|  |  |
| --- | --- |
| **POLITECHNIKA ŚWIĘTOKRZYSKA W KIELCACH**  **Wydział Elektrotechniki, Automatyki i Informatyki**  **Projekt Technologii Obiektowych** | |
| Temat projektu:  41 – Porównanie rozwiązań związanych z testowaniem | Autorzy:  Daniel Szwajkowski, Kacper Jurek |

**Podział pracy:**

Daniel Szwajkowski – testy jednostkowe (JUnit, JTest, TestNG),

Kacper Jurek –testy e2e (Cypress, Selenium, Cucumber).

1. **Testy E2E**

Głównym celem testowania End-To-End jest testowanie na podstawie doświadczenia użytkownika końcowego poprzez symulację rzeczywistego scenariusza użytkownika i walidację testowanego systemu wraz z komponentami pod kątem integracji i integralności danych. Ten rodzaj testów staje się coraz bardziej istotny, gdyż oprogramowanie jest coraz bardziej złożone i skomplikowane. Możemy dostrzec coraz większą liczbę integracji z innymi systemami, podsystemami, czy też usługami. W przypadku awarii jednego z takich elementów istnieje poważne ryzyko, że cały system może ulec awarii, czy też może nie spełniać swoich głównych założeń w sposób prawidłowy.  
Testy E2E są zwykle przeprowadzane na gotowych systemach (w końcowej fazie developmentu – ponieważ muszą już w pełni działać wszystkie funkcjonalności, które będziemy testować).

.

|  |  |
| --- | --- |
| **Testy funkcjonalne** | **Testy End to End** |
| Testowanie jest ograniczone do jednego modułu, systemu, aplikacji | Testowanie obejmuje wiele aplikacji, usług i bardziej opiera się na pełnej integracji oprogramowania |
| Zapewnia, że ​​testowane oprogramowanie spełnia kryteria akceptacji | Zapewnia, że ​​procesy biznesowe działają prawidłowo, zgodnie z założeniami |
| Testuje sposób, w jaki pojedynczy użytkownik korzysta z aplikacji | Testuje sposób pracy wielu użytkowników w aplikacjach |
| Wykonuje się je na różnych etapach developmentu | Wykonuje się je, wtedy, gdyż faza developmentu zbliża się do końca, lub gdy jest już zakończona. |

**Jak wygląda cały proces testów E2E**

Na standardowy proces E2E składaj się następujące fazy:

1. Zbieranie wymagań testowych (w kontekście pełnych, biznesowych funkcjonalności)
2. Projektowanie scenariuszy testowych (do wcześniej zebranych wymagań procesów biznesowych)
3. Faza developmentu – tworzenie programowania
4. Testowanie End to End – wykonywanie wcześniej opracowanych scenariuszy testowych

Każda aplikacja komunikuje się z wieloma systemami i bazami danych poza własnym środowiskiem. To sprawia, że ​​całościowy przepływ pracy aplikacji jest dość skomplikowany do przetestowania.

Testy End to End określają, czy różne zależności, integracje aplikacji działają poprawnie. Sprawdzają również, czy między wieloma komponentami systemu przesyłane i odbierane są poprawnie informacje.

* **Backend** : testy End to End weryfikują warstwy bazy danych i integracji backendowych. Jest to konieczne, ponieważ podstawowe główna logika systemu dzieje się właśnie „pod spodem”
* **System wielowarstwowy** : jeśli aplikacja ma złożoną architekturę, której przepływ pracy działa na wielu warstwach, testy E2E są niezbędne do weryfikacji ogólnych funkcji, a także interakcji między poszczególnymi warstwami w architekturze.
* **Środowisko rozproszone** : jeśli aplikacja jest oparta na architekturze SOA (architektura zorientowana na usługi) lub środowiskach chmurowych, konieczne jest testowanie End to End. Jest również niezbędny w przypadku aplikacji składających się z wielu komponentów, które muszą działać w tandemie, aby zapewnić pomyślne działanie.
* **Spójne doświadczenie użytkownika** : ponieważ testowanie E2E obejmuje frontend, zapewnia również, że aplikacja zapewnia wrażenia użytkownika działające na wielu urządzeniach, platformach i środowiskach. Na przykład testy zgodności przeglądarek jest ważną częścią testowania E2E w tym zakresie.

**Cypress**

Cypress to javascriptowy framework przede wszystkim do testów end to end. Twórcy tego frameworka zapewniają nas, że dzięki niemu: konfiguracja, tworzenie, uruchamianie oraz debugowanie testów będzie proste.

Idea cypressa jest zbliżona do koncepcji JEST – to znaczy zebranie wszystkich niezbędnych narzędzi takich jak framework, biblioteki asercji, wrappera czy dodatkowych bibliotek w jeden tool.

### Co odróżnia cypressa od innych narzędzi?

* Nie opiera się na Selenium – zbudowany jest na nowej architekturze. Cypress działa w tej samej pętli uruchomieniowej, co twoja aplikacja.
* Koncentruje się na sprawnym wykonywaniu testów – jego użycie jest intuicyjne i proste.
* Jest uniwersalny – działa prawdopodobnie na wszystkim co może być uruchomione w przeglądarce. Dobrze powinien sobie poradzić nawet na starszych aplikacjach, nie tylko tych używających nowoczesnych frameworków.
* Testy są pisane w javascripcie – dla jednych wada dla innych zaleta ale cypress jest oparty na javascripcie.
* Wszystko czego potrzeba w 1 narzędziu – Nie trzeba instalować osobnych narzędzi aby uruchomić testy.
* Narzędzie zarówno dla programistów jak i QA.
* Cypress jest szybki – pozwala na równoczesny development i testowanie.

**Instalacja**

Instalacja Cypressa nie wydaje się niczym skomplikowanym, tym bardziej jeśli miało się już do czynienia z instalacją z użyciem npm.

cd /scieżkaProjektu/

Zainstaluj cypressa poleceniem:

npm install cypress --save-dev

W ten sposób instalujemy najnowszą wersję cypressa lokalnie. I tak naprawdę, gdy instalacja dobiegnie do końca możemy zacząć działać. Możemy też ściągnąć paczkę instalacyjną ze strony  cypress.io instalacja z konsoli jest dużo szybsza i wygodniejsza.

Po zainstalowaniu w folderze projektu pojawi się folder node\_modules oraz plik package-lock.json.

By uruchomić Cypressa wystarczy użyć komendy:

node\_modules\\.bin\cypress open

Efektem wykonania tego polecenia powinno być uruchomienie Test Runnera.  
Po kliknięciu przycisku ‘OK, got it’ który się pojawił na ekranie runnera pojawi się stworzona struktura danych w naszym folderze projektowym. Niech to będzie nasza baza do nauki Cypressa.

**Struktura plików**

To jeszcze słów kilka o strukturze. Stworzone zostały 4 główne katalogi, które są godne uwagi:

* **Fixtures** – pliki które zawierają statyczne dane, mogące być użyte podczas testów. Zwykle odnosimy się do nich z użyciem cy.fixture(). Używamy gdy chcemy przetestować funkcjonalność opartą na żądaniach sieciowych (xhr,/ ajax) , framework może sztucznie wstawić dane podczas testu czyli pełnić rolę Stub’a.
* **Test files** – pliki testów domyślnie są umiejscowione w folderze integration. Cypress wygenerował dodatkowo folder examples z przykładami użycia.
* **Plugin files** – umożliwiają korzystanie, modyfikowanie, rozszerzanie wewnętrznych zachowań frameworku. Domyślnie Cypress używa pluginów zawartych w katalogu plugins w pliku index.js. Można również pobrać bardzo dużo pluginów dostępnych na stronie cypress, często współtworzonych przez użytkowników.
* **Support files** – tu można wrzucać reużywalne funkcje, niestandardowe polecenia itp które mają być dostępne dla wszystkich plików testowych.

**Selenium**

Selenium to popularne narzędzie do automatyzowania operacji wykonywanych przez przeglądarkę. Głównym zastosowaniem Selenium są testy aplikacji webowych, w szczególności ich frontentu. Za pomocą tego narzędzia możemy pokryć testami bardziej frontendową część aplikacji, np. kod JavaScript - coś czego zwykłe testy nie są w stanie obsłużyć.

Selenium 2 Webdriver to zewnętrzna biblioteka, która łączy w sobie sterownik do danej przeglądarki oraz serwer Selenium. Webdriver dostarcza użytkownikowi gotowe API pozwalające na interakcję z przeglądarką. Dostarczony interfejs jest minimalistyczny i bardzo prosty w użyciu.

### ****Zalety****

• Darmowe narzędzie (licencja open-source)  
• Ogromne internetowe Community  
• Kompatybilność z najpopularniejszymi systemami operacyjnymi (Windows, Linux, Max OS)  
• Pełna obsługa wielu popularnych języków programowania  
• Kompatybilność z najpopularniejszymi przeglądarkami internetowymi (Chrome, FireFox, Edge, Safari)  
• Kompatybilność z wieloma szeroko stosowanymi narzędziami jak np. Maven, Jenkins, Docker  
• Stosunkowo liczne grono specjalistów deklaruje chęć pracy w Selenium

### ****Wady****

• Brak wbudowanych mechanizmów raportowych  
• Do pracy niezbędne są wysokie umiejętności programowania  
• Czasochłonny proces tworzenia skryptów testowych

Selenium to rozwiązanie służące do automatyzacji testów aplikacji webowych. Tak ja w przypadku Cypress – jest zupełnie darmowe, oparte na licencji open source. Był przez wiele lat i jest nadal niekwestionowanym liderem rynku narzędzi do automatyzacji testów aplikacji internetowych. Rozwiązanie to zapewnia nam współpracę ze wszystkim popularnymi przeglądarkami internetowymi- wykorzystując do tego celu ich natywne sterowniki (drivery) np. ChromeDriver (przeglądarka Chrome), czy GeckoDriver (przeglądarka FireFox). Warto podkreślić, że do pracy z Selenium Webdriver możemy korzystać z wielu popularnych języków programowania np. Java, C#, JavaScript, Python czy Ruby. Dla mnie jedną z największych zalet tego narzędzia jest też bardzo pokaźne internetowe Community (którego tak bardzo brakuje Cypressowi). Na szereg popularnych problemów można szybko i łatwo znaleźć odpowiedź w sieci.