JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 7 (2018) 205-209
Wysłane: 2017-12-03
Przyjęte: 2017-12-12

Analiza porównawcza zastosowania szkieletów Angular2 i Ember.js

Jan Palak*, Małgorzata Plechawska-Wójcik
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Celem artykułu jest analiza porównawcza bibliotek Angular2 z Ember.js. Na potrzeby testów została opracowana specjalna aplikacja internetowa. W świetle kryteriów porównawczych analizie poddano: szybkość generowania strony, wyświetlanie dużej liczby danych, obsługa multimediów, mechanizmy zabezpieczeń, dostępność dokumentacji, obsługę różnych formatów danych, integrację z portalami społecznościowymi. Ponadto celem jest również prezentacja cech wspólnych oraz różnic obu bibliotek.

Słowa kluczowe: Angular2; Ember.js; porównanie bibliotek Javascript

*Autor do korespondencji.

Adres e-mail: jpalak@o2.pl

Comparative analysis of the usage of Angular2 and Ember.js frameworks

Jan Palak*, Małgorzata Plechawska-Wójcik
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The focus of this article is the comparative analysis of the usage of the Angular2 and Ember.js frameworks. The requirement for tests of these frameworks has been developed though special web applications. The analysis was performed with the following comparative criteria: the speed of the website generation, presentation of large amount of data, media services, security mechanism, documentation availability, different data format handling, and integration with social media. Also, this paper aims to present common features and differences between the two frameworks regarding their characteristic features.

Keywords: Angular2; Ember.js; comparative analysis; Javascript frameworks

*Corresponding author.

E-mail address: jpalak@o2.pl

1. Wstęp

Możliwości JavaScript w porównaniu do zastosowania kiedyś a dziś różnią się diametralnie. Jeszcze nie dawno, bo 10 lat temu wykorzystywano go głównie do prostych operacji na poziomie formularza. Dzisiaj już jest o wiele bardziej zaawansowany i rozbudowany, nie tylko w przeglądarkach internetowych ale także w telefonach [1]. Model podwójnego wiązania elementów używanych w Javascript jest teraz powszechnie używany w dużych bibliotekach [2]. Ponadto służy do sterowania modułami jak i całymi stronami. sterowane jest za pomocą niego moduły a także całe strony [3].

W niniejszym artykule porównano ze sobą dwie popularne biblioteki Angular 2 oraz Ember.js. W obu technologiach została opracowana specjalna aplikacja testowa, która pobiera dane z serwera. Zbadano wydajność pracy bibliotek, ich możliwości oraz popularność wśród programistów.

2. Angular 2

Angular 2 to następca równie popularnej pierwszej wersji oznaczonej pod nazwą AngularJS [4]. Największą nowością w drugiej wersji Angular jest pełne wsparcie biblioteki ECMAScript 6 oraz języka TypeScript [5]. Sam język TypeScript cechami przypomina język Java: posiada klasy, interfejsy, adnotacje. Głównym zadaniem TypeScript jest kompilacja kodu do zwykłego JavaScriptu [6]. Proces pozwala szybciej pisać kod przez programistę, który jest odpowiednio zamieniany na postać najbardziej zrozumiałą dla przeglądarki [7].

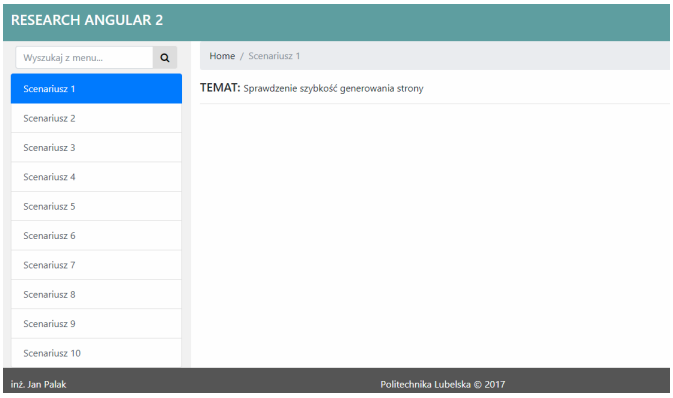
3. Ember.js

Otwarta biblioteka Javascript do tworzenia dużych aplikacji internetowych wykorzystująca wzorzec MVC (Model-Widok-Kontroler) [8]. Jest jednym z najpopularniejszych bibliotek Javascript do kontrolowania widoku na stronie. Wiele popularnych firm tj. Yahoo, Groupon, Discourse wykorzystuje obecnie tą bibliotekę w swoich aplikacjach. Biblioteka umożliwia tworzenie strony single-page-application opartej o szablony HTMLBars, które służą do automatycznego generowania widoku [9]. Są one kontrolowane przez warstwę kontrolera oraz routing aplikacji [10].

4. Opis aplikacji testowych

Do przeprowadzenia eksperymentu został przygotowany szablon (rys. 1), na podstawie którego opracowano logikę w Angular 2, Ember.js. Obydwie aplikacje mają charakter strony single page application, która polega na dynamicznym poruszaniu się stronie internetowej bez przeładowywania całej

witryny. Szablon zawiera w menu odniesienia do 10 scenariuszy, których opis znajduje się w rozdziale „Opis procedury badawczej". Nad menu znajduje się wyszukiwarka do odpowiedniego scenariusza po nazwie, a po prawej stronie panel do wyświetlenia głównej części strony.



Rys. 1. Szablon aplikacji internetowej

Szablon został wykonany w najnowszej wersji biblioteki CSS: Bootstrap, która z dniem dzisiejszym została oznaczona jako v4.0.0-alpha.6.

Wspólna część back-end obydwu aplikacji została opracowana w technologii REST API w implementacji Java. Za pomocą odpowiednich URI z aplikacji internetowej można wywoływać, pobierać czy też autoryzować się po stronie serwera aplikacyjnego Tomcat. Odpowiedzi z serwera za pomocą badanych bibliotek można przetwarzać i sterować na stronie za pomocą kontrolera.

5. Opis procedury badawczej

Wszystkie testy zostały przeprowadzone według 11 scenariuszy badawczych.

Scenariusz 1: Szybkość generowania strony

W tym przypadku sprawdzona została szybkość ładowania strony. Wielkość jest wyrażona w liczbie wiadomości na stronie (od 10 do 100) Na jedną wiadomość składa się 100 wyrazów, obrazek i dwa filmy zewnętrzne z dwóch różnych źródeł.

Scenariusz 2: Obsługa popularnych zabezpieczeń

W tym scenariuszu sprawdzono czy dane biblioteki obsługują zabezpieczenia takie jak: Ciasteczka, LocalStorage (pamięć wewnętrzna przeglądarki), reCAPTCHA, Szyfrowanie, JWT (Javascript WebToken). Są to najpopularniejsze technologie, w których można przechowywać ukryte dane, niedostępne dla innych użytkowników.

Scenariusz 3-6: Obsługa różnych formatów danych

Te scenariusze miały za zadanie sprawdzenie czy badane technologie obsłużą formaty danych takie jak: JSON, XML, YAML.

Scenariusz 7: Integracje z portalami społecznościowymi

Scenariusz ten miał za zadanie sprawdzenie czy możliwa jest autoryzacja z portalami społecznościowymi tj. Facebook.

Scenariusz 6: Sprawdzenie obsługi cache

Ten eksperyment miał na celu sprawdzenie czy w danej bibliotece został zaimplementowany mechanizm do przechowywania danych w celu eliminacji pobrania ich ponownie bądź wprowadzenie przez użytkownika.

Scenariusz 7: Dostęp do bazy Firebase

To zadanie miało na celu sprawdzenie czasów operacji typu odczyt, zapis i usunięcie danych z bazy Firebase. Są to proste dane generowane przez określoną liczbę razy. Kolejno 10, 100, 1 tys., 2 tys. i 3 tysiące.

Scenariusz 8: Dostęp do bazy typu SQLite

W tym przypadku podobnie jak w scenariuszu 7 zbadane zostały czasy operacji na bazie przeglądarkowej typu SQLite.

Scenariusz 9: Szybkość współpracy z innymi frameworkami – Google Charts

Ten scenariusz pokazuje, która z analizowanych technologii jest najszybsza pod względem komunikacji z innymi bibliotekami. W tym przypadku jest to popularna biblioteka do generowania wykresów. Sprawdzono czas generowania liczby punktów do czasu jego wyświetlenia, a następnie sprawdzono w której technologii najszybciej się one wygenerują.

Scenariusz 10: Szybkość ładowania obrazków

Rozpatrzono jak technologie porównywane poradzą sobie z przetwarzaniem i wyświetlaniem na stronie dużej liczby obrazków.

Scenariusz 11: Popularność użycia

W tym zadaniu pokazana została liczba projektów wykonanych w danej technologii w serwisie repozytorium GitHub.

Tezę badawczą sformułowano następująco: *Ember.js jest szybszą biblioteką w porównaniu do Angular 2 pod względem renderującym stronę, najlepiej współpracującą z innymi bibliotekami i najbardziej bezpieczną*

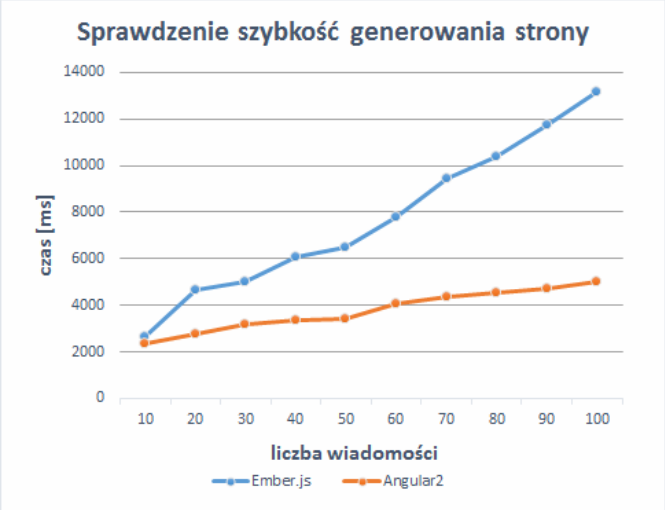
6. Wyniki badań

Do badań został wykorzystany laptop ASUS N551JM (Intel i7-4710HQ/16GB/250GB SSD/Win8 GTX860) oraz przeglądarka internetowa Chrome w wersji 62.0.3202.94 (64-bitowa).

W zależności od scenariusza została powtórzona określona liczba prób. Wyniki zostały uśrednione. W artykule przedstawiono 3 najistotniejsze scenariusze a wyniki wszystkich podano w podsumowaniu. W opisywanym przypadku rozpatrzono szybkość generowania strony, dostęp do bazy Firebase oraz szybkość współpracy z Google Charts.

6.1. Testy szybkości generowania strony

Pierwszym testem była szybkość wyświetlenia wiadomości, na którą składa się 100 wyrazów różnej długości, obrazek, film zewnętrzny i film wewnętrzny. Te informacje zwraca serwer REST, który działa lokalnie na komputerze. Czas ładowania strony jest zdefiniowany od momentu przeładowania do zakończenia wykonania wszelkich skryptów na stronie (Rys. 2).



Rys. 2. Wyniki testów szybkości aplikacji wyświetlania wiadomości strony.

Na podstawie rys. 2 można wyraźnie odczytać, że aplikacja napisana w Angular 2 o wiele szybciej wczytuje wiadomości ze zdjęciami i filmami niż Ember.js. Przyczynia się do tego sumaryczny czas, w którym czynniki składają się z różnych czasów: ładowania, wykonujących skrypt, renderowania, wyświetlania i bezczynności. W scenariuszu pełne ładowanie strony zajmuje 70% czasu wykonania skryptów. Jak dowodzi powyższe badanie, sumaryczne algorytmy wykorzystane w Angular 2 są szybsze.

Kolejnym testem jest szybkość wyświetlania dużej liczby obrazków (Rys. 3).

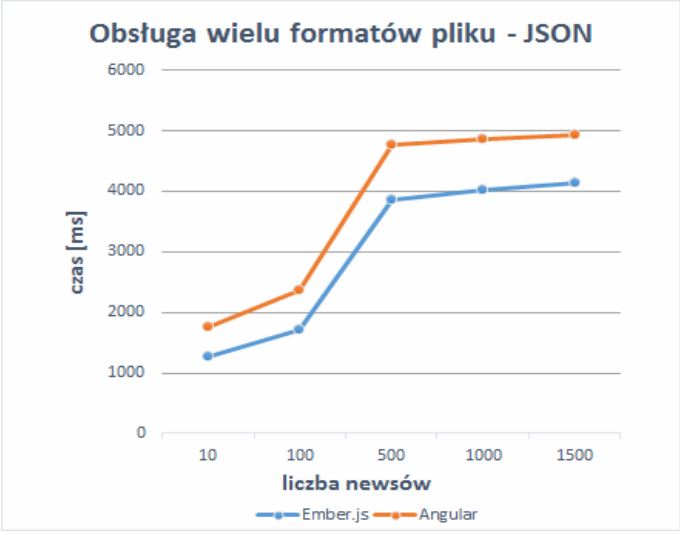
Z przeprowadzonych badań wynika, że w przypadku Ember.js informacje bajtowe są szybciej przetwarzane i wyświetlane. Przewaga ta zaczyna się w momencie, gdy wyświetlanych jest więcej jak 100 obrazków średniej wielkości tzn. 500px na szerokość i wysokość.

Rys. 3. Wyniki testów szybkości ładowania obrazków.

6.2. Bezpieczeństwo aplikacji internetowych

Pod względem zabezpieczeń obie technologie reprezentują prawie wszystkie formy popularnych zabezpieczeń . Wyjątkiem jest szyfrowanie w Ember.js. Nie ma żadnych modułów, które mogłyby zabezpieczyć w prosty sposób dane. Angular 2 za to udostępnia wszystkie rodzaje szyfrowań jak również resztę zabezpieczeń tj. reCAPTCHa, JWT (JSON Web Token) LocalStorage (wewnętrzna pamięć przeglądarki), Ciasteczka w przeglądarce internetowej.

W następnej kolejności zbadano przetwarzanie tekstu, na ten czynnik składa się parsowanie tekstu z formatu JSON do postaci modelu DOM (Document Object Model) wyświetlanych na stronie. Wyniki przedstawiono na rys. 4.



Rys. 4. Wyniki testów przetwarzania tekstu z formatu typu JSON.

Z rys. 4 wynika, że Ember.js przetwarza tekst szybciej niż Angular 2.

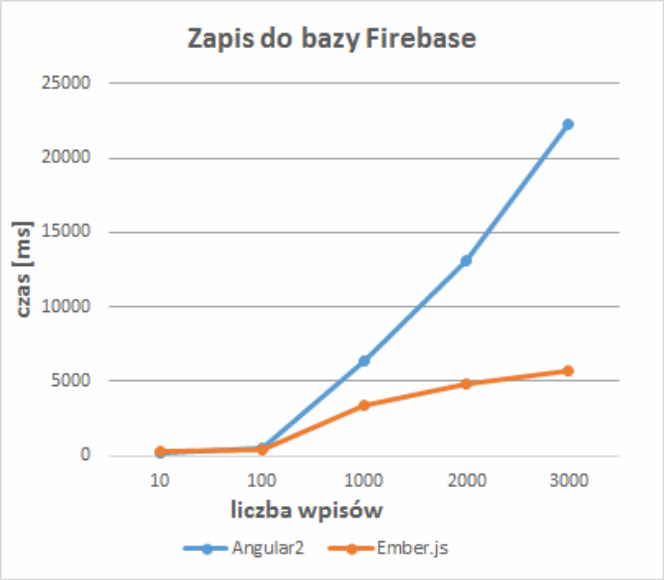
6.3. Wsparcie różnych formatów danych oraz jej szybkość parsowania.

Inne formaty danych tj. XML czy YAML to Ember.js nie umożliwiają ich wsparcia w swojej architekturze przetwarzania informacji. Cała architektura tej technologii została stworzona w taki sposób, że komunikacja odbywa się tylko i wyłącznie za pomocą notacji JSON. Angular 2 jest tutaj wyjątkiem, gdyż jego architektura pozwala tworzyć obejścia i parser, który może różną notację sprowadzać do postaci postawowego typu JSON.

Badając współpracę bibliotek z integracją z portalami społecznościowym sprawdzono najpopularniejszy na świecie Facebook. Obie porównywane technologie wspierają wewnętrzną aplikacje Facebook API. Wsparcie to można wykorzystać do wielu czynnosci: od pobierania podstawowych danych do zdjęć z galerii użytkownika. Oczywisty jest fakt, że użytkownik musi zaakceptować zgodę na wykorzystanie tych danych w aplikacji.

6.4. Dostęp do bazy danych

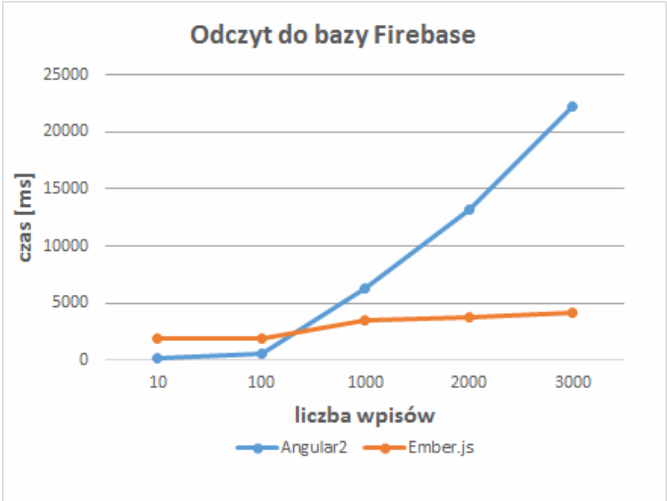
W kolejnym teście zbadano szybkość podstawowych operacji na zenętrznej bazie danych typu Firebase (Rys. 5, 6, 7). Do tej bazy ładuje się określoną ilość danych oraz mierzy czas jaki potrzebny jest do jej wykonania.



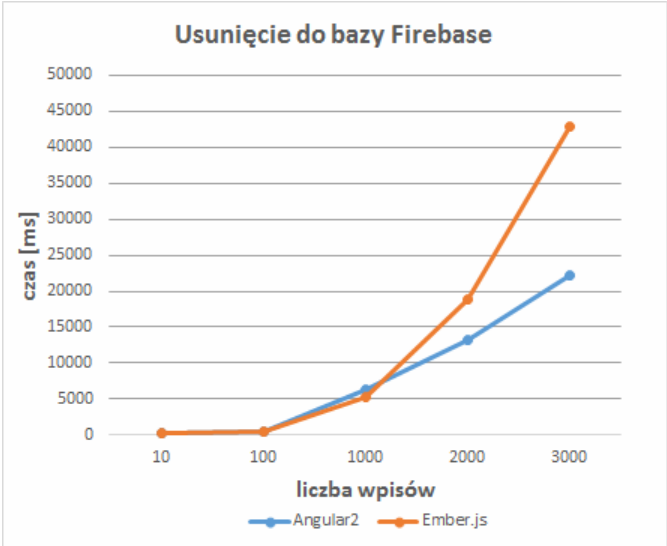
Rys. 5. Wyniki testów zapisu danych do bazy danych Firebase.

Z rysunków 5, 6 oraz 7 wynika, że zapis jak i odczyt w Ember.js jest znacznie szybszy niż w Angular 2. Z kolei, jeżeli chodzi o usunięcie danych, w Angular jest to znacznie szybsze. Wynika to z tego, że w Ember.js każdy rekord jest identyfikowany po konkretnym Id. Każdy z nich należy iterowac i czyścić. Natomiast w przypadku Angular 2 sprawa sprowadza się do przypisania pustej tablicy do bazy danych co powoduje jej wyczyszczenie.

Badając inny typ dostępu do bazy danych typu SQLite można dojść do wniosku, że Angular 2 nie wspiera bazy danych w przeglądarce. Można tylko symulować taki zbiór danych na emulatorach telefonów komórkowych typu Android [11]. W przypadku Ember.js otrzymać można wszelkie podstawowe wsparcie [12].



Rys. 6. Wyniki testów odczytu danych z bazy danych Firebase.



Rys. 7. Wyniki testów usunięcia danych z bazy danych Firebase.

6.5. Szybkość współpracy z innymi bibliotekami

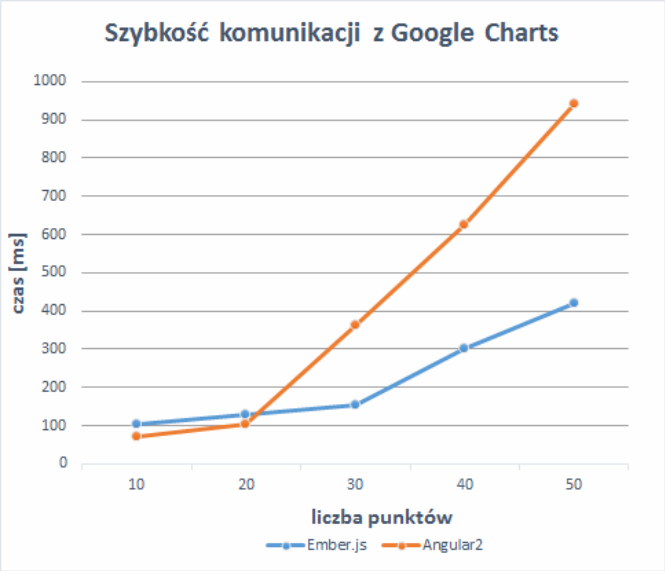
Kolejne badanie przedstawia szybkość komunikacji z inną biblioteką do generowania wykresów typu Google Charts. Wyniki przedstawia rysunek 8. Wyniki te pokazują sporą przewagę Ember.js nad Angular.

6.6. Popularność

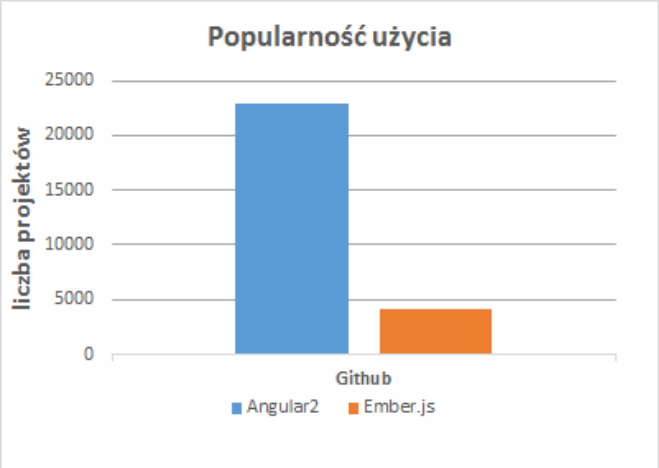
Ostatnim testem, któremu poddane zostały technologie jest ich popularność wśród programistów oraz popularnść w sieci.

Rys. 9 przedstwia aż 5-krotną przewagę Angular 2 nad Ember.js. Dotyczy to również stron typu stackoverflow, gdzie programiści proszą o pomoc i ją otrzymują, co nadaje takim

serwisom funkcję źródła wiedzy dla programistów. Ponadto z przerowadzonych badań wynika, że więcej tematów jak i rozwiązań można otrzymać z Angulara 2 niż z Ember.js.



Rys. 8. Wyniki szybkości komunikacji z Google Charts.



Rys. 9. Diagram kolumnowy ilości projektów w danych technologiach.

7. Wnioski

Zaimplementowano aplikację webową typu SPA (Single-Page-Application) Angular 2 i Ember.js. Funkcjonalność tych aplikacji służyła do porównania określonych funkcji w obu technologiach z wykorzystaniem specjalnie przygotowanych scenariuszy. Za pomocą deweloperskich ustawień w przeglądarce Chrome można z dużą dokładnością zmierzyć czas wykonania określonych operacji.

Podsumowując wszystkie scenariusze i zestawiając je według wag o różnym stopniu przydatności: 3 – istotne, 2 – średnio istotne, 1 – mało istotne.

Można za pomocą funkcji sum wagowych wyznaczyć lepszą technologię.

Tabela 1. Zestawienie metodą funkcji sum wagowych obu technologii.

Scenariusz	Ember.js	Angular 2	Waga	Ember.js z wagą	Angular2 z wagą
1	0	1	3	0	3
2	4	5	3	12	15
3	1	0	3	3	0
4	0	1	1	0	1
5	0	1	1	0	1
6	1	1	3	3	3
7	2	1	3	6	3
8	1	0	2	2	0
9	1	0	3	3	0
10	1	0	3	3	0
11	0	1	3	0	3
Suma				**32**	**29**

Przeprowadzone badania dowodzą, że Ember.js jest szybszą biblioteką od Angular 2, widać to szczególnie w komunikacji z Google Charts. Angular 2 jest bardziej bezpieczny pod względem szyfrowania wiadomości. Podsumowując w niektórych scenariuszach Angular 2 jest lepszą technologią od Ember.js. Różnica ta jest jednak niewielka. Pod względem innych czynników np. popularności czy wsparcia technicznego przez innych programistów większą przewagę posiada Angular 2. Natomiast pod względem funkcjonalnym i szybkości znacznie lepiej sprawdza się Ember.js. Wobec czego jest potwierdzeniem tezy badawczej, że Ember.js jest szybszym rozwiązaniem od konkurenta.

Literatura

[1] D. Magnusson, E. Grenmyr, An Investigation of Data Flow Patterns Impact on Maintainability When Implementing Additional Functionality, Linnaeus University, Faculty of Technology, Department of Computer Science, 2016.

[2] U. Shakya, Using a Framework to develop Client-Side App A Javascript Framework for cross-platform application, Helsinki Metropolia University of Applied Sciences, 6 Nov 2014.

[3] J. Koetsier, Evaluation of JavaScript frameworks for the development of a web-based user interface for Vampires, Informatica — Universiteit van Amsterdam, June 8, 2016.

[4] G. Kunz, Angular 2. Tworzenie interaktywnych aplikacji internetowych, Helion 2017.

[5] P. Deeleman, Learning Angular 2, Packt-Publishing 2016.

[6] M. Nayrolles, Angular 2 Design Patterns and Best Practices, Packt-Publishing 2017.

[7] M. Gechev, Switching to Angular 2, Packt-Publishing 2016.

[8] S. Shrestha, Ember.js front-end framework – SEO challenges and frameworks comparison, Haaga-Helia University of Applied Science 2015.

[9] M. Bodmer, Instant Ember.js Application Development How-to, Packt-Publishing 2013.

[10] J. Cravens, T. Q. Brady, Ember.js dla webdeveloperów, Helion 2015.

[11] www.github.com/litehelpers/Cordova-sqlite-storage [08.12.2017].

[12] www.pmjs.com/package/ember-sqlite-adapter [08.12.2017].