Wyższa Szkoła Technologii Informatycznych w Katowicach

REFERAT PRACY DYPLOMOWEJ

Temat pracy: Projekt i implementacja środowiska do automatyzacji przeprowadzania testów aplikacji internetowych w oparciu o metodykę Behavior Driven Development.

Autor: Stepowany Sebastian

Promotor: dr Jadwiga Bakonyi

Kategorie: testowanie oprogramowania
Słowa kluczowe: testowanie, platforma testowa, BDD, aplikacje internetowe, generowanie raportu.

1. Cel i podstawowe założenia

Głównym celem pracy jest przygotowanie środowiska ułatwiającego pracę z automatyzacją testów aplikacji internetowych. Połączenie technologii zastosowanych do zbudowania platformy jest rozwiązaniem własnym autora. Najważniejsze aspekty pracy to zastosowanie metodyki Behavior Driven Development oraz stworzenie od podstaw generowania raportu z automatycznie przeprowadzonych testów. Kolejnym aspektem pracy jest przedstawienie jej pod kątem edukacyjnym. W pracy przedstawiono opis tworzenia środowiska wraz z jego testowaniem oraz użytkowaniem.

2. Realizacja projektu

Celem projektu było samodzielne stworzenie platformy do tworzenia i przeprowadzania testów automatycznych aplikacji internetowych. Projekt zakładał stworzenie automatycznie generowanego raportu z testów, który powiązany był by z metodyką Behavior Driven Development.

Realizacja projektu odbywała się zgodnie z wszelkimi zasadami sztuki informatycznej obowiązującymi w następujących podstawowych etapach:

- zapoznanie się z istniejącymi na rynku rozwiązaniami i analiza ich zalet oraz wad,
- opracowanie wymagań funkcjonalnych i niefunkcjonalnych,
- zaprojektowanie architektury platformy,
- zaprojektowanie i implementacja automatycznego raportowania testów,
- opracowanie wymagań funkcjonalnych i niefunkcjonalnych aplikacji testowej,
- napisanie aplikacji testowej,
- konfiguracja platformy,

- testowanie i weryfikacja poprawności działania platformy,
- utworzenie instrukcji tworzenia podstawowego testu oraz jego wykonanie.

W trakcie tworzenia pracy wykorzystano niżej wymienione narzędzia oraz technologie:

- język programowania „Python”,
- bibliotekę „Lettuce”,
- serwer ciągłej integracji „Jenkins”,
- system kontroli wersji „SVN”,
- narzędzie „Tortoise SVN’
- składnia języka „Gherkin”,
- język programowania „JavaScript”,
- serwer „NodeJS”
- bibliotekę „AngularJS”,
- bilbiotekę „ExpressJS”
- bazę danych „MongoDB”.

3. Produkt końcowy – stworzona platforma

3.1 Podstawowe wymagania aplikacji

Platforma testowa została przygotowana w taki sposób aby jej instalacja jak i konfiguracja nie wymagały dużego nakładu pracy. Wymaga ona jednak podstawowej wiedzy z zakresu programowania w celu automatyzacji testów.

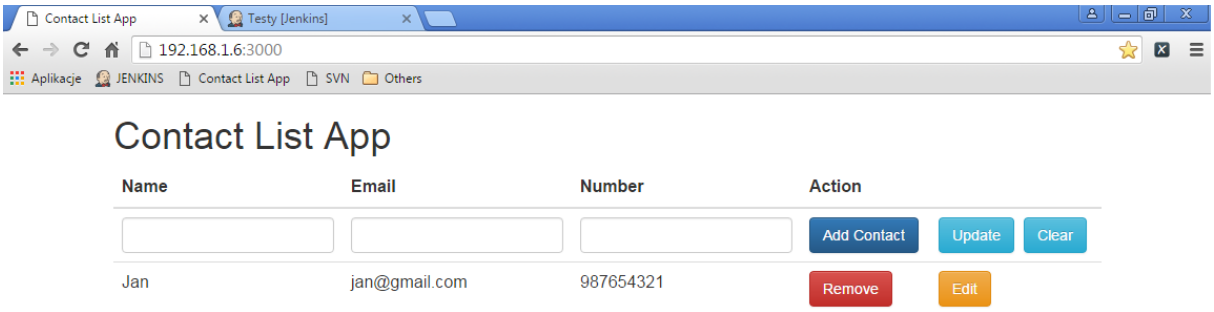
Ważnym aspektem platformy jest zastosowanie metodyku Behavior Driven Development. Ułatwia ona komunikację pomiędzy interesariuszami tworzonego i testowanego oprogramowania w celu zapewnienia największej jakości produktu końcowego.

3.2 Funkcjonalność platformy

3.2.1 Aplikacja testowa

Funkcjonalności przygotowanej aplikacji testowej „Książka kontaktowa” (Rysunek 1):

- widok listy kontaktów,
- dodanie kontaktu,
- edycja kontaktu,
- usunięcie kontaktu,
- czyszczenie pól kontaktu.



Rysunek 1. Interfejs graficzny aplikacji testowej

3.2.2 Platforma testowa

Uruchomienie platformy polega na instalacji oraz konfiguracji elementów wchodzących w jej skład: serwera ciągłej integracji, systemu kontroli wersji oraz języka programowania „Python". Ponad to należy użyć implementacji generowania automatycznego raportu z przeprowadzonych testów.

Osoby korzystające z platformy można podzielić na grupy. Pierwsza z nich to grupa osób tworzących scenariusze testowe napisane w składni języka „Gherkin". Do tej grupy należało by zaliczyć klienta tworzonego oprogramowania lub osoby oddelegowane do pracy z funkcjonalnościami aplikacji. Kolejną grupą są osoby tworzące programistyczną implementację przygotowanych wcześniej scenariuszy testowych. Do tej grupy zaliczyć można programistów lub testerów piszących testy automatyczne.

3.2.2.1 Wykorzystanie platformy

Posługując się aplikacją testową oraz jej wymaganiami funkcjonalnymi można przygotować scenariusz testowy. Jest to pierwszy etap tworzenia automatycznego testu, który zostanie wykonany na platformie. Wybrane wymaganie funkcjonalne:

„Wpis do spisu powinien posiadać: nazwę, adres emaila, numer telefonu. Dodanie wpisu zatwierdza się dedykowanym przycisku."

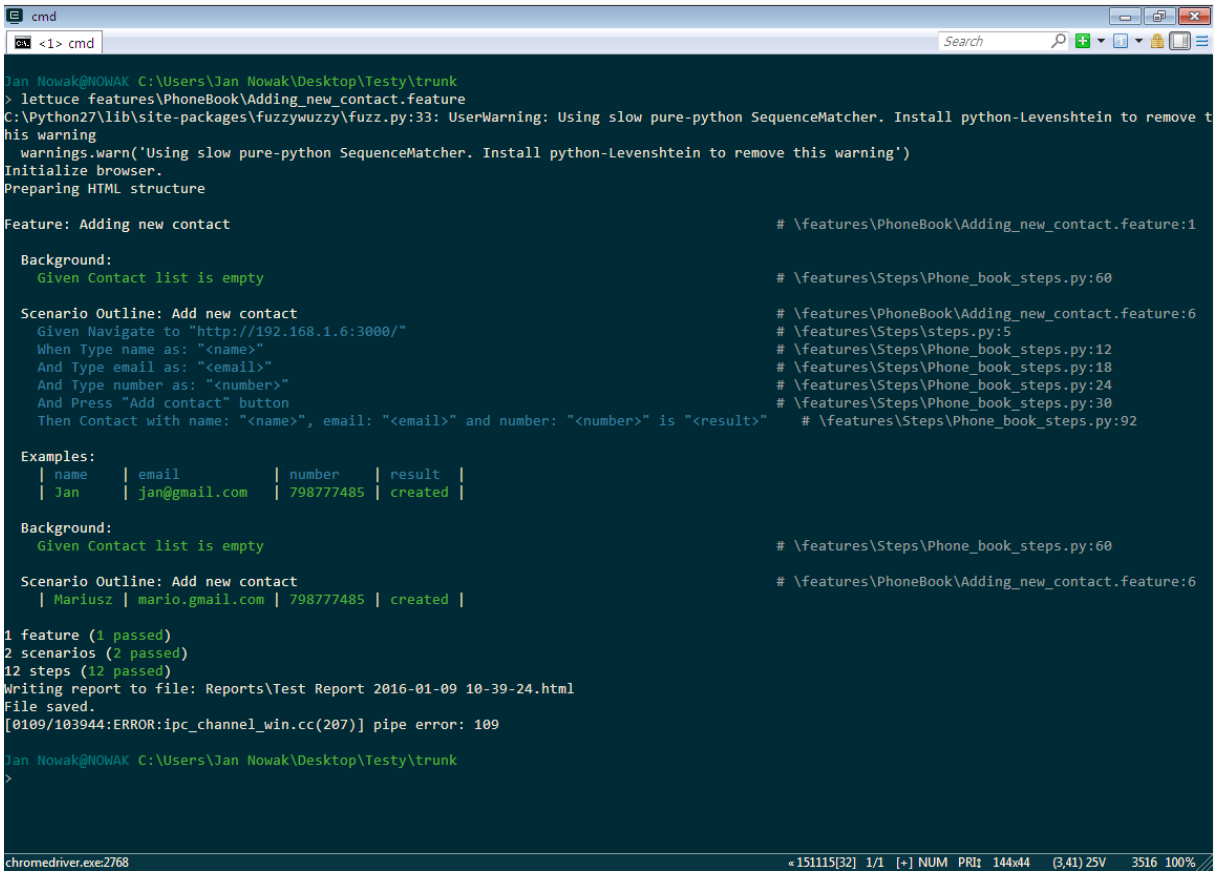
Scenariusz testowy wykorzystany zostaje do automatyzacji testu. Test zostanie użyty przez testera, programistę lub serwer ciągłej integracji. Przykładowy przypadek testowy podanego wcześniej wymagania funkcjonalnego przedstawiono w tabeli poniżej (Tabela 1).

Tabela 1. Przykładowy przypadek testowy

Przypadek testowy	Opis przypadku
Feature: Adding new contact **Background**: **Given** Contact list is empty **Scenario Outline**: Add new contact **Given** Navigate to the test application **When** Type name as: "<name>" **And** Type email as: "<email>" **And** Type number as: "<number>" **And** Press "Add contact" button **Then** Contact with name: "<name>", email: "<email>" and number: "<number>" is "<result>" **Examples**: | name | email | number | result | | Jan | jan@gmail.com | 798777485 | created | | Mariusz | mario.gmail.com | 798777485 | not created |	Wszystkie pogrubione słowa są dedykowane dla składni języka „Gherkin”. **Feature** – nazwa przypadku testowego (nazwa funkcjonalności). **Background** – warunki jakie należy spełnić przed wykonaniem każdego scenariusza. **Scenario Outline** – nazwa złożonego scenariusza. Jest to scenariusz, który zostaje wykonany wielokrotnie w zależności od ilości zestawów danych jakie ma uruchomić. **Given** – początkowe kroki, które prowadzą do przygotowania wykonania scenariusza. Mogą być rozbudowane o kolejne kroki słowem kluczowym „And”. **When** – krok, który wykonuje akcję najczęściej na GUI (graficzny interfejs użytkownika). Może również być rozbudowany o kroki słowem kluczowym „And”. **Then** – definiuje kroki sprawdzające poprawne wykonanie scenariusza. Może również być rozbudowany o kroki słowem kluczowym „And”. **Examples** – tabela z zestawami danych, które wiersz po wierszu będą używane do kolejnych iteracji scenariusza.

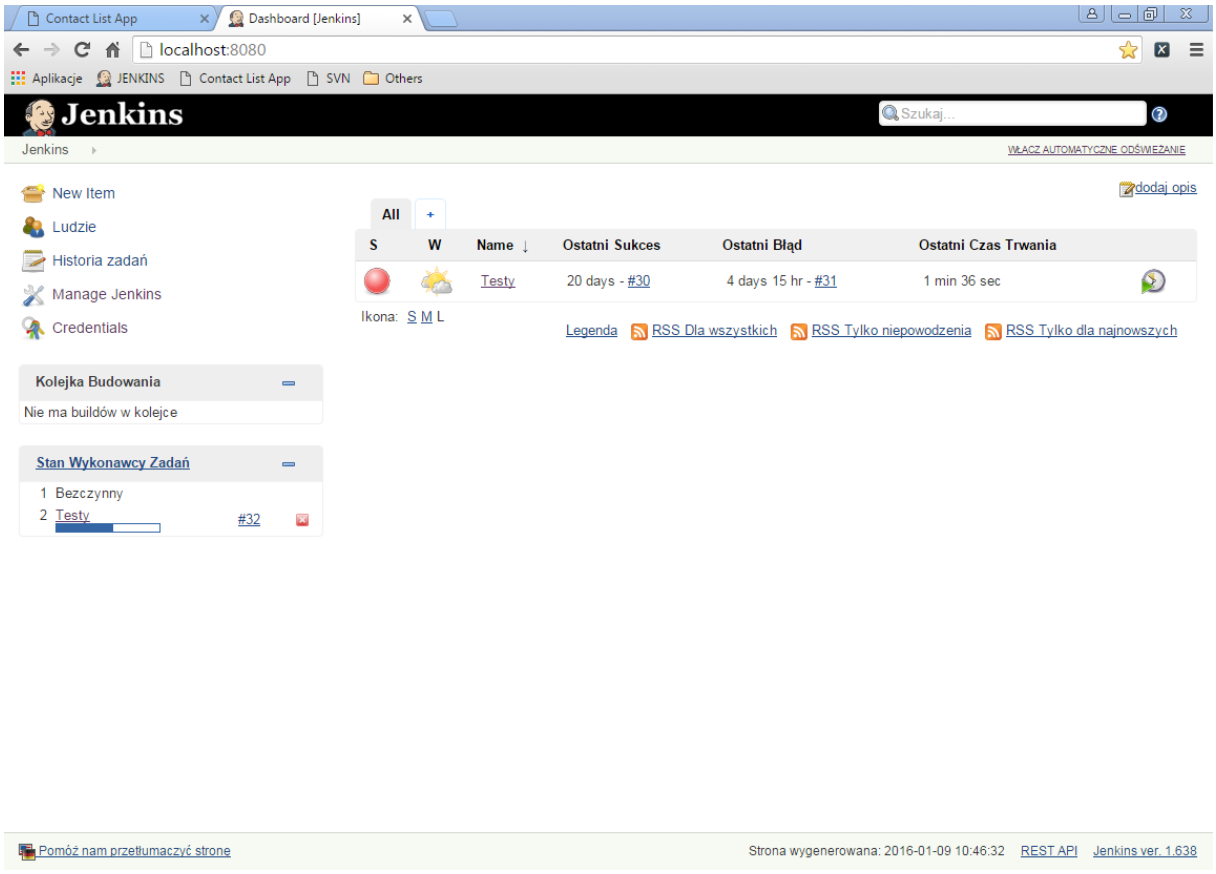
Kolejnym etapem jest implementacja programistyczna kroku testowego. Implementacja kroku zależy od wiedzy i umiejętności testera, programisty jednakże wymagane jest napisanie go w języku programowania „Python”.

Napisany i gotowy test do uruchomienia można uruchomić na dwa sposoby. Pierwszy z nich to uruchomienie na lokalnej stacji roboczej gdzie wyniki przedstawione są w konsoli danej stacji roboczej (Rysunek 2).



Rysunek 2. Wynik testu – lokalnie

Kolejnym sposobem jest automatyczne uruchomienie testów na serwerze ciągłej integracji (Rysunek 3). Z tego etapu może skorzystać każdy interesariusz projektu. Szczególnie ważne jest to dla klienta gdyż dostaje on raport z testów bez konieczności kontaktu z zespołem projektowym. W przypadku tego sposobu wyniki testu widoczne są w konsoli konkretnego zadania serwera ciągłej integracji.

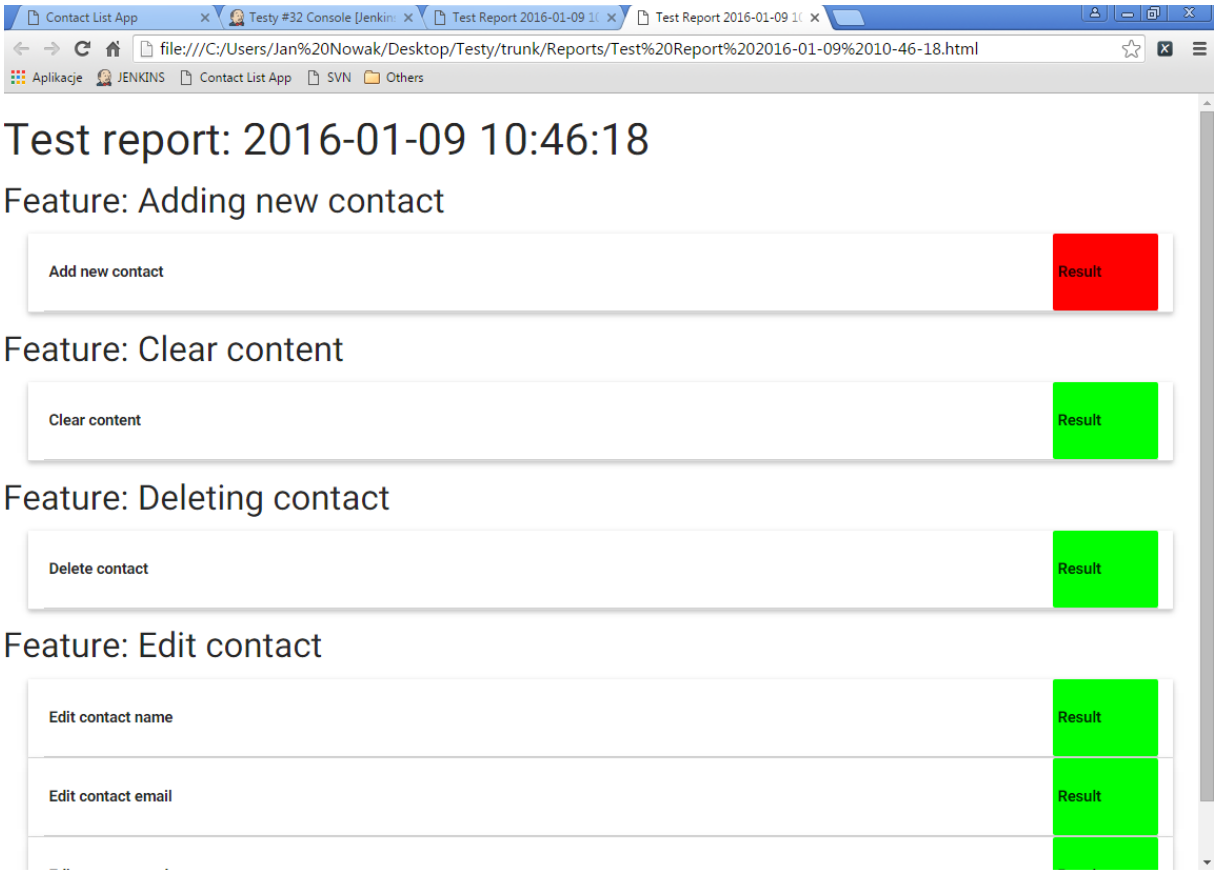


Rysunek 3. Uruchomione zadanie na Jenkinsie

Oba przedstawione sposoby uruchamiania dają informacje na temat poprawności działania testowanej aplikacji. Poza wynikami w konsoli, z obu sposobów uruchomienia za każdym uruchomieniem testów generowany zostaje raport z testów. Raport zostaje umieszczany w systemie kontroli wersji skąd może być odczytany przez osoby mające do niego dostęp. Raport generowany jest w formacie HTML i może dla jego czytelności należy przeglądać go w przeglądarce internetowej.

Na rysunku (Rysunek 4) przedstawiono przykładowy raport z testów, w którym jeden ze scenariuszy testowych wykonany został z błędami. Błędy zaznaczone zostają po przez odpowiednie kolorowanie co w przypadku przygotowanej platformy jest kolorem czerwonym.

Raport przygotowany w takiej postaci jest czytelny przede wszystkim dla osób, które nie są doświadczone w korzystaniu konsoli systemowej. Dzięki temu klienci nie muszą uczyć się odczytywać wyników z konsoli. Co ważne taki raport może służyć jako dokument, który zostaje dostarczany dla klienta.



Rysunek 4. Raport z testów

4. Informacje o możliwości wykorzystania / wykorzystaniu pracy

Przygotowana platforma jak i aplikacja testowa jest przykładem rozwiązania problemu automatycznego testowania aplikacji internetowych. Zarówno platforma jak i aplikacja testowa skierowane są do wszystkich zainteresowanych tego typu platformami oraz aplikacjami internetowymi.