JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 8 (2019) 82-87
Wysłane: 2019-10-10
Przyjęte: 2019-10-29

Analiza wydajnościowa wybranych narzędzi do budowy aplikacji Single Page Application

Yehor Timanovskyi*, Małgorzata Plechawska-Wójcik
Politechnika Lubelska, Katedra Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. W artykule przedstawiono analizę wydajnościową z wykorzystaniem wybranych narzędzi do budowy Single Page Application. Do oceny wydajności aplikacji testowej używana została przeglądarka Google Chrome z narzędziem DevTools. Całkowita liczba wszystkich testów wynosiła 112. W ramach przeprowadzonego badania powstała aplikacja testowa z wykorzystaniem różnych szkieletów JavaScript - szkieletu Angular i szkieletu Vue.js.

Słowa kluczowe: SPA; szkielet JavaScript; Angular; Vue.js; wydajność

*Autor do korespondencji.

Adres e-mail: yehor.timanovskyi@pollub.edu.pl

Performance analysis of selected tools for building a Single Page Application

Yehor Timanovskyi*, Małgorzata Plechawska-Wójcik
Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The article presents analysis of the performance of selected tools to build a Single Page Application. Chrome browser with DevTools tool was used to evaluate the performance of the test application. The total number of all tests was 112. As part of the study, a test application was created using different JavaScript frameworks - the Angular framework and the Vue.js framework.

Keywords: SPA; frameworks JavaScript; Angular; Vue.js; performance

*Corresponding author.

E-mail address: yehor.timanovskyi@pollub.edu.pl

1. Wstęp

Aplikacje internetowe są wykorzystywane jako skuteczne rozwiązanie dla szerokiej gamy zadań biznesowych. W ostatnich latach aplikacje internetowe rozwijają się niezwykle szybko, stopniowo wypierając stacjonarne rozwiązania i stając się najważniejszym elementem biznesu w dzisiejszym świecie.

Jednostronicowe aplikacje (ang. *Single Page Applications*, zwane dalej SPA) są to aplikacje internetowe, których składniki są pobierane raz na jednej stronie i aktualizowane w razie potrzeby. W miarę wzrostu popularności jednostronicowych aplikacji, duża część przetwarzania danych jest skoncentrowana po stronie klienta.

Szkielety (ang. *frameworks*) są tworzone przede wszystkim, aby języki programowania (na przykład, *JavaScript*) ewoluowały. Szkielety to specjalne narzędzia, definiowane jako duże zestawy bibliotek wielokrotnego użytku do implementacji podstawowej struktury aplikacji. Szkielety JavaScript są używane do rozszerzenia możliwości programisty i upraszczania programowania poprzez dostarczanie zestawu gotowych komponentów dla bardziej skomplikowanych funkcjonalności. Korzystanie ze szkieletów JavaScript za pomocą natywnego JavaScript sprawia, że proces tworzenia interfejsu jest w dłuższej perspektywie szybszy.

Głównym problemem stojącym przed programistami jest odpowiedni dobór szkieletu lub języka do wykonywania określonej pracy. Dlatego w artykułe poruszono zagadnienie analizy wydajnościowej szkieletów JavaScript. Wyniki badania pozwolą ustalić, czy jest szkielet Vue.js jest bardziej wydajny od szkieletu Angular.

2. Cel i teza

Celem niniejszego artykułu jest bezpośrednie porównanie wydajności szkieletów JavaScript Vue.js oraz Angular.

Za tezę badawczą przyjęto:

Szkielet Vue.js jest bardziej wydajny od szkieletu Angular.

3. Przegląd literatury

Artykuł [1] poświęcony jest wydajnościowej analizie trzech współczesnych szkieletów JavaScript używanych w kontekście JSS (ang. *Sitecore JavaScript Services*). Szczegółowo opisana została architektura wzorców Model-Widok-Kontroler (ang. *Model-View-Controller*), MVP (ang. *ModelView-Presenter*) i MVVM (ang. *Model-View-ViewModel*).

W artykule [2] została opisana historia JavaScriptu oraz technologia ReactJS / Native wraz z ich zaletami i wadami.

Autorzy opisują również jak technologie React integrują się z innymi rozwiązaniami opartymi o język JavaScript.

Celem badań przedstawionych w artykule [3] jest zidentyfikowanie i zrozumienie czynników, które wpływają na wybór frameworka JavaScript spośród innych dostępnych na rynku. Badania realizowane są, aby odpowiedzieć na następujące pytanie badawcze: «Jakie czynniki prowadzą do podjęcia decyzji o wyborze konkretnego szkieletu JavaScript?».

Każdy z artykułów publikowanych [4], [5], [6] przedstawia poszczególne badania, dotyczące wspólnego tematu - porównania szkieletów JavaScript. W badaniach tych zostały podane szczegółowe informacje, których wyniki pozwoliły na ilościową ocenę pomiarów wskaźników porównawczych poprzez przeprowadzenie różnych eksperymentów. Autorzy opisują wpływ środowiska, w którym została uruchomiona aplikacja testowa, która korzysta ze szkieletu JavaScript.

Autor artykułu [7] przedstawia dokładny opis wykorzystania nowoczesnej architektury aplikacji internetowych, różnice między aplikacjami wielostronicowymi, a aplikacjami jednostronicowymi oraz wyjaśnia zasadę działania SPA.

Przegląd literatury pozwolił ocenić szkielety z różnych stron, zwrócić uwagę na popularne kryteria oceny szkieletów. Popularne kryteria to wydajność i rozmiar aplikacji.

4. Opis narzędzi badawczych

Obecnie JavaScript jest najbardziej popularnym językiem programowania aplikacji klienckich, dlatego otrzymał szerokie uznanie całej społeczności programistów. Korzystanie ze szkieletów JavaScript staje się coraz bardziej popularne, co uzasadnia fakt, że tworzenie szkieletów zmniejsza obciążenie procesu tworzenia aplikacji internetowych.

Analizując dane z Internetu, można zauważyć, że istnieją setki platform do tworzenia aplikacji internetowych. Każdy szkielet ma wiele atrakcyjnych funkcji i dodatków, dlatego wybór odpowiedniego jest dość trudny, a niewłaściwa decyzja może być główną przyczyną porażki projektu. Często pomijanym aspektem jest wydajność środowiska JavaScript, co jest istotne szczególnie w przypadku przedsiębiorstw tworzących złożone aplikacje.

4.1 Wady i zalety korzystania ze szkieletów JavaScript

Podczas wyboru szkieletów mogą wystąpić pewne trudności związane ze zdefiniowaniem zadań, które może wykonywać, dlatego należy ostrożnie podejść do kwestii wyboru i rozważyć wszystkie wady i zalety poszczególnych rozwiązań.

Poniżej znajdują się zalety korzystania ze szkieletów JavaScript [8]:

1) Elastyczność i skalowanie — elastyczne rozwiązanie niestandardowych zadań i możliwość dalszego rozszerzenia funkcjonalności poprzez połączenie bibliotek stron trzecich lub oddzielnych klas.
2) Zastosowanie architektury Model-Widok-Kontroler (ang. Model-View-Controller, zwany dalej MVC), znacznie poprawia funkcjonalność i elastyczność projektu. Jest to przydatne podczas projektowania małych aplikacji, ale niezbędne do prawidłowej funkcjonalności większych projektów.
3) Bezpieczeństwo. Szkielety mają wbudowane zabezpieczenia przed podstawowymi atakami (przykładowo, SQL injection).
4) Szkielet pozwala skoncentrować się na rozwiązywaniu problemów architektonicznych, a nie podstawowych, jak w projektowaniu bez jego użycia.
5) Jakość materiałów źródłowych. Na etapie tworzenia projektu szkielet pomaga pozbyć się typowych błędów, ponieważ istnieją pewne podstawowe zasady korzystania ze szkieletów.

Główną zaletą szkieletów, jest to, że są one idealne do tworzenia skalowalnych, unikalnych projektowo stron internetowych. Żaden duży projekt nie został rozwinięty przy użyciu gotowych CMS.

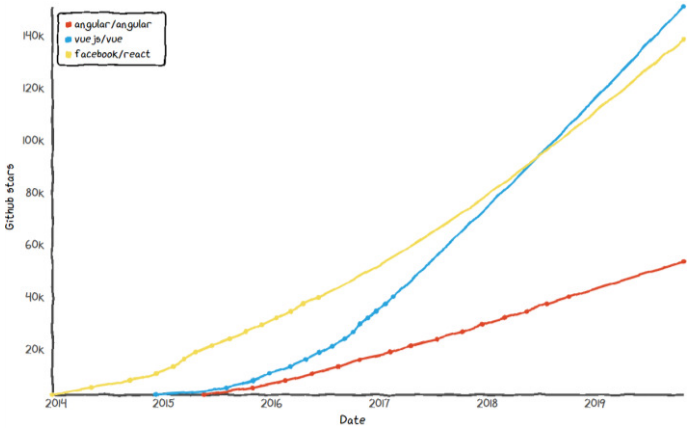
Wady korzystania ze szkieletów są dosyć umowne i niewielkie w porównaniu z korzyściami [8]:

1) Trudności w obsłudze.
2) Cena opracowania — koszt standardowego serwisu wykonanego na szkielecie od podstaw będzie wyższy niż na gotowych CMS, ponieważ zajmuje to kilka razy więcej czasu na opracowanie.
3) Trudności w procesie uczenia się.
4) Brak gotowych modułów i komponentów, które mógłby zainstalować klient. Wszystkie modyfikacje należy zgłaszać do deweloperów.

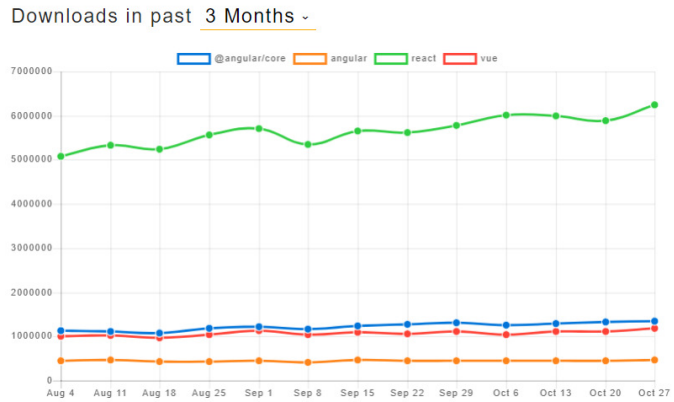
Opierając się na takich źródłach internetowych jak HotFrameworks i GitHub, statystyki dotyczące pobierania NPM [9], Stack Overflow, Google Trends [10], Developer Framework Satisfaction, można określić najbardziej popularne szkielety.

Dla porównania popularności szkieletów, jedną z pierwszych stron internetowych, które muszą być wymienione jest GitHub. Z punktu widzenia statystyki, system gwiazdkowy GitHub jest najbardziej przydatnym narzędziem do efektywnego wyszukiwania potrzebnego szkieletu (rys. 1).

Innym źródłem oceny pozycji szkieletów są statystyki pobierania dotyczące NPM (ang. *Node Package Manager*) [11]. Ponieważ NPM jest najpopularniejszym miejscem do instalowania bibliotek, jego liczby pobrań mogą dostarczyć wiarygodnych informacji na temat tego, jak dobrze technologia jest postrzegana i wykorzystywana przez społeczność (rys 2).



Rys. 1. Statystyka GitHub stars szkieletów [11]



Rys 2. Statystyka pobierania szkieletów [9]

W ramach przeprowadzonej analizy powyżej wymienionych źródeł można wyróżnić dwa najczęściej używane szkielety: React i Angular. Do późniejszego badania zdecydowano się użyć Angular oraz Vue.js, który nie tylko zyskuje na popularności, ale również jest uważany za jeden z najbardziej obiecujących.

4.2 Szkielet Angular

Angular to popularna platforma programistyczna umożliwiająca tworzenie aplikacji mobilnych i stacjonarnych [12]. Został on pierwotnie stworzony przez pracowników Google Misko Hevery i Adama Abronsa w 2008 roku.

Szkielet jest zbudowany w całości w języku TypeScript i użycie go w rozwoju JavaScript jest zalecane, choć nie obowiązkowe. Język TypeScript to nadzbiór JavaScript, który dodaje opcjonalne pisanie statyczne do JavaScript oraz został pierwotnie wprowadzony i jest nadal obsługiwanym przez Microsoft oprogramowaniem typu open-source.

Angular, podobnie jak wiele innych szkieletów, jest oparty na komponentach. Oznacza to, że komponenty są głównymi elementami składowymi - mogą wyświetlać informacje, renderować szablony i wykonywać działania na danych.

Powiązanie danych w ramach komponentu jest również godne uwagi. Zasadniczo dotyczy to wymiany danych między widokiem (szablon HTML) a modelem (plikiem TypeScript).

Routing odgrywa ważną rolę dla użyteczności SPA. Definiując trasy w osobnym pliku lub module, Angular obsługuje logikę, dla której składnik powinien być wyświetlany w zależności od aktualnie aktywnej ścieżki adresu URL.

W tym badaniu jest wykorzystany i oceniany Angular w wersji 6.4.7.

4.3 Szkielet Vue.js

Vue.js to innowacyjna platforma JavaScript typu open-source. Szkielet jest łatwo dostępny, wszechstronny i produktywny [13]. Wydany w 2014 r. przez Ewana You, byłego pracownika Google, który miał duże doświadczenie w pracy z AngularJS. Jednak wersja 1.0.0 pojawiła się dopiero w październiku 2015 r.

Vue nazywany jest często środowiskiem progresywnym, którego można używać do tworzenia interfejsów użytkownika. Chociaż nie był ściśle powiązany z szablonem Model-ViewViewModel (MVVM), został częściowo zainspirowany zasadami projektowania.

Vue korzysta z wielu przydatnych funkcji React i Angular, takich jak wirtualny DOM (ang. *Document Object Model*, zwany dalej DOM) i szablony. Wiele z tych funkcji zostało zaimplementowanych w jeszcze bardziej złożony sposób. Skalowalność jest jedną z głównych zalet. Jedną z cech Vue jest to, że jest w pełni rozwijany przez społeczność open source, a nie duże przedsiębiorstwo.

Podobnie jak Angular, Vue używa dyrektyw w swoich szablonach. Są one wykorzystywane do wiązania danych, obsługi zdarzeń i nie tylko oraz wizualnie oznaczone prefiksem *v*.

Rekwizyty Vue przypominają React. Aby wysłać dane z komponentów nadrzędnych do podrzędnych, jedną z opcji jest użycie rekwizytów.

W tym badaniu jest wykorzystany i oceniany Vue.js w wersji 2.6.10.

5. Plan badań

W SPA, gdzie stosowane są narzędzia JavaScript, szybkość reagowania i interakcji z klientem w dużej mierze zależy od wydajności platformy. W związku z tym wydajność szkieletów jest ważnym aspektem, który powinien być brany pod uwagę przy ocenie. Do oceny wydajności aplikacji testowej będzie używana przeglądarka Google Chrome z narzędziem DevTools [14].

Do weryfikacji wydajności będą wykorzystywane dwie strony aplikacji:

1) strona z listą dostępnych treningów wybranej kategorii (rys. 3);
2) strona ustawień użytkownika (rys. 4 – rys. 5).

Wykorzystane zostaną podane strony, ponieważ istnieje możliwość wyświetlenia nieograniczonej liczby treningów w pierwszym przypadku, i historii treningów w innym.

Zdecydowano się użyć cztery scenariusze badania, które różnią się liczbą treningów: 5, 100, 1000, 10000.

Biorąc pod uwagę, że termin „Wydajność" jest kompleksowym i wspólnym kryterium, w badaniu wykorzystane zostaną konkretne metryki. Poniżej znajduje się lista wybranych metryk. Wszystkie one mierzone w milisekundach (ms).

1) czas ładowania HTML (ang. *Loading time*),
2) całkowity czas ładowania strony (ang. Total time),
3) czas działania procesora graficznego (ang. *Graphics Processing Unit*, zwany dalej GPU).

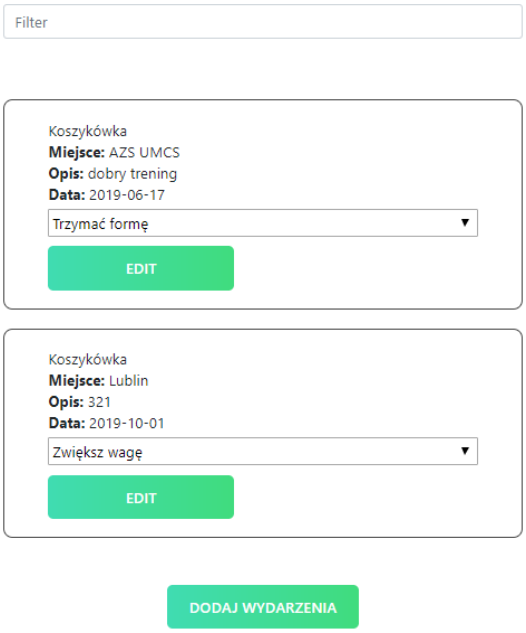
Porównanie się odbędzie w następujący sposób:

1) Do każdego szkieletu i scenariusza test będzie uruchomiony pięć razy, aby znaleźć wartość średnią i 2 razy, aby porównać tzw. trottling (ang. *Throttling*, zwany dalej trottling). Termin ten odnosi się do zjawiska polegającego na obniżeniu taktowania karty graficznej (GPU). Narzędzie DevTools wspiera czterokrotny i sześciokrotny poziom obniżenia taktowania. Odpowiednio, będzie wykorzystany każdy poziom do każdego scenariusza. Całkowita liczba wszystkich testów wynosi 112. Należy zauważyć, że scenariusze do porównania trottlingu będą używane do zrozumienia ograniczeń przepustowości aplikacji w celu określenia niezawodności systemu podczas ekstremalnych obciążeń (maksymalnej liczbie DOM elementów). Również dlatego, żeby odpowiedzieć na pytania o wystarczającej wydajności systemu w przypadku, gdy bieżące obciążenie znacznie przekracza przewidywaną maksymalną liczbę.
2) Wszystkie uzyskane dane zostaną wprowadzone do każdego szkieletu i osobnej tabeli, a następnie dla każdej metryki obliczona wartość średnia. Po otrzymaniu wartości średniej dane będą zgrupowane do jednej tabeli. Również metryki każdego szkieletu zostaną przedstawione w postaci wykresów.

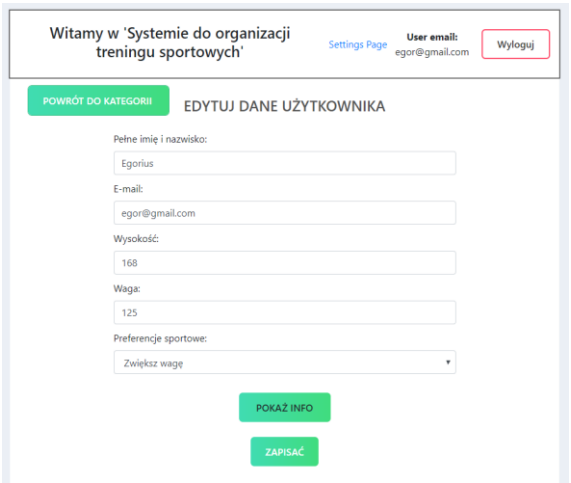
6. Aplikacja badawcza

Na potrzeby badania zrealizowano aplikację testową: system do organizacji treningów sportowych. Zadaniem systemu jest dostarczenie użytkownikowi optymalnego treningu dobranego ze względu na jego potrzeby i wymagania. Aplikacja oferuje kompleksowe formy treningów, pozwalające uzyskać dobrą formę sportową. Ponadto system umożliwia dokonanie oceny stanu fizycznego użytkownika, monitorowanie skuteczności treningu, a także dobór optymalnych ćwiczeń w celu uzyskania pożądanego efektu.

Do przeprowadzenia analizy zostały stworzone dwie aplikacje z maksymalnie podobnym UI. Każda z nich zawiera trzy główne moduły: logowanie i rejestracja, ustawienia użytkownika, stworzenie i wyświetlanie listy treningów. Na rysunkach 3 – 5 przedstawione są strony, które będą potrzebne podczas analizy wydajności.



Rys. 3. Strona z listą dostępnych treningów wybranej kategorii



Rys. 4. Formularz edycji danych strony ustawień użytkownika

Rys. 5. Lista ukończonych i przyszłych treningów strony ustawień użytkownika

7. Realizacja badań

Przed rozpoczęciem analizy wydajnościowej została przeprowadzona implementacyjna analiza porównawcza. Wyniki analizy porównawczej pozwalają stwierdzić, że projekt stworzony za pomocą szkieletu Vue.js jest mniejszy o 41 kB od projektu zrealizowanego przy użyciu szkieletu Angular. Dla tworzenia małych projektów otrzymana różnica nie jest zasadnicza. Jednak, w przypadku dużych aplikacji - każdy megabajt danych jest istotny. Należy zauważyć, że im mniejszy rozmiar gotowej aplikacji, tym szybciej się ładuje strona, i tym mniej czasu potrzeba na analizę w przeglądarce.

Implementacyjna analiza zawiera jeszcze jedną metrykę - liczba linii kodu. Całkowita liczba linii kodu użytych w aplikacji z wykorzystaniem szkieletu Angular wynosi 1801, do szkieletu Vue.js – 1558.

Podczas prowadzenia badań wydajnościowych ważnym aspektem, który bezpośrednio wpływa na ocenę, jest moc obliczeniowa komputera. Testy zostały uruchomione na komputerze o następujących parametrach:

1) procesor - Intel Core i7 - 4400U;
2) pamięć RAM - 12 GB;
3) dysk - 256GB SSD;
4) karta graficzna - Intel HD Graphics 4400;
5) system operacyjny - Windows 10 Pro.

W trakcie badań okazało, że obliczeniowa moc komputera nie pozwala na renderowanie 10000 treningów na jednej stronie, dlatego zdecydowano się zmniejszyć liczbę do tego scenariusza dwukrotnie.

Poniżej przedstawiono wyniki analizy wydajności dla każdej ze stron w postaci tabel (tabela 1 - 2). Im krótszy czas, tym lepiej działa aplikacja. Do wygodnego przeglądania tabel kolorem żółtym zaznaczono szkielet Angular. Niebieskim kolorem zaznaczona kolumna szkieletu Vue.js. Pomarańczowym kolorem zaznaczona kolumna różnicy procentowej pomiędzy szkieletami Vue.js a Angular.

Porównanie wyników odbywa się między scenariuszami na podstawie podania wartości średniej, odchylenia standardowego, różnicy procentowej każdej metryki bez uwzględniania wartości zerowych.

Po przeprowadzeniu wszystkich 112 testów zwrócono uwagę na to, jak trottling wpływa na wyniki. Wybrany wskaźnik (4x lub 6x) w porównaniu do testów bez wykorzystania trottlingu, zwiększa wyniki średnio o 37% - 56% dla 4x, i odpowiednio 125% - 410% dla 6x.

W niektórych testach z powodu niewystarczającej mocy komputera wyniki nie zostały przedstawione w narzędziu DevTools, ale wyświetlone na stronie z poprawną liczbą treningów. W takim przypadku dla takiego testu do tabeli podano wartość 0.

Na podstawie uzyskanych wyników, można powiedzieć, że najczęściej takie testy zostały uruchomione do scenariusza nr 4. W przypadku otrzymania procentowej wartości ujemnej to oznacza, że szkielet Vue.js jest bardziej wydajny w porównaniu do szkieletu Angular.

Analizując wyniki badań, można stwierdzić, że dla metryk "Czas działania procesora graficznego", "Całkowity czas ładowania strony" szkielet Vue.js wykazuje lepszą wydajność.

Tabela 1. Wyniki analizy strony z listą dostępnych treningów [ms]

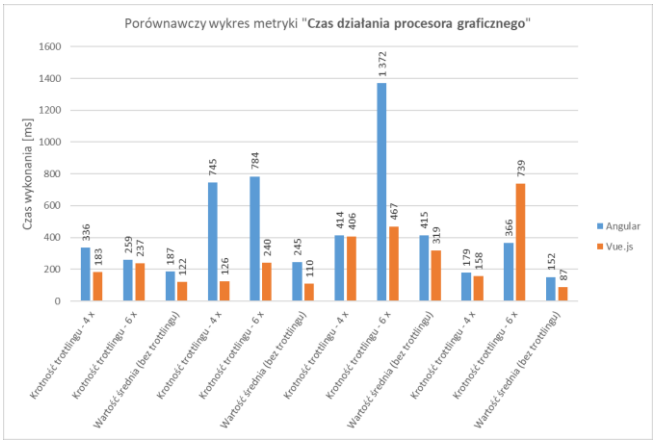
Scenariusz		Metryki								
		Czas ładowania HTML			Całkowity czas ładowania strony			Czas działania procesora graficznego		
		Angular	Vue.js	Różnica [%]	Angular	Vue.js	Różnica [%]	Angular	Vue.js	Różnica [%]
№ 1 (5 treningów)	Krotność trottlingu - 4 x	145,0	169,0	17%	18954,0	6061,0	-68%	336,0	183,0	-46%
	Krotność trottlingu - 6 x	248,0	243,0	-2%	27467,0	8023,0	-71%	259,0	237,0	-8%
	Wartość średnia (bez trottlingu)	44,0	42,2	-4%	5258,3	1427,4	-73%	187,0	122,2	-35%
№ 2 (100 treningów)	Krotność trottlingu - 4 x	189,0	163,0	-14%	30408,0	5071,0	-83%	745,0	126,0	-83%
	Krotność trottlingu - 6 x	332,0	276,0	-17%	40938,0	8028,0	-80%	784,0	240,0	-69%
	Wartość średnia (bez trottlingu)	42,4	48,0	13%	5033,8	1702,0	-66%	245,4	110,4	-55%
№ 3 (1000 treningów)	Krotność trottlingu - 4 x	150,0	170,0	13%	21927,0	16327,0	-26%	414,0	406,0	-2%
	Krotność trottlingu - 6 x	242,0	324,0	34%	68559,0	29931,0	-56%	1372,0	467,0	-66%
	Wartość średnia (bez trottlingu)	44,2	52,0	18%	8038,8	4268,2	-47%	414,6	318,8	-23%
№ 4 (5000 treningów)	Krotność trottlingu - 4 x	0,0	0,0	0%	24815,0	15094,0	-39%	179,0	158,0	-12%
	Krotność trottlingu - 6 x	0,0	0,0	0%	43091,0	125709,0	192%	366,0	739,0	102%
	Wartość średnia (bez trottlingu)	0,0	0,0	0%	7704,4	5179,8	-33%	152,0	87,2	-43%

Tabela 2. Wyniki analizy strony ustawień użytkownika [ms]

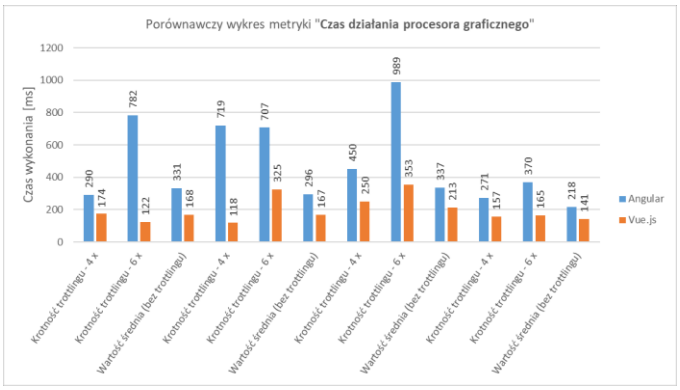
Scenariusz		Metryki								
		Czas ładowania HTML			Całkowity czas ładowania strony			Czas działania procesora graficznego		
		Angular	Vue.js	Różnica [%]	Angular	Vue.js	Różnica [%]	Angular	Vue.js	Różnica [%]
№ 1 (5 treningów)	Krotność trottlingu - 4 x	162,0	176,0	9%	17178,0	4209,0	-75%	290,0	174,0	-40%
	Krotność trottlingu - 6 x	180,0	271,0	51%	28234,0	6366,0	-77%	782,0	122,0	-84%
	Wartość średnia (bez trottlingu)	41,6	44,8	8%	4912,0	1778,0	-64%	331,2	168,2	-49%
№ 2 (100 treningów)	Krotność trottlingu - 4 x	114,0	178,0	56%	19252,0	4171,0	-78%	719,0	118,0	-84%
	Krotność trottlingu - 6 x	231,0	315,0	36%	31935,0	7264,0	-77%	707,0	325,0	-54%
	Wartość średnia (bez trottlingu)	47,4	44,4	-6%	5116,0	1973,0	-61%	296,0	166,8	-44%
№ 3 (1000 treningów)	Krotność trottlingu - 4 x	184,0	169,0	-8%	22640,0	6155,0	-73%	450,0	250,0	-44%
	Krotność trottlingu - 6 x	248,0	283,0	14%	48300,0	9126,0	-81%	989,0	353,0	-64%
	Wartość średnia (bez trottlingu)	46,2	45,0	-3%	7373,0	3121,6	-58%	336,8	213,2	-37%
№ 4 (5000 treningów)	Krotność trottlingu - 4 x	160,0	165,0	3%	25561,0	7530,0	-71%	271,0	157,0	-42%
	Krotność trottlingu - 6 x	0,0	0,0	0%	41478,0	10639,0	-74%	370,0	165,0	-55%
	Wartość średnia (bez trottlingu)	61,6	42,0	-32%	9871,8	4472,8	-55%	217,8	140,6	-35%

Metryka czas działania procesora graficznego została przedstawiona w postaci wykresów słupkowych (rys. 6 - 7). Na wykresach widać, że z 32 przeprowadzonych testów

wyłącznie w 2 szkielet Vue.js pokazuje gorsze wyniki, co wskazują na widoczną przewagę tego szkieletu.



Rys. 6. Porównawczy wykres metryki „*Czas działania procesora graficznego*" strony z listą dostępnych treningów [ms]



Rys. 7. Porównawczy wykres metryki „*Czas działania procesora graficznego*" strony ustawień użytkownika [ms]

8. Wnioski

Cele postawione w artykule zostały spełnione. Aby zrealizować badania została stworzona aplikacja sportowa. Program spełnia wszystkie wymagania funkcjonalne i niefunkcjonalne. Do tworzenia aplikacji wykorzystano narzędzia w najnowszych wersjach.

Do weryfikacji wydajności zostały wykorzystywane dwie strony aplikacji i cztery scenariusze badania, które różnią się liczbą treningów. Całkowita liczba wszystkich testów wynosiła 112.

Zarówno Vue.js jak i Angular to bardzo wydajne narzędzia, które ułatwiają tworzenie SPA. Wyniki przeprowadzonego badania pozwalają na potwierdzenie tezy o tym, że Vue.js jest bardziej wydajny w porównaniu do szkieletu Angular.

Literatura

[1] Petukhova, E. (2019). Sitecore JavaScript Services Framework Comparison.

[2] Boaler, S. (2019). ReactJS a Viable Career Option?.

[3] Pano, A., Graziotin, D., & Abrahamsson, P. (2018). Factors and actors leading to the adoption of a JavaScript framework. Empirical Software Engineering, 23(6), 3503-3534.

[4] Nägele, T., Hooman, J., Zigterman, R., & Brul, M. (2015). Client-side performance profiling of JavaScript for web applications. Universidad de Radbound.

[5] Ferreira, J. (2018). A JavaScript Framework Comparison Based on Benchmarking Software Metrics and Environment Configuration.

[6] Mariano, C. L. (2017). Benchmarking javascript frameworks.

[7] Voutilainen, J. (2017). Evaluation of Front-end JavaScript Frameworks for Master Data Management Application Development.

[8] Проценко, М. М. (2016). Аналіз фреймворків як засобів розробки Web-додатків. Міжнародний науковий журнал, (6 (1)), 86-89.

[9] Npm trends, https://www.npmtrends.com/@angular/core-vs-angular-vs-react-vs-vue [09.2019]

[10] Google Trends, https://trends.google.pl/trends/explore cat=31&date=2016-03-01%202019-03-01&q=React,Angular,Vue [08. 2019]

[11] Star history, http://www.timqian.com/star-history/#angular/angular&vuejs/vue&facebook/react [08. 2019]

[12] Angular, https://angular.io/docs [08.2019]

[13] Vue.js, https://vuejs.org [08.2019]

[14] Chrome DevTools, https://developers.google.com/web/tools/chrome-devtools [09.2019]