unity GROUP

Efektywne testy integracyjne w PHP

/ 25.06.2022

/ Kamil Pińkowski

> Testy integracyjne vs. biznes

- "Testy, a na co to komu potrzebne?"
- "Przecież nie mamy żadnej integracji"
- "Impreza integracyjna?

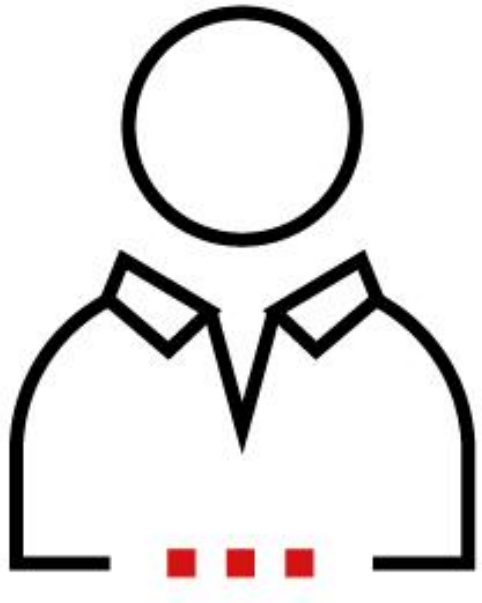
> Testy integracyjne vs. programiści

- "Nigdy nie słyszałem"
- "Czym się różnią od innych?"
- "Problemy z setupem"
- "Do niczego się nie przydają"
- "Zbyt wolne"

*źródło: https://knowyourmeme.com/photos/1900244-gigachad

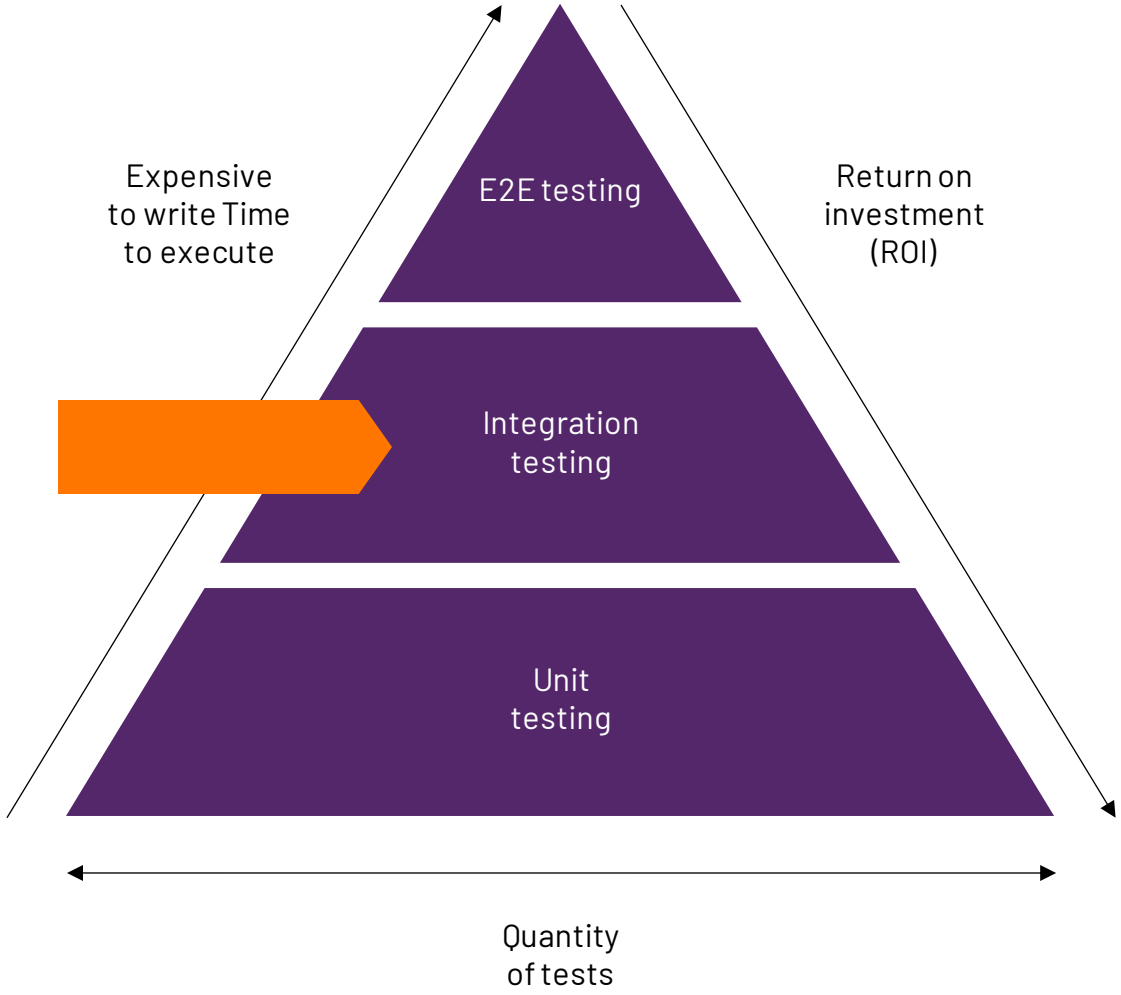
> Parę słów o ... :)

- Kamil Pińkowski
- PHP / Symfony
- Unity Group
  - Tech Lead
  - PHP Dev
- Dolnośląska Szkoła Wyższa
  - Pracownik dydaktyczny
- PIM
  - Akeneo
  - Pimcore



> Testy integracyjne w modelu piramidy testów

- Średni czas wykonania
- Średni koszt utrzymania
- Średnia złożoność



> Testy jednostkowe

- Testują pewien wycinek (jednostkę) oprogramowania
- Konieczne zastosowane izolacji od innych części programu
  - Osiągane przy pomocy test doubles ("mocki")
- Celem jest wykrycie defektu w wybranej jednostce

> Testy integracyjne

- Testują kilka modułów / komponentów jednocześnie
- Używają "prawdziwych" komponentów (np. Bazy danych)
- Celem jest wykrycie defektów w interakcjach pomiędzy klasami, modułami lub komponentami

> Zalety testów integracyjnych

Testy integracyjne

ZALETY

Umożliwiają wykrycie błędów na poziomie integracji między modułami / klasami / komponentami aplikacji

Umożliwiają znalezienie defektów w miejscach, których jawnie nie testujemy (np. Błędne definicje serwisów)

> Wady testów integracyjnych

Testy integracyjne

WADY

Wysokie pokrycie jest ciężkie do osiągnięcia

- 3 klasy, każda z 3 edge case to:
- 3*3 = 9 testów jednostkowych do napisania (każdy z osobna)
- 3^3 = 27 testów integracyjnych do napisania (każda kombinacja), lub

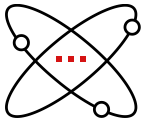
Setup bywa problematyczny

- Dla każdego frameworka / platformy setup się różni

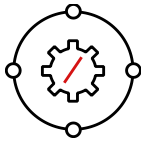
Wolniejsze od testów jednostkowych

Droższe w utrzymaniu od testów jednostkowych

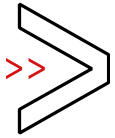
> Cechy efektywnych testów integracyjnych



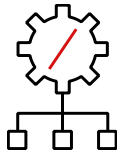
Uruchamiane ze wszystkimi komponentami aplikacji (dependencies, DB, cache etc.)



Posiadają mechanizm do kreowania danych testowych (fixtures)



Bezstanowe



Posiadają czytelną strukturę

Mockują integrację z systemami trzecimi

Deterministyczne

Uruchamiane równolegle (parallel)

> Przykładowy setup testów integracyjnych

> Podsumowanie

- Omówienie koncepcji testów integracyjnych
- Setup odseparowanego środowiska na potrzeby testów
- Oddelegowanie odpowiedzialności w testach (Assert, Fixture, Utll)
- Test integracyjny napisany w serwisie, który posiada jakieś zależności
  - Serwisy bez zależności testujemy jednostkowo
- Mechanizm do kreowania danych testowych (Fixtures)
- Użycie losowych generatorów danych, tak, aby nie zaburzyły determinizmu testów
- Użycie transakcji i rollbacku w celu osiągnięcia bezstanowości
- Zastosowanie biblioteki paratest w celu zoptymalizowania czasu uruchomienia testów

unity GROUP

Dziękuję!

Kamil Pińkowski
kamil.pinkowski@unitygroup.pl
https://www.linkedin.com/in/kamil-pinkowski/
https://github.com/kpinkowski/summit2022