Calorie Tracker

Proiect - Metode de dezvoltare software

Documentație Backend

Pentru dezvoltarea backend-ului din cadrul proiectului Calorie Tracker pentru cursul „Metode de Dezvoltare Software” am avut atât nevoia, cât și dorința de a folosi o varietate de tool-uri pentru a-mi ușura munca și a lucra la un standard cât mai apropiat de industrie. În acest document voi expune funcționalitatea acestor tool-uri și utilitatea lor în cadrul proiectului.

- ## Docker

Docker este o platformă open-source care permite dezvoltatorilor să creeze, să livreze și să ruleze aplicații în containere. Un container Docker este o unitate standardizată de software care include toate componentele necesare pentru a rula o aplicație, inclusiv codul, dependențele și bibliotecile. Docker facilitează izolarea aplicațiilor în containere pentru a asigura portabilitatea, fiabilitatea și securitatea acestora.

Docker Compose este un instrument suplimentar care permite dezvoltatorilor să definască și să ruleze aplicații multi-container. Acesta utilizează un fișier YAML pentru a defini serviciile, rețelele și volumele necesare pentru a rula aplicația într-un mediu izolat. Docker Compose permite dezvoltatorilor să configureze și să gestioneze întregul stack de aplicații, inclusiv toate dependențele, într-un mod simplu și eficient.

În cadrul proiectului am folosit un feature recent adăugat în Docker, numit **MultiStaging**. Astfel, cu un singur Dockerfile am definit mai multe stagii: de development, testare și producție. Am urmat standarde de industrie și am folosit **Python-Alpine** (cea mai mică imagine ce conține python), evitând, astfel, posibile breșe de securitate.

- ## Task Runner

Pentru a rula task-uri am folosit un tool-ul **poethepoet**[1]. Acesta a facilitat foarte mult procesul de dezvoltare, ajutându-mă să evit linii de terminal repetitive și consumatoare de timp. În cadrul proiectului am configurat următoarele comenzi: construirea imaginii docker, rularea aplicatie in containere orchestrate folosind docker-compose, curatarea proiectului de artefacte, generarea migrarilor si rularea testelor (aceasta consta in mai multe comenzi apelate secvential: ridicarea bazei de date, asteptarea acesteia, rularea testelor, oprirea bazei de date, curatarea de __pycache__, curatarea de artefacte nedorite). De asemenea, un alt aspect important al acestui tool constă în ușurința cu care pot schimba environment-ul în care rulez diferite comenzi, tot ce este necesar fiind specificarea argumentului env și pasarea unui path către un fisier .env.

[1] https://github.com/nat-n/poethepoet

- ## EnvFiles

Pentru a nu expune secrete în GitHub sau în Deploy, am folosit **EnvFiles**, care m-a ajutat la crearea cu ușurință a mai multor environment-uri: CI/CD, Dev, Localm Production și Test.

- ## Deploy

Pentru deploy am folosit o platformă free numită **Render**[2], în care hostez atât baza de date, cât și aplicația, care este construită și deployată prin intermediul imaginii de Docker scrisă de mine.

- ## CI/CD

Un best practice în industrie constă în configurarea pipeline-urilor în diferite platforme (Jenkins, GitLab, GitHub). Pentru acest proiect am folosit **GitHub** și, prin urmare, pipeline-ul a fost configurat în **GitHub Actions**. Acest pipeline are 2 pași pentru schimbările apărute într-un pull request și 3 pași pentru schimbările apărute în main branch (primii doi pași coincid)

1. Rularea unui **linter** pentru a menține lizibilitatea codului.
2. Generarea unei **baze de date nepersistente**, legarea acesteia de aplicația noastră rulată în pipeline și rularea testelor în cadrul acesteia.
3. Apelarea unui link ce declanșează deploy-ul. Acesta preia schimbările din ultimul commit din main branch și dă rebuild + rerun imaginii din deploy.

- ## Pre-commit

Pre-commit este un tool ce trebuie instalat local. La activarea acestuia pe un repository local, sunt rulate programele configurate în config file la fiecare commit. În proiectul nostru, am folosit **end-of-file-fixer**, **trailing-whitespaces**, **check-yaml**, **check-added-large-files** și **fourmat**. Toate acestea sunt lintere ce ajuta la menținerea unui cod mai curat. În urma unui commit, daca toate aceste programe se termină cu succes, commit-ul este efectuat. Altfel, acestea schimbă greșelile găsite și este necesară readăugarea lor în partea de staged, apoi efectuarea din nou a commit-ului.

[2] https://render.com/