JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 20 (2021) 205–209
Received: 16 June 2021
Accepted: 29 June 2021

Comparative analysis of the Angular 10 and Vue 3.0 frameworks

Analiza porównawcza szkieletów programistycznych Angular 10 i Vue 3.0

Jarosław Kyć*, Piotr Lipski*, Beata Pańczyk

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

The aim of this article is to perform a comparative analysis of the Angular v10 and Vue v3.0 frameworks. The basis of the comparison is the performance tested with two applications that are similar in terms of functionality. The view with a variable number of displayed elements was examined, and the time was measured from the moment the number of components was indicated to the end of rendering. The amount of disk space occupied by the final applications and application segments was also compared in relation to the method of implementing their functionality. The results of the research allowed to formulate the conclusions that Vue is more efficient than Angular and additionally the Vue application takes up less disk space.

Keywords: Angular; Vue.js; efficiency; comparison

Streszczenie

Celem artykułu jest przeprowadzenie analizy porównawczej dwóch szkieletów programistycznych Angular w wersji 10 oraz Vue w wersji 3.0. Podstawą porównania jest wydajność zbadana za pomocą dwóch podobnych co do funkcjonalności aplikacji. Badaniu poddano widok ze zmienną liczbą wyświetlanych elementów, a czas mierzono od momentu wskazania liczby komponentów do zakończenia renderowania. Porównano także ilość przestrzeni dyskowej jaką zajmują finalne aplikacje oraz modułów aplikacji względem sposobu realizacji ich funkcjonalności. Wyniki badań pozwoliły sformułować wnioski, że Vue jest bardziej wydajny od Angulara a dodatkowo aplikacja Vue zajmuje mniej przestrzeni dyskowej.

Słowa kluczowe: Angular; Vue.js; wydajność; porównanie

*Corresponding author

Email address: jaroslaw.kyc@pollub.edu.pl (J. Kyć), piotr.lipski2@pollub.edu.pl (P. Lipski)

1. Wstęp

Środowisko przeglądarek i języki programowania zorientowane na aplikacje internetowe rozwijały się na przestrzeni lat. Szkielet strony internetowej w postaci plików HTML w połączeniu z kaskadowymi arkuszami stylów i językiem JavaScript dają coraz więcej możliwości. Dzięki rozwojowi technologii aplikacje już nie tylko przedstawiają treści, ale wchodzą w interakcje z użytkownikiem.

Aplikacje internetowe można podzielić ze względu na wielkość. Z reguły mniejsze aplikacje są aplikacjami statycznymi [1]. Mniejsze aplikacje, które nie posiadają skomplikowanej logiki mogą zostać zbudowane od podstaw przy użyciu czystego kodu JavaScript, plików HTML oraz arkuszy stylów. Przy rozbudowanych projektach bardziej opłaca się skorzystać z pomocy szkieletów programistycznych.

Korzystanie z gotowych szkieletów stało się powszechną praktyką w branży IT. Szkielety znacząco ułatwiają i przyspieszają początkowe prace nad projektem. Oczywiście z każdym nawet najprostszym szkieletem programistycznym należy się zaznajomić, poznać jego budowę i zasady działania. Należy jednak zastanowić się jak taki pakiet gotowych rozwiązań wpływa na proces tworzenia aplikacji i jej końcowy wynik. Czym różnią się szkielety programistyczne i którego użyć do danego projektu?

Technologie rozwijają się w bardzo szybkim tempie i pojawiają się coraz to nowsze ich wersje Nasuwa się jednak pytanie jak pokrewne rozwiązania wypadają na tle wydajności oraz procesu tworzenia aplikacji internetowych.

2. Cel i zakres badań

Celem badań przeprowadzonych w niniejszym artykule jest analiza wydajności szkieletów programistycznych Vue.js w wersji 3.0 oraz Angular w wersji 10.0 i wybranie korzystniejszego rozwiązania. W tym celu stworzona została identyczna aplikacja internetowa za pomocą obydwu szkieletów programistycznych. Porównanie wydajności będzie polegać na zmierzeniu czasu rysowania elementów od momentu wyboru ich liczby, aż do zakończenia rysowania. Zostanie to zmierzone za pomocą narzędzi deweloperskich dostępnych w przeglądarce Google Chrome i Firefox.

W artykule postawiono następującą tezę badawczą: Szkielet programistyczny Vue.js jest bardziej wydajny niż Angular.

3. Obiekty badań

Testowane szkielety programistyczne zostały wybrane ze względu na swoją popularność. Są to dwa szkielety programistyczne, które plasują się w ścisłej czołówce popularności.

Wedle ankiety [2] przeprowadzonej na jednym z najbardziej znanych serwisów społecznościowych programistów StackOverflow - Vue i Angular znajdują się wśród czołówki najczęściej używanych sieciowych szkieletów programistycznych przez deweloperów z całego świata.

Zarówno Vue.js jak i Angular są potężnymi narzędziami zdolnymi do tworzenia zaawansowanych stron typu SPA (ang. Single Page Application). Oba rozwiązania stosują podobny styl pisania komponentów, które można użyć w różnych kontekstach. Jednak mimo podobieństw mają kilka znaczących różnic.

3.1. Vue

Vue to progresywny szkielet programistyczny do budowania interfejsów użytkownika. W przeciwieństwie do innych monolitycznych szkieletów programistycznych, Vue jest projektowany od podstaw tak, aby można go było stopniowo adaptować. Podstawowa biblioteka skupia się tylko na warstwie widoku i jest łatwa do pobrania i zintegrowania z innymi bibliotekami lub istniejącymi projektami. Z drugiej strony, Vue jest również przystosowany do zasilania zaawansowanych jednostronicowych aplikacji, gdy jest używany w połączeniu ze wspierającymi bibliotekami [3].

3.2. Angular

Angular jest szkieletem programistycznym do projektowania aplikacji i platformą programistyczną do tworzenia wydajnych i zaawansowanych aplikacji typu SPA. Angular w przeciwieństwie do Vue i Reacta jest rozwiązaniem kompletnym i nie polega na zewnętrznych bibliotekach. Implementuje on podstawowe i opcjonalne funkcje jako zestaw bibliotek TypeScript, które są importowane do aplikacji. Podstawowymi elementami konstrukcyjnymi szkieletu programistycznego Angular są komponenty, które są zorganizowane w moduły. Moduły zbierają powiązany kod do zestawów funkcyjnych. Aplikacja Angular jest zdefiniowana przez zestaw modułów. Zawsze posiada co najmniej główny moduł i zazwyczaj wiele innych modułów funkcjonalnych [4].

3.3. Podobieństwa i różnice

W tabeli 1 porównano cechy obu szkieletów programistycznych [5, 6].

Tabela 1: Porównanie frameworków Vue i Angular

	Angular	Vue.js
Podobieństwa	Wykorzystanie podwójnych nawiasów klamrowych do interpolacji kodu w szablonach.	
	Szkielety posiadają zaawansowane narzędzie cli służące do generacji kodu/projektu.	
	Analogiczny jest sposób korzystania z techniki data binding (np. v-bind vs ng-bind).	
Różnice	Modyfikacje realizowane na rzeczywistym obiekcie DOM.	Wykorzystanie virtual-DOM do modyfikacji dokumentu.
	Wykorzystuje komponenty klasowe.	Wsparcie dla komponentów obiektowych oraz funkcyjnych.
	Konieczność korzystania z języka TypeScript,	Dowolność wyboru między językiem który jest nadzbiorem JavaScript. JavaScript i TypeScript
	Konieczność korzystania z rxjs.	Szerokie wsparcie dla zewnętrznych bibliotek.

4. Przegląd literatury

W artykule [7] zostało przeprowadzone badanie szkieletów Vue.js w wersji 2 oraz Angular 6 pod kątem wydajności wyrażonej jako czas wysyłania żądania i renderowania komponentów. Autorzy stwierdzili, że szkielet programistyczny Vue.js lepiej nadaje się dla początkującego programisty niż Angular, ponieważ łatwiej jest się go nauczyć. Dodatkowym wymienionym powodem jest wydajność przemawiająca za szkieletem programistycznym Vue.js pod warunkiem, że tworzona aplikacja ma średnią złożoność. W podsumowaniu autorzy doszli do wniosku, że na podstawie przeprowadzonych badań szybkość pracy obu szkieletów programistycznych jest do siebie zbliżona.

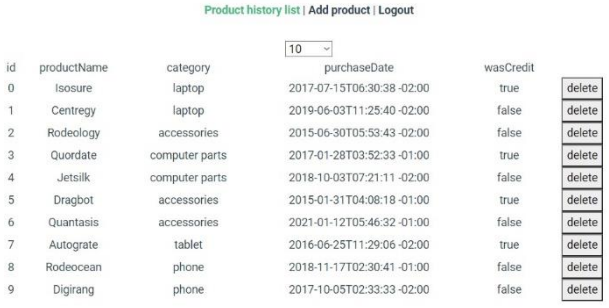
Artykuł [8] dotyczy szkieletów aplikacyjnych Vue 2, Angular 4 i React 16 oraz metodyki tworzenia jednostronicowych SPA i wielostronicowych MPA (ang. Multi Page Application) aplikacji internetowych. Najlepszy wynik w przeprowadzonych tam badaniach uzyskał Vue (w kategorii SPA oraz MPA), potem React i Angular.

5. Metoda badań

Na bazie takich samych co do funkcjonalności aplikacji (typu SPA) zaimplementowanych w obu szkieletach programistycznych, przeprowadzono badanie wydajności.

Podstawą porównania był sposób przygotowania aplikacji, implementacja jej komponentów w odniesieniu do dwóch szkieletów oraz finalna wydajność, czyli czas renderowania głównego komponentu.

Obiektem prowadzonych badań był widok z tabelą (Rys. 1), która zawierała szereg danych do wyświetlenia. Na stronie znajduje się rozwijana lista z możliwością wyboru: 10, 100, 1000 i 10000 wierszy do wyświetlenia w tabeli. Zmierzono czasy między kliknięciem w konkretną liczbę z listy, a momentem wyrenderowania danych w tabeli. Dla każdej opcji z listy, testy powtórzono po 10 razy.



Rysunek 1: Interfejs graficzny aplikacji testowych.

Aplikacje zostały uruchomione na dwóch przeglądarkach: Google Chrome w wersji 90.0.4430.93 (64-

bitowa) i Firefox Developer w wersji 89.0b6 (64-bitowa). Przeglądarki zostały wybrane ze względu na różne silniki renderowania co mogło wpływać na wyniki. Obie przeglądarki posiadają własne narzędzia deweloperskie. Google Chrome został zbudowany na podstawie projektu chromium, który korzysta z silnika blink [9]. Na jego podstawie zaimplementowane są także takie przeglądarki jak Opera, Edge, czy Vivaldi. Firefox z kolei korzysta z autorskiego silnika Gecko [10].

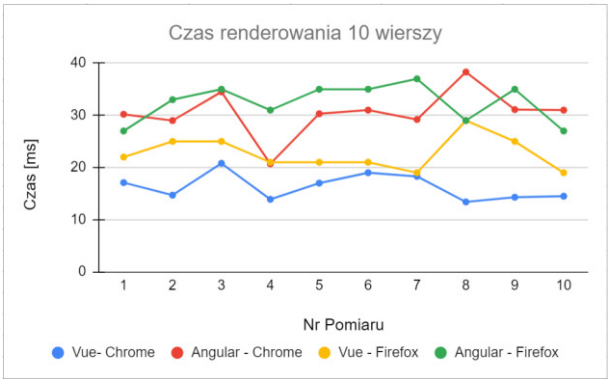
6. Platforma testowa

Do uruchomienia aplikacji został użyty komputer klasy PC z następującą specyfikacją:

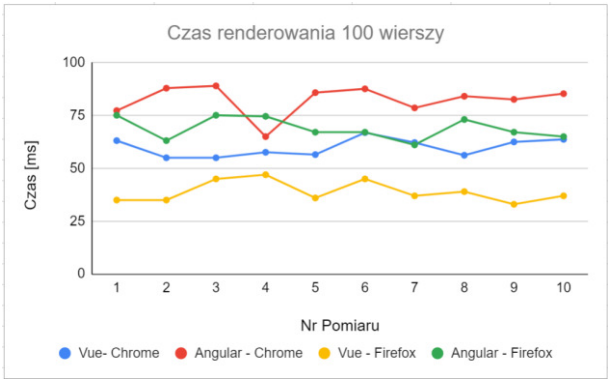
- procesor: AMD Ryzen 5 1600 AF 3.2GHz, 6 rdzeni, 12 wątków,
- pamięć: 16384MB DDR4 3000MHz,
- dysk: SSD 512GB M.2,
- system operacyjny: Windows 10 Pro,
- rozdzielczość ekran: 1920x1080px,
- układ graficzny: NVIDIA GeForce RTX 2060.

7. Wyniki

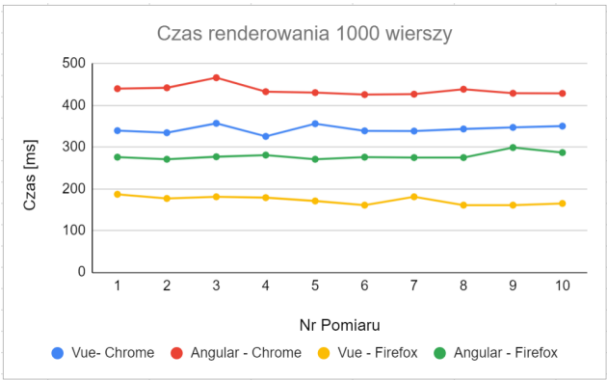
Wyniki przeprowadzonych testów przedstawiono na rysunkach 2-5.



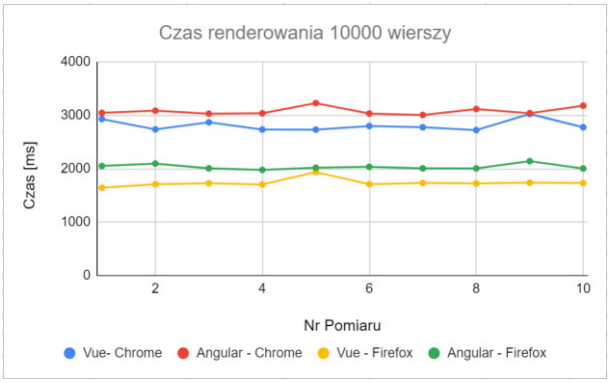
Rysunek 2: Wyniki pomiarów dla 10 wierszy.



Rysunek 3: Wyniki pomiarów dla 100 wierszy.

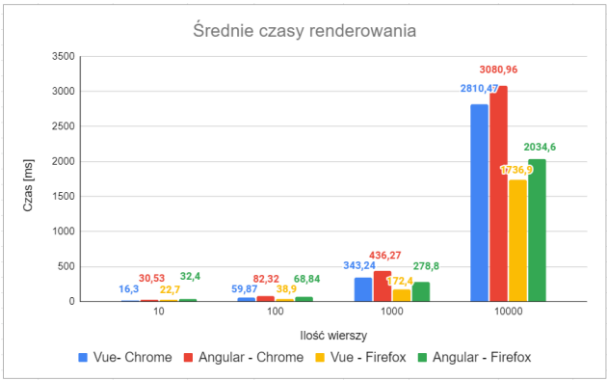


Rysunek 4: Wyniki pomiarów dla 1000 wierszy.



Rysunek 5: Wyniki pomiarów dla 10000 wierszy.

Średnie czasy przedstawia rysunek 6.



Rysunek 6: Średnie czasy pomiarów.

8. Analiza

Wyniki dla 10 wierszy (Rys. 2) przedstawiają niższe czasy renderowania dla Vue niż w przypadku Angular. Takie wyniki otrzymano zarówno w przeglądarce Google Chrome jak i Firefox. Wyniki pojedynczych pomiarów wykazywały największe wahania i odstępstwa od średniej właśnie dla wyświetlania 10 wierszy.

Najniższe czasy renderowania dla 100 wierszy (Rys. 3) uzyskał ponownie Vue, jednak najlepszy czas odnotowano tym razem dla przeglądarki Firefox (55ms). Angular w najlepszym pomiarze potrzebował 61ms. Wahania pojedynczych pomiarów zmalały względem pomiarów dla 10 wierszy.

W przypadku 1000 wierszy (Rys. 4) można zaobserwować wzrost przewagi aplikacji Vue nad Angular.

Najniższe czasy uzyskał Vue w przeglądarce Firefox (161ms). Dla porównania najniższy czas Angular osiągnął również w przeglądarce Firefox 271ms. Jest to aż 160% czasu uzyskanego przez Vue.

Dla 10000 wierszy (Rys. 5) wyniki są zbliżone do przypadku z 1000 wierszy. Najkrótszy czas uzyskał Vue w Firefox - 1645ms w jednym pomiarze. W przypadku aplikacji Angular najlepszy wynik to 1978 ms. Przewaga Vue jest widoczna na obydwu przeglądarkach.

Wszystkie wyniki przedstawiają dość spójny obraz wydajności porównywanych szkieletów programistycznych. W każdym przypadku to Vue jest szybszy w renderowania widoku z tabelą. Dodatkowo widać, że Vue uzyskał lepsze wyniki w przeglądarce Firefox.

W tabeli 2 porównano rozmiary aplikacji i czasy ich kompilacji.

Tabela 2: Porównanie rozmiarów aplikacji

	Angular	**Vue**
Rozmiar aplikacji	482 MB	239 MB
Czas kompilacji	26539 ms	2729 ms

9. Wnioski

Oba badane szkielety programistyczne są zaawansowanymi narzędziami tworzenia aplikacji typu SPA. Wykazują podobieństwa i różnice, które przedstawiono w tabeli 1.

Pod względem czasu renderowania się komponentów to Vue okazał się lepszy niż Angular (Rys. 2-5). Wyniki dla dużej liczby wierszy, przedstawione w niniejszym artykule mogą nie znaleźć odzwierciedlenia w rzeczywistym użyciu, gdyż wtedy stosowane są ograniczenia co do maksymalnej liczby wyświetlanych rekordów oraz stronicowanie wyników. Mimo tego dla celów badawczych zostały one poddane testom i również wykazały lepszą wydajność Vue, co potwierdza postawioną tezę badawczą iż Vue.js jest bardziej wydajny niż Angular.

Dodatkowym atutem przemawiającym po stronie Vue.js jest finalny rozmiar aplikacji na dysku. Jest on dwukrotnie mniejszy niż aplikacji Angular. Dla szkieletu programistycznego Vue.js zostały wykorzystane pokrewne biblioteki do gotowych elementów graficznych i należy wziąć to pod uwagę. Wybór różnych bibliotek wynika z tego, że Vue material nie obsługuje wersji Vue 3.0.

Jeśli chodzi o łatwość przyswajania danego szkieletu programistycznego oraz preferencje programistów to Angular może wydawać się bardziej przyjazny programistom części serwerowych aplikacji (back-end) z powodu zastosowania podobnych konwencji programowania, na przykład techniki wstrzykiwania zależności. Często spotykanym zjawiskiem na stanowisku fullstack developer jest programowanie w Angular części klienckiej i programowanie w Spring (Java) w części serwerowej aplikacji.

Z kolei Vue może wydawać się bardziej przystępny dla nowych programistów, bez doświadczenia programistycznego po stronie serwera, ze względu na możliwość pisania w czystym języku JavaScript oraz ze względu na konieczność przyswojenia mniejszej liczby technologii. Z tych powodów wydajność nie powinna być jedynym kryterium wyboru. Różnice rzędu kilkudziesięciu milisekund będą niezauważalne dla użytkownika końcowego, zaś wybór odpowiednich narzędzi dla zespołu programistycznego może znacznie zwiększyć jego komfort oraz szybkość pracy.

W artykule [7] stwierdzono, że wydajności aplikacji Angular i Vue są bardzo zbliżone. Natomiast w wyniku badań prezentowanych w niniejszym artykule Vue.js uzyskał zdecydowanie lepsze wyniki. Wnioski dotyczące przystępności szkieletów programistycznych są natomiast zbliżone do tych przedstawionych w publikacji z artykułu [7].

Różnice w wydajności w stosunku do badań z pozycji [7] mogą być spowodowane konkretną funkcjonalnością aplikacji testowych oraz różnymi wersjami szkieletów programistycznych, na których realizowano testy wydajnościowe.

Literatura

[1] Porównanie stron statycznych z dynamicznymi, https://about.gitlab.com/blog/2016/06/03/ssg-overview-gitlab-pages-part-1-dynamic-x-static/ , [25.06.2021]

[2] Ankieta dotycząca popularnych technologii, https://insights.stackoverflow.com/survey/2020#technology-web-frameworks-all-respondents2 [31.05.2021].

[3] Dokumentacja Vue.js, https://vuejs.org/v2/guide/ , [30.11.2020].

[4] Dokumentacja Angular, https://angular.io/docs , [30.11.2020].

[5] M. Frisbie, Angular 2 Cookbook, Packt Publishing, 2017.

[6] A. Passaglia, Vue.js 2 Cookbook, Packt Publishing, 2017.

[7] R. Baida, M. Andriienko, M. Plechawska-Wójcik, Perfomance analysis of frameworksAngular and Vue.js, Journal of Computer Sciences Institute, 14 (2020) 59-64.

[8] M. Kaluža, K. Troskot, B. Vukelić, Comparison of front-end frameworks for webapplications development. Zbornik Veleučilišta u Rijeci, 2018.

[9] Silnik renderowania Blink, http://www.chromium.org/blink , [25.06.2021].

[10] Silnik renderowania Gecko, https://developer.mozilla.org/en-US/docs/Glossary/Gecko , [25.06.2021].

[11] M.S. Mikowski, J.C. Powell, Single Page Web Applications, Manning Publications, 2013.

[12] TypeScript Notes for Professionals, GoalKicker.com, 2018.

[13] Angular Notes for Professionals, GoalKicker.com, 2018.

[14] JavaScript Notes for Professionals, GoalKicker.com, 2018.

[15] Dokumentacja MDN web docs, https://developer.mozilla.org/en-US/docs/Learn/Performance , [30.11.2020].

[16] Dokumentacja biblioteki Vuex, https://vuex.vuejs.org/ , [30.11.2020].

[17] Dokumentacja biblioteki vue-router, https://router.vuejs.org/ , [30.11.2020].

[18] Dokumentacja biblioteki Ngrx, https://ngrx.io/docs , [30.11.2020].