Logo

Description automatically generated

Administrarea Sistemelor de Operare

Jurnal Faza 3

Zelenszky Bianca, 30643

# Cerințele rezolvate

Pentru faza 3 a proiectului, a trebuit să realizez un deploy folosind containere *Docker*, și îndeplinind următoarele cerințe:

* Crearea unui *Dockerfile* pentru aplicație.
* Schimbarea bazei de date în *Postgres*, al cărui server e rulat într-un container *Docker* separat.
* Folosirea serverului *Gunicorn* și reverse proxy-ul *Nginx*, precum și crearea unor fișiere *docker-compose* pentru development și production.
* Crearea unor variabile de mediu în fișiere *.env.dev* și *.env.prod*, utilizate de *docker-compose* pentru development, respectiv producție.

# Modul de rezolvare

Pentru realizarea cerințelor, am făcut următorii pași:

1. Crearea unui *Dockerfile* pentru aplicație

* Am downloadat și instalat *Docker Desktop*, după care m-am asigurat că a pornit cu succes
* În folder-ul proiectului (*project*), am creat un fișier text numit *requirements.txt,* unde am specificat dependențele care sunt necesare pentru a fi instalate în containerul *docker* ale aplicației web:
  + *Django* versiunea 4.1.4
  + *Channels* versiunea 4.0.0
  + *Daphne* versiunea 4.0.0
  + *DjangoRestFramework* versiunea 3.14.0
* În folder-ul project, am creat un *Dockerfile*, care va fi utilizat în continuare pentru development. În fișier am scris comenzi pentru următoarele operații:
  + Am făcut pull la imaginea *python:3.10.7-alpine*
  + Am setat directorul de lucru (work directory)
  + Am setat variabilele de mediu *PYTHONDONTWRITEBYTECODE* și *PYTHONUNBUFFERED*
  + Am instalat dependențele din fișierul *requirements.txt*
  + Am făcut o copie la proiect
* Pentru a vedea dacă *Dockerfile*-ul funcționează, am creat un fișier *docker-compose.dev.yml*, pentru development, în care am menționat versiunea, imaginea care va fi construită, comanda de rulare a aplicației, portul expus. La final, l-am rulat cu comanda *docker-compose -f docker-compose.dev.yml up -d.*

1. Schimbarea bazei de date în *Postgres*

* Am șters fișierul *db.sqlite3*, care conține baza de date a aplicației folosind *Sqlite*
* Am completat *docker-compose.dev.yml* prin adăugarea imaginii bazei de date, și anume *postgres:13.0-alpine*, crearea unui volum *postgres\_data*, și setarea variabilelor de mediu *POSTGRES\_USER*, *POSTGRES\_PASSWORD* și *POSTGRES\_DB*.
* În *settings.py*, la variabila *DATABASES*, am adăugat configurațiile pentru *NAME*, *USER*, *PASSWORD*, *HOST* și *PORT*
* În requirements.txt am adăugat *psycopg2-binary* versiunea 2.9.1
* Am creat în folder-ul *chat*, folder-ul *management*.*commands*, iar apoi în *commands* fișierul *ensure\_adminuser.py*, care conține un script python pentru a crea un superuser folosit în aplicație, fiind dat username-ul, email-ul și parola.
* Am creat un fișier în folder-ul project numit entrypoint.sh, pentru a verifica dacă Postgres e healthy înainte de a aplica comenzile:
  + *python manage.py flush --no-input* = pentru a face refresh la baza de date
  + *python manage.py migrate* = pentru a popula baza de date cu tabele
  + *python manage.py ensure\_adminuser --username=user1 --email=user1@user.com --password=user1* = pentru a crea superuser-ul user1
  + *python manage.py ensure\_adminuser --username=user2 --email=user2@user.com --password=user2* = pentru a crea superuser-ul user2
* În *Dockerfile* am adăugat o comandă pentru a instala dependențele pentru *psycopg2*: *RUN apk update && apk add postgresql-dev gcc python3-dev musl-dev*. De asemenea, am făcut o copie la *entrypoint.sh* și i-am dat drept de execuție, ca la final să fie rulat în container.

1. Serverul *Gunicorn*, reverse proxy-ul *Nginx* și fișiere *docker-compose* diferite pentru development și producție

* Am adăugat în requirement.txt pachetele *gunicorn* versiunea 20.1.0, *uvicorn[standard]* versiunea 0.20.0 și *websockets* versiunea 10.4
* Am creat scriptul entrypoint.prod.sh, în care am scos comenzile de python.
* Am creat *Dockerfile.prod*, unde am folosit un builder ca imagine temporară de python care a făcut lint la proiect și a instalat dependențele, și o imagine finală de python, care a creat directoare pentru fișierele media și a rulat *entrypoint.prod.sh*.
* Am creat un folder *nginx* cu propriul său *Dockerfile* și configurările *nginx.conf*.
* În *nginx.conf* am creat un server care ascultă pe portul 80, i-am pus ca locație */* pentru endpoint-urile folosite de aplicația web și websocket, și locația */media/* ca să păstreze fișierele media în acel folder
* În fișierul *settings.py*, am adăugat variabila de mediu *CSFR\_TRUSTED\_ORIGINS* pentru *http://localhost:1337*
* Am creat un fișier *docker-compose.prod.yml* în care am creat imaginea aplicației web, am menționat dockerfile-ul să fie *Dockerfile.prod* și am schimbat comanda în *gunicorn project.asgi:application -k uvicorn.workers.UvicornWorker --bind 0.0.0.0:8000*. De asemenea, am adăugat imaginea *nginx*, cu volum pentru fișiere media (*media\_volume*) și portul 1337:80.
* Pentru a face deploy în modul production, se vor rula următoarele comenzi:
  + *docker-compose -f docker-compose.prod.yml down -v*
  + *docker-compose -f docker-compose.prod.yml up -d --build*
  + *docker-compose -f docker-compose.prod.yml exec web python manage.py migrate --noinput*
  + *docker-compose -f docker-compose.prod.yml exec web python manage.py ensure\_adminuser --username=user1 --email=user1@user.com --password=user1*
  + *docker-compose -f docker-compose.prod.yml exec web python manage.py ensure\_adminuser --username=user2 --email=user2@user.com --password=user2*

1. Fișierele *.env.dev* și *.env.prod*

* Am creat în folder-ul proiectului fișierul *.env.dev*, unde am setat variabilele de mediu *DEBUG* la 1, cheia secretă *SECRET\_KEY*, *DJANGO\_ALLOWED\_HOSTS* și informații legate de baza de date
* Am creat fișierul *.env.prod* cu aceleași variabile de mediu, numai că altă bază de date, altă cheie secretă, iar *DEBUG* e 0
* Am creat fișierul *.env.prod.db*, unde am ținut variabilele de mediu ale bazei de date pentru a fi înlocuite în *docker-compose.prod.yml*
* În settings.py am înlocuit folosind datele din *env.dev* și *env.prod* în variabilele de mediu *SECRET\_KEY* și *DATABASES*
* În *docker-compose.dev.yml* am adăugat env\_file-ul .*env.dev* la serviciul imaginii web
* În *docker-compose.prod.yml* am adăugat env\_file-ul *.env.prod* la serviciul imaginii web, iar la db am adăugat *.env.prod.db* în loc să menționez variabilele de mediu direct.

# Probleme întâlnite și modul de rezolvare

Cu toate că nu am avut probleme majore în rezolvarea cerințelor, am făcut câteva observații cu privire la rezolvarea lor care ar putea fi de folos în viitor:

* În cazul în care, după crearea containerelor, apare o eroare precum ”*module X not found*”, trebuie adăugat modulul la fișierul *requirements.txt*.
* Pentru că am lucrat cu websocket-uri, am avut nevoie de un server ASGI în loc de WSGI. Astfel, a trebuit să folosesc serverul *uvicorn* împreună cu *gunicorn* la comanda de rulare a containerului pentru ca aplicația web să funcționeze. De asemenea, am întâmpinat o eroare în care trebuia să mut mai sus niște linii de cod din fișierul *asgi.py* (*import django, os.environ.setdefault('DJANGO\_SETTINGS\_MODULE','project.settings'),django.setup()*).
* Pentru fișiere media, a trebuit să măresc size-ul default pentru imaginile încărcate în nginx la 100M.
* La crearea de superuseri necesară pentru ca să te autentifici în aplicație, am creat un script *python ensure\_adminuser.py* care primește ca parametru *username*, *email* și *parolă*, și creează un nou superuser. Am făcut acest lucru deoarece vroiam să automatizez comanda *createsuperuser*. [3]

# Concluzii

Această fază a proiectului mi-a dezvoltat capabilitățile de a crea containere în docker, m-am familiarizat cu ideea de proxy-uri pentru aplicație și am învățat despre importanța separării modului de deploy development și production.

## Referințe

[1] Github, <https://github.com/Biancazel/ASO_Project/tree/phase3>

[2] Tutorial deployment, <https://testdriven.io/blog/dockerizing-django-with-postgres-gunicorn-and-nginx/#production-dockerfile>

[3] ensure\_adminuser.py, <https://stackoverflow.com/questions/39744593/how-to-create-a-django-superuser-if-it-doesnt-exist-non-interactively>