JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 6 (2018) 82-86
Wysłane: 2017-11-08
Przyjęte: 2017-11-16

Porównanie szkieletów aplikacji AngularJS i React.js na przykładzie aplikacji internetowej

Łukasz Capała*, Maria Skublewska-Paszkowska
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Artykuł zawiera porównanie szkieletów aplikacji AngularJS i React.js. Dla potrzeb badawczych zostały utworzone dwie aplikacje oferujące identyczne funkcjonalności. Zmierzony i porównany został czas wykonania typowych zadań aplikacji internetowej: wstawianie, sortowanie i usuwanie elementów z listy obiektów JavaScript.

Słowa kluczowe: javascript, angular, react, wydajność frameworka

*Autor do korespondencji.

Adresy e-mail: lukaszc24@gmail.com, maria.paszkowska@pollub.pl

Comparison of AngularJS and React.js frameworks based on a web application

Łukasz Capała*, Maria Skublewska-Paszkowska[a]
[a] Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The paper contains comparison between AngularJS and React.js frameworks. For research purposes two applications that provides the same functionalities were created. Time of typical tasks execution: elements insertion, sorting and deletion from JavaScript list of objects was measured and compared.

Keywords: javascript, angular, react, framework performance

*Corresponding author.

E-mail addresses: lukaszc24@gmail.com, maria.paszkowska@pollub.pl

1. Wstęp

Standard ECMAScript 6 wprowadził do języka JavaScript elementy charakterystyczne dla popularnych języków wysokiego poziomu, takich jak Java, C++ czy C#. Zamiast używania funkcji jako zamienników dla obiektów zaimplementowano charakterystyczne słowo kluczowe *Class*. Język JavaScript znajduje zastosowanie w prostej manipulacji drzewem dokumentu strony internetowej w celu interakcji z użytkownikiem. Można go wykorzystać do budowy skomplikowanych aplikacji działających w przeglądarce internetowej a także aplikacje typu *server-side* (działające po stronie serwera) dzięki Node.js. Połączenie JS z responsywnymi elementami języka HTML pozwala tworzyć rozbudowane projekty, które wchodzą w interakcję z użytkownikiem bez potrzeby przeładowania okna przeglądarki internetowej (ang. *single page application*) [1].

Na polu *front-endu*, czyli warstwy prezentacji również nastąpiły duże zmiany na przestrzeni lat. Oparte o JavaScript szkielety aplikacji (ang. *framework*) umożliwiają na rozległą, dynamiczną modyfikację zawartości strony a pod względem implementacji są proste w obsłudze. Operowanie na danych, takie jak wstawianie, sortowanie, usuwanie są teraz wykonywane przez program klienta, nie powodując obciążenia po stronie serwera, poprzez wyeliminowanie zbędnych zadań. Same szkielety opierają się również o nakładki standaryzujące i rozszerzające możliwości języka JavaScript (JSX, TypeScript), jednocześnie ułatwiając programiście budowanie aplikacji. Tak utworzone projekty są bardzo interaktywne i podobne do tych działających bezpośrednio w środowisku systemu operacyjnego, bez połączenia z Internetem.

Wśród najpopularniejszych frameworków można wymienić AngularJS (od nowszych wersji nazywany Angular) oraz React.js (lub prościej - React) [2]. Angular jest rozwijany przez Google, React przez firmę Facebook.

Artykuł podejmuje temat porównania powyższych szkieletów. W każdym z nich została utworzona identyczna aplikacja. Oferowała te same funkcjonalności. Jedno z porównań dotyczyło Angular w wersji 1 [3]. Liczba zmian zawartych w wersjach 2 lub wyższych jest duża. Zmienia to całkowicie budowę aplikacji opartej o Angular.

Porównanie nie będzie pokrywało takich kryteriów jak przyswajalność (prędkość opanowania zasad tworzenia aplikacji przy pomocy frameworka) czy rodzaj wzorca projektowego.

2. AngularJS

Angular posiada rozbudowane możliwości, jeżeli chodzi o kwestię implementacji obszaru frontowego aplikacji. Najnowsza wersja 4.2.4 używa do tego języka TypeScript, który stanowi swoistą nakładkę na język JavaScript. Wprowadza on obsługę typowania oraz inne rzeczy charakterystyczne dla znanych języków programowania. Wersja 1 i niższe były pod względem składni kodu źle oceniane przez programistów [4]. Tutaj istnieje wyraźny podział na komponenty, szablony, dyrektywy oraz serwisy. Zrezygnowano z charakterystycznej terminologii MVC (Model Widok Kontroler, *ang. Model View Controller*) znanej z poprzednich wersji narzędzia.

Atutem Angular wciąż pozostaje możliwość dwustronnego wiązania danych [5]. Umożliwia ona aktualizację pola obiektu komponentu poprzez interfejs użytkownika i odwrotnie – zmienna zaktualizowana w komponencie zmieni swoją wartość również na warstwie prezentacji. Proces ten jest zautomatyzowany przez framework i nie wymaga żadnej ingerencji programisty do jego wywołania.

Prosty system szablonów opartych na plikach HTML pozwala łatwo mieszać dyrektywy ze znanymi znacznikami hipertekstowymi, co ułatwia przystosowanie aplikacji pod kątem dostosowania zawartości do różnych urządzeń i przekątnych ekranu (ang. *responsive web design*). [6]

Metody zaimplementowane we frameworku można bezproblemowo testować za pomocą narzędzi testów automatycznych Javascript, takich jak Karma [7][8].

Dodatkowo dla AngularJS możliwe jest utworzenie profilu UML w celu projektowania aplikacji sterowanych modelami. Tak zaimplementowany projekt wymaga jedynie wypełnienia formularza generującego gotową aplikację. 87% kodu pokrywa się z ustalonymi koncepcjami co skraca czas programowania [9].

3. React.js

React klasyfikowany jest również do bibliotek JS. Ma on jednak bardzo specyficzne cechy frameworka. Jego uporządkowanie i hierarchia powoduje, że programista ma z góry narzuconą drogę implementacji swojej aplikacji w tym narzędziu. Tak jak w Angular tutaj również istotną rolę posiadają komponenty [10].

Komponenty są obiektami klasy, która dziedziczy po *React.Component*. Każdy moduł musi zawierać funkcję *render()*, która rysuje komponent w drzewie dokumentu. Zawierają się w niej zwykle szablony oraz inne subkomponenty. Te drugie układają się w kompozycje. Zaczynając od komponentu nadrzędnego, kończąc na subkomponentach małych elementów sterujących aplikacją. Postać komponentu może zawierać elementy jego stanu oraz parametry (ang. *props*) [10].

Parametry są dla danego modułu niezmienne, stanowią jego właściwości, które może zmieniać jedynie komponent nadrzędny o ile taki istnieje. Parametrem może być na przykład wielkość fontu napisu, liczba możliwych ocen na liście ratingowej. Stan jest zbiorem aktualnych, dynamicznych wartości przechowywanych w ciele komponentu. Jako przykład można podać aktualnie przeglądaną stronę książki elektronicznej czy chociażby stan rozgrywki w szachy (mat, szach).

Pisanie szablonów i wyrażeń w składni JSX może zastąpić tworzenie kodu bezpośrednio w języku JavaScript. Pozwala to uniknąć skomplikowanych struktur złożonych z zagnieżdżonych w sobie metod *React.createElement* [11]. Dzięki temu kod jest czytelny, łatwiejszy w modyfikacji i recenzji.

Składnia JSX przypomina połączony JavaScript z HTML. Widoki wypisywane są w postaci standardowych znaczników (plus znaczniki komponentów), ale można takie struktury przypisywać do zmiennych, tablic i pól obiektów [11].

4. Wybór odpowiedniego szkieletu

Istotnym czynnikiem w wyborze szkieletu są przede wszystkim jego koszty, programiści wolą polegać na darmowych narzędziach. Framework powinien być jak najbardziej rozszerzalny za pomocą prostej modyfikacji kodu. Aplikacja na nim oparta, powinna zajmować mało przestrzeni dyskowej i wykonywać szybko swoje zadania [12].

Istotne są również:

1) modułowość;
2) przenośność;
3) możliwość manipulacji DOM (Obiektowy Model Dokumentu, *ang. Document Object Model*);
4) czytelność kodu;
5) częste aktualizacje [12].

5. Metoda i platforma testowa

Aplikacje zostały sprawdzone pod kątem prędkości wykonywanych operacji – wstawiania, sortowania oraz usuwania jednego lub wielu elementów przechowywanych w tablicy obiektów JavaScript. Operacja usuwania pojedynczego wiersza sprowadzona została do usunięcia środkowego elementu listy. Badanie zostało wykonane na komputerze typu laptop z procesorem Intel® Core™ i7-6700HQ 2.60GHz w przeglądarce Google Chrome. Ta sama przeglądarka posiada wygodne narzędzie deweloperskie które ułatwiło pomiar czasu (od wyprowadzenia żądania do finalnej metody rysującej).

Obie aplikacje korzystały z tej samej bazy danych (przechowywanej w pliku tekstowym), zawierającej 100000 rekordów. Każdy rekord składał się z trzech kolumn zawierających ciągi znaków. Baza danych została pobrana do tablicy obiektów JavaScript. Każda z aplikacji pobierała rekordy z tablicy do mniejszej i wyświetlała zawartość

nowej macierzy w postaci tabeli HTML (funkcjonalność wstawiania rekordów). Sortowanie oraz usuwanie odbywa się na wyświetlonej porcji danych. Kasowanie dzieliło się na wymazywanie wszystkich elementów oraz usuwanie pojedynczego wiersza (rekordu ze środka utworzonej tablicy).

Listing 1. Wyświetlanie tabeli rekordów – Angular

```
<table>
<thead>
<tr>
<th>id</th>
<th>gender</th>
<th>phone</th>
</tr>
</thead>
<tbody>
<tr *ngFor='let item of records'>
<td>{{item.index}}</td>
<td>{{item.gender}}</td>
<td>{{item.phone}}</td>
</tr>
</tbody>
</table>
```

Listing 2. Wyświetlanie tabeli rekordów - React

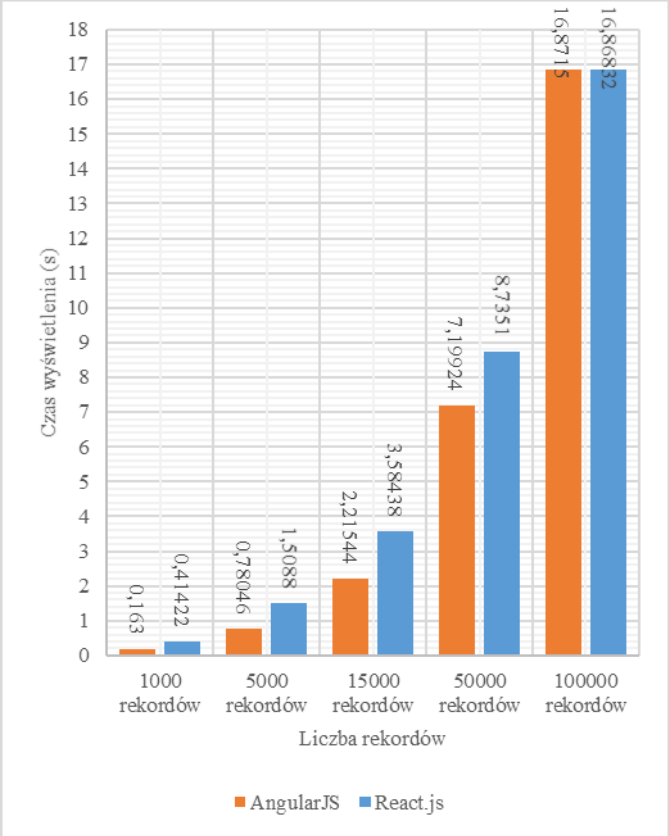
```
<table>
<thead>
<tr>
<th>id</th>
<th>gender</th>
<th>phone</th>
</tr>
</thead>
<tbody>
{this.state.records}
</tbody>
</table>
```

Listing 3. Metoda sortująca

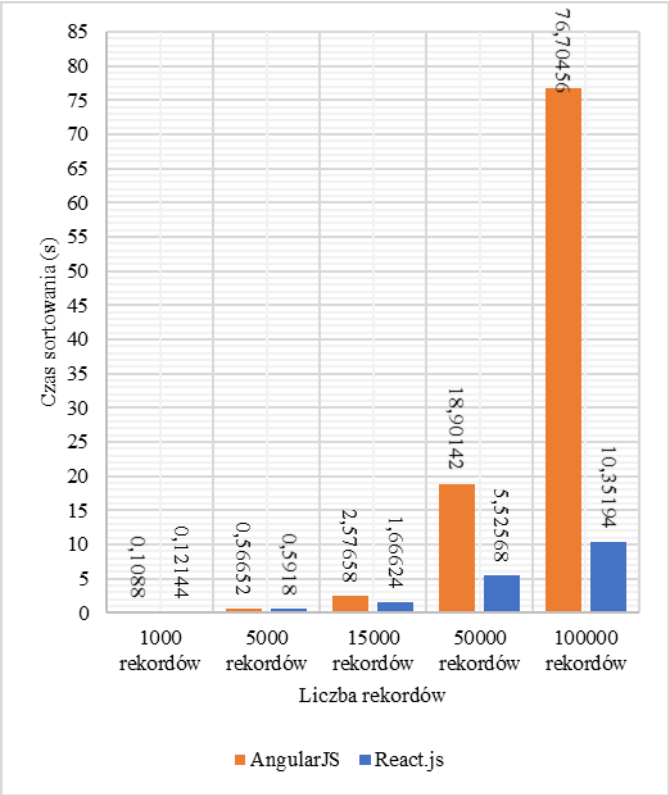
```
var newrecords = this.records.sort(function (a, b) {
var a_key = parseInt(a.index);
var b_key = parseInt(b.index);

if (a_key < b_key)
return -1 * order;

if (a_key > b_key)
return 1 * order;

return 0;
});
```

6. Wyniki badań

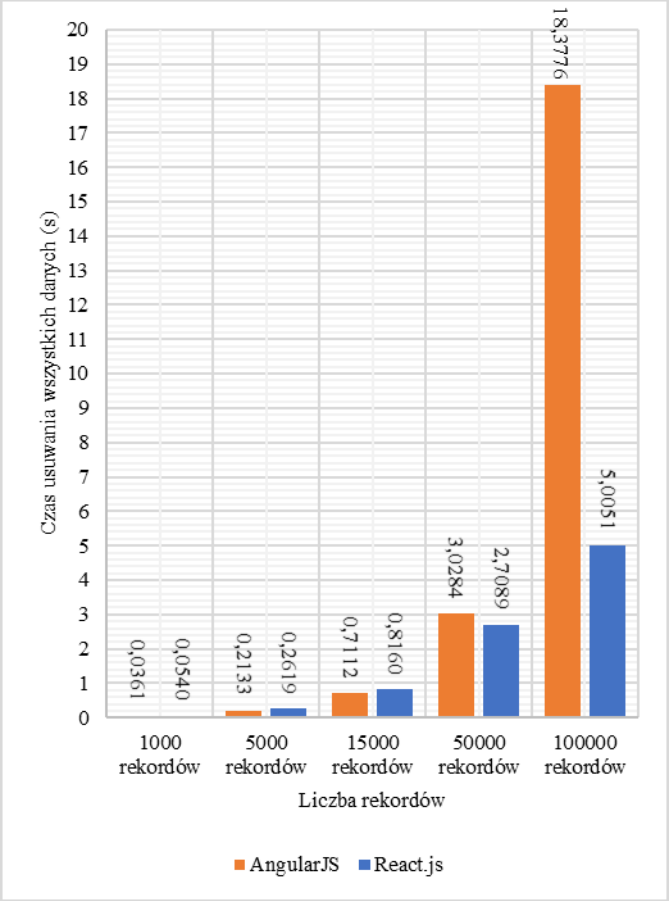
Badania zostały przeprowadzone pod kątem prędkości wykonywanych operacji. Zadaniem aplikacji było wczytywanie, sortowanie i usuwanie dużej liczby rekordów (lub pojedynczego rekordu) z tablicy JavaScript zawierającej łącznie 100000 wierszy. Każda z aplikacji musiała wyświetlić, posortować, usunąć jeden lub wszystkie elementy załadowane poprzez przyciski sterujące działaniem aplikacji. Wyniki zostały zaprezentowane zbiorczo na rysunkach od 1 do 4.



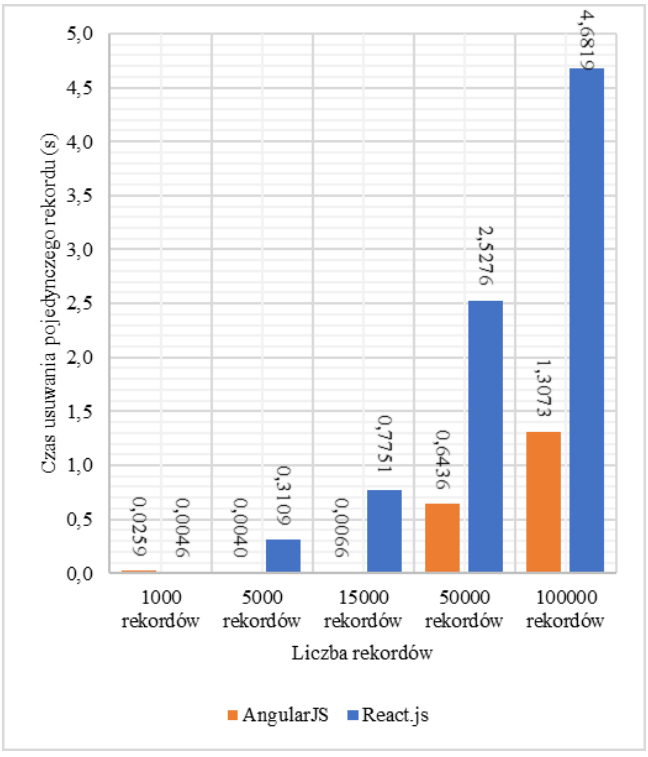
Rys. 1. Wykres średniego czasu dla operacji ładowania rekordów



Rys. 2. Średnie czasy sortowania dla każdej serii badanych rekordów



Rys. 3. Uśrednione czasy usuwania wszystkich rekordów dla obu szkieletów



Rys. 4. Uśrednione czasy dla operacji usuwania jednego rekordu

7. Dyskusja wyników.

Rysunek 1 pokazuje przewagę Angular dla czterech pierwszych zestawów rekordów. Dla liczby stu tysięcy czas wstawiania jest krótszy dla frameworka React. Mimo gorszego wyniku drugiego frameworka nie można jednoznacznie stwierdzić że taka minimalna różnica byłaby odczuwalna przez użytkownika w standardowym używaniu aplikacji – zwykle wprowadza się podział na strony przez co można uniknąć opóźnień w wyświetlaniu dużej liczby danych. Zbliżony wynik dla 100000 rekordów może świadczyć o obciążeniu maszyny testowej. Manipulacja DOM w takim zakresie jest kosztowna czasowo, niezależnie od stosowanej technologii. Platforma, na której działa aplikacja ma znaczenie również w przypadku tysiąca wierszy – niewielkie zmiany wynikają z działania samej przeglądarki, systemu operacyjnego, pracy dysku twardego.

Listing 4. Implementacja metody kopiującej obiekt

```
copy(current_object: any) {
var object_copy = current_object;
if (current_object && typeof current_object === "object") {
object_copy =
Object.prototype.toString.call(current_object) === "[object
Array]" ? [] : {};
for (var property in current_object) {
object_copy[property] =
this.copy(current_object[property]);
}
}
return object_copy;
}
```

Średnie czasy sortowania (rysunek 2) dużej liczby rekordów pokazują że Angular radzi sobie kilkukrotnie gorzej od React. Zmiany kolejności elementów w Angular.js przeprowadzane są bezpośrednio na widoku – kilkukrotnie uruchamiana jest iteracja w celu ponownego wypisania elementów za pomocą *ngFor* (Listing 1). React zmieniając stan komponentu tworzy tak zwane *deep copy* obiektu, czyli kopiuje wszystkie jego atrybuty przez wartość, nie referencję. Pozwala to na niemal natychmiastową podmianę zawartości listy. Angular w wersji 2 i wyższej nie posiada wbudowanego kopiowania (wcześniej istniała funkcja *copy*). Implementacja własnej metody kopiującej (Listing 4) pomogła skrócić czas sortowania do około 30 sekund, mimo to był to wynik 3 razy wolniejszy niż ten osiągnięty przez React. Dodatkowo takie rozwiązanie nie kwalifikowało się jako poprawny test, ponieważ używane jest dodatkowe obejście, nie zaimplementowane bezpośrednio w danym frameworku.

Taki sam problem istnieje w przypadku usuwania wszystkich rekordów. Rysunek 3 pokazuje, że średnie czasy wykonywania tej operacji są krótsze dla React dla liczby rekordów [50000; 100000]. Kolejny raz w React zawartość listy podmieniana jest na jej kopię, w tym wypadku pustą. Angular wykonuje dodatkowe iteracje co spowalnia cały proces.

Angular osiąga krótsze średnie czasy dla usuwania pojedynczego rekordu. Rysunek 4 pokazuje że React jest szybszy jedynie dla 1000 rekordów. Może to być

spowodowane pracą maszyny testowej oraz czynnikami niezależnymi od aplikacji. Dla kolejnych serii danych jest kilkukrotnie wolniejszy. Kopiowanie obiektu podczas zmiany stanu powoduje powstanie nadmiaru wykonywanych operacji – potrzebne jest ponowne uzupełnienie listy z wyłączeniem usuwanej pozycji. Angular jedynie modyfikuje wyświetlaną tabelę – usuwa element z drzewa dokumentu, jeżeli ten nie występuje już w modelu jego komponentu.

8. Wnioski.

Operacje na dużych zbiorach danych zawsze są kosztowne czasowo. Aplikacja działająca po stronie przeglądarki klienta powinna operować jedynie na porcjach danych, wprowadzić podział na strony których zawartość będzie ładowana z bazy danych tylko na wyraźne żądanie użytkownika. Rendering długich list powoduje spowolnienie pracy przeglądarki, zmniejszenie responsywności na akcje dokonywane przez korzystającego z aplikacji.

Angular oraz React dla małej liczby rekordów osiągają zbliżone wyniki, nie wpływające na efektywność użytkowania strony. Dopiero operowanie na wielkich listach pokazuje przewagę React – dla sortowania oraz usuwania dużej kolekcji. Mimo takiej prezentacji wyników używanie Angular do budowy aplikacji wcale nie jest niezalecane. Konieczna jest przemyślana implementacja funkcjonalności w aplikacji zarządzającej dużą liczbą danych. Bez odpowiedniego podejścia nawet bardzo szybkie frameworki, a nawet sam JavaScript nie są w stanie zoptymalizować zarządzania strukturą hipertekstu. Długie dokumenty HTML nigdy nie będą działały tak samo dobrze jak ich krótsze odpowiedniki, wynika to z natury samego języka – jest interpretowany przez przeglądarkę.

Warto korzystać z obu narzędzi, a wybór konkretnego zależy od potrzeb i preferencji. Dostarczają one kompletny zestaw funkcji do zbudowania dynamicznej aplikacji internetowej typu *single page*. Dzisiejsze możliwości wręcz wymuszają na producentach oprogramowania używanie tych narzędzi. Mimo swoich zalet i wad ułatwiają pracę i skracają czas potrzebny na zbudowanie skomplikowanego projektu.

9. Literatura

[1] Stępniak, W.; Nowak, Z.; Performance analysis of SPA web systems, Advances in Intelligent Systems and Computing 521, s. 235-247, 2017.

[2] 5 Best JavaScript Frameworks in 2017 https://da-14.com/blog/5-best-javascript-frameworks-2017 (dostęp 08.06.2017)

[3] Nowacki, R.; Plechawska-Wójcik, M.; Analiza porównawcza narzędzi do budowania aplikacji Single Page Application – AngularJS, ReactJS, Ember.js, Politechnika Lubelska, Lublin, Polska, 2016.

[4] Kumar, A.; Singh Kumar, R.; Comparative Analysis of AngularJS and ReactJS, International Journal of Latest Trends in Engineering and Technology, nr 7, s. 225-227.

[5] Angular Docs https://angular.io/docs/js/latest/index.html (dostęp 08.06.2017).

[6] Patel S. K. - Responsive Web Design with AngularJS, 2014.

[7] Fat, N.; Vujovic, M.; Papp, I.; Novak, S.; Comparison of AngularJS framework testing tools, Zooming Innovation in Consumer Electronics International Conference, ZINC 2016.

[8] Mesbah, A.; Chapter Five – Advances in Testing JavaScript-Based Web Applications, Advances in Computers, nr 97, s. 201-235, 2015.

[9] Chansuwath, W.; Senivongse, T.; A Model-Driven Development of Web Applications Using AngularJS Framework, IEEE/ACIS 15TH INTERNATIONAL CONFERENCE ON COMPUTER AND INFORMATION SCIENCE (ICIS), s. 683-688, 2016.

[10] Components, Props and State https://facebook.github.io/react-vr/docs/components-props-and-state.html (dostęp 08.06.2017)

[11] Vipul A M; Sonpatki, P.; ReactJS by Example – Building Modern Web Applications with React, 2016.

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