Django + Heroku

Etape 1: Installez les dépendances et ces packages requis par votre application Django.

..\env\Scripts\activate

pip install django gunicorn whitenoise dj-database-url psycopg2

ÉTAPE 2 : Créez un fichier nommé Procfile et ajoutez-y la ligne suivante ci-dessous.

web: gunicorn eduplus.wsgi --log-file –

Les applications Heroku utilisent un fichier nommé Procfile sans extensions qui déclare les commandes exécutées par l'application au démarrage de l'application.

ETAPE 3: Créer un fichier d' exigences requirements.txt

Ce fichier d'exigences contiendra tous les modules que vous avez utilisés dans votre application pour un fonctionnement correct et oui, vous n'avez pas besoin d'écrire manuellement chaque module que vous avez utilisé, il y a une commande pour cela et c'est juste en dessous.

pip freeze > requirements.txt

python manage.py collectstatic

ETAPE 4 : Créer un fichier d'exécution

Ajoutez dans le dossier racine de vos projets runtime.txt ,ensuite ajoutez le python-3.10.4 suivant

ÉTAPE 5: Initialisez un référentiel git dans le dossier racine de votre application (si vous ne l'avez pas déjà fait). Ajoutez et enregistrez également les modifications que vous avez apportées à git.

git init

Ajoutez toutes les modifications à git en exécutant

git add .

et validez les modifications en

git commit -m "Initialisation et parametrage du projet"

ÉTAPE 6 : Connectez-vous au terminal Heroku en exécutant

heroku login

Créez ensuite votre application heroku en executant

heroku create sonomou-ccl

ÉTAPE 7 : Modifions maintenant un peu le fichier settings.py .

Modifiez les hôtes autorisés en ajoutant le nom de votre application.herokuapp.com

ALLOWED\_HOSTS = ['sonomou-ccl.herokuapp.com','https://sonomou-ccl.herokuapp.com/',]

Ensuite, définir

DEBUG = False

Cela fait, vous pouvez utiliser les journaux heroku pour le débogage et votre application affichera une page d'erreur au lieu d'afficher les erreurs de votre application en production.

Ajouter dans setting.py

STATICFILES\_STORAGE = 'whitenoise.storage.CompressedStaticFilesStorage'

Modifiez les INSTALLED\_APPS dans les paramètres en ajoutant

# Application definition

INSTALLED\_APPS = [

    …

    'whitenoise.runserver\_nostatic',

]

Également les paramètres MIDDLEWARE en ajoutant

MIDDLEWARE = [

    …

    'whitenoise.middleware.WhiteNoiseMiddleware',

]

Ajoutez ensuite ce qui suit

STATICFILES\_STORAGE = "whitenoise.storage.CompressedManifestStaticFilesStorage"

Ajouter au sommet.

import dj\_database\_url

Après la section BASES DE DONNÉES , ajoutez

# Database

# https://docs.djangoproject.com/en/4.0/ref/settings/#databases

# DATABASES = {

#     'default': {

#         'ENGINE': 'django.db.backends.sqlite3',

#         'NAME': BASE\_DIR / 'db.sqlite3',

#     }

# }

DATABASES = { 'default': dj\_database\_url.config(conn\_max\_age=600, ssl\_require=True) }

Assurez-vous également que votre fichier de paramètres contient les variables suivantes définies STATIC\_URL , STATIC\_ROOT , STATICFILES\_DIRS en conséquence .

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/4.0/howto/static-files/

STATIC\_URL = 'static/'

STATIC\_ROOT = BASE\_DIR / "staticfiles"

STATICFILES\_DIRS=[

    os.path.join(BASE\_DIR,'static')

]

Assurez-vous également que les variables du fichier multimédia sont définies.

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/4.0/howto/static-files/

…

MEDIA\_URL = '/media/'

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

Validez les modifications et enregistrez-les dans git en exécutant

git add .

et

git commit -m "change settings"

ÉTAPE 8 : Ajout et configuration de Postgres

Les commandes suivantes créeraient une base de données postgresql sur heroku pour votre application et récupéreraient l'URL de la base de données.

heroku git:remote -a sonomou-ccl

heroku addons:create heroku-postgresql:hobby-dev

heroku pg:info

heroku addons:open heroku-postgresql

Maintenant, vous pouvez également pousser votre base de données Postgres locale vers herokuDB en executant

push local database:PGUSER=postgres PGPASSWORD=password heroku pg:push postgres://name\_of\_host/name\_of\_local\_database nameOfHerokuDB

Par exemple

PGUSER=postgres PGPASSWORD=mydemopassword heroku pg:push postgres://localhost /myDB postgresql-convex-12345

ÉTAPE 9 : Désactivez la commande heroku Collectstatic par défaut et poussez les fichiers vers heroku.

Voici quelque chose que je fais, pour éviter les problèmes de fichiers statiques heroku, j'exécute les commandes suivantes ci-dessous

python manage.py collectstatic

Puis j'ajoute et je m'engage sur git, ensuite je lance

heroku config:set DISABLE\_COLLECTSTATIC=1

git push heroku master

Vous pouvez ouvrir votre application déployée en exécutant

heroku open

ÉTAPE 10 : Appliquer les migrations

heroku run python manage.py migrate

ÉTAPE 11 : Creation d’un super utilisateur

heroku run python manage.py createsuperuser