JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 14 (2020) 59-64
Wysłane: 2019-10-20
Przyjęte: 2019-10-29

Analiza porównawcza wydajności frameworków Angular oraz Vue.js

Roman Baida*, Maksym Andriienko*, Małgorzata Plechawska-Wójcik
Politechnika Lubelska, Katedra Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Celem artykułu jest porównanie wydajności popularnych frameworków JavaScriptowych Angular i Vue.js w kontekście tworzenia gier oraz wybór lepszego z nich. Kryteria porównawcze są następujące: czas wymiany danych z serwerem oraz renderowania różnych komponentów aplikacji, ilość zajmowanej pamięci podczas odświeżania informacji o przebiegu gry i przywróceniu użytkownika do bieżącej gry, stopień obciążenia przeglądarki oraz rozmiar plików końcowych. Na podstawie wyników z przeprowadzonych badań można stwierdzić, że bardziej wydajny jest framework Vue.js.

Słowa kluczowe: Angular; Vue; analiza porównawcza; gra przeglądarkowa

*Autorzy do korespondencji.

Adresy e-mail: r.baida@pollub.edu.pl, maksym.andriienko@pollub.edu.pl

Perfomance analysis of frameworks Angular and Vue.js

Roman Baida*, Maksym Andriienko*, Małgorzata Plechawska-Wójcik
Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The aim of the paper is to compare the performance of popular JavaScript frameworks Angular and Vue.js in the context of game development. The comparative criteria are as follows: time of data exchanging with server and rendering of various application components, memory consumption during refreshing the current game information and restoring the user to the current game, browser load level and size of the final application files. The test results show that the Vue.js framework is more efficient.

Keywords: Angular; Vue; performance analysis; browser game

*Corresponding authors.

E-mail addresses: r.baida@pollub.edu.pl, maksym.andriienko@pollub.edu.pl

1. Wstęp

W dzisiejszych czasach aplikacje oraz strony internetowe rzadko są tworzone bez zastosowania frameworków, a tylko i wyłącznie za pomocą zwykłych języków programowania, takich jak JavaScript, PHP, CSS. Wraz z rozwojem technologii rozwijają się także strony internetowe. Strony statyczne zostały zastąpione stronami dynamicznymi z dużą liczbą elementów interaktywnych, które mogą się zmieniać wielokrotnie w bardzo krótkim czasie. Do tworzenia takich stron potrzebne są frameworki [1], które pomagają programistom poradzić sobie ze złożonością takich projektów. Istnieje wiele różnych frameworków JavaScriptowych, ale na potrzeby niniejszego artykułu zostały porównane tylko dwa z nich, Angular i Vue.js. Angular, który został utworzony przez firmę Google, jest jednym z najpopularniejszych frameworków JavaScriptowych [2]. Każdego roku framework Vue.js staje się coraz bardziej popularny. Został on utworzony został przez Evana You [3], który w przeszłości pracował w Google, wykorzystując AngularJS. Vue.js powstał jako framework, który zawiera w sobie najlepsze cechy frameworka Angular, skupując się na prostocie pisania kodu i małej ilości zajmowanego miejsca plików wykonywalnych na dysku.

Celem badania jest analiza wydajności frameworków JavaScriptowych Angular oraz Vue.js. Porównanie wydajności frameworków będzie polegać na porównaniu średniego czasu wysyłania żądań do serwera i renderowania komponentów stron oraz ilości zajmowanej pamięci podczas wykonywania tych działań. W tym celu stworzona została aplikacja internetowa implementująca grę planszową Munchkin. Na podstawie wyników badań zostanie wykonana analiza porównawcza tych frameworków.

Postawiono następującą tezę:

Framework Vue.js jest lepszym środowiskiem programistycznym do budowy aplikacji internetowych dla początkującego programisty niż framework Angular ze względu na łatwość nauczenia się go oraz wydajność przy założeniu, że tworzona aplikacja internetowa ma średni poziom złożoności.

2. Przegląd literatury

Po przeanalizowaniu dostępnych materiałów można wywnioskować, że większość prac związanych z frameworkami Angular i Vue.js polega na analizie technicznej tych frameworków lub ma za cel charakterystykę tych frameworków.

W artykule *Speed Performance Comparison of JavaScript MVC Frameworks* [4] autor przedstawił analizę wydajności ośmiu frameworków. Przeprowadzone badania polegały na

mierzeniu czasu tworzenia, modyfikowania i usuwania elementów HTML na stronie internetowej. Porównanie wyników tych badań pokazuje, który z frameworków jest najszybszy. Porównując średnie wyniki czasowe, Vue.js jest szybszy od Angular tylko w przypadku usuwania elementów HTML. We wnioskach autor stwierdza, że Angular posiada najlepsze wyniki i zajmuje pierwsze miejsce w prawie wszystkich przeprowadzonych testach. Vue jest jednym z najszybszych frameworków w przypadku tworzenia i modyfikowania elementów, chociaż jest i tak wolniejszy od Angular w wykonywaniu tego typu operacji.

Głównym celem artykułu *Sitecore JavaScript Services Framework Comparison* [5] jest określenie frameworku, który najlepiej nadawałby się do stworzenia serwisu w systemie zarządzania treścią Sitcore. Do wyznaczenia najodpowiedniejszej technologii, autor dokonuje porównania takich metryk jak liczba linii kodu źródłowego, ilość miejsca na dysku zajmowana przez pliki wykonywalne, złożoność pisania kodu, ograniczenia architektury wprowadzane przez framework i wydajność. Na podstawie tych badań wysunięto wniosek, że w celu stworzenia tego typu aplikacji, najlepiej użyć biblioteki React. Autor zauważa również, że framework Vue.js ma ogromne perspektywy na pokonanie React, ze względu na niski próg wejścia, minimalną konfigurację podstawową oraz łatwość rozszerzenia architektury kodu. W przypadku złożonego i bogatego w funkcjonalność projektu Angular jest głównym konkurentem React ze względu na elastyczność kodu i szybkość renderowania stron.

W artykule *Comparison of front-end frameworks for web applications development* [6] dokonano porównania frameworków Angular, Vue.js i React w celu wyznaczenia, który z nich najbardziej nadaje się do tworzenia aplikacji typu SPA (*ang. Single Page Application*), który do tworzenia aplikacji typu MPA (*ang. Multi Page Application*) lub do tworzenia obu rodzajów aplikacji. Przy porównaniu wybranych frameworków, autor porusza takie zagadnienia, jak możliwość generowania strony po stronie serwera, możliwość napisania i zarządzania dużą ilością złożonego kodu oraz możliwości frameworka w przypadku kompresji i minimalizacji plików źródłowych. W wyniku przeprowadzonych badań autor wywnioskuje, że w celu tworzenia MPA najbardziej nadaje się Vue.js, zaś Angular jest najgorszym wyborem ze względu na jego rozmiar i złożoność, jak i również jego nieprzystosowanie do sterowania małymi częściami struktury DOM. W przypadku SPA Angular, zdaniem autora, jest najlepszym wyborem. Na końcu autor stwierdza, że Vue.js jest dobrym wyborem zarówno w przypadku tworzenia SPA, jak i MPA.

Po przeanalizowaniu publikacji naukowych dotyczących tematu porównania frameworków JavaSctiptowych można stwierdzić, że większość z badań przeprowadzonych w tych pracach polega na porównaniu czasu wykonania prostych operacji po stronie klienta. Nie udało się znaleźć prac naukowych, które dokonywałyby porównania czasów wymiany informacją z serwerem albo, na przykład, renderowania całych stron lub poszczególnych ich elementów.

3. Obiekty badań

3.1. Angular 6

Angular 6 jest nową wersją popularnego frameworka do tworzenia dynamicznych stron internetowych typu SPA. Ten framework jest stworzony i wspierany przez firmę Google. Pierwsza wersja framework'u bazowała na języku JavaScript, a od wersji 2+ bazuje na języku TypeScript [7].

Zaletami szkieletu Angular są:

- łatwość pisania kodu aplikacji,
- dwustronne powiązanie zmiennych kodu TWDP (*ang. Two-way data binding*),
- skalowalność,
- przydatność do pracy z interfejsem Rest API,
- podział na komponenty.

Wadami szkieletu Angular są:

- wyższy próg wejścia ze względu na Observable (RxJS) i Dependency Injection [9],
- wymagane jest poświęcenie dużej ilości czasu na optymalizacje w celu uzyskania szybkiego i poprawnego kodu,
- trudność realizacji dynamicznego tworzenia komponentów,
- duży rozmiar zbudowanego kodu ze względu na zbędne funkcję frameworku,
- jest wspierany przez firmę Google, która gwarantuje częste aktualizację i długi rozwój frameworku.

Angular umożliwia tworzenie dużych i złożonych pod względem logiki biznesowej aplikacji internetowych. Angular po wersji AngularJS został stworzony całkowicie na nowo [8]. W wyniku tego, framework ten zyskał większe możliwości skalowania, a jego kod wykonuje się szybciej w porównaniu ze starszą wersją.

W porównaniu do Vue.js, framework Angular posiada więcej możliwości pracy z interfejsem Rest API. Na przykład, umożliwia dokonanie asynchronicznej modyfikacji danych lub monitorowanie zmiany stanów zmiennych za pomocą Observable (RxJS) lub Subject.

3.2. Vue.js

Vue.js jest frontendową biblioteką, która została napisana w języku JavaScript. Biblioteka ta umożliwia łatwe tworzenie aplikacji internetowych opartych o MVVM (*ang. Model-View View Model*). Każda aplikacja napisana na podstawie Vue.js składa się z komponentów, które zawierają kod JavaScript do wykonania w obrębie komponentu, kod stylów CSS i kod HTML warstwy widoku.

Główne zalety Vue.js to:

- niski próg wejścia,
- wydajność uzyskana dzięki wirutalnemu DOM,
- wysoka elastyczność,

- prosta obsługa,
- minimalne rozmiary zbudowanego kodu,
- wysoka szybkość wykonywanego kodu i renderowania stron aplikacji.

Wadami szkieletu Vue.js są:

- realizacja dwustronnego powiązania zmiennych kodu TWDP wymaga implementacji dodatkowych funkcji *computed* lub *watch*,
- trudność rozszerzenia gotowej aplikacji,
- tworzony głównie przez jednego programistę, przy współpracy z niezależną społecznością programistyczną, co powoduje rzadkie aktualizację oraz nie zapewnia długoterminowego rozwoju frameworku.

Praca z Vue.js rozpoczyna się od stworzenia nowej instancji *new Vue* przy pomocy konstruktora. Umożliwia to obserwowanie zmian, które występują w danym komponencie. Następnie instancja Vue przyjmuje zmienną *template*, z której będzie ona korzystać w celu pobrania widoku do renderowania. W atrybucie *data* są przechowywane wszystkie zmienne komponentu, którego stan śledzi Vue.

Komponenty, które zasadniczo są częściami interfejsu użytkownika, pomagają w rozwinięciu podstawowych elementów HTML i implementowaniu kodu wielokrotnego użytku. Na etapie projektowania aplikacja zostaje podzielona na niezależne części w celu uzyskania drzewiastej struktury komponentów, ale ze względu na to, że kod HTML, CSS i JavaScript znajduje się w jednym pliku, rozszerzenie gotowej aplikacji internetowej jest trudniejsze w porównaniu do Angular.

4. Metoda badań

Porównanie wydajności frameworków będzie polegać na porównaniu średniego czasu wysyłania żądań do serwera oraz renderowania komponentów. Każde badanie zostało przeprowadzone dziesięć razy, a z otrzymanych wyników został obliczony wynik średni. W tym celu została stworzona aplikacja internetowa implementująca grę planszową Munchkin. Aplikacja ta wspiera grę wieloosobową, zarządzanie kontem użytkownika, tworzenie pokojów do gry.

Rozgrywka w grze Munchkin polega na użyciu dwóch talii kart, a cały proces tej gry polega na zbieraniu i wymianie pewnych kart pomiędzy graczami oraz rozegraniu tych kart we właściwych momentach gry. Aplikacja internetowa zawiera wszystkie niezbędne karty, a powiadomienie użytkowników o akcjach z nimi związanych odbywa się za pomocą okien typu pop-up. Dlatego jednymi z najważniejszych badaniami są badania określające czas renderowania kart oraz okien typu pop-up.

Do testowania i porównania możliwości wybranych frameworków zostało wykorzystane narzędzie Google Chrome DevTools [10], które pomaga przeglądać i debugować pliki HTML, CSS oraz witryny JavaScript. Pomaga ono również w sprawdzaniu ruchu sieciowego pobieranego przez witrynę, sprawdzeniu szybkości witryny, analizie wydajności środowiska wykonawczego.

Badania zostały przeprowadzone na komputerze HP EliteBook 840 G3. Komputer ten posiada procesor czterordzeniowy Intel Core i5-6300U o częstotliwości ~2.5GHz oraz ma zainstalowaną pamięć RAM DDR4 8GB. Systemem operacyjnym jest Windows 10 Enterprise 64-bit (10.0). Do badań użyto przeglądarki Google Chrome w wersji 77. Serwery aplikacji zostały uruchomione lokalnie. Część backendowa została uruchomiona za pomocą .Net Core CLI (*ang. Command-Line Interface*), a części frontendowe Angular i Vue.js zostały, odpowiednio, uruchomione przy użyciu Angular CLI oraz Vue CLI.

Kryteria według których zostaną porównane obrane frameworki są następujące:

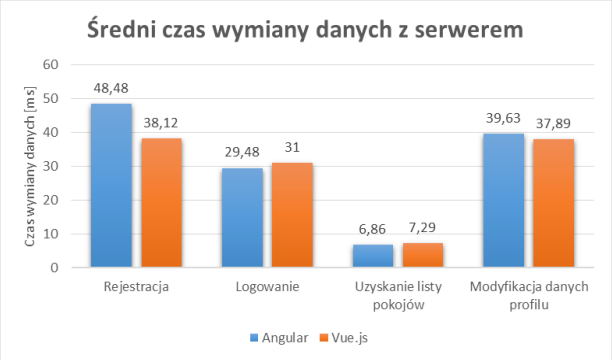
- czas wymiany danych z serwerem,
- czas renderowania komponentów stron,
- czas renderowania kart,
- renderowanie okien typu pop-up
- przywrócenie użytkownika do bieżącej gry,
- odświeżanie informacji o przebiegu gry,
- stopień obciążenia przeglądarek,
- rozmiar plików końcowych.

5. Wyniki badań

5.1. Czas wymiany danych z serwerem

Badanie to zostało przeprowadzone na takich funkcjach aplikacji jak: rejestracja, logowanie, pobieranie listy pokojów dostępnych do gry oraz modyfikacja danych profilu użytkownika. Badane funkcje dokonują wymiany danymi z serwerem, a po otrzymaniu odpowiedzi odświeżają bieżącą stronę lub powiadamiają użytkownika o uzyskanym błędzie. Podczas przeprowadzenia tych badań, liczony był czas od momentu wywołania akcji do wysyłania danych na serwer oraz od momentu odebrania danych do uzyskania reakcji. Czas transferu danych przez sieć nie był uwzględniony.

Na rysunku 1 przedstawiono diagram słupkowy. Oś rzędnych reprezentuje czas w milisekundach, a oś odciętych wskazuje na badaną funkcję. Różnica między średnim czasem wykonywania funkcji dla obu frameworkach jest dość mała. Minimalna średnia różnica wynosi 0,5 milisekundy dla komponentu listy dostępnych pokojów do gry, natomiast maksymalna stanowi 10 milisekund dla komponentu rejestracji. Spowodowane to jest tym, że do komunikacji sieciowej, Angular korzysta ze swojego modułu HttpClient, a Vue.js korzysta z modułu Axios, przy czym HttpClient jest stworzony na bazie Axios. Z tego powodu wyniki są bardzo podobne. Z wyników badań można wywnioskować, że główna różnica polega na czasie wykonania żądań typu GET, gdzie Angular jest wolniejszy.

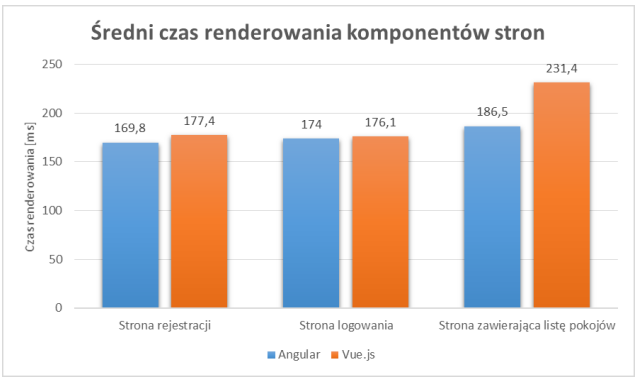


Rys. 1. Diagram pokazujący czas średni wywołania różnych funkcji wybranymi frameworkami

5.2. Czas renderowania komponentów stron

W tym badaniu będą porównywane identyczne funkcje napisane w Angular i Vue.js. To badanie potrzebne jest do porównywania czasów renderowania oddzielnych komponentów aplikacji internetowej. Zostało ono przeprowadzone na następujących komponentach aplikacji: komponent logowania, komponent rejestracji, komponent w którym odbywa się przebieg gry.

Na rysunku 2 przedstawiono diagram słupkowy. Oś rzędnych reprezentuje czas w milisekundach, a oś odciętych wskazuje na badaną funkcję. Różnica między średnim czasem renderowania stron jest bardzo mała. Na podstawie wykresu można wywnioskować, że Vue.js ma dłuższy czas renderowania komponentu zawierającego listę dostępnych pokoi dla gry w porównaniu do Angular. Ta różnica jest spowodowana tym, że Vue.js tworzy instancje do monitorowania bieżącego stanu zmiennej zawierającej listę, co powoduje wydłużenie czasu renderowania. Żeby zwiększyć szybkość renderowania listy pokojów do gry trzeba użyć specjalnego modyfikatora *v-once*, która nie pozwala ustawiać dla zmiennej listy instancję monitorowania. W takim przypadku lista zostanie wyrenderowana jednokrotnie.

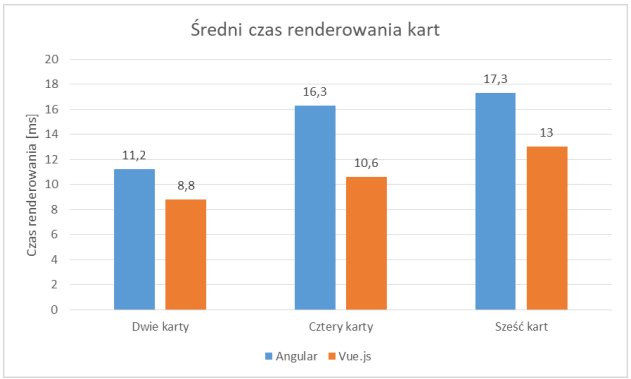


Rys. 2. Diagram pokazujący czas średni renderowania stron wybranymi frameworkami

5.3 Czas renderowania kart

Wyniki tego badania pokazują, w jakim stopniu renderowanie kart obciąża zasoby przeglądarki. Karty są wyświetlane w przypadku ich wyciągnięcia z talii, gdy użytkownik chce zobaczyć karty, które posiada inny gracz lub podczas innych akcji. Jest to bardzo istotne badanie, które pokaże, kompilator którego frameworku jest lepszy pod względem optymalizacji zasobów używanych aplikacją.

Rysunek 3 przedstawia diagram słupkowy. Oś rzędnych reprezentuje czas w milisekundach, a oś odciętych wskazuje na liczbę kart renderowanych w każdym badaniu. Z wyników badań średniego czasu renderowania kart do gry można wywnioskować, że kompilator optymalizacji plików dodatkowych oraz zdjęć jest lepszy w frameworku Vue.js, gdzie dla aplikacji testowej średni czas jest o ~4,2 milisekundy krótszy od czasu renderowania kart w Angular.



Rys. 3. Średni czas renderowania różnej liczby kart

5.4. Renderowanie okien typu pop-up

Ponieważ prawie każda aplikacja internetowa implementuje okna typu pop-up, badanie to jest jednym z najważniejszych w niniejszym artykule. Pozwoli ono wyznaczyć ilość zajmowanej pamięci oraz czas potrzebny do renderowania takich okien. Na potrzeby artykułu zostało przebadane okno, które powstaje podczas akcji sprzedaży niepotrzebnych kart. Analiza wyników tego badania pokaże, który framework lepiej radzi z renderowaniem okien typu pop-up.

Okna typu pop-up stanowią ważną część aplikacji internetowej, ponieważ wyświetlają one ważną informację o przebiegu gry. W zakresie jednej rundy składającej się z czterech etapów, okno typu pop-up pojawia się co najmniej dwa lub trzy razy, a w niektórych przypadkach ich liczba wzrasta wielokrotnie. Dlatego ważne jest to, ile czasu zajmuje renderowanie jednego takiego okna oraz ile pamięci zostaje wykorzystane przy jego pojawieniu się.

W tabeli 1 znajdują się wyniki badań określających średni czas renderowania okien typu pop-up.

Tabela 1. Czas renderowania okien typu pop-up w milisekundach

Framework	Angular	Vue.js
Wartość średnia	55,5	58,4

Tabela 2 pokazuje użytą pamięć w kilobajtach. Ilość pamięci używanej frameworkiem Vue.js jest o trzy razy mniejsza w porównaniu do Angular.

Tabela 2. Ilość zajmowanej pamięci podczas renderowania okien typu pop-up w kilobajtach

Framework	Angular	Vue.js
Wartość średnia	179,9	50,69

Z wyników badań średniego czasu renderowania i wywołania okien typu pop-up wynika, że czas średni jest bardzo podobny i nie ma dużej różnicy, ale przy badaniu pamięci potrzebnej do renderowania widoczne jest to, że Vue.js potrzebuje faktycznie 3 razy mniej pamięci niż Angular. Jest to związane z tym, że we frameworku Vue.js został zaimplementowany lepszy sposób modyfikacji DOM strony, w odróżnieniu do Angular.

5.5. Przywrócenie użytkownika do bieżącej gry

Badanie to umożliwia wyznaczenie czasu potrzebnego na powrót do aktualnego stanu gry użytkownikowi, który przez przypadek zamknął okienko z grą, stracił połączenie z serwerem lub odświeżył stronę. Wtedy, po przejściu na stronę z listą pokojów, użytkownik automatycznie zostaje przywrócony do gry, w której on uczestniczy, a informacja o grze aktualizuje się do aktualnego stanu. Wyniki badania posłużą do określenia, który framework szybciej podłącza się do WebSocket, pobiera dużą ilość danych od serwera oraz renderuje niezbędne komponenty stron.

Tabela 3. Czas przywrócenia użytkownika do bieżącej gry

Framework	Angular	Vue.js
Wartość średnia	88,7	51,3

Tabela 4. Ilość zajmowanej pamięci podczas przywrócenia użytkownika do bieżącej gry w kilobajtach

Framework	Angular	Vue.js
Wartość średnia	53,45	51,55

Jak widać z tabel wyżej (tab. 3 oraz tab. 4), pamięć potrzebna do przywrócenia użytkownika do bieżącej gry w Angular i we Vue.JS jest faktycznie jednakowa, ale czas potrzebny na to działanie w przypadku Angular jest o 50% dłuższy. Jest to przede wszystkim związane z pracą modułu odpowiadającego za dwustronnie przesyłanie danych w zmiennej, gdzie w Angular dana funkcja jest zbyt rozbudowana.

5.6. Odświeżanie informacji o przebiegu gry

To badanie umożliwia wyznaczenie lepszego frameworku pod względem pobierania informacji z serwera i wyrenderowania niezbędnych danych. Na końcu każdej tury oraz podczas wykonywania różnych akcji, serwer wysyła każdemu graczowi informację o aktualnym przebiegu gry w czasie rzeczywistym. Odświeża się bieżący etap rundy, informacja o graczach oraz ich kartach.

Tabela 5. Czas odświeżania informacji o przebiegu gry

Framework	Angular	Vue.js
Wartość średnia	47	31,7

Tabela 6. Ilość zajmowanej pamięci podczas odświeżania informacji o przebiegu gry w kilobajtach

Framework	Angular	Vue.js
Wartość średnia	43,85	43,32

Wyniki tego badania odpowiadają wynikom poprzedniego badania "Przywrócenie użytkownika do bieżącej gry". Z tych tabel (tab. 5 oraz tab. 6) można wywnioskować, że pamięć potrzebna do przywrócenia użytkownika do bieżącej gry w Angular i we Vue.JS jest faktycznie jednakowa, ale czas potrzebny na to działanie w przypadku Angular jest dwukrotnie dłuższy. Jest to głównie związane z pracą modułu odpowiadającego za dwustronnie przesyłanie danych w zmiennej, gdzie w Angular dana funkcja jest zbyt rozbudowana.

5.7. Stopień obciążenia przeglądarek

Badanie to pozwala zweryfikować, ile pamięci wykorzystuje aplikacja. Pamięć podręczna przeglądarki była czyszczona przed każdą kolejną próbą. Potrzebne to jest do określenia frameworku, który obciąża przeglądarkę w mniejszym stopniu. Stopień obciążenia przeglądarek był mierzony podczas uruchomienia aplikacji oraz w przeciągu jednej godziny z próbami co 30 minut. Po skończeniu badań nie wykryto dużych różnic w obciążeniu zależnych od czasu życia aplikacji.

Kod aplikacji i wszystkie pliki graficzne pobierane są przy pierwszym uruchomieniu aplikacji w celu zwiększenia jej wydajności. Dlatego stopień obciążenia przez cały czas jest niemal jednakowy. Nieznaczna różnica rzędu 5-10 kB jest spowodowana sytuacją w przebiegu gry, w której każdy z graczy posiada dużą ilość kart.

Tabela 7. Stopień obciążenia pamięci przeglądarką w zależności od czasu życia aplikacji w kilobajtach

	Angular	**Vue.js**
Uruchomienie	140,9	73
Półgodziny	146,2	77,8
Godzina	151,4	82,5

Jak widać w tabeli 7, średnia ilość pamięci wykorzystanej podczas uruchomienia aplikacji dla Angular wynosi 140,9 kB, a dla Vue.JS 73 kB. Jest to przede wszystkim związane z tym, że framework Angular jest bardziej rozbudowanym i posiada wiele dodatkowych modułów, które od razu zaczynają działać, nawet, wtedy, gdy nie jest to potrzebne. Inną przyczyną uzyskania takich wyników jest to, że framework Vue.js jest nowszy w porównaniu z Angular i jego optymalizacja kodu jest lepsza. To badanie pokazuje, że przy uruchomieniu na urządzeniach mobilnych badanej aplikacji, większe ryzyko zawieszenia urządzenia występuje w przypadku frameworka Angular.

5.8. Rozmiar plików końcowych

Rozmiar plików końcowych został zbadany przy użyciu wbudowanych narzędzi systemu Windows, po zbudowaniu paczki aplikacji internetowej odpowiedniego frameworku. Na podstawie otrzymanych wyników rozmiaru zbudowanej paczki aplikacji można wywnioskować, dla którego frameworku aplikacja zostanie szybciej załadowana przez urządzenie.

Po zbudowaniu aplikacji internetowej gry Munchkin, rozmiar plików końcowych w przypadku użycia Angular wynosi 31,7 MB, a w przypadku użycia Vue.js wynosi 22,1 MB. Rozmiar plików końcowych przy użyciu frameworku Vue.js jest prawie o 10 MB mniejszy. Tak wielka różnica jest powodowana tym, że Angular jest bardziej rozbudowany w porównaniu do Vue.js. Na przykład, Angular posiada bardzo rozbudowany system kontroli życia aplikacji, co powoduje zwiększenie rozmiaru plików końcowych oraz wydłuża czas wykonywania dowolnej funkcji. To, że Vue.js ma mniejszy rozmiar plików końcowych może być spowodowane tym, że Vue.js jest nowszym frameworkiem w porównaniu do Angular i twórcy Vue.js uwzględnili wszystkie tak zwane „wąskie gardła" oraz zbędne funkcje, które występują w Angular i dokonali w stosunku do nich zmian i optymalizacji lub całkowicie z nich zrezygnowali. Dlatego Vue.JS jest szybszy od Angular i wykorzystuje mniej pamięci.

6. Wnioski

Analizując wyniki przeprowadzonych badań można stwierdzić, że szybkość pracy tych dwóch frameworków jest bardzo podobna. Prawdopodobnie związane to jest z tym, że Vue.js rozpoczął swoje istnienie jako odnoga Angular, i jest on napisany w bardzo podobnym do Angular stylu. Dzięki użyciu dobrych technik programowania i uwzględnieniu problemów, które posiada Angular, programistom Vue.js udało się zmniejszyć rozmiar kodu wykonywalnego projektu o 50%. I chociaż na początku te dwa frameworki były bardzo do siebie podobne, Angular w wersji drugiej został napisany w TypeScript, a Vue.js nadal pozostaje rozwijany za pomocą HTML i JavaScript.

Ze względu na te uwagi można wywnioskować, że framework Angular jest bardziej rozbudowany i przez to trudniejszy do nauki, ale dzięki twardej typizacji Angular jest bardzo przydatny w przypadku tworzeniu dużych projektów. Vue.js posiada mniej możliwości i wydaje się być lepszym wyborem dla początkujących programistów, ale jednocześnie jest on bardziej elastycznym frameworkiem niż Angular i może zostać użyty do napisania zarówno dużych, jak i małych projektów.

Po przeanalizowaniu otrzymanych wyników można wywnioskować, że dla początkującego programisty framework Vue.js stanowi lepsze rozwiązanie do tworzenia aplikacji internetowych, dlatego że jest on łatwiejszy do nauki, potrzebuje mniej zasobów komputerowych do uruchomienia i działania napisanej za jego pomocą aplikacji internetowej i jest szybszy w przypadku projektu o niewielkim stopniu złożoności. Warto zauważyć, że wraz ze wzrostem doświadczenia zawodowego programisty i przy tworzeniu dużych projektów framework Angular może stać się bardziej przydatny. Pomimo faktu, że TypeScript, w którym jest napisany Angular, jest bardziej złożony i trudniejszy do nauki, rozbudowanie i wsparcie aplikacji stworzonej przy jego pomocy jest łatwiejsze. Na podstawie wyżej wymienionych faktów można stwierdzić, iż teza mówiąca że *framework Vue.js jest lepszym środowiskiem programistycznym do budowy aplikacji internetowych dla początkującego programisty niż framework Angular ze względu na łatwość nauczenia się go oraz wydajność przy założeniu, że tworzona aplikacja internetowa ma średni poziom złożoności* jest prawdziwa.

Literatura

[1] https://fructcode.com/ru/blog/features-of-popular-frameworks-html-css-php-and-python-frameworks/, Framework — ważne narzędzie programistyczne, [24.07.2019]

[2] https://divante.com/blog/top-10-popular-javascript-frameworks-2019/, Top 10 most popular JavaScript frameworks in 2019, [24.07.2019]

[3] https://en.wikipedia.org/wiki/Vue.js, Vue.js, [27.07.2019]

[4] Svensson A.: Speed Performance Comparison of JavaScript MVC Frameworks. Blekinge Institute of Technology, 2015

[5] Petukhova E.: Sitecore JavaScript Services Framework Comparison. Åbo Akademi University, 2019

[6] Kaluža M., Troskot K., Vukelić B.: Comparison of front-end frameworks for web applications development. Zbornik Veleučilišta u Rijeci, 2018

[7] Seshadri S.: Angular: Up and Running: Learning Angular, Step by Step. O'Reilly Media, 2018

[8] https://habr.com/ru/post/320014/, AngularJS vs Angular, [25.07.2019]

[9] Wilken J.: Angular in Action. Manning Publications, 2018

[10] https://developers.google.com/web/tools/chrome-devtools, Chrome Dev Tools, [12.08.2019]