JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 8 (2018) 226-229
Wysłane: 2018-01-24
Przyjęte: 2018-01-30

Efektywność tworzenia warstwy prezentacji aplikacji we frameworkach AngularJS, Angular2, BackboneJS

Monika Tobiańska*, Jakub Smołka

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

Streszczenie. Niniejszy artykuł poświęcony jest analizie porównawczej trzech frameworków służących do tworzenia warstwy prezentacji aplikacji. Przeprowadzone zostały trzy rodzaje badań na dwóch przeglądarkach, Google Chrome oraz Mozilla Firefox. Wzięto pod uwagę złożoność kodu, szybkość generowania widoku, płynność działania aplikacji przy obciążeniu danymi, ilość przesyłanych danych potrzebnych do uruchomienia aplikacji oraz zużycie pamięci zajmowanej przez program w zależności od liczby elementów na liście. Do pomiarów wykorzystano aplikacje TodoMVC napisane przy użyciu narzędzi: AngularJS, Angular2 i BackboneJS. Przeprowadzone eksperymenty wykazały, że framework Angular2 w przeglądarce Google Chrome uzyskał najlepszy wynik. BackboneJS natomiast był faworytem dla przeglądarki Mozilla Firefox.

Słowa kluczowe: AngularJS; Angular2; BackboneJS; Wydajność, Framework frontendowy

*Autor do korespondencji.

Adres e-mail: monika.tobianska@gmail.com

Efficiency of creating application's presentation layer with frameworks AngularJS, Angular2, BackboneJS

Monika Tobiańska*, Jakub Smołka

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

Abstract. This article is focused on comparative analysis of three frameworks to create presentation layer of application. Three analysis have been conducted on two browsers, Google Chrome and Mozilla Firefox. Code complexity, the speed of view rendering, the smoothness of working of overstretched application, the amount of sent data needed to start application and the amount of memory used by application depending on the number of elements on list was took into consideration. In research purposes TodoMVC applications made with AngularJS, Angular2 and BackboneJS were used. Conducted experiments shown that Angular2 in Google Chrome achieve the best result. BackboneJS was the winner in Mozilla Firefox.

Keywords: AngularJS; Angular2; BackboneJS; Performance, Frontend framework

*Corresponding author.

E-mail address: monika.tobianska@gmail.com

1. Wstęp

Obecnie Internet stał się najpopularniejszym źródłem informacji. Strony internetowe wypierane są przez bardziej funkcjonalne programy komputerowe dostępne przez przeglądarki internetowe.

Frameworki określają strukturę aplikacji, jej mechanizm działania oraz dostarczają biblioteki i komponenty przydatne podczas tworzenia programów. Od kilku lat na rynku dostępne są nowe szkielety do budowy aplikacji. Głównym celem twórców jest dostosowanie ich narzędzi do różnego rodzaju systemów. Co więcej chcą także sprostać wymaganiom programistów oraz wyeliminować niedogodności napotkane podczas korzystania z poprzednich wersji frameworków.

Wielu programistów korzysta z frameworków dlatego znalezienie literatury dotyczącej omawianych narzędzi nie było trudne.

W [1] porównany został AngularJS, EmberJS i BackboneJS. Autor stwierdził, że dwa pierwsze frameworki są podobne pod względem ilości dostarczanych funkcjonalności. Dwa ostatnie natomiast w zbliżony sposób obsługują szablony widoków.

W [2,3] zaprezentowano różnice między AngularJS a Angular2. Porównano sposób tworzenia i używania kontrolerów oraz konfigurację aplikacji. Te same technologie omówione zostały w [4]. Z artykułu wynika, że AngularJS nie został stworzony, aby być wydajnym rozwiązaniem, lecz aby łatwo się go używało. W pewnym momencie efektywność zaczęła odgrywać ważną rolę dlatego wydano Angular2, który zwiększa szybkości działania aplikacji 3-10 krotnie..

Jeden z pracowników Juniper Networks napisał artykuł zestawiający ze sobą frameworki AngularJS, Angular2, ReactJS i BackboneJS z MarionetteJS [5]. Przeprowadzona analiza pozwoliła mu na pokazanie wyższości Angular2 nad innymi narzędziami.

Ciekawe porównanie przedstawiono w [6]. Artykuł dotyczący EmberJS, AngularJS oraz BackboneJS przedstawia aspekty, na które należy zwrócić uwagę podczas wyboru narzędzia. Jednym z najważniejszych aspektów okazała się społeczność, im większa tym łatwiej znaleźć odpowiedź

na pojawiające się pytania. Równie ważny był czas ładowania strony ponieważ użytkownikom zależy na jak najszybszym znalezieniu interesujących ich treści.

Celem opisanej w niniejszym artykule pracy badawczej było stworzenie porównania wybranych frameworków do tworzenia warstwy prezentacji aplikacji webowej. Porównanie to ma ułatwić wybór narzędzia, z którego programiści będą korzystać przy pisaniu systemów. Ponadto zestawienie takie będzie uzupełnieniem informacji dostępnych w Internecie oraz książkach, ponieważ wiedza taka nie jest zgromadzona w jednym miejscu.

Zakres badań obejmował opisanie obiektów badań, czyli narzędzi AngularJS, Angular2 oraz BackboneJS. Następnie przeprowadzone zostały analizy złożoności kodu, szybkości generowania widoku, płynności działania aplikacji obciążonej danymi, ilość przesyłanych danych potrzebnych do uruchomienia aplikacji oraz zużycie pamięci zajmowanej przez program. Po zestawieniu wyników wytypowano najlepsze narzędzie.

2. Obiekt badań

Obiektem badań były trzy frameworki frontendowe. AngularJS, Angular2 oraz BackboneJS. Pierwszy został stworzony w celu zmniejszenia nakładów pracy potrzebnych do rozwoju i testowania aplikacji internetowych poprzez wdrożenie wzorca MVC [7]. Drugi jest nowszą wersją AngularJS, wprowadzone zostały ulepszenia i modyfikacje w celu wyeliminowania problemów napotkanych przez programistów. BackboneJS natomiast jest najmniejszym z frameworków. Jego zadaniem jest zwiększenie wydajności aplikacji.

3. Metoda badań

Badania przeprowadzone zostały przy użyciu projektu TodoMVC, który jest zbiorem identycznych aplikacji napisanych w różnych frameworkach do tworzenia warstwy prezentacji. Jako pierwsza przeprowadzona została analiza złożoności kodu. Następnie zmierzono szybkość generowania widoku, płynność działania aplikacji przy obciążeniu danymi, ilość przesyłanych danych potrzebnych do uruchomienia aplikacji oraz zużycie pamięci zajmowanej przez program w zależności od liczby elementów na liście. Zasymulowane zostało obciążenie aplikacji danymi. Lista Todo zawierała kolejno 100, 500 i 1000 elementów. Poniżej znajduje się konfiguracja sprzętowa maszyny, na której wykonano testy:

- Procesor: Intel® Core™ i7-4790K,
- Pamięć: 16GB,
- Dysk: 2000GB + 1000GB,
- Grafika: Intel® HD Graphics 4600, NVIDIA GeForce GTX 750 Ti,
- System operacyjny: Windows 10 Pro 64-bit.

Wszystkie przypadki pomiaru czasu i zużycia pamięci rozpatrzone zostały przy użyciu przeglądarek:

- Google Chrome, wersja 62.0.3202.94 (Oficjalna wersja) (64-bitowa),
- Mozilla Firefox, wersja 57.0.3 (64-bitowa).

Pomiary wykonano przy użyciu narzędzi programistycznych wymienionych w kolejnym punkcie.

3.1. Badane parametry

- **Złożoność kodu** – zestawienie liczby linii jakie zostały napisane do działania programu. Pominięto puste linie, pliki ze stylami, plik ze zdefiniowanymi bibliotekami, folder z narzędziami do budowania oraz folder zawierający testy aplikacji.

- **Szybkość generowania widoku** – czas ładowania obiektowego modelu dokumentu. Pomiary wykonano przy użyciu dodatków doinstalowanych do przeglądarek. W przypadku Google Chrome był to *Page load time*, Mozilla Firefox natomiast *app.telemetry Page Speed Monitor*.

- **Płynność działania przy obciążeniu danymi** – zmiana czasu ładowania obiektowego modelu dokumentu w zależności od liczby elementów na liście Todo.

- **Czas uruchomienia programu** - czas uruchomienia aplikacji od momentu przeładowania strony, do pojawienia się listy.

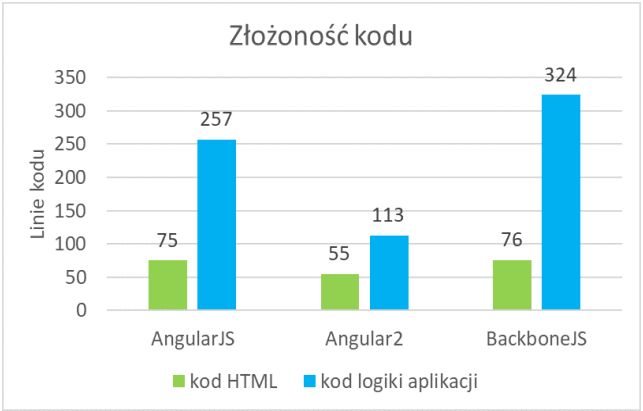
- **Ilość przesyłanych danych** – ilość danych jaka jest potrzebna do uruchomienia aplikacji. Wyniki uzyskano dzięki narzędziu *Network* przeglądarki internetowej.

- **Zużycie zasobów** – ile pamięci zajmowała aplikacja w zależności od obciążenia danymi. W tym przypadku wykorzystano narzędzie *Memory* przeglądarki.

4. Wyniki badań

4.1. Złożoność kodu

Na rysunku 1 zestawiono liczbę linii kodu dla aplikacji napisanej w trzech frameworkach.



Rys. 1. Zestawienie złożoności kodu wybranych frameworków

W tym przypadku faworytem został Angular2. Zawiera on bardzo dużo gotowych funkcjonalności, które ułatwiają i przyspieszają proces tworzenia systemów. Co więcej powoduje to redukcję liczby linii kodu.

4.2. Szybkość generowania widoku oraz płynność działania przy obciążeniu danymi

W tabeli 1 zestawiono średnie czasy generowania widoku dla dwóch przeglądarek przy zmiennym obciążeniu danymi.

Tabela 1. Średnie czasy generowania widoku dla wszystkich pomiarów

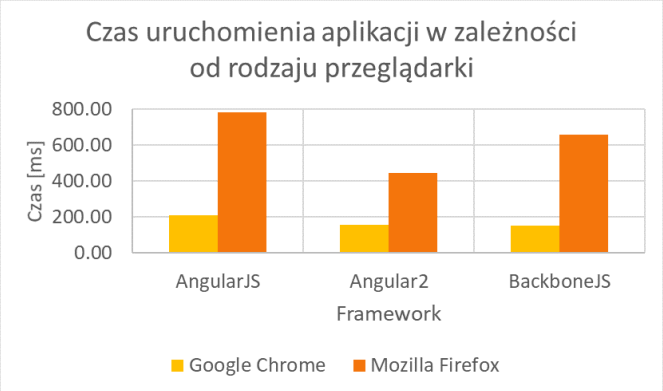
	Liczba elementów	Google Chrome	Mozilla Firefox
AngularJS	100	86,01ms	106,13ms
	500	87,77ms	111,65ms
	1000	92,04ms	119,71ms
Angular2	100	37,38ms	52,04ms
	500	44,73ms	63,02ms
	1000	51,7ms	81,26ms
BackboneJS	100	44,16ms	61,85ms
	500	50,25ms	69,08ms
	1000	62,75ms	99,9ms

Bez względu na rodzaj przeglądarki najdłuższy czas uzyskał AngularJS. Spowodowane jest to sposobem w jaki zaimplementowane zostało dwukierunkowe wiązanie danych w tym frameworku. Najszybszy okazał się Angular2, jednak BackboneJS osiągnął zbliżone rezultaty. Obydwa narzędzia nastawione zostały przez ich twórców na szybkość działania.

Zauważalna jest także duża rozbieżność pomiędzy Google Chrome a Mozilla Firefox. Pierwsza z przeglądarek o wiele szybciej generowała obiektowy model danych. Jednak obciążenie danymi nie miało dużego wpływu na działanie aplikacji.

4.3. Czas uruchomienia aplikacji

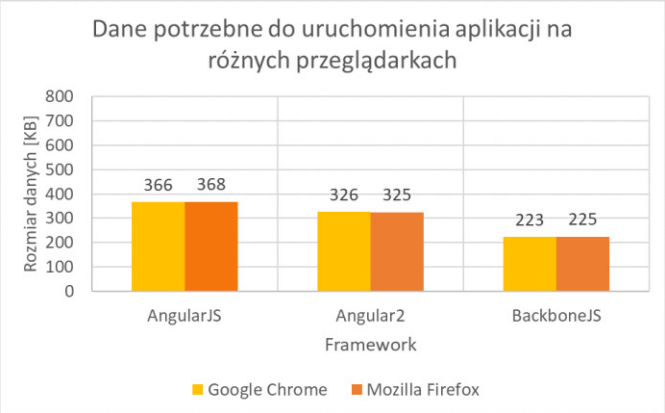
Poniżej zestawiono średnie czasy uruchomienia aplikacji w zależności od rodzaju przeglądarki (Rys. 2). Z łatwością da się także zauważyć różnice między programami napisanymi przy użyciu tych samych narzędzi. Największą odnotowano w przypadku aplikacji korzystających z AngularJS oraz BackboneJS. W przeglądarce Google Chrome programy uruchamiały się prawie cztery razy szybciej niż w Mozilla Firefox. Najmniej odbiegają od siebie wyniki jakie uzyskał Angular2. Framework ten okazał się trzykrotnie szybszy w Google Chrome.



Rys. 2. Czas uruchomienia aplikacji w zależności od rodzaju przeglądarki

4.4. Ilość przesyłanych danych

Rysunek 3 pokazuje jaka ilość danych musiała zostać przesłana w celu uruchomienia aplikacji.



Rys. 3. Dane potrzebne do uruchomienia aplikacji w przeglądarce

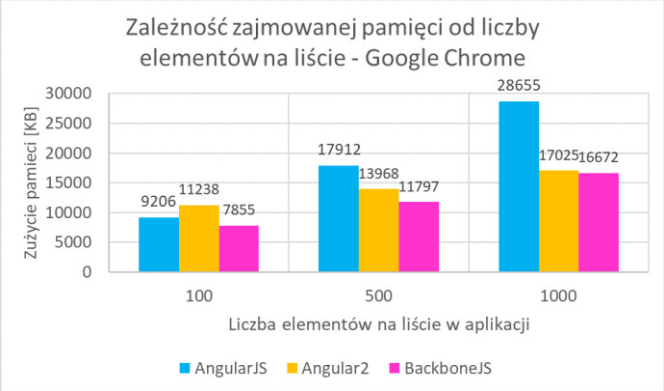
Niezależnie od przeglądarki wyniki były praktycznie identyczne. Nieznaczne różnice wynikać mogą z odmiennego sposobu interpretacji styli CSS, dodatków w aplikacji oraz obrazków przez Google Chrome i Mozilla Firefox.

Aplikacja napisana przy użyciu BackboneJS mimo dużej liczby dodatków jakie są potrzebne do działania systemu, wymagała najmniej danych. Spowodowane jest to niewielkim rozmiarem frameworka. Angular2 dzięki funkcjonalnościom dostarczanym z narzędziem nie wymaga tylu danych co AngularJS.

Twórcy narzędzia Angular2 długo pracowali nad zmniejszeniem liczby ładowanych plików, wersja *Candidate* dla projektu typu „Hello world" pobierała aż 594 kilobajty danych [8].

4.5. Zużycie zasobów

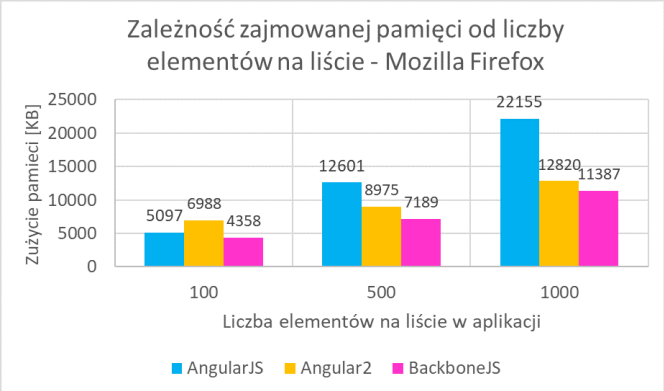
W tym podrozdziale zestawione zostało zużycie pamięci zajmowanej przez aplikacje w zależności od liczby elementów na liście. Wyniki zestawiono na poniższych wykresach (Rys. 4 -5).



Rys. 4. Zależność zajmowanej pamięci od liczby elementów na liście - Google Chrome

W przypadku Google Chrome AngularJS dla stu elementowej listy uzyskał drugi wynik. W kolejnych pomiarach wraz ze wzrostem liczby zadań na liście w aplikacji następował gwałtowny wzrost zajmowanej pamięci.
Dla Angular2 wraz ze wzrostem obciążenia danymi, parametr ten poprawia się. W pierwszym pomiarze framework ten miał najgorszy wynik, dla tysiąca elementów natomiast prawie zrównał się z faworytem.

BackboneJS okazał się najlepszy.



Rys. 5. Zależność zajmowanej pamięci od liczby elementów na liście - Mozilla Firefox

W przeglądarce Mozilla Firefox AngularJS również z drugiego miejsca spadł na ostatnią pozycję.
Angular2 po najgorszym wyniku dla stu elementowej listy uzyskał drugi wynik i do najlepszego narzędzia zabrakło mu jedynie 1,5MB, chociaż wartość ta w przypadku Google Chrome była znacznie mniejsza.
Framework BackboneJS bez względu na liczbę elementów na liście w aplikacji zużywał najmniej pamięci przeglądarki.
Mozilla Firefox uzyskała mniejsze wartości niż Google Chrome, ponieważ każda przeglądarka w inny sposób interpretuje aplikacje i bierze pod uwagę różne elementy.

5. Wnioski

Na podstawie przeprowadzonych analiz można stwierdzić, że Angular2 i BackboneJS uzyskały najlepsze rezultaty.

Pierwszy z nich dostarcza najwięcej wbudowanych funkcjonalności. Umożliwia to szybsze i łatwiejsze pisanie skomplikowanych programów przy użyciu mniejszej ilości kodu. Dzięki temu był w stanie osiągnąć najlepsze czasy generowania obiektowego modelu danych niezależnie od obciążenia aplikacji danymi.
Wielkość narzędzia spowodowała, że framework zajmował najwięcej pamięci w przeglądarce dla stu elementowej listy. Jednak posiada on modularną strukturę, co powoduje, że ładowane są tylko te partie aplikacji, które są potrzebne. Przy przejściu do innych części systemu następuje pobranie kolejnych elementów [9].

Drugi faworyt dzięki swoim niewielkim rozmiarom zajmował najmniej pamięci aplikacji oraz wymagał najmniejszej liczby danych do uruchomienia programu. Co więcej narzędzie osiągnęło bardzo dobry czas uruchomienia aplikacji, ponieważ wspiera mechanizm szablonów, który może być wcześniej ładowany na serwerze. Zwiększa to szybkość strony zwłaszcza na urządzeniach z niską mocą obliczeniową [10].

AngularJS okazał się najsłabszym frameworkiem. Przez sposób w jaki zaimplementowano dwukierunkowe wiązanie danych miał najgorsze czasy generowania widoku. Nie obsługuje on leniwego ładowania danych, co spowalnia proces uruchamiania. Ponadto narzędzie przechowuje dodatkową kopię każdego obiektu, dlatego wraz ze wzrostem obciążenia aplikacji danymi zwiększało się zużycie pamięci.

Aplikacje uruchamiane w przeglądarce Google Chrome działały dużo szybciej. Zaobserwowano mniejsze opóźnienia w generowaniu widoków i uruchamianiu aplikacji niż w przypadku Mozilla Firefox.

Literatura

[1] B. Dybowski, Angular, Backbone czy Ember? Wybieramy framework JavaScript, 20.07.2014, https://www.nafrontendzie.pl/ [30.09.2017].

[2] A. Moorthy, Angular vs AngularJS 2: A detailed comparison, 11.07.2016, https://www.cubettech.com/ [01.10.2017].

[3] M. Fakiolas, I have a new project, should I use AngularJS 1.x or Angular2?, 18.01.2016, https://angularjs-recipes.com/ [01.10.2017].

[4] D. Rathore, Angularjs vs Angular2 | what's the difference ?, 13.08.2016, https://www.dunebook.com [1.10.2017].

[5] F. He, Angular2, next generation UI solutions. 02.04.2017, https://www.linkedin.com/pulse [1.10.2017].

[6] S. Karn, Difference : AngularJS vs. Backbone.js vs. Ember.js, 08.06.2016, , https://www.linkedin.com/pulse [1.10.2017].

[7] A. Kalbarczyk, D. Kalbarczyk, AngularJS. Pierwsze krok, Helion, Gliwice 2015, 15.

[8] M. Miszczyszyn, Wstęp do Angular 2, 03.06.2016, https://typeofweb.com/ [2.11.2017].

[9] Praca zbiorowa: Rangle's Angular Training Book, GitBook, 2017, 297-300.

[10] https://versus.com/pl/angularjs-vs-backbone-js [2.12.2017].