Automatyczna platforma testowa dla front-end stron internetowych oraz aplikacji sieciowych w modelu SaaS – BrowserSpot

Szymon Binek, Jakub Góral
ClickRay Sp. z o.o., Halicka 9, 31-036 Kraków

Streszczenie: W artykule zaprezentowano nowoczesne narzędzie w modelu SaaS do automatyzacji testów front-end stron internetowych oraz aplikacji sieciowych, BrowserSpot. Przedstawione zostały kluczowe elementy i funkcje narzędzia oraz przesłanki i problemy branży, które zainicjowały rozpoczęcie prac nad rozwiązaniem. Omówiono szczegółowo wszystkie osiem etapów prac badawczych i rozwojowych projektu, jak również rezultaty i kluczowe osiągnięcia każdego z nich. Narzędzie stworzone zostało w modelu SaaS w celu zmaksymalizowania jego przystępności dla każdego rodzaju użytkownika i jest dostępne na rynku jako produkt firmy ClickRay Sp. z o.o.

Słowa kluczowe: automatyzacja testowania, Software as a Service, model SaaS, narzędzie testowe, luka testowa, BrowserSpot

1. Wprowadzenie

Tworząc każdego rodzaju stronę internetową lub aplikację sieciową konieczne jest jej testowanie od strony front-end. Ma to na celu identyfikację potencjalnych błędów na różnych urządzeniach oraz rozdzielczościach, ale również zapewnienie najwyższej jakości i płynności użytkowania dla użytkownika końcowego. Do tej pory proces ten był w głównej mierze zadaniem manualnym pochłaniającym ogromną ilość czasu oraz zasobów. W ostatnich latach zaczęły pojawiać się narzędzia i rozwiązania, których zadaniem jest zautomatyzowanie tego procesu. Jednym z takich narzędzi jest platforma testowa BrowserSpot opracowana i wdrożona przez ClickRay Sp. z o.o., działająca w modelu SaaS oraz korzystająca z mocy obliczeniowej chmury.

Celem artykułu jest prezentacja wyników prac badawczo-rozwojowych przeprowadzonych pod kątem opracowania i przyszłego wdrożenia platformy testowej BrowserSpot oraz przedstawienie podstawowej literatury przedmiotu, poszczególnych etapów projektu, a także rezultatów każdego z nich. Projekt badawczo rozwojowy realizowany był w ramach dofinansowania z Funduszy Europejskich. Jest odpowiedzią na potrzebę rynkową dotyczącą rozwiązań automatyzujących testowanie front-endu stron internetowych oraz aplikacji sieciowych.

Autor korespondujący:
Jakub Góral, goral.jakub99@gmail.com

Artykuł recenzowany
nadesłany 11.08.2023 r., przyjęty do druku 15.01.2024 r.

Zezwala się na korzystanie z artykułu na warunkach licencji Creative Commons Uznanie autorstwa 3.0

2. Model SaaS

Modelem SaaS (ang. *Software as a Service*) można nazwać sposób dystrybucji oprogramowania, w ramach którego producent oprogramowania hostuje je w sieci za pomocą zewnętrznego dostawcy lub usługodawcy. Model ten jest najczęściej utożsamiany z subskrypcyjnymi modelami płatności lub płatnościami według zużycia pay-as-you-go [1]. Narzędzia i aplikacje korzystające z tego modelu programowane są specjalnie na potrzeby działania oraz pełnego dostępu dla użytkowników w środowisku internetowym [2].

Najnowsze rozwiązania SaaS mogą same w sobie działać na modelach PaaS (ang. *Platforma as a Service*), które z kolei mogą działać na modelach IaaS (ang. *Infrastructure as a Service*), tym samym jeszcze bardziej zakorzeniając się w wielopoziomową strukturę, charakterystyczną pod względem swojego usługowego podejścia do dystrybucji [3].

Aplikacje w modelu SaaS stają się coraz bardziej popularne między innymi ze względu na liczne korzyści zarówno dla użytkowników, jak i dla dostawcy usługi. Wśród korzyści dla użytkownika można wyróżnić między innymi [4]:
- niskie obciążenie kosztami ogólnymi,
- brak konieczności wdrożenia rozwiązania w siedzibie firmy,
- przyjazne modele finansowania np. pay-as-you-go lub subskrypcja,
- możliwość dostosowania funkcji do własnych potrzeb,
- brak konieczności korzystania z własnych zasobów komputerowych (w przypadku korzystania z chmury obliczeniowej).

Główne korzyści dla dostawców usługi [4]:
- skalowalność aplikacji,
- łatwość w utrzymaniu i konserwacji,
- wysoka efektywność,
- wysoka stabilność kodu źródłowego aplikacji,

– wysoki poziom możliwości dostosowywania funkcji, np. interfejsu użytkownika, nadawania uprawnień, konfiguracji modułów rozwiązania, rozszerzalna i niestandardowa integracja z różnymi zewnętrznymi systemami,
– aplikacja jako usługa,
– elastyczne koszty zależne od zużycia.

Elastyczność charakterystyczna dla aplikacji opracowanych w modelu SaaS jest atrakcyjna przede wszystkim dla mniejszych biznesów. Wynika to z braku konieczności zaangażowania dużej ilości zasobów, co z kolei nie przekłada się na stratę efektywności działania w porównaniu z innymi modelami dystrybucji.

3. Problemy w testowaniu

Pomysł na opracowanie narzędzia BrowserSpot powstał w wyniku identyfikacji luki rynkowej dotyczącej takiego produktu na polskim rynku oraz wieloletniej działalności firmy ClickRay w branży IT. Zidentyfikowano również potrzebę automatyzacji procesu testowania front-endu stron internetowych i aplikacji sieciowych [7], który do tej pory w większości przypadków był głównie manualny. Proces ten polegał na ręcznej weryfikacji poszczególnych elementów UI na różnych urządzeniach oraz w różnych rozdzielczościach. W niektórych przypadkach pisany był również dedykowany dla danego problemu kod, którego zadaniem było wyszukiwanie potencjalnych błędów testując różną kombinację urządzeń i rozdzielczości [8].

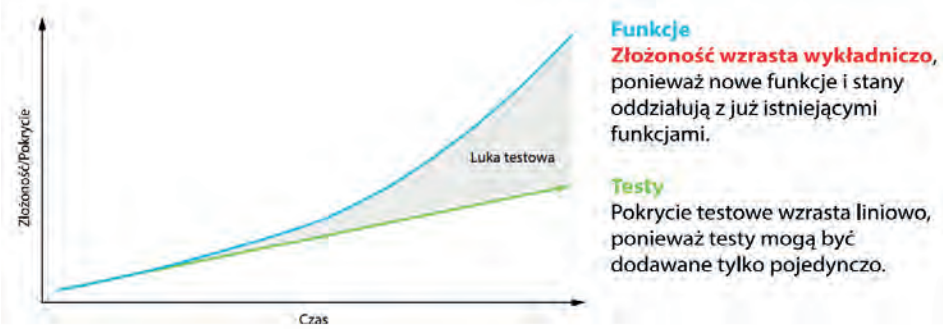
Problem ten najlepiej został udokumentowany i przedstawiony graficznie przez CEO firmy Appdiff Jason'a Arbon w publikacji z 2017 r. [9]. Zauważył on stale rosnący trend w ilości danych na nowych stronach internetowych, co znacznie zmniejszało efektywność testów manualnych, oraz stale rosnącą kompleksowość stron i aplikacji sieciowych, co z kolei zwiększało wymagane nakłady pracy testerów. Teoria ta została zobrazowana na wykresie (Rys. 1).

Zarówno teoria luki testowej, jak i doświadczenie firmy ClickRay w branży potwierdzają coraz mniejszą skuteczność, wydajność i efektywność manualnych testów front-end stron internetowych i aplikacji sieciowych. Z tego właśnie względu podjęto się prac nad narzędziem automatyzującym ten proces, tym samym rozwiązującym problem luki testowej.

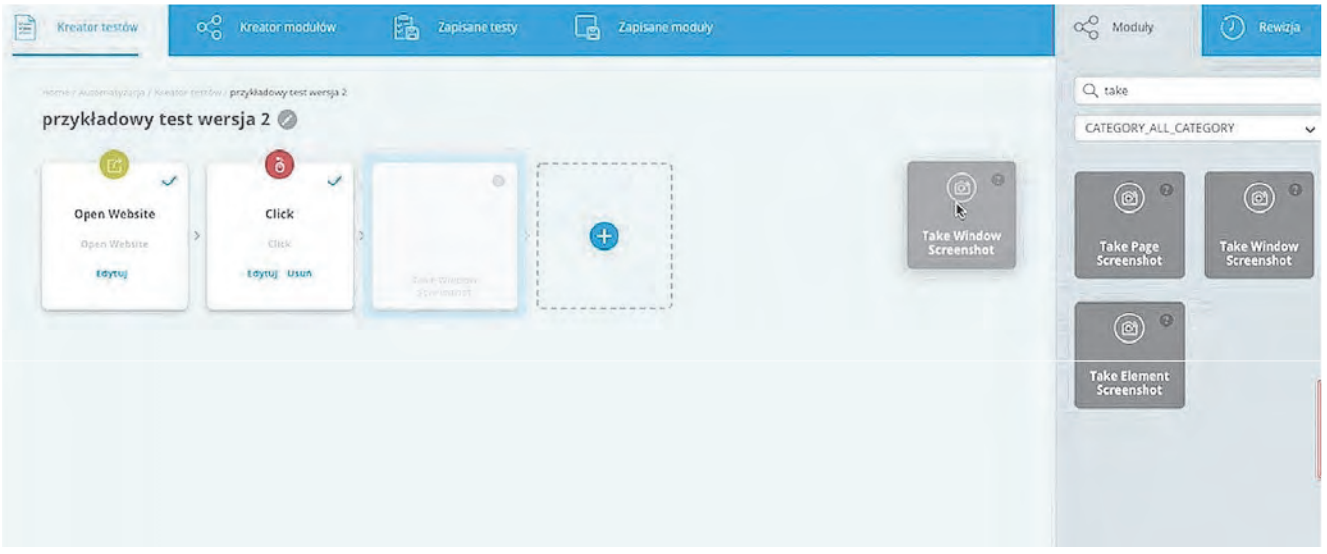
4. Charakterystyka narzędzia BrowserSpot

Narzędzie BrowserSpot opracowane zostało w modelu SaaS oraz korzysta z chmury obliczeniowej w celu zapewnienia jak największej wygody, oraz łatwości użycia użytkownikom. W literaturze przedmiotu pojawia się również pojęcie modelu TaaS (ang. *Testing as a Service*) [5], który działa na podobnej zasadzie co model SaaS, ale dotyczy konkretnie działań związanych z testowaniem. Nie jest ono jednak tak spopularyzowane, jak SaaS. Jedyną różnicą między tymi modelami dotyczy przeznaczenia ich produktów końcowych.

W celu maksymalizacji dostępności, narzędzie zostało wyposażone w wiele funkcjonalności ułatwiających jego działanie oraz komfort pracy użytkownika. Są to między innymi: możliwość testowania na różnych urządzeniach oraz w różnych rozdzielczościach, brak konieczności nadzorowania pracy narzędzia w trakcie testów, bezkodowe testowanie (ang. *Codeless testing*). Dodatkowo interfejs użytkownika jest przyjazny dla osób nieposiadających umiejętności kodowania dzięki wykorzystaniu technologii Drag & Drop (Rys. 2), która pozwala na dodawanie kolejnych kroków testów w formie kafelków. Takie rozwiązanie zapewnia



Rys. 1. Teoria luki testowej J. Arbone
Fig. 1. Testing gap theory by J. Arbone



Rys. 2. Funkcja bezkodowego testowania oraz Drag & Drop
Fig. 2. Codeless Testing and Drag & Drop Feature

łatwy i przejrzysty dostęp do rozwiązania również osobom nieposiadającym doświadczenia w testowaniu lub w pisaniu kodu.

5. Analiza rozwiązań rynkowych

W odpowiedzi na dynamikę środowiska biznesowego, na rynku istnieje, i stale pojawia się coraz więcej rozwiązań ułatwiających, oraz automatyzujących proces testowania stron internetowych oraz aplikacji webowych. Narzędzia te są nieodłącznym elementem wielu zespołów programistycznych, przez co kluczowa jest ich jakość oraz skuteczność. W celu dostrzeżenia oraz zrozumienia różnic między dostępnymi na rynku rozwiązaniami analizie poddane zostanie pięć najpopularniejszych narzędzi. Są to:
– Katalon,
– Selenium,
– Appium,
– TestComplete,
– Cypress.

Spośród wskazanych rozwiązań jedynie Katalon oraz TestComplete oferują możliwość testowania zarówno aplikacji webowych, jak i mobilnych oraz desktopowych. W przypadku obsługiwanych platform, większość rozwiązań dostępna jest na wszystkich platformach. Wyjątkiem jest jedynie TestComplete, który dostępny jest tylko na platformie Windows, oraz Appium, które nie jest dostępne dla użytkowników systemu Linux. Jeśli chodzi o funkcję korzystania z opcji low-code lub no-code, ponownie, jedynie Katalon oraz TestComplete zapewniają taką możliwość. Pozostałe trzy narzędzi wymagają umiejętności sprawnego kodowania, co w pewnym sensie ogranicza grupę docelową ich potencjalnych konsumentów. Warto dodać, że zarówno Katalon, jak i TestComplete są stosunkowo proste w konfiguracji i ustawianiu. W procesie tym niewymagane są umiejętności kodowania, co dodatkowo potwierdza fakt, iż narzędzia te są stworzone z myślą o prostocie użytkowania.

6. Etapy projektu badawczego

Prace nad narzędziem podzielone zostały na osiem oddzielnych etapów. Ten rozdział skupi się na zaprezentowaniu każdego z etapów oraz prac, jakie zostały przeprowadzone w jego ramach. Każdy z etapów składał się zarówno z poszczególnych zadań, jak i badań koniecznych do kontynuacji dalszych prac. Głównym zadaniem pierwszego etapu było opracowanie fundamentalnej struktury kodu dla serwera aplikacji. Następnie w etapie drugim i trzecim prowadzono badania odpowiednio nad środowiskiem wirtualnym oraz integracją chmury obliczeniowej z aplikacją. Celem etapów 4–6 było zwiększenie możliwości technicznych najważniejszych modułów rozwiązania. Były to odpowiednio moduł monitorowania w etapie czwartym, moduł poprawności w etapie piątym oraz moduł porównywania w etapie szóstym. Kolejnym krokiem było przeprowadzenie badań pod kątem konwersji, komunikacji i prezentacji danych w ramach etapu siódmego, oraz zintegrowanie wszystkich wcześniej opracowanych elementów wraz z testami w etapie ósmym.

6.1. Etap pierwszy

Pierwszym etapem projektu było stworzenie fundamentalnej struktury kodu dla serwera Selenium, a następnie połączenie go z planowanymi do wdrożenia przeglądarkami i urządzeniami. W ramach tego etapu przeprowadzono następujące działania:
— opracowanie algorytmu działania przeglądarek na komputerach stacjonarnych,
— opracowanie algorytmu komponentu do użytku przeglądarek mobilnych,
— przygotowanie emulatorów do użytku z urządzeniami i platformami,
— opracowanie algorytmu niezbędnego do zintegrowania serwera Selenium z urządzeniami mobilnymi.

W trakcie tego etapu podjęto również działania konieczne do weryfikacji następujących kwestii:
— prawidłowego sposobu działania technologii,
— zdolności i możliwości działania poszczególnych przeglądarek na serwerze Selenium.

6.2. Etap drugi

Celem drugiego etapu było zbadanie, czy możliwe jest skuteczne oddzielenie technologii środowiska wirtualnego (stworzonej w etapie 1) na maszyny fizyczne. Ważnym elementem tego etapu było określenie, czy płynna komunikacja i wymiana danych między serwerem a urządzeniami będzie możliwa. Badania te pozwoliły poznać techniczne parametry i ograniczenia serwera we współdziałaniu z urządzeniami fizycznymi. Stworzono lokalne środowisko testowe, w którym zostały przetestowane wcześniej wymienione możliwości techniczne serwera.

6.3. Etap trzeci

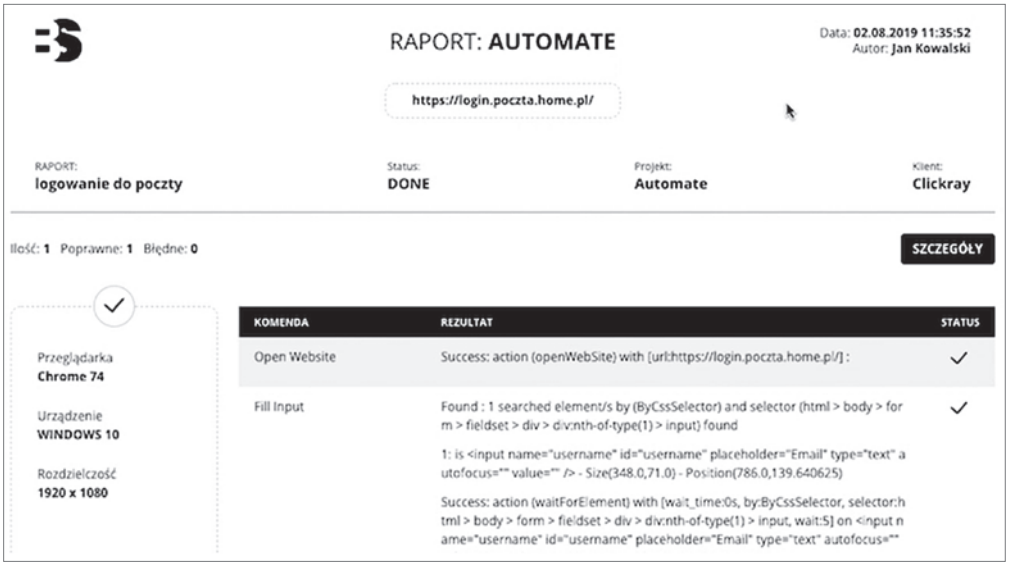
Podczas trzeciego etapu projektu przeprowadzono badania dotyczące możliwości wprowadzenia zadań obliczeniowych za pomocą chmury do aplikacji. W ramach tego etapu opracowano też interfejs użytkownika niezbędny do uruchomienia modelu SaaS. Przeprowadzone badania miały na celu dostarczyć jednoznaczną odpowiedź, czy wykorzystanie modelu w chmurze jest najbardziej optymalnym rozwiązaniem dla narzędzia BrowserSpot. Pod koniec etapu opracowano raport prezentujący wydajność rozwiązania działającego w oparciu o chmurę obliczeniową.

6.4. Etap czwarty

Celem czwartego etapu było zwiększenie możliwości technicznych modułu do badania poprawności witryny stanowiącego element składowy narzędzia. Główne prace koncentrowały się na stworzeniu komponentu dedykowanego monitorowaniu wydajności wyświetlania strony. Pozwala to zweryfikować poprawność działania linków na danej stronie, sprawdzić jakość wyrenderowanej strony, weryfikować i ocenić wykryte błędy w wyświetlaniu, a także przekształcić je w formę opisową, oraz ostatecznie wygenerować szczegółową listę informacji i danych dotyczących ładowania każdego elementu na testowanej stronie. Do głównych zadań wykonanych w trakcie tego etapu należą:
— stworzenie i weryfikacja baz danych niezbędnych do poprawnego działania opracowanego modułu,
— wprowadzenie możliwości zapisu danych z modułu do bazy danych,
— weryfikacja i wdrożenie technicznych możliwości przygotowania danych interfejsu front-end do eksportu,
— weryfikacja i wdrożenie projektu komponentu systemowego w obszarze front-end,
— implementacja dotychczasowych efektów projektu za pomocą algorytmu API do części front-end narzędzia,
— testowanie użyteczności i poprawności zaimplementowanego w tym etapie fragmentu projektu,
— testowanie użyteczności oraz poprawnego działania modułów pod względem opracowanej technologii,
— wdrożenie technicznej zdolności modułu do generowania testów indywidualnych.

6.5. Etap piąty

Głównym celem piątego etapu projektu było zwiększenie możliwości oraz zakresu technologicznego modułu do testowania poprawności językowej narzędzia BrowserSpot. Komponent



Rys. 3. Funkcja generowania raportów na temat znalezionych błędów
Fig. 3. Error Report Generation Feature

ten pozwala na przeprowadzanie weryfikacji witryn internetowych pod kątem poprawności językowej oraz proste generowanie raportu pokazującego wykryte na stronie błędy (Rys. 3).

W ramach omawianego etapu przeprowadzone zostały też badania dotyczące opracowania wcześniej wspomnianego komponentu. Moduł miał zostać połączony z kilkoma zewnętrznymi bazami danych, dzięki którym możliwe było badanie następujących założeń:
- poprawności gramatycznej tekstów umieszczonych na stronie,
- poprawności stylistycznej zdań,
- poprawności adaptacji treści strony.

Poza wskazanymi wyżej badaniami, w ramach etapu przeprowadzono następujące działania:
- opracowanie i sprawdzenie zapisów danych z algorytmu modułu oraz przesłanie go do bazy danych,
- opracowanie i przetestowanie możliwości eksportu danych do sekcji front-end narzędzia,
- wdrożenie wcześniej opracowanego algorytmu, którego celem jest przeprowadzenia weryfikacji witryny pod kątem poprawności językowej,
- dalsza praca i weryfikacja postępów projektach komponentu systemowego pod względem front-end narzędzia,
- wdrożenie wcześniej wspomnianego projektu za pomocą algorytmu API do sekcji front-end narzędzia,
- integracja i standaryzacja informacji zawartych w sekcji front-end na podstawie opracowanego interfejsu API,
- testowanie modułu pod kątem poprawności działania,
- testowanie zaimplementowanej sekcji narzędzia pod kątem poprawności działania,
- rozwinięcie oraz dalsza praca nad algorytmem bazy danych, niezbędnym do poprawnego działania modułu.

6.6. Etap szósty

Celem szóstego etapu był rozwój oraz zwiększenie zakresu technologicznego modułu aplikacji dotyczącego porównywania elementów na stronie, które zostały wyrenderowane w przeglądarkach. Opracowano komponent, którego głównym zadaniem jest wykrywanie błędów dotyczących poszczególnych elementów aktualnie testowanej witryny lub aplikacji sieciowej, oraz automatyczne generowanie raportów prezentujących opisowo różnice w wyświetlaniu poszczególnych elementów renderowanych na stronie na różnych urządzeniach i w różnych przeglądarkach.

Moduł ten miał za zadanie pozyskiwanie wcześniej pobranych z serwera danych oraz przetwarzanie ich na konkretne informacje dotyczące błędów związanych z poszczególnymi elementami stron internetowych lub aplikacji sieciowej. Prace nad modułem koncentrowały się wokół realizacji poniższych zadań:
- nadawanie pozwoleń na generowanie testów w odniesieniu do znajdowania błędów w elementach pojawiających się na stronie,
- prezentowanie wykrytych błędów w formie opisowej (stworzenie algorytmu przekształcającego informację o konkretnym błędzie na tekst) w odniesieniu do wykrytych różnic w położeniu, kolorze, wymiarach i proporcjach poszczególnych elementów,
- porównywanie poszczególnych elementów na stronie lub w aplikacji sieciowej,
- integracja do narzędzia przygotowanego interfejsu API i części front-end w dziedzinie testowania i wykrywania błędów oraz generowania testów i raportów dla użytkownika,
- przeprowadzenie weryfikacji poprawności technologii,
- rozwój algorytmu.

6.7. Etap siódmy

Głównym celem siódmego etapu było przeprowadzenie badań, które miały pomóc w procesie zdobywania wiedzy i umiejętności z dziedziny konwersji, komunikacji i prezentacji danych. Prace badawcze obejmowały w głównej mierze rozwinięcie algorytmu komunikacji i protokołu do konwersji i przesyłania informacji z renderowanej strony internetowej lub aplikacji sieciowej oraz zaprojektowanie sposobu ich prezentacji w części front-end interfejsu użytkownika. Przeprowadzone w ramach etapu prace ukierunkowane były na integrację dotychczas opracowanych modułów i funkcjonalności narzędzia, m.in. miały umożliwić automatyczne generowanie zbiorczego raportu scalającego dane i wnioski z wszystkich modułów, tj. modułu poprawności wyświetlania strony, modułu badania poprawności językowej strony oraz modułu poprawności działania poszczególnych elementów strony (modułu testowania wydajności i SEO).

Otrzymane w wyniku prac rozwiązania techniczne miały stanowić jeden z komponentów końcowej technologii niezbędnej do uruchomienia prototypu narzędzia BrowserSpot, tym samym umożliwiając rozpoczęcie testów świadczenia usługi.

Prace projektowe w trakcie tego etapu obejmowały również rozwinięcie algorytmu odpowiedzialnego za pobieranie danych z serwera oraz eksportowanie ich do części front-end interfejsu użytkownika. Wspomniany wcześniej stworzony algorytm modułu ma wbudowany interfejs API, za pomocą którego dane z serwera były konwertowane na wybrany wcześniej przez użytkownika język.

6.8. Etap ósmy

Celem ósmego etapu projektu były prace rozwojowe dotyczące zintegrowania wszystkich wcześniej opracowanych i stworzonych elementów technologii oraz testy gotowego produktu w postaci narzędzia BrowserSpot w warunkach rzeczywistych. Zadania w ramach tego etapu dotyczyły w głównej mierze ukończenia prototypu narzędzia dla usługi BrowserSpot.

W ramach etapu przetestowano opracowany wcześniej prototyp narzędzia w warunkach rzeczywistych przez demonstracje technologiczne wśród zamkniętej grupy wybranych użytkowników. Działania poprzedzające demonstracje obejmowały:

- integrację stworzonych wcześniej komponentów narzędzia,
- poprawę błędów w interfejsie API,
- wprowadzenie końcowych zmian we front-end narzędzia,
- weryfikację poziomów obciążenia narzędzia,
- weryfikację działania narzędzia pod kątem generowania tekstu na temat wykrytych błędów,
- korektę zauważonych błędów w procesie generowania informacji z przeprowadzonych weryfikacji oraz w działaniu przeglądarek.

Celem demonstracji prototypu było również zweryfikowanie poprawności działania całokształtu narzędzia oraz jego poszczególnych komponentów, sprawdzenie wydajności i potencjalnych problemów związanych z liczbą obsługiwanych jednocześnie użytkowników, oraz weryfikacja i usunięcie tak zwanych „wąskich gardeł" w procesie działania narzędzia. Wszystkie założenia zostały zweryfikowane, przetestowane i potwierdzone w warunkach rzeczywistych, co oznacza, że faza badań i rozwoju projektu zakończyła się sukcesem.

7. Rezultaty poszczególnych etapów

W ramach niniejszego rozdziału przedstawione zostaną rezultaty prac badawczych i prac rozwojowych dla każdego z ośmiu etapów projektu. Ma to między innymi na celu zapewnienie lepszego i dokładniejszego wglądu w wyniki prac badawczych oraz przedstawienie osiągnięć projektu.

7.1. Rezultaty pierwszego etapu

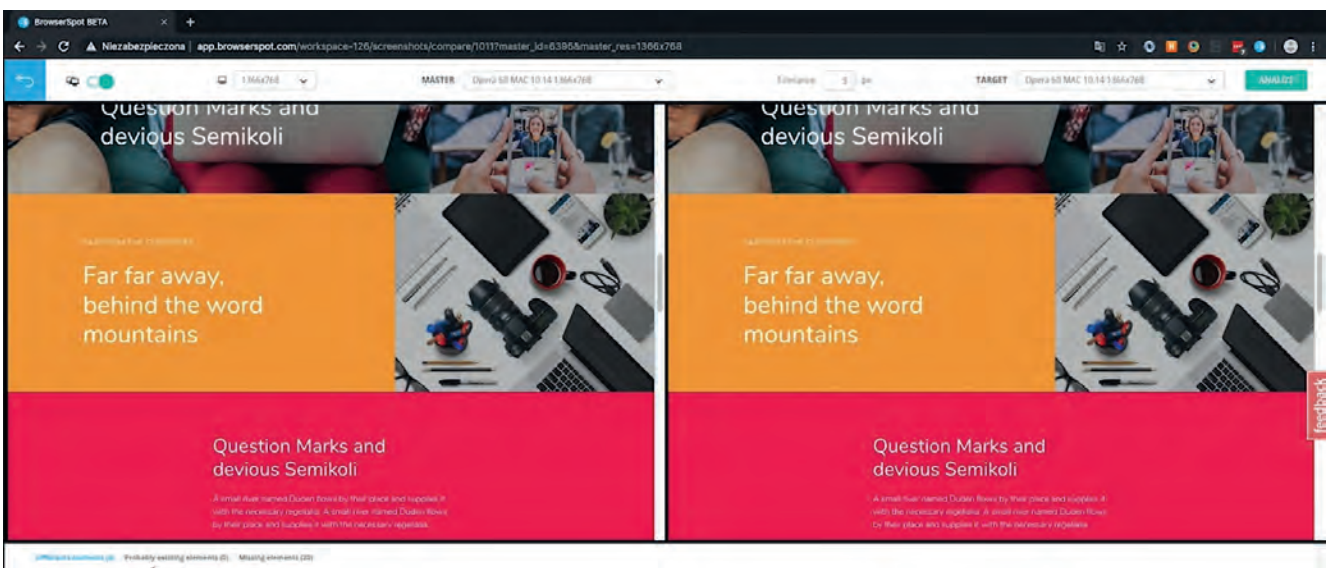
W pierwszym etapie projektu została stworzona maszyna wirtualna będąca środowiskiem uruchomieniowym i rdzeniem całego projektu. Wirtualna maszyna to izolowane środowisko działające na komputerze gospodarza, symulujące działanie fizycznego urządzenia z oddzielnym systemem operacyjnym, dowolnie wybranym niezależnie od systemu komputera gospodarza [6]. Głównym komponentem działającym w tak przygotowanym środowisku był silnik wykorzystujący technologię Selenium.

Selenium to framework automatyzujący zadania związane z testowaniem funkcjonalnym aplikacji internetowych. Pozwala on na kontrolowanie przeglądarek internetowych na poziomie kodu i definiowanie akcji, które mają wykonać. W projekcie użyto jednego z narzędzi Selenium – WebDriver. Narzędzie oraz silnik zintegrowane zostały za pomocą języka Java. Zadania przeprowadzone w ramach tego etapu posłużyły jako fundament dla narzędzia BrowserSpot, który mógł być dalej rozwijany i wzbogacany o nowe moduły i komponenty.

7.2. Rezultaty drugiego etapu

W trakcie kolejnego etapu prac, system mógł już wykorzystywać wirtualne maszyny obsługiwane przez standardowe systemy operacyjne dla komputerów stacjonarnych oraz zestaw emulowanych urządzeń mobilnych. Działanie emulatorów oferowało możliwość odtworzenia zachowania się danych urządzeń do pewnego stopnia i tym samym umożliwiało testowanie działania witryn w ich środowiskach, ale nie odpowiadało 1:1 rzeczywistym przypadkom użycia z powodów wielu wad wynikających z używania emulacji. W wyniku takiej sytuacji na tym etapie wiele emulatorów, w których występowało najwięcej wad zastąpiono urządzeniami fizycznymi. Przedsięwzięcie to wymagało zaprojektowania autorskich algorytmów do integracji urządzeń fizycznych z narzędziem. Algorytm musiał zapewnić niezawodną dwukierunkową komunikację z urządzeniem, aby jakość użytkowania pozostała na jak najwyższym poziomie. Zaimplementowane rozwiązanie umożliwiało obsługę urządzeń fizycznych z komputera stacjonarnego i wyświetlanie zawartości ekranu urządzenia mobilnego na ekranie komputera oraz zdolność do wykonywania operacji w czasie rzeczywistym, przez co opóźnienie zostało zminimalizowane. Podobne rozwiązanie zastosowano później w przypadku odpowiadających systemów komputerowych. Ponadto zaprojektowano i zaimplementowano nowe funkcje, takie jak pobieranie zrzutów ekranu stron internetowych renderowanych przez różne przeglądarki, pobieranie kodu testowanych stron oraz porównywanie sposobu wyświetlania stron w różnych środowiskach (Rys. 4).

Funkcja porównywania stron oferuje możliwość wizualnego podglądu strony w danym środowisku. W zależność od urządzenia, wersji oprogramowania i przeglądarki zmiany te mogą



Rys. 4. Funkcja jednoczesnego porównywania stron internetowych
Fig. 4. Simultaneous Website Comparison Feature

być mniejsze lub większe. W większości przypadków od strony wizualnej widać jedynie drobne i nieznaczące różnice.

Implementacja wspomnianych funkcjonalności i rozwinięcie istniejących pozwoliło na utworzenie pierwszej bazy danych w projekcie. Ze względu na rozmieszczenie funkcji systemu oraz połączenie nowych urządzeń z systemem, zidentyfikowana została potrzeba zaimplementowania metod równoważenia obciążenia. W tym celu wykorzystano rozwiązanie oparte na wymienionych metodach, którego zadaniem była odpowiednia dystrybucja zadań między urządzeniami w infrastrukturze.

7.3. Rezultaty trzeciego etapu

W trakcie trzeciego etapu projektu opracowano moduł porównawczy wydajności między aplikacją działającą w tradycyjnym modelu (kiedy użytkownik instaluje oprogramowanie na swoim komputerze) a oprogramowaniem dostarczanym w modelu usługi (model SaaS). Analiza wydajności została przeprowadzona z perspektywy użyteczności użytkownika i została przedstawiona w tabeli (Tab. 1).

Tabela 1 Porównanie modelu tradycyjnego i modelu SaaS z perspektywy użytkownika
Table 1 Traditional and SaaS model comparison from user's perspective

Model tradycyjny	Model SaaS
– Blokuje użytkownikowi możliwość przeprowadzania innych aktywności podczas przeprowadzania testów (testy nie działają w tle), – Użytkownik musi manualnie utrzymywać cały system w celu przeprowadzenia testów (różne wersje przeglądarek itp.), – Wydajność oprogramowania (szybkość przeprowadzania testów i wykonywania zadań) jest ograniczona przez moc obliczeniową komputera użytkownika.	– Model można skalować przez „wzbogacanie" chmury o nowe urządzenia; pojedyncze zadanie jest dystrybuowane i wykonywane jednocześnie na wielu komputerach, – Komputer użytkownika nie jest obciążony, ponieważ aplikacja natywnie uruchamia przeglądarki, a testy są wykonywane w tle, – Użytkownik otrzymuje cały pakiet, w tym oprogramowanie i infrastrukturę (urządzenia w chmurze z różnymi wersjami przeglądarek, różnymi rozdzielczościami, bez ponoszenia kosztów utrzymania urządzeń).

Obliczenia wydajności przeprowadzono dla wielu wariantów testowych. Szacowana wydajność oprogramowania przedstawiona jako szybkość testowania i wykonywania zadań, dla przykładowego przypadku testowego:

– test strony składającej się z 57 podstron na trzech przeglądarkach.

Założenia użyte do obliczeń:

– średni czas testowania jednej podstrony za pomocą jednego komputera – 35 s (średnia wartość, dokładny czas zależy od parametrów urządzenia, takich jak wielkość pamięci RAM, prędkość procesora itp.).

Wyniki odnotowane na opracowanym modelu SaaS wykazały drastyczną poprawę wydajności. W zależności od złożoności strony internetowej i składników tradycyjnego modelu komputera, zauważono wzrost wydajności nawet do 70 % przy użyciu prezentowanego rozwiązania w modelu SaaS względem tradycyjnego modelu.

Na podstawie otrzymanych wyników podjęto decyzję o przeniesieniu usługi do chmury i udostępnieniu jej w modelu SaaS. Kluczowe zalety przeniesienia usługi do modelu SaaS obejmują: brak konieczności instalowania oprogramowania przez użytkownika, co skutkuje niższym progiem wejścia dla potencjalnych klientów, brak wymagań dotyczących konkretnego sprzętu, łatwiejsze rozpowszechnianie rozwiązania, łatwiejsze utrzymanie usługi w dłuższej perspektywie, zwiększona skalowalność systemu. Aby umożliwić działanie usługi w tym modelu, konieczne było stworzenie interfejsu użytkownika. Ta część pracy obejmowała m.in. przedsięwzięcia takie jak ustalanie wizerunku aplikacji, projektowanie i wdrażanie układów strony, projektowanie i wdrażanie mechanizmów sterowania zdalnymi urządzeniami, integrację interfejsu użytkownika z interfejsem API serwera, testowanie interfejsu w celu znalezienia błędów i wad, oraz rozpoczęcie prac nad stałym utrzymaniem usługi.

7.4. Rezultaty etapów 4–6

Etapy 4–6 obejmowały między innymi wdrożenie i przetestowanie modułu testowania wydajności, modułu testowania językowego i modułu porównywania wyników renderowania. Automatyczny moduł testowania wydajności strony pozwala użytkownikowi na zweryfikowanie witryny pod kątem wydajności oraz poprawnego zastosowania praktyk SEO, a także sugeruje możliwe ulepszenia w oparciu o znalezione błędy. Testy są przeprowadzane w pełni automatycznie, użytkownik wprowadza jedynie adres strony internetowej i przypisuje danemu testowi informacje takie jak: klient, projekt oraz założone kamienie milowe. Po udanym przeprowadzeniu testu wydajności generowany jest automatycznie raport zawierający szereg danych, w tym informacje o ilościowym udziale *response codes* serwera http (200, 301, 404), informacje o ilości pobranych danych z serwera wraz z dokładnymi adresami, z których te dane zostały pobrane, informacje o ilości pobranych danych z serwera uwzględniające formaty danych (html, css, js itp.), podstawowe informacje podsumowujące o czasach odpowiedzi i ładowania strony, sugestie dotyczące ulepszeń strony (YSLOW), uwzględniające optymalizację kodu JavaScript, kompresję plików i liczbę żądań HTTP.

Drugą wprowadzoną funkcjonalnością w ramach wcześniej wspomnianych etapów był moduł testowania poprawności językowej witryny. Moduł ten działa w oparciu o narzędzie LanguageTool. Algorytm sprawdza cały tekst widoczny dla użytkownika pod kątem: gramatyki, pisowni, frazeologii, interpunkcji i składni. Po pozytywnie wykonanym teście generowany jest raport z sugestiami ulepszeń w oparciu o wykryte błędy. Kolejną opracowaną w ramach tych etapów funkcjonalnością był moduł porównywania renderowania różnych przeglądarek. Narzędzie to opiera się na wcześniej opracowanych funkcjonalnościach: module pobierania zrzutów ekranu, kodzie HTML i narzędziu do szukania różnic. Po uruchomieniu moduł zwraca wynik zawierający podgląd strony renderowanej przez każdą z wybranych przeglądarek oraz urządzeń. Każdy z podglądów można swobodnie obejrzeć i porównać ręcznie. Głównym elementem narzędzia jest jednak automatyczne porównywanie stron renderowanych. Narzędzie umożliwia: wybranie dwóch renderów i zestawienie ich ze sobą, synchroniczne przewijanie stron renderowanych w celu wizualnego porównania, automatyczne znajdowanie i listowanie różnic, ustawienie tolerancji dla różnic w wymiarach elementów, przeglądanie i lokalizowanie różnic, szukanie elementów brakujących, szukanie elementów, które pojawiły się w kodzie, ale nie pojawiają się w podglądzie użytkownika itp.

7.5. Rezultaty siódmego etapu

Podczas siódmego etapu projektu opracowany został komponent odpowiedzialny za generowanie gotowych oraz pełnych raportów dla końcowego użytkownika. Zaimplementowane zostały funkcje związane z przechowywaniem i udostępnianiem historycznych i bieżących raportów. Wszystkie raporty prze-

chowywane są automatycznie w zabezpieczonej bazie danych w celu zapewnienia jak największego bezpieczeństwa dla użytkowników. Repozytorium zawiera raporty z trzech rodzajów testów: testów renderowania, testów wydajności i zgodności z SEO oraz testów poprawności językowej. Każdy z raportów jest dostępny w systemie BrowserSpot pod zakładką „Raporty", a także istnieje możliwość ich pobrania w formie sformatowanego dokumentu PDF do prezentacji poza usługą lub w formie papierowej.

7.6. Rezultaty ósmego etapu

W ramach ósmego etapu projektu połączone zostały komponenty technologiczne opracowane we wcześniejszych etapach. Ponadto przeprowadzona została seria testów i stworzono prototyp narzędzia BrowserSpot, który został udostępniony zamkniętej grupie użytkowników w celu przeprowadzenia testów UX. W rezultacie prac przeprowadzonych w ramach projektu, stworzony został kompletny produkt IT działający jako narzędzie do testowania stron internetowych oraz aplikacji sieciowych pod kątem front-end, w modelu SaaS. Wszystkie opracowane i wcześniej wspomniane funkcjonalności narzędzia przeszły cykl testów i poprawek, aby wyeliminować możliwe błędy zarówno na etapie projektowania aplikacji, jak i w trakcie jej implementacji. Wyniki badania User Experience narzędzia na zamkniętej grupie użytkowników zostały zebrane za pomocą dedykowanego kwestionariusza. Źródłem danych były testy zadań z ramowego scenariusza – każdy uczestnik otrzymał zadanie zastosowania BrowserSpot w warunkach rzeczywistych.

Wymagania dla zadania zostały sformułowane następująco: zadanie dotyczy testowania dowolnej usługi, obiektem zadania jest aplikacja internetowa o średnim poziomie złożoności operacyjnej, zadanie obejmuje pojedynczą aplikację lub jej komponent. Grupa respondentów została wybrana spośród programistów, pracowników QA (testerów oprogramowania) oraz menedżerów projektów IT. Badania przeprowadzono asynchronicznie, bez moderatora. Analizie zostały poddane trzy obszary: jakość dostarczanych rozwiązań, nadmiarowość lub braki funkcjonalności, oraz błędy i ogólna wydajność usługi. Uczestnikom badania dostarczono instrukcję oraz podstawowe materiały pomocnicze dla użytkownika, nie udzielając wskazówek ani porad podczas testu. Badanie było anonimowe, jedynymi danymi uczestników dostarczanymi w trakcie badania była ich pozycja i staż pracy. Uzyskane wyniki zostały wykorzystane do udoskonalenia usługi BrowserSpot przed jej pełnym wprowadzeniem na rynek.

8. Podsumowanie

W wyniku przeprowadzonego projektu badawczo-rozwojowego opracowano kompleksowe narzędzie do testowania, umożliwiające znaczną automatyzację i standaryzację działań testowych, zwłaszcza w małych i średnich projektach dostępne na rynku pod nazwą handlową BrowserSpot. Narzędzie działa w modelu SaaS, dzięki czemu zapewnia łatwość użycia oraz nie wymaga wykorzystywania zasobów własnego komputera do przeprowadzania kompleksowych operacji. Wyniki poszczególnych etapów projektu potwierdziły założenia zdefiniowane na początku projektu B+R. Wyniki badań zostały wprowadzone na rynek w postaci gotowego do użytku narzędzia. Oferowane jest przez firmę ClickRay Sp. z o.o. pod komercyjną nazwą BrowserSpot w modelu SaaS. Narzędzie jest bezpośrednią odpowiedzią na kluczowe wyzwania stojące aktualnie przed branżą testowania, głównie związane z infrastrukturą techniczną, wysokimi kosztami utrzymania laboratoriów urządzeń i zasobów ludzkich oraz zjawiskiem luki testowej potwierdzającej efektywnie coraz wyższe trudności wynikające z testowania manualnego. Opracowane narzędzie pomaga rozwiązywać problemy z zatrudnieniem pracowników i jest przystępnym rozwiązaniem dla osób, które nie posiadają rozległych umiejętności programistycznych. Po zakończeniu projektu B+R narzędzie było dalej rozwijane, w celu lepszej adaptacji do potrzeb i oczekiwań użytkowników, czemu służyło m.in. wzbogacenie platformy o technologię Drag & Drop czy udostępnienie e-booka przedstawiającego w przystępnym sposób zagadnienia z obszaru automatyzacji testowania. BrowserSpot zawiera ułatwienia i wygodne rozwiązania takie jak między innymi funkcję tworzenia testów automatycznych bez konieczności pisania kodu, za pomocą interfejsu graficznego i intuicyjne funkcje Drag & Drop w interfejsie użytkownika na etapie tworzenia scenariusza testowego.

Podziękowania

Projekt badawczo-rozwojowy był współfinansowany przy zaangażowaniu dotacji z Unii Europejskiej. Tytuł projektu: „Rozwój unikalnego mechanizmu renderowania w celu uruchomienia usługi prototypowej przeglądarki BrowserSpot". Numer projektu badawczego: RPMP.01.02.01-12-0487/16, współfinansowany z funduszy Regionalnego Programu Operacyjnego dla Województwa Małopolskiego na lata 2014–2020, Oś Priorytetowa Gospodarka Wiedzy, Działanie 1.2 Badania i innowacje w przedsiębiorstwach, Poddziałanie 1.2.1 Projekty B+R przedsiębiorstw współfinansowany przez Europejski Fundusz Rozwoju Regionalnego.

Bibliografia

1. Satyanarayana S., *CLOUD COMPUTING : SAAS*. "Computer Science and Telecommunications", Vol. 36, No. 4, 2012, https://fenix.tecnico.ulisboa.pt/downloadFile/1126518382178096/1986.pdf.
2. Ju J., Wang Y., Fu J., Wu J., Lin Z., *Research on Key Technology in SaaS*. International Conference on Intelligent Computing and Cognitive Informatics. 2010, DOI: 10.1109/icicci.2010.120.
3. Tsai W., Bai X., Huang Y., *Software-as-a-service* (*SaaS*)*: perspectives and challenges*. "Science China Information Sciences", Vol. 57, No. 5, 2014, 1–15, DOI: 10.1007/s11432-013-5050-z.
4. Kumar K., *Software As A Service For Efficient Cloud Computing*. "International Journal of Research in Engineering and Technology", Vol. 3, No. 1, 2014, 178–181, DOI: 10.15623/ijret.2014.0301028.
5. Ali A., Maghawry H.A., Badr N., *Automated Parallel GUI testing as a service for mobile applications*, "Journal of Software: Evolution and Process", Vol. 30, No. 10, 2018, DOI: 10.1002/smr.1963.
6. Barham P., Dragovic B., Fraser K., Hand S., Harris T., Ho A., Neugebauer R., Pratt I., Warfield A., *Xen and the art of virtualization*. "ACM SIGOPS Operating Systems Review", Vol. 37, No. 5, 2003, 164–177, DOI: 10.1145/1165389.945462.

Inne źródła

7. Capgemini, Micro Focus, Sogeti, *World Quality Report*, 2018-2019 Tenth Edition, https://www.capgemini.com/news/press-releases/world-quality-report.
8. BrowserStack, Unadkat J., Community Contributor, *Manual Testing Tutorial for Beginners*, 2023, https://www.browserstack.com/guide/manual-testing-tutorial.
9. Arbon J., *AI for Software Testing* [In:] Pacific NW Software Quality Conference, PNSQC, 2017 http://uploads.pnsqc.org/2017/papers/AI-and-Machine-Learning-for-Testers-Jason-Arbon.pdf.

An Automatic Testing Platform for Front-end of Websites and Web Applications in a SaaS Model – BrowserSpot

Abstract: The article presents a modern tool created in a SaaS model for automating tests of front-end for websites and network applications, BrowserSpot. The key elements and functions of the tool were presented, as well as the industry premises and issues that initiated the start of work on the solution. All eight stages of research and development project were discussed in detail, along with the results and key achievements of each of them. The tool was created in a SaaS model to maximize its accessibility for every type of user, and is available on the market as a product by ClickRay Sp. z o.o.

Keywords: testing automation, Software as a Service, SaaS model, testing tool, testing gap, BrowserSpot tool

mgr Szymon Binek
s.binek@clickray.eu
ORCID: 0009-0001-7936-8056

Jest głównym pomysłodawcą i współzałożycielem ClickRay. Jest specjalistą w zakresie usług rozwoju on-line oraz API HubSpot, a także w szerokim zakresie cyfrowych przedsięwzięć i eksperymentów dla różnego rodzaju klientów. Nadzorował projekt badawczo-rozwojowy dofinansowany ze środków europejskich, który zaowocował platformą BrowserSpot. Zainteresowania badawcze: sztuczna inteligencja i machine learning, automatyzacja testów oprogramowania.

lic. Jakub Góral
goral.jakub99@gmail.com
ORCID: 0009-0003-9634-4915

W 2022 r. uzyskał tytuł licencjata na Uniwersytecie Ekonomicznym w Krakowie w dziedzinie Zarządzania ze specjalizacją w Zarządzaniu Small-Businessem. Następnie kontynuował swoje studia i obecnie znajduje się na ostatnim roku studiów magisterskich na kierunku Zarządzania Międzynarodowego. Zainteresowania badawcze obejmują technologię, Przemysł 4.0, sztuczną inteligencję oraz nauki o danych.