JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 5 (2017) 204-207
Wysłane: 2017-06-29
Przyjęte: 2017-07-01

Porównanie efektywności szkieletów AngularJS i VueJS

Nazar Patrylo*, Marek Miłosz

Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. W artykule przedstawiono rezultaty analizy porównawczej szkieletów JavaScript AngularJS i VueJS dla tworzenia interfejsów użytkownika. Badanie wydajności zostało zrealizowane poprzez implementację, z użyciem obu szkieletów, aplikacji korzystających z języka JavaScript. Dla każdej z nich wykonano pomiary wydajności i na ich podstawie sporządzono analizę wyników. Analizę wykonano przy pomocy mechanizmu badawczego zawartego w opracowanym kodzie oraz przy użyciu narzędzi developerskich przeglądarek internetowych.

Słowa kluczowe: AngularJS; VueJS; JavaScripr szkielety; wydajność

*Autor do korespondencji.

Adres/adresy nazarpatrylo@gmail.com

Comparison of AngularJS and VueJS frameworks efficiency

Nazar Patrylo*, Marek Miłosz

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

Abstract. The article presents the results of a comparative analysis of JavaScript AngularJS and VueJS frameworks to create user interfaces. The study of productivity has been achieved through the implementation of JavaScript applications using both frameworks. Each of them was executed and the performance was measured. The results was a basis is the analysis. The analysis was done using the code build in the applications as well as the developer tools of web browsers.

Keywords: AngularJS; VueJS; JavaScript frameworks; efficiency

* Corresponding author.

1. Wstęp

Web-programowanie staje się coraz bardziej popularne ze względu na bardzo szybkie tempo rozwoju technologii informatycznych, w tym internetowych. Wraz z tym rozwojem, front-end programowanie przejmuje coraz więcej funkcji strony serwerowej, zmniejszając obciążenie serwera, a tym samym zwiększając ogólną wydajność aplikacji internetowych.

Najlepszym rozwiązaniem do automatyzacji i usprawnienia procesu tworzenia aplikacji stało się korzystanie przez programistów z różnych szkieletów JavaScript. Porządkuje to strukturę aplikacji oraz pozwala na ponowne wykorzystanie już istniejącego kodu. Pozwala to również zwiększyć skalowalność aplikacji [1].

Deweloperzy często wykorzystują wiele szkieletów i bibliotek do wykonywania swojej pracy, szczególnie przy opracowywaniu dużych i złożonych aplikacji. Jedną z ich głównych zalet jest to, że umożliwiają ponowne użycie kodu. Pozwala to na poświęcenie więcej czasu i uwagi na projektowanie i testowanie aplikacji internetowych. Z drugiej strony, istnieje również niebezpieczeństwo, że wybór nieodpowiedniego szkieletu spowoduje obniżenie jakości aplikacji oraz wydłuża terminy jej realizacji.

W ostatnim czasie JavaScript został nazwany jednym z najlepszych języków programowania, jakich można uczyć się w 2017 roku [2].

Popularność języka JavaScript szybko rośnie. W 2016 roku nastąpiły w nim wielkie zmiany. Całkowicie zmodernizowano i wdrożono drugą wersję angularjs, rozwinięto biblioteki jQuery, której używa aż 96,5% wszystkich stron internetowych wykorzystujących JavaScript, zmodernizowano standard ecmascript oraz zaktualizowano Node.js [3].

W tym artykule przedstawiony został proces analizy dwóch szkieletów AngularJS, VueJS. Kryterium wyboru tych frameworków jest ich ogromna popularność oraz sukces prestiżowych aplikacji napisanych za ich pomocą.

2. Cel i hipotezy pracy

Celem pracy jest analiza i porównanie szkieletów AngularJS i VueJS w procesie tworzenia interfejsów użytkownika za pomocą języka programowania JavaScript.

W celu zrealizowania założeń pracy sformułowano dwie hipotezy:

Hipoteza 1: Szkielet AngularJS 2 pokazuje lepsze wyniki przetwarzania danych aplikacji internetowych niż framework VueJS.

Hipoteza 2: Tworzenie aplikacji w szkielecie AngularJS 2 jest bardziej pracochłonne od tworzenia z wykorzystaniem szkieletu VueJS.

By udowodnić lub odrzucić hipotezy badawcze powstała specjalna aplikacja testowa. Przy jej pomocy przetestowano szybkości wykonania różnych operacji. W celu weryfikacji drugiej hipotezy poddano analizie utworzony kod (wraz ze szkieletem) i wyznaczono jego wielkość, mierzoną liczbą linii kodu.

3. Metodyka badań

3.1. Analiza metryk kodu

Zastosowanie testów metryk kodu pozwala kierownikom projektów i deweloperom oceniać różne aspekty tworzonego lub istniejącego produktu, przewidzieć wielkość wykonanej pracy, charakteryzować złożoność oprogramowania i wydajność zespołów je tworzących, oszacować ilość czasu i liczbę osób niezbędnych do napisania programu [8].

Najprostszą metryką jest liczba wierszy w kodzie (ang. *SLOC, Source Lines Of Code*) [7]. Użycie tej metryki jest utrudnione, ze względu na fakt, że jedna i ta sama funkcjonalność może być zapisana w jednym wierszu lub podzielona na kilka. Spowodowało to konieczność rozróżnienia pojęć fizyczna i logiczna linia kodu. Liczba linii kodu oprogramowania zależy także od stylu programowania i używanego języka programowania [4].

SLOC jako miara wielkości oprogramowania może być z powodzeniem użyta do porównania dwóch programów tworzonych przez tego samego programistę.

3.2. Analiza efektywności

Efektywność pracy, jest najważniejszym parametrem programu komputerowego. Dla porównania wydajności aplikacji internetowej zrealizowanej w języku JavaScript najlepiej skorzystać z funkcjonalności samej przeglądarki, ponieważ przeglądarka jest interpreterem kodu aplikacji webowej i ma dostęp bezpośrednio do charakterystyk wydajności jej wykonania.

Przeglądarka Google Chrome posiada zaawansowane narzędzie do sprawdzania wydajności strony – Dev tool. To narzędzie jest używane, m.in. do szczegółowej analizy czasu ładowania elementów na stronie. Program zawiera wszystkie niezbędne opcje do wglądu w pobieranie zasobów i wykonywanie kodu języka JavaScript.

Można stwierdzić, że wyżej opisane narzędzie daje duże możliwości testowania, debugowania, a także analizy poszczególnych elementów na stronie. W procesie realizacji eksperymentu skorzystano z dwóch narzędzi:

- Console, pozwalającego na przeglądanie różnego rodzaju ostrzeżeń i zaleceń, a także błędów.
- Performance, umożliwiającego przegląd czasu poświęconego na ładowanie strony.

3.3. Kryteria i metody porównania

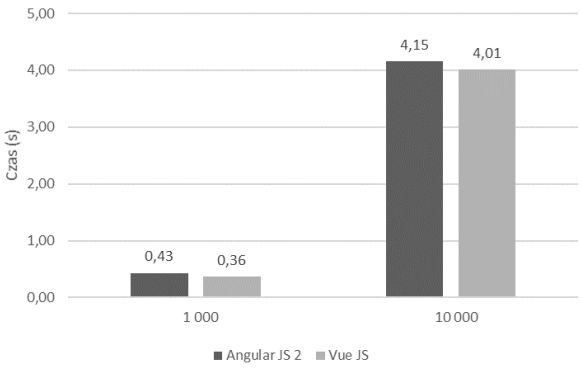
W celu porównania dwóch szkieletów zaimplementowano kilka typowych operacji aplikacji internetowej oraz przeprowadzono testy ich wydajności. Do typowych operacji zaliczone zostały: tworzenie, aktualizowanie, usuwanie obiektów na stronie i mierzenie czasu reakcji przy kliknięciu w obiekt. Badania przeprowadzono czterokrotnie w dwóch etapach: na małej oraz na dużej próbce danych. Z zebranych wartości wyliczono wartości średnie wykonania każdej z wyżej wymienionych operacji.

Drugi etap, czyli porównanie metryki kodu pozwala programiście określić przybliżony czas wykonania zadania, jego stopień trudności oraz scharakteryzować zastosowane wzorce projektowe.

4. Wyniki testów wydajności

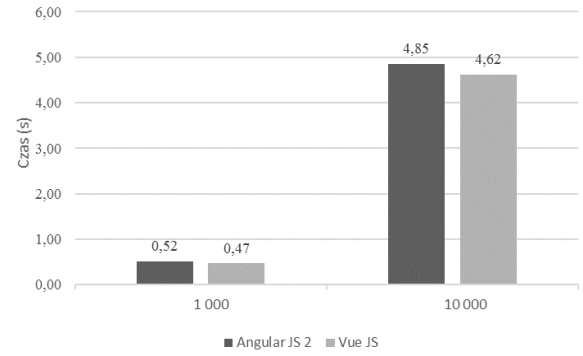
W testach zestawiono dwa programy napisane w różnych szkieletach, co pozwala porównać ich skuteczność i przeanalizować zalety i wady frameworków .

Po przeprowadzeniu eksperymentu i uzyskanych wyników można przystąpić do statystycznej analizy. Jej rezultaty dla obydwóch wersji aplikacji przedstawiono na rys. 1-6.



Rys. 1. Średnie czasowe charakterystyki tworzenia obiektów ekranowych dla różnych ich ilości

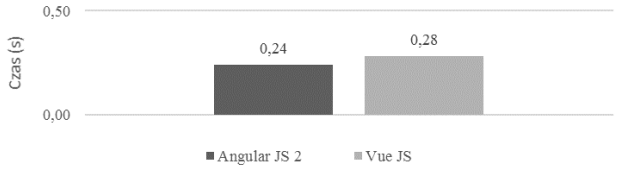
W teście przeprowadzonym na małych liczbach obiektów (informacji) różnica w wydajności szkieletów jest bardzo niewielka, ale przy teście na dużej liczbie różnica ta jest już zauważalna. Znacznie lepiej radzi sobie w tym zakresie szkielet VueJS Dotyczy to zarówno procesu tworzenia nowych obiektów na stronie (rys. 1), jak i dodawania dużej ilości informacji na stronie (rys. 2).



Rys. 2. Średnie czasowe charakterystyki dodawania obiektów ekranowych dla różnych ich wartości

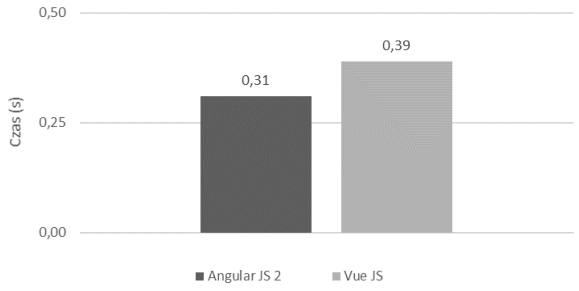
Drugim testowanym procesem jest aktualizacja elementów na stronie oraz testowanie aktualizacji każdego dziesiątego elementu w aplikacji (rys. 3), a także eksperyment zamiany kolejności obiektów w aplikacji (rys. 4).

Jak widać z rysunku 3, dla procesu aktualizacji każdego dziesiątego elementu, aplikacja napisana z wykorzystaniem szkieletu AngularJS 2 wykazywała lepsze rezultaty.



Rys. 3. Średnie czasy aktualizacji elementów interfejsu

W eksperymencie zmian sekwencji elementów aplikacja napisana we frameworku AngularJS 2, podobnie jak w poprzednim teście, osiągnęła lepszy wynik niż aplikacja oparta na szkielecie VueJS (rys. 4).

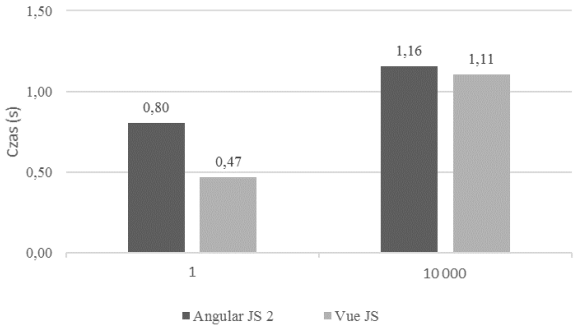


Rys. 4. Średnie czasy zmiany pozycjonowania elementów interfejsu

Trzecim testowanym elementem jest szybkość usuwania obiektów ze strony. Ten eksperyment był podzielony na dwie części.

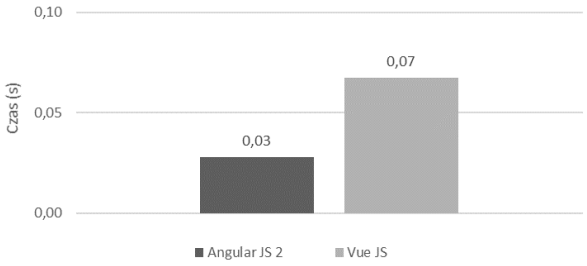
W pierwszej części zbadano szybkość usuwania pojedynczych elementów. Jak widać z rysunku 5 lepsze wyniki osiągnęła aplikacja napisana we frameworku AngularJS 2.

W drugiej części przeprowadzono analogiczny test, jednak liczba usuniętych elementów na stronie wynosiła 10000 sztuk. W tym przypadku lepsze wyniki osiągnęła aplikacja napisana w szkielecie VueJS.



Rys. 5. Średnie czasy usuwania obiektów

Czwartym testowanym elementem był czas reakcji na zdarzenie kliknięcia w elementy aplikacji. Eksperyment został zrealizowany przy zastosowaniu dużej liczby elementów na stronie. Jak widać z rysunku 6 aplikacja napisana na szkielecie AngularJS 2 osiągnęła ponad dwa razy lepszy wynik wydajności niż aplikacja oparta na szkielecie VueJS.



Rys. 6. Średnie czasy reakcji na kliknięcie na obiekcie ekranowym

Framework AngularJS 2 okazał się lepszy w przypadku: aktualizacji danych i czasie reakcji po naciśnięciu w elementy aplikacji, a framework VueJS w sytuacji tworzenia obiektów na stronie i usuwania elementów.

5. Wyniki eksperymentu analizy metryki kodu

Eksperyment badania liczby linii kodu obydwu aplikacji został zrealizowany za pomocą NodeJS wtyczki cloc - pozwalającej przeskanować projekt i przedstawić wyniki metryki kodu. W tabele 1 przedstawione zostały wyniki dla obu frameworków.

Tabela 1. LOC aplikacji testowych

Typ pliku	**VueJS**	**Angular JS 2**
JavaScript	140	416
TypeScript	-	206
Vuejs Component	137	-
JSON	32	61
HTML	12	55
Razem	**321**	**738**

Z wyników przeprowadzonego testu (tab. 1 i tab. 2) można wyciągnąć wnioski, że aplikacje napisane z wykorzystaniem szkieletu AngularJS 2 są bardziej pracochłonne niż aplikacje zrealizowane za pomocą szkieletu VueJS.

6. Wnioski

Na podstawie przeprowadzonych badań zostały wyciągnięte wnioski dotyczące efektywności pracy i wytwarzania aplikacji wykorzystujących szkielety AngularJS 2 i VueJS.

Pierwszym istotnym parametrem, na który trzeba zwrócić uwagę to jest analiza wartości metryki kodu (w tym przypadku LOC). Z jej wyników można wykryć, że aplikacje napisane na szkielecie VueJS są mniej pracochłonne niż aplikacje zrealizowane przy pomocy szkieletu AngularJS 2.

Na podstawie przeprowadzonych testów wydajności można zaobserwować, że szkielet AngularJS 2 lepiej radzi sobie w przypadku aktualizacji danych i reakcji na zdarzenie naciśnięcia w elementy programu. Z kolei VueJS poradził sobie znacznie lepiej w przypadku tworzenia nowych elementów na stronie i ich usuwania, co jest kryterium dużo istotniejszym, niż lekkość samego szkieletu.

Początkowo zakładano, że szkielet AngularJS 2 osiąga lepsze wyniki w przypadku przetwarzania danych w aplikacji internetowej niż VueJS (Hipoteza 1) i że aplikacje napisane na szkielecie AngularJS 2 są bardziej czasochłonne niż aplikacje zrealizowane za pomocą szkieletu VueJS (Hipoteza 2). Na podstawie przeprowadzonych badań otrzymano wyniki dementujące prawdziwość hipotezy pierwszej i potwierdzające słuszność hipotezy drugiej.

Literatura

[1] Mariano C. L.: Benchmarking JavaScript Frameworks. Masters dissertation, 2017. doi:10.21427/D72890

[2] Cristine Felt: Which Are The Best Programming Languages to Learn in 2017 , https://www.ibm.com/developerworks/community/blogs/d13a8e32-0870-49cb-9b0dba0e34fa6561/entry/Which_Are_The_Best_Programming_Languages_to_Learn_in_2017?lang=en

[3] Noon H.: JavaScript Frameworks in 2017, https://hackernoon.com/5-best-javascript-frameworks-in-2017-7a63b3870282

[4] Программный код и его метрики: https://habrahabr.ru/company/intel/blog/106082 [28.04.2017]

[5] Метрики сложности кода: http://www.ispras.ru/preprints/docs/prep_25_2013.pdf [01.05.2017]

[6] Source lines of code: https://en.wikipedia.org/wiki/Source_lines_of_code [07.06.2017]

[7] Oviedo E.J.: Control flow, data flow and program complexity, Chicago, 1980.

[8] Rentrop J.: Software Metrics as Benchmarks for Source Code Quality of Software Systems, Amsterdam, 2006