Documentație pentru Testele K6

Preliminarii

- **K6** este un instrument open-source pentru testarea performanței și a încărcării (load testing).
- **Modul de rulare:** Pentru a rula un script, folosește comanda:

k6 run nume_script.js

- **Importuri utilizate:**
    - http – modulul folosit pentru efectuarea de cereri HTTP (GET, POST etc.).
    - sleep – funcția care simulează perioada de „gândire” a utilizatorului (pauză între iterații).
    - check – funcția de verificare a unor condiții pe răspunsul primit de la server.

Testul 1: Load Test de Bază

Explicații

- **Scopul Testului:**
  Acest script efectuează un load test de bază asupra endpoint-ului http://localhost:5081/api/curs.
  Se simulează comportamentul unui utilizator care face cereri GET.
- **Opțiuni (options):**
    - **stages:** Permite definirea treptată a numărului de utilizatori virtuali (VUs):
        - **Ramp-up:** Se crește numărul de VUs de la 0 la 20 în primele 30 de secunde.
        - **Sustenabilitate:** Se mențin 20 VUs timp de 1 minut.
        - **Ramp-down:** Se reduc VUs la 0 în următoarele 30 de secunde.
- **Funcția principală (default function):**
    - Efectuează o cerere GET către API.
    - Verifică dacă statusul răspunsului este 200 (folosind check).
    - Introduce o pauză de 1 secundă pentru a simula timpul de așteptare al utilizatorului.

Testul 2: Soak Test

Explicații

- **Scopul Testului:**
  Acest script realizează un **soak test** (test de rezistență pe termen lung), menit să verifice comportamentul sistemului sub o sarcină constantă prelungită.
- **Opțiuni (options):**
    - **scenarios:** Permite definirea scenariilor de testare.

- **executor 'ramping-vus':** Permite creșterea/decreșterea treptată a VUs.
- **startVUs:** Se începe cu 0 utilizatori virtuali.
- **stages:**
  - **Ramp-up:** Creșterea numărului de VUs până la 100 în 10 minute.
  - **Sustenabilitate:** Menținerea a 100 VUs timp de 1 oră.
  - **Ramp-down:** Reducerea la 0 VUs în ultimele 10 minute.
- **gracefulStop:** Permite un timp de 30 de secunde pentru oprirea grațioasă a VUs, asigurând finalizarea iterărilor curente înainte de terminare.

- **Funcția principală:**
  - Efectuează o cerere GET către API.
  - Verifică dacă răspunsul are status 200.
  - Simulează o pauză de 1 secundă între iterații.

Testul 3: Stress Test

Explicații

- **Scopul Testului:**
  Acest script este destinat **stress testing-ului** – testarea comportamentului sistemului sub o sarcină foarte mare, prin creșterea numărului de utilizatori virtuali la un nivel foarte ridicat.
- **Opțiuni (options):**
  - **scenarios:** Definește scenariul de test.
  - **executor 'ramping-vus':** Permite creșterea progresivă a numărului de VUs.
  - **startVUs:** Pornim cu 0 utilizatori.
  - **stages:**
    - Într-o singură etapă de 10 minute se crește numărul de VUs până la 1000.
    - Nu există o etapă explicită de ramp-down în acest scenariu, dar testul se va opri după ce s-a atins și menținut numărul țintă (sau se poate opri prin alte mecanisme).
  - **gracefulStop:** Se asigură că VUs au la dispoziție 30 de secunde pentru a-și finaliza iterațiile curente înainte de oprire.
- **Funcția principală:**
  - Efectuează o cerere GET la API.
  - Verifică statusul răspunsului pentru a se asigura că este 200.

- Include o pauză de 1 secundă între iterații, simulând un timp de gândire.

Considerații Finale

- **Verificări (Checks):**
  În toate scripturile se utilizează funcția check pentru a valida că serverul răspunde cu statusul 200. Acest lucru este important pentru a identifica eventualele erori sau comportamente neașteptate sub sarcină.
- **Simularea Utilizatorilor:**
  Fiecare script folosește fie opțiunea stages (testul 1) fie scenarios (testele 2 și 3) pentru a simula comportamentul utilizatorilor în funcție de tipul testului:
  - **Load Test:** Simulează un număr moderat de utilizatori pe o perioadă scurtă.
  - **Soak Test:** Menține o sarcină constantă pe o perioadă lungă pentru a testa stabilitatea sistemului.
  - **Stress Test:** Crește brusc sarcina pentru a determina punctul de cedare sau comportamentul sub presiune extremă.
- **Pauza (sleep):**
  Introducerea unei pauze de 1 secundă între iterații ajută la simularea unui comportament realist al utilizatorului și la evitarea suprasolicitării serverului cu cereri continue fără oprire.

Descriere Pipeline CI/CD pe Branch-ul Main

Pipeline-ul nostru CI/CD, configurat pe branch-ul **main**, este conceput pentru a asigura consistența și calitatea codului printr-o serie de etape automate. În cadrul acestui pipeline, se inițializează un mediu containerizat care garantează izolarea și reproducibilitatea mediului de execuție. Ulterior, se configurează variabilele de mediu necesare pentru aplicația .NET, asigurându-se astfel un context optim pentru rularea testelor. În final, se execută testele de performanță K6, care validează integritatea și eficiența aplicației înainte de implementarea în producție.