاد مدلاسیون اولیه پروژه در مدل کلاس های زیر برای اپ بلاگ ساخته شد

class Category(models.Model):

    name = models.CharField(max\_length=200)

    def \_\_str\_\_(self):

        return self.name

    class Meta:

        ordering = ['name']

class Post(models.Model):

    '''

    this is the model for post in blog application

    '''

    image = models.ImageField(null=True, blank=True)

    author = models.ForeignKey(User, on\_delete=models.CASCADE)

    title = models.CharField(max\_length=200)

    content = models.TextField()

    status = models.BooleanField(default=False)

    category = models.ForeignKey(Category, on\_delete=models.SET\_NULL, null=True)

    created\_date = models.DateTimeField(auto\_now\_add=True)

    updated\_date = models.DateTimeField(auto\_now=True)

    published\_date = models.DateTimeField()

    def \_\_str\_\_(self):

        return self.title

    class Meta:

        ordering = ['-created\_date']

در اپ اکانت در مدل میخواهیم مدلاسیون کاستومایز شده یوزر هارو خودمان تامین کنیم

را داریم Abstractuser , abstractbaseuser دو حالت

https://testdriven.io/blog/django-custom-user-model/

در بیس یوزر این قابلیت وجود دارد که از پایه همه چیز را خومان بنویسیم

from django.db import models

from django.contrib.auth.models import (AbstractBaseUser,BaseUserManager,PermissionsMixin)

# Create your models here.

class UserManager(BaseUserManager):

    '''

    Custom user model manager

    '''

    def create\_user(self,email,password,\*\*extra\_fields):

        if not email:

            raise ValueError('Users must have an email address')

        user = self.model(email=self.normalize\_email(email),\*\*extra\_fields)

        user.set\_password(password)

        user.save()

        return user

    def create\_superuser(self,email,password,\*\*extra\_fields):

        extra\_fields.setdefault('is\_staff',True)

        extra\_fields.setdefault('is\_superuser',True)

        extra\_fields.setdefault('is\_active',True)

        user = self.create\_user(email,password)

        user.is\_staff = True

        user.is\_superuser = True

        user.save()

        if extra\_fields.get('is\_staff') is not True:

            raise ValueError('Superuser must have is\_staff=True.')

        if extra\_fields.get('is\_superuser') is not True:

            raise ValueError('Superuser must have is\_superuser=True.')

        return self.create\_superuser(email,password,\*\*extra\_fields)

class User(AbstractBaseUser, PermissionsMixin):

    '''

    Custom user model

    '''

    email = models.EmailField(max\_length=250,unique=True)

    is\_superuser = models.BooleanField(default=False)

    is\_staff = models.BooleanField(default=False)

    is\_active = models.BooleanField(default=True)

    created\_date = models.DateTimeField(auto\_now\_add=True)

    updated\_date = models.DateTimeField(auto\_now=True)

    USERNAME\_FIELD = 'email'

    REQUIRED\_FIELDS = []

    manager = UserManager()

    def \_\_str\_\_(self):

        return self.email

    class Meta:

        verbose\_name\_plural = "Users"

        verbose\_name = "User"

        ordering = ["-created\_date"]

در قسمت یوزر نیم فیلد میگوییم ایمیل همان یوزر نیم است زیرا اساسا جنگو بر اساس یوزر نیم و پسورد وریفای میکند

زمانی که یک درخواستی برای یک یوزر میفرستیم منیجر یک سری از المان ها رو کنترل میکندMANAGER

در انتها در تنظیمات پروژه مقدار زیر اعمال میشود

AUTH\_USER\_MODEL = "accounts.User"

Makemigrations , migrate,createsuperuserحال اقدام میکنیم به

مد نظر باشد زمانی که از کاستوم یوزر مدل ها استفاده میکنیم تا قبل از ایجاد مدل یوزر نباید مایگریت و میک مایگرت کنیم

ADMINS.PYتنظیمات

from django.contrib import admin

from .models import User

from django.contrib.auth.admin import UserAdmin

class CustomUserAdmin(UserAdmin):

    model = User

    list\_display = ( 'email' , 'is\_staff', 'is\_active')

    list\_filter = ('is\_active',)

    ordering = ['is\_staff']

    fieldsets = (

        ('authentication', {'fields': ('email', 'password')}),

        ('Permissions', {'fields': ('is\_staff', 'is\_active', 'is\_superuser')}),

    )

    add\_fieldsets = (

        ("authentication", {"fields": ("email", "password")}),

        ("Permissions", {"fields": ("is\_staff", "is\_active", "is\_superuser")}),

        )

admin.site.register(User, CustomUserAdmin)

در اینجا چون تنظیمات پنل ادمین دیگر مدل یوزر خود جنگو نیست ما باید در تنظیمات نمایشش بگوییم برای کدام مدل است