JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 7 (2018) 200-204
Wysłane: 2017-12-09
Przyjęte: 2017-12-12

Porównanie szkieletów aplikacji Angular i BackboneJS na przykładzie aplikacji internetowej

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

Streszczenie. Artykuł zawiera porównanie szkieletów aplikacji Angular i BackboneJS. Dla potrzeb badawczych zostały utworzone dwie aplikacje jednostronicowe, oferujące identyczne funkcjonalności. Zmierzony i porównany został czas ładowania elementów aplikacji, liczba linii napisanego kodu, rozmiar skompilowanych aplikacji, społeczność zgromadzona wokół szkieletu oraz czas potrzebny na wykonanie aplikacji. Otrzymane wyniki nie wykazały rozwiązania wyraźnie lepszego do zastosowania w aplikacjach jednostronicowych.

Słowa kluczowe: JavaScript; Angular; BackboneJS; aplikacja jednostronicowa

*Autor do korespondencji.

Adres/adresy e-mail: mateusz.moczulski@dzielo.pl, m.plechawska@pollub.pl

A comparative analysis of selected Java Script frameworks in the context of web applications on the example of Angular and BackboneJS

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

Abstract. The paper contains comparison between Angular and BackboneJS frameworks. For research purposes, two single page applications were created. Both providing the same functionalities. In this article such elements were examined and compared: time of loading elements, number lines of code, size of compiled application. community gathered around frameworks and time needed to create the application using frameworks. The obtained results did not indicate a solution that was clearly better to be used in single page applications.

Keywords: JavaScript; Angular; BackboneJS; Single Page Application

*Corresponding author.

E-mail address: mateusz.moczulski@dzielo.pl, m.plechawska@pollub.pl

1. Wstęp

JavaScript jest jednym z najpopularniejszych języków programowania, a wraz z rozwojem Node.js zyskał na popularności zarówno w zakresie front-end, jak i back-end [1]. Framework JavaScript różni się od biblioteki JavaScript w swoim strumieniu kontrolnym. Biblioteka oferuje funkcje, które mają być wywoływane przez swój kod nadrzędny, podczas gdy framework definiuje cały projekt aplikacji. Framework wywołuje oraz używa kodu napisanego przez developera w pewien szczególny sposób. Większość szkieletów JavaScript jest zgodnych z paradygmatem model-widok-kontroler, zaprojektowanym w celu segregowania aplikacji WWW na ortogonalne jednostki w celu poprawy jakości kodu i łatwości konserwacji [2]. Takimi szkieletami są Angular oraz Backbone.

Angular oraz Backbone wykorzystywane są zazwyczaj do wytwarzania aplikacji jednostronicowych (ang. Single Page Application). Aplikacja jednostronicowa to aplikacja internetowa lub witryna internetowa, która współdziała z użytkownikiem poprzez dynamiczne przepisywanie bieżącej strony, zamiast ładowania całych nowych stron z serwera. Takie podejście pozwala uniknąć przerw w korzystaniu użytkownika z kolejnych stron, dzięki czemu aplikacja zachowuje się bardziej jak aplikacja desktopowa. W SPA wszystkie niezbędne kody - HTML, JavaScript i CSS - są pobierane przy pojedynczym załadowaniu strony lub odpowiednie zasoby są dynamicznie ładowane i dodawane do strony w razie potrzeby, zwykle w odpowiedzi na działania użytkownika [3]. Strona nie wczytuje się ponownie w żadnym momencie procesu ani nie kontroluje transferu na inną stronę, chociaż za pomocą skrótu lokalizacji lub interfejsu API historii HTML5 można zapewnić percepcję i możliwość nawigacji osobnych stron logicznych w aplikacji. Aplikacja taka jest znacznie szybsza od aplikacji, w której każda strona jest ładowana oddzielnie [4].

Frameworkami dostępnymi obecnie na rynku są także Backbone oraz Angular. Artykuł podejmuje temat porównania powyższych szkieletów. W każdym z nich została utworzona identyczna aplikacja, która oferowała te same funkcjonalności.

2. Angular

Angular 4 jest jednym z frameworków dostępnych na rynku wspomagających wytwarzanie SPA. Jest wspierany i rozwijany przez developerów firmy Google. Pierwsze wydanie Angular miało miejsce w 2009r. Cały czas produkt był rozwijany i w 2017 roku weszła wersja 4.0. Od wersji 2.0 Angular korzysta z TypeScript zamiast JavaScript [5].

Angular to zestaw narzędzi do tworzenia aplikacji internetowych. Jest w pełni rozszerzalny i działa dobrze z innymi bibliotekami. Każda cecha może zostać zmodyfikowana lub wymieniona, aby pasowała do potrzeb. HTML jest bardzo dobry do deklarowania statycznych dokumentów, jednak nie jest skuteczny, podczas próby użycia go do deklarowania dynamicznych widoków w aplikacjach internetowych [6]. Angular umożliwia rozszerzenie słownictwa HTML w aplikacji. Powstałe środowisko jest niezwykle wyraziste, czytelne oraz szybkie. Niewątpliwą zaletą Angular jest powiązanie danych. Jest to automatyczny sposób aktualizowania widoku, gdy model się zmienia, a także aktualizuje model, gdy zmienia się widok. Jest to bardzo użyteczne, ponieważ eliminuje manipulacje DOM (Obiektowy Model Dokumentu, *ang. Document Object Model*); z listy rzeczy, na które trzeba zwracać uwagę [7][6].

3. BackboneJS

Backbone.js to lekka biblioteka JavaScript, która dodaje strukturę do kodu klienckiego. Ułatwia to zarządzanie aplikacją, pozostawiając kod, który jest dłużej możliwy do utrzymania.

Backbone jest dojrzały, popularny i ma zarówno tętniącą życiem społeczność deweloperską, jak i mnóstwo wtyczek i rozszerzeń [8]. Korzysta on z wzorca MVC. Wykorzystywany jest do tworzenia nietypowych aplikacji przez firmy takie jak Disqus, Walmart, SoundCloud i LinkedIn.

Backbone dostarcza minimalny zestaw do strukturyzacji danych (modele, kolekcje) i interfejsów użytkownika (widoki, adresy URL), które są przydatne podczas tworzenia dynamicznych aplikacji przy użyciu kodu JavaScript. Można użyć zalecanej architektury oferowanej przez twórców lub rozszerzyć ją, aby spełnić wymagania developera. Biblioteka nie skupia się na widżetach ani nie zastępuje sposobu, w jaki są tworzone obiekty. Dostarcza ona narzędzia do manipulacji i kwerendy danych w aplikacji [9].

4. Wybór odpowiedniego szkieletu

Frameworki JavaScript stają się bardzo pomocne przy szybkim tworzeniu aplikacji internetowych. Dlatego ważne jest, aby wybrać najlepszy szkielet dostosowany do stawianych wymagań. Istotnym czynnikiem, który decyduje o wyborze szkieletu są jego koszty, developerzy wolą korzystać z rozwiązań darmowych. Ważnym czynnikiem wyboru szkieletu jest także przestrzeń dyskowa jaka jest zajmowana przez gotową aplikację. Istotne znaczenie podczas wyboru szkieletu odgrywają również:

1) modułowość;
2) przenośność;
3) możliwość manipulacji DOM,
4) częste aktualizacje [10].

5. Procedura badawcza

Przed przystąpieniem do realizacji tematu należało określić kryteria analizy, jaka będzie przeprowadzona podczas porównywania frameworka Angular oraz Backbone. W artykule szkielety będą badane pod względem praktycznym. Analiza praktyczna ma na celu analizę działającego produktu. Pozwala ona określić czy napisana aplikacja spełnia wymagania wydajnościowe oraz oczekiwania klienta. Do zastosowanych kryteriów porównawczych należą: analiza czasu ładowania elementów, analiza liczby linii kodu oraz analiza rozmiaru skompilowanego kodu.

Na potrzeby badań założono również testowe Rest API, z którego aplikacje po załadowaniu pobierają potrzebne dane. W przypadku obu aplikacji pobieranie danych odbywa się od razu po włączeniu.

Do wyświetlenia badanej tabeli w przypadku Angular posłużyła metoda *ngFor* (Przykład 1). Jest to metoda umożliwiająca wyświetlenie elementów kolekcji w postaci HTML. W przypadku Backbon posłużyła do tego metoda *mapObject* (Przykład 2). Działa ona analogicznie jak opisana wcześniej metoda *ngFor.*

Przykład 1. Wyświetlanie tabeli rekordów – Angular

```
<tr *ngFor="let carss of cars; let i = index">
<td>
{{carss.id}}
</td>
<td>
{{carss.mark}}
</td>
<td>
{{carss.model}}
</td>
<td>
{{carss.price}}
</td>
<td>
{{carss.yearBook}}
</td>
<td>
{{carss.carMileage}}
</td>
<td width = 150 px>
<button id={{i+1}} type="button" class="btn btn-success"
(click)="onClick($event)">Pokaż szczegóły</button>
</td>
</tr>
```

Przykład 2. Wyświetlanie tabeli rekordów - Backbone

```
<%
_.mapObject(cars, function(cars) { %>
<tr>
<td>
<%- cars.get('id') %>
</td>
```

```
<td>
<%- cars.get('mark') %>
</td>
<td>
<%- cars.get('model') %>
</td>
<td>
<%- cars.get('price') %>
</td>
<td>
<%- cars.get('yearBook') %>
</td>
<td>
<%- cars.get('carMileage') %>
</td>
<td width = 150 px>
<button id="button" class="btn btn-success" data-
carsId=<%- cars.get('id') %>>Pokaż szczegóły</button>
</td>
</tr>
```

6. Wyniki badań

Pierwszym badanym elementem będzie czas ładowania elementów aplikacji (Tabela 1). Jest to ważne kryterium badawcze szkieletów. Przed wyborem frameworka warto się zastanowić jakie elementy będą szybciej działały w jakiej technologii. Jest to szczególnie istotne w przypadku pisania rozbudowanej aplikacji, z której będzie korzystało wielu użytkowników.

Tabela 1. Czasy ładowania elementów aplikacji

	Angular [ms]	Backbone [ms]
Ładowanie całego projektu	1867	615
Widok - kontakt	1.24	0.71
Widok - rejestracja	0.95	0.98
Widok - logowanie	1.37	0.84
Widok - wszystkie samochody	3.27	7.89
Widok - strona główna	1.63	1.93
Widok – szczegóły samochodu	2.32	3.42

Badania zostały przeprowadzone także pod kątem szybkości ładowania elementów do tabeli (Tabela 2). Przeprowadzono badania na różnej liczbie rekordów, aby osiągnąć bardziej widoczne różnice.

Tabela 2. Czasy ładowania tabeli z rekordami

	Angular [ms]	Backbone [ms]
15 rekordów	2.98	7.89
100 rekordów	3.16	10,78
5000 rekordów	3381.70	3209.50
20000 rekordów	8928.80	9872.90
40000 rekordów	14829.58	12022.91
70000 rekordów	29742.29	26927.98
120000 rekordów	54687.29	49782.98

Zbadano również liczbę linii kodu wykorzystanej do wykonania aplikacji (Tabela 3). Wykonanie niektórych elementów aplikacji w różnych frameworkach wymaga napisania różnej liczby kodu. Mniejsza liczba linii kodu często przekłada się na mniejszy czas wymagany na napisanie danego elementu. W badaniu liczona jest liczba linii kodu oraz liczba znaków wykorzystana do wykonania danego elementu.

Tabela 3. Liczba linii kodu wykorzystana do napisania elementu

	Angular (liczba linii kodu/liczba znaków)	Backbone (liczba linii kodu/liczba znaków)
Slider na stronie głównej	20/491	25/859
Przełączanie pomiędzy widokami	14/437	26/835
Pobranie danych z samochodami	4/131	15/253
Wyświetlenie tabeli z samochodami	66/1154	66/1891
Wyświetlanie zdjęć samochodu	9/249	14/257
Wyświetlanie danych samochodu	31/719	34/828
Wysłanie danych wprowadzonych podczas rejestracji do serwera	8/301	6/161
Wyświetlenie komunikatu o zalogowaniu	11/218	19/377
Cała aplikacja	232/4315	311/6198

Zbadano również rozmiar aplikacji (Tabela 4). Im większy projekt tym dłużej będzie się ładował na urządzeniach użytkowników. Dlatego warto zabiegać, aby cały projekt miał jak najmniejszą wagę.

Tabela 4. Rozmiar skompilowanych projektów

	Angular	Backbone
Rozmiar całego projektu	520 kb	32 kb

Następnym badanym elementem będzie badanie społeczności korzystającej z danego szkieletu (Tabela 5). Większa liczba użytkowników świadczy o większej możliwości pomocy na forach internetowych. Często też doświadczeni użytkownicy nagrywają swoje materiały i udostępniają w sieci.

Tabela 5. Społeczność frameworków w wybranych serwisach

	Angular	Backbone
stackoverflow.com [11]	66119 zapytań	20413 zapytań
gitter.im [12]	800tys. użytkowników	-
github.com [13]	27 tys. gwiazd	26 tys. gwiazd
youtube.com [14]	700 tys. materiałów	30 tys. materiałów

Następnym elementem jaki będzie analizowany jest czas jaki został wykorzystany na wykonanie danego elementu (Tabela 6). Czas był mierzony za pomocą aplikacji Licznik

czasu zadań dostępnej w przeglądarce Chrome. Czasy zostały zaokrąglone do dziesiętnych części godziny.

Twórcy frameworków starają się jak najbardziej skrócić czas jaki jest niezbędny do tworzenia elementów aplikacji przez deweloperów. Czas zaoszczędzony na aplikacji często przekłada się na niższe koszty wytworzenia aplikacji, Czas potrzebny na wykonanie aplikacji w danym frameworku jest podstawowym kryterium branym pod uwagę do wyceny przez firmy zajmujące się wytwarzaniem oprogramowania. Zaoszczędzony czas można przeznaczyć na większą liczbę testów co przekłada się nieraz na jakość wytworzonego oprogramowania.

Tabela 6. Czas pracy wykorzystany na wykonanie elementów aplikacji

	Angular [h]	Backbone [h]
Utworzenie strony głównej	1	1,2
Utworzenie slidera	2.5	2.2
Utworzenie przejścia pomiędzy widokami	0.5	1.8
Utworzenie pozostałych widoków	2.5	4.2
Wyświetlenie tabeli z wszystkimi samochodami	0.8	3
Wyświetlenie zdjęć w szczegółach samochodu	1.5	1.5
Wyświetlenie widoku z podglądem zdjęcia	2.6	2
Pobranie danych do modelu	1.6	2.2

W tabeli 7 przedstawiono zbiorczą ocenę od 1 do 5 (5 – najlepsza ocena, 1 – najgorsza) szkieletów jakie uzyskały w badanych kryteriach. Jest to subiektywna ocena autorów artykułu.

Tabela 7. Zbiorcza ocena szkieletów w poszczególnych kryteriach

	Angular	Backbone
Czas ładowania elementów aplikacji	4	5
Czas ładowania tabeli z rekordami	4	5
Liczba linii kodu	5	3
Rozmiar skompilowanego kodu	2	5
Społeczność frameworków	5	2
Czas wykorzystany na wykonanie aplikacji	5	3
Suma	25	23

7. Analiza wyników

Angular oraz Backbone osiągają podobne wyniki czasu ładowania elementów (Tabela 1 i 2). Różnice między nimi mogą pojawiać się w przypadku ładowania bardzo dużych ilości danych. Czasy ładowania widoków w aplikacji są zbliżone. Lepszy czas ładowania całej aplikacji osiąga Backbne. W przypadku ładowania rekordów do tabeli lepsze wyniki osiąga Backbone, jednak aby zauważyć różnicę w czasie ładowania elementów muszą być to wielkości rzędu co najmniej kilkudziesięciu tysięcy rekordów. W przypadku ładowania zdjęć do aplikacji nieco lepiej radzi sobie Angular, jednak również aby różnica była zauważalna musi być to bardzo duża liczba zdjęć.

Backbone jest frameworkiem, który wymaga napisania większej ilości kodu (Tabela 3) w porównaniu do Angular. Wykorzystanie narzędzia dla programistów Angular CLI służącego do budowania elementów aplikacji za pomocą komend ułatwia oraz przyśpiesza pracę nad aplikacją. Napisanie aplikacji w Backbone wymagało ok 43% większej liczby znaków niż w Angular

Rozmiar skompilowanego kodu (Tabela 4) w Angular jest znacznie większy niż w przypadku Backbone. Ma na to wpływ to co jest dodawane przez sam szkielet do projektu oraz sama waga frameworka. Rozmiar kodu może mieć wpływ na czas wczytywania aplikacji co potwierdza tabela 1. Różnica może być jeszcze bardziej odczuwalna przez użytkowników ze słabym połączeniem internetowym.

Angular posiada większą społeczność (Tabela 5), co powoduje, że łatwiej będzie znaleźć użytkownikowi rozwiązanie problemów, oraz uzyskać pomoc od innych użytkowników. Istnieje więcej materiałów do nauki Angular dostępnych bezpłatnie w sieci. Producenci Angular organizują również specjalne spotkania dla społeczności zrzeszonej wokół swojego produktu.

Czas potrzebny na wykonanie aplikacji (Tabela 6) za pomocą Backbone był o ok 26% większy niż w przypadku Angular Jest to dość znaczna różnica, która w przypadku bardziej skomplikowanej aplikacji mogła by mieć decydujący wpływ na wybór szkieletu. Różnica ta może znacznie przekładać się na koszty potrzebne na wykonanie aplikacji.

8. Wnioski

W dzisiejszych czasach technologie, które są stosowane do wytwarzania oprogramowania cały czas ulegają zmianie. Często zastępują je rozwiązania nowsze i bardziej rozbudowane, dające developerom więcej możliwości. Obecnie na rynku dostępnych jest wiele frameworków typu open-source, które umożliwiają tworzenie aplikacji prostych oraz bardziej zaawansowanych. Wraz z zastosowaniem nowych szkieletów praca developerów jest szybsza oraz wymaga napisania mniejszej liczby kodu. Niniejsza praca porównuje wybrane rozwiązania, pozwala programiście na wybór odpowiedniego szkieletu do zastosowania we własnej aplikacji.

Otrzymane wyniki pozwoliły stwierdzić, że oba frameworki nadają się do zastosowania w aplikacji jednostronicowej co pokazuje tabela 7. Angular posiada większą społeczność, powoduje to że programiście łatwiej będzie znaleźć rozwiązanie problemów, uzyskać pomoc od innych użytkowników, oraz znaleźć materiały do nauki. Analiza czasu ładowania elementów pozwoliła stwierdzić, że nie ma dużej różnicy w wynikach obu szkieletów. Backbone osiągnął lepszy wynik w przypadku ładowania tabeli z różną liczby rekordów. Różnice te były by jednak

zauważalne dopiero w przypadku dziesiątek tysięcy elementów. Analiza złożoności kodu pokazała, że wykonanie aplikacji z wykorzystaniem szkieletu Angular wymaga mniejszej ilości kodu do napisania. Czas potrzebny na wykonanie aplikacji z wykorzystaniem Angular był znacznie mniejszy.

Wybór konkretnego szkieletu zależy więc od potrzeb i preferencji. Oba badane frameworki dostarczają kompletny zestaw funkcji do wykonania aplikacji internetowej typu single page aplication. Oba szkielety nie są pozbawione wad oraz zalet. Ułatwiają one oraz skracają czas jaki jest potrzebny na wykonanie rozbudowanej aplikacji. Główną zaletą Angular jest duża społeczność zgromadzona wokół szkieletu oraz niewielka ilość czasu i liczba linii kodu potrzebna na wykonanie aplikacji. Największą wadą szkieletu jest duży rozmiar skompilowanego projektu. Największą zaletą Backbone jest mały rozmiar skompilowanego projektu oraz niewielka ilość czasu potrzebne na załadowanie elementów aplikacji. Największą wadą tego szkieletu jest mała społeczność korzystająca z frameworka, wysoka liczba linii kodu potrzebnego na wykonanie aplikacji oraz dłuższy czas potrzebny na wykonanie aplikacji niż w przypadku Angular.

Literatura

[1] Larry Ullman,: *Nowoczesny język JavaScript*, Helion, Gliwice 2013.

[2] Tim Amber, Nicholas Cloud,: *JavaScript Frameworks for Modern Web Dev,* Apress, Nowy Jork 2015.

[3] Michael S. Mikowski, Josh C. Powell*,: Single Page Web Applications*, Manning, Nowy Jork 2014.

[4] Mikito Takada,: *Single Page Application in Depth*, ebook, 2012.

[5] Ari Lerner, Felipe Coury, Nate Murray, Carlos Taborda,: *ng-book 2 The Complete Book on Angular 4*, Nowy Jok 2017.

[6] Jesse Palmer, Corinna Cohn, Michael Giambalvo, Craig Nishina ,; *Testing Angular Applications*, Manning, Nowy Jork 2017.

[7] Angular 4 documentation - https://angular.io/docs/ts/ latest/ [01.12.2017]

[8] Addy Osmani ,: *Developing Backbone.js Applications*, O'Reilly, 2013.

[9] BackboneJS documentation - http://backbonejs.org/ documentation/ [01.12.2017]

[10] Pano, A.; Graziothin, D.; Abrahamsson, P.; *What leads developers towards the choice of a JavaScript* framework?, 2016.

[11] stackoverflow.com [01.12.2017]

[12] https://gitter.im [01.12.2017]

[13] https://github.com/ [01.12.2017]

[14] https://www.youtube.com/ [01.12.2017]