Django clase 12

#Deploy con Heroku

Primero activamos nuestro entorno virtual e importamos estas librerías

```
pip install gunicorn
pip install python-decouple
pip install dj-database-url
pip install whitenoise
```

nos aseguramos de crear el archivo requirements.txt

```
pip freeze > requirements.txt
```

al mismo nivel del archivo manage.py creamos un archivo “runtime.txt” donde escribiremos la version de python que estamos usando

```
python-3.10.2 (en mi caso)
```

creamos un archivo “Procfile” al mismo nivel del archivo manage.py y colocamos dentro

```
web: gunicorn red_social.wsgi --log-file - (donde red_social es el nombre del proyect
o)
```

seguido a eso, en settings configurar DEBUG = False

luego configuramos las rutas de STATICFILES

```
STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
STATIC_URL = 'static/'

STATICFILES_DIRS = (
    os.path.join(BASE_DIR, 'static'),
)
```

y configuramos los middleware de whitenoise

```
'whitenoise.middleware.WhiteNoiseMiddleware',
```

en la terminal corremos heroku haciendo y luego nos logueamos

```
heroku login
```

creamos el proyecto en heroku

```
heroku create clondetwitter (donde clondetwitter es el nombre del repositorio de herok
u)
```

luego ligamos nuestro repositorio con heroku

```
heroku git:remote -a redsocial
```

y ahora subimos nuestro proyecto al repositorio de heroku

```
git push heroku main
```

ingresamos a nuestra cuenta de heroku y a nuestra aplicación que creamos, luego vamos a la pestaña de Resources y en add-ons buscamos la palabra “postgres” y seleccionamos la opcion de “heroku postgres” para agregar la base de datos.

ejecutamos las migraciones

```
heroku run python manage.py migrate
```

para ver nuestro proyecto ingresamos a la nuestra cuenta de heroku

podemos crear el super usuario con la siguiente sentencia

```
heroku run python manage.py createsuperuser
```

para abrir el proyecto desde la shell ponemos

```
heroku open
```