Performance tests

Nasze testy sprawdzają już "***co***" robi nasza aplikacja. W celu upewnienia się, że w przyszłości **performance** (czyli to "***jak szybko***", **"jak stabilnie"** działa aplikacja) nie ulegnie pogorszeniu stworzymy dziś podstawowy performance test.
Test będzie sprawdzał ile czasu zajmuje naszej aplikacji wysłanie odpowiedzi na API request.

Zadanie:

Stwórz nowy folder a w nim plik z dwoma testami API:

- Test który używając API stworzy i usunie jeden po drugim 100 kont. Test ma sprawdzić czy otrzymanie responsu w każdym przypadku będzie krótsze niż 0.5s
- Test który używając API stworzy konto i wywoła zaksięgowanie 100 przelewów przychodzących. Odpowiedź w każdym requescie powinna być krótsza niż 0.5s. Test powinien sprawdzić również ostateczne saldo

Najprostszym sposobem jest użycie pętli i ustawienie timeoutu w naszym requescie (jak dodać timeout).
Ważne również jest aby sprawdzać czy response code jest poprawny.

Co może wpływać na szybkość działania naszej aplikacji?
Czemu testowanie "jednowątkowe" może nie wykryć realnych problemów w naszej aplikacji?

Jest to olbrzymie uproszczenie. Jeżeli chcesz zgłębić temat polecam narzędzia takie jak Locust lub k6 (js stack).

Github pipelines

Gdy wszystkie nasze testy działają lokalnie pora dodać je również w github pipeline.
Nasze repozytorium powinno zawierać 3 github workflows:

1. unit testy i sprawdzanie coverage (pipeline powinien już działać)
2. API testy
3. Gherkin scenarios
4. API Performance

Podpowiedzi:

- nie zapomnij nadawać odpowiednich nazw pipelinom (pierwsza linijka pliku .yaml)
- pamiętaj o dodaniu wszystkich używanych bibliotek w pliku requirements.txt
- pamiętaj, że testy API oraz Gherkin potrzebują odpalonej aplikacji (flask)
- flask powinien być odpalano w oddzielnym kroku. Aby nie blokować przejścia do kolejnych kroków pamiętaj o dodaniu
  & na końcu komendy: `flask --app app/api.py run &`