Wyższa Szkoła Ekonomii
i Informatyki w Krakowie

Testy integracyjne/e2e

Why oh why? A gdzie unity?

- Koncentracja na perspektywie użytkownika
- Ochrona przed regresją funkcjonalności
- "Spokojniejszy", pewniejszy refaktor kodu
- Redukują koszty - "czynnik białkowy" mniej klika
- W przypadku błędów pokazują która kluczowa funkcjonalność ucierpiała
- Testy integracyjne/e2e są znacznie mniej podatne na zmiany kodu niż unit testy (który to kod nie zmienia funkcjonalności)

WSEI

Testy End to End

- Weryfikują kluczowe procesy biznesowe
- Weryfikują współpracę wszystkich części/modułów aplikacji (front/back ale również np. płatności/koszyk/produkty w sklepie internetowym)
- Weryfikują integrację z zewnętrznymi systemami (płatności, kurier, broker informacji etc)
- Symulują zachowanie użytkownika
- Mają zazwyczaj długi czas wykonania (zależy to w głównej mierze od aplikacji i testowanego procesu)

WSEI

Testy integracyjne

- Weryfikują procesy wewnątrz aplikacji (bez zewnętrznych zależności)
- Mogą obejmować różne warstwy/części aplikacji, np.:
  - działanie pojedynczego komponentu (np. progress bar)
  - uzupełnienie całego formularza (bez wysyłki danych)
  - czy ikonka z ilością produktów w koszyku aktualizuje się po dodaniu/usunięciu produktu z koszyka?
  - pełen proces zakupowy aż do kliknięcia w "złóż zamówienie"
- Mockując scenariusze brzegowe można badać stabilność/bezpieczeństwo systemu

WSEI

Budowa testów

- Zanim zaczniesz pisać testy - zapisz co chcesz mieć przetestowane
- Test powinien weryfikować dokładnie jeden proces.
- Testy nie powinny się powielać (np. jeden test jest częścią testu innego procesu)
- Pilnuj nazewnictwa - nazwa ma dokładnie mówić co się wywróciło
- Testy są niezależne od innych testów
- …i od czynników zewnętrznych (np. czasu - wywala się po 18:00;))

WSEI

Budowa testów

- Test na dzień dobry powinien się wywalić. Będziesz wiedział że w ogóle działa:)
- Zmiana kodu (bez zmiany funkcjonalności) wywala test? Test jest zależny od implementacji:(
- Test często się wywala? Może testuje kilka procesów lub za duży proces?
- Pisz test jakbyś nie znał(-a) kodu. Masz to co użytkownik.

WSEI

Narzędzia

- Playwright
- Cypress
- Puppeteer
- Selenium
- PyTest
- …i kg innych

WSEI

Playwright

- API do tworzenia testów e2e
- Obsługuje wszystkie popularne przeglądarki (Chromium, Firefox, WebKit)
- Pracuje na Windows, Linux i macOS
- Testy można pisać w JS, TS, Python, Java i .NET
- Możliwe jest korzystanie z wielu instancji różnych przeglądarek równolegle

WSEI

Playwright

- Dla każdego testu powstaje nowy kontekst przeglądarki
- Umożliwia przechowywanie stanu kontekstu pomiędzy testami (np. jednorazowe logowanie użytkownika)
- Testy mogą ze sobą łączyć procesy z wielu kart przeglądarki (i różnych origins)
- Mamy do dyspozycji szerokich zestaw dodatkowych narzędzi - m.in. execution trace, screenshoty, video, możliwość inspekcji/debugowania testowego kodu oraz testowanej aplikacji

WSEI

Przygotowanie testów

- Przygotowanie wymagań testowych (środowisko oraz metryki testów)
- Wybranie procesów biznesowych do testów - tworzenie scenariuszy
- Development
- Uruchomienie
- Raportowanie

WSEI

Instalacja

```
npm init playwright
```

WSEI

Podstawowe polecenie CLI

- Uruchomienie testów: **playwright test**
- Testy w trybie headed**: playwright test —headed**
- Testy z widocznym UI testowym: **playwright test —ui**
- Testy z wybranym projektem: **playwright test —project=chromium**
- Debugowanie testów**: playwright test —debug**

- Wyświetlanie raportu z ostatnio uruchomionych testów: **playwright show-report**

WSEI

Wyższa Szkoła Ekonomii
i Informatyki w Krakowie