JCSI 3 (2017) 33-37
Journal of Computer Sciences Institute

Wysłane: 2016-11-29
Przyjęte: 2016-11-30

Zwiększenie efektywności procesu tworzenia aplikacji internetowych poprzez połączenie frameworków Meteor JS i Angular JS

Viacheslav Nishtuk*, Elżbieta Miłosz[a]
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Celem artykułu było przeprowadzenie analizy porównawczej frameworków AngularJS i MeteorJS, oraz próba ich połączenia w procesie tworzenia aplikacji internetowej Analiza porównawcza została wykonana na podstawie dokumentacji technicznej wybranych narzędzi oraz na podstawie oceny procesu tworzenia przykładowej aplikacji internetowej z ich udziałem. Wybrane zostały mocne strony każdego z narzędzi, zostały one wykorzystane w procesie tworzenia aplikacji internetowej, w którym połączono frameworki przydzielając każdemu z nich okresloną funkcjonalność aplikacji. Połaczenie frameworków miało na celu zwiększenie efektywności procesu wytwarzania aplikacji internetowej.

Słowa kluczowe: MeteorJS; AngularJS; JS - frameworki; połączenie frameworków.

*Autor do korespondencji.

Adres e-mail: viacheslav@ideaua.com

Increasing an efficiency of the web-applications developing the process through the combine of frameworks MeteorJS and AngularJS

Viacheslav Nishtuk*, Elżbieta Miłosz[a]
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The comparative analysis was made on the basis of technical documentation selected tools and on the process of creating a sample web application. Strengths of each tool were selected and have been used in the process of developing a web application that combines frameworks by assigning each of them a specific application functionality. Combining of frameworks aimed increasing the efficiency of the developing process of web application.

Keywords: MeteorJS; AngularJS; JS - frameworks; an unification frameworks.

*Corresponding author.

E-mail address: viacheslav@ideaua.com

1. Wstęp

Celem artykułu było przeprowadzenie analizy porównawczej dwóch obecnie popularnych frameworków do tworzenia stron internetowych w języku Java Script: AngularJS i MeteorJS, wyjaśnienie ich wad i zalet na podstawie oceny procesów tworzenia z ich pomocą aplikacji internetowej oraz próba ich połączenia tj. wykorzystania obydwu frameworków w jednej aplikacji. Połączenie miało na celu zwiększenie efektywności procesu wytwarzania aplikacji.

2. Metody i narzędzia tworzenia stron internetowych

Pierwszym ważnym zadaniem w procesie tworzenia strony internetowej jest planowanie jej układu zawartości i wyglądu . Proces ten można podzielić na kilka etapów: generowanie pomysłów, tworzenie struktury projektu, wybór technologii i badanie układu projektu. Przykładowy szkicowy projekt serwisu przedstawiono na rys.1.

Metody tworzenia strony WWW są podzielone na dwie grupy: ręczne pisanie kodu strony internetowej i automatyczne metody tworzenia strony internetowej z wykorzystaniem narzędzi typu framework [1]. Do drugiej grupy należą takie narzędzia jak Joomla i Wordpress, które z kolei pozwalają szybko stworzyć prostą stronę internetową bez znajomości programowania.

Technologie sieci Web, które umożliwiają tworzenie stron internetowych czy aplikacji sieci Web to [2]:

- Języki znaczników (HTML, XML);
- Kaskadowe arkusze stylów (CSS, SCSS, SASS, LESS);
- Język skryptowy JavaScript;
- Język skryptowy PHP;
- Obiektowy język programowania JAVA;
- Technologia NodeJS;

Narzędzia typu frameworki, są dostępne w różnych językach programowania, najczęściej wykorzystywane są js-frameworki lub php-frameworki.

3. Cel, hipoteza i metody badań

Celem badań było porównanie efektywności tworzenia prostej aplikacji internetowej w 2 frameworkach, ocena słabych i silnych stron frameworków oraz próba połączenia

tych frameworków w jednej aplikacji w celu. zwiększenia efektywności procesu wytwarzania aplikacji.

W artykule postawiono następujące *pytania badawcze*:

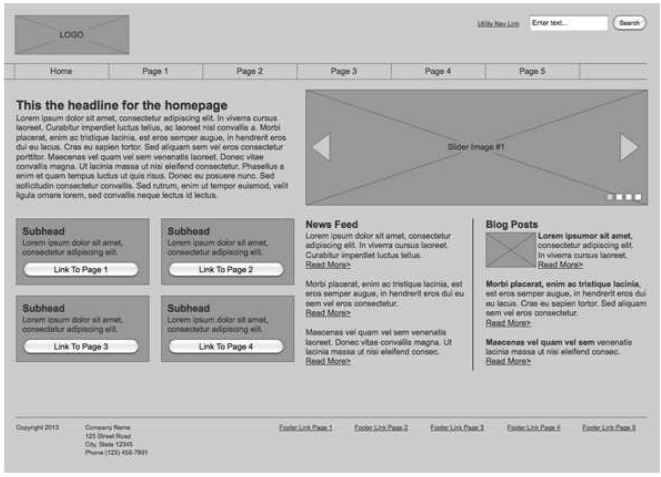
1. Który z frameworków łatwiej realizuje wybrane funkcje aplikacji?
2. Czy możliwe jest połączenie 2 frameworków w celu zwiększenia efektywności tworzenia aplikacji?

Dla uzyskania odpowiedzi na postawione pytania badawcze sformułowano następującą *hipotezę badawczą*:

Możliwe jest połączenie frameworków: Meteor JS i Angular JS w procesie tworzenia aplikacji internetowej.

Dla potwierdzenia hipotezy wykorzystano następujące *metody badań*:

- Analiza porównawcza frameworków
- Eksperyment połączenia frameworków.



Rys. 1. Szkicowy zarys serwisu [3]

4. Analiza porównawcza frameworków

Porównanie frameworków jest istotne dla społeczności programistów podczas podejmowania decyzji: "Jaki framework mam wybrać dla następnego projektu?". Każdy programista przed tym, jak ma zacząć korzystać z nowego frameworka, porównuje go z tym, który wcześniej używał.

Analizie porównawczej poddano dwa najbardziej popularne obecnie frameworki. Pierwszym był MeteorJS, który przyciąga dewelopera dostępnością, powłoką i logiką biznesową NodeJS. Drugim został wybrany AngularJS, który jest aktualnie najbardziej popularnym frameworkiem [4]. Wyniki porównania frameworków przedstawiono w tabeli 1.

W rzeczywistości wybór frameworka zależy od potrzeb programisty i zadań do realizacji, jedne frameworki mogą radzić sobie lepiej z określonym zadaniem, a inne, odpowiednio lepiej z innym zadaniem. Nie ma idealnego frameworka, wszystkie one koncentrują się na określonych sprofilowanych zadaniach.

5. Eksperyment połączenia frameworków

Eksperyment połączenia frameworków wykonano dla przykładowej aplikacji internetowej, w której użytkownik mógł samodzielnie tworzyć notatki, a przy tym uzyskać do nich dostęp z dowolnego urządzenia w dowolnym miejscu z dostępem do Internetu. Aplikacja powinna zapewnić możliwość ustawienia przypomnienia dla użytkownika, tworzenia notatek, udostępniania ich innym użytkownikom w postaci określonych grup notatek, albo konkretnych wybranych notatek.

Do projektowania witryny sieci Web wybrano szereg kryteriów, a mianowicie:

- prosty trójkolumnowy układ strony,
- zapewnienie logiki biznesowej (back-end) – szybka realizacja zapytań do bazy danych.

Tabela 1. Porównanie frameworków AngularJS i MeteorJS [5, 6, 7]

Kryterium porównania	**AngularJS**	**MeteorJS**
Full-stack framework	Nie, tylko MVC w kliencie	Tak
Logika biznesowa (Back-end)	Dowolny	NodeJS
Preprocesor	Nie	Tak
Dynamiczne połączenie html z danymi klienta	Tak	Tak
Renderowanie html na serwerze	Nie	Tak
Społeczność	Npm, Bower	Atmospherejs, Npm, Bower
Reaktywność	Działa tylko w $scope	Wszystko
Systememy szablonów	Oficjalnego wsparcia nie ma, ale można użyć innych	Oficjalne wsparcie Blaze, Handerbars, Jade
Baza danych	Wszystko zależy od Back-end (dowolna)	NodeJS obsługuje mongoDB, ale można użyć dowolnej
Synchronizacja danych między klientami	Nie	Są (Optimistic UI)
Kanał synchronizacji danych	Nie	DDP protokół (web-sockets)
Aktualizacja aplikacji bez konieczności ponownego uruchamiania (wygoda rozwoju)	Nie	Tak — html, css, js

Projekt układu strony przedstawiono na rys.2.

Nawigacja: -Logowanie -Notatki osobiste -Import/ export -Powiado-mienia -Notatki publiczne -Wyjście	Graficzne notatki	Szukaj  Lista notatek

Rys. 2. Projekt układu strony do eksperymentu

Do realizacji badania przedstawiony projekt został napisany w każdym z dwóch frameworków, a następnie utworzono aplikację, w której połączono 2 frameworki, które realizowały w optymalny dla siebie sposób wybrane funkcje. W aplikacjach zaimplementowano następujące funkcjonalności:

- autoryzację i rejestrację użytkowników (logowanie)
- ustawienia notatek osobistych
- import/export notatek
- powiadomienia o notatkach
- ustawienia notatek publicznych
- wyjście z aplikacji.

Realizację usługi autoryzacji i rejestracji do strony internetowej zrealizowaną w AngularJS, można zobaczyć na listingu 1. Logowanie i rejestracja działa na "websocket", który jest bezpośrednio podłączony do back-end. Tak jak większość funkcji witryny potrzebują oprogramowania logiki biznesowej, więc do AngularJS był podłączony NodeJS. AngularJS w odróżnieniu od MeteorJS ma wbudowany back-end (NodeJS synchroniczny). W MeteorJS podobną funkcjonalność realizowano w łatwiejszy sposób, ponieważ MeteorJS ma wbudowane moduły, które pracują z identyfikacją i rejestracją użytkownika (przykład 2).

Przykład 1. Usługa odpowiedzialna za autoryzację i rejestrację użytkownika w AngularJS [8]

```
angular.module('noteApp').service('AuthService',
function($rootScope, $cookies, $routeSegment, $location, mySocket)
{
     $rootScope.bool.isLoading    = true;
     this.status = {
    authorized: false,
   };
   this.check = function(data, callback){
     mySocket.emit('authCheck', $cookies.get('token'), function(data)
{
     if(data){
          $rootScope.hideBlocks = true;
               callback(self.status.authorized = true);
          }else{
               $rootScope.hideBlocks = false;
               callback(self.status.authorized = false);
          }
               $rootScope.bool.isLoading    = false;
          });
   }
   this.logIn = function(key, callback){
     mySocket.emit('authLogin', {key: key}, function(data) {
               if(data.token){
                    $cookies.put('token', data.token);
                    $rootScope.hideBlocks = true;
                    callback(self.status.authorized = true);
               }else{
                    $rootScope.hideBlocks = false;
                    callback(self.status.authorized = false);
               }
               $rootScope.bool.isLoading    = false;
          });
   }
   this.register = function(key, callback){
     mySocket.emit('authRegister', {key: key}, function(data) {
               if(data.token){
                    $cookies.put('token', data.token);
                    $rootScope.hideBlocks = true;
                    callback(self.status.authorized = true);
               }else{
                    $rootScope.hideBlocks = false;
                    callback(self.status.authorized = false);
               }
               $rootScope.bool.isLoading    = false;
          });
   }
})
```

Po wykonaniu aplikacji w dwóch frameworkach podsumowano doświadczenie z procesu tworzenia aplikacji, oceniono, który framework pozwalał w łatwiejszy sposób realizować funkcje aplikacji. Duża różnica między AngularJS i MeteorJS - to obecność gotowego „back-end-a” w MeteorJS - NodeJS (synchroniczny). Dlatego napisanie tej funkcji z wykorzystaniem AngularJS zajmuje więcej czasu niż z wykorzystaniem MeteorJS. MeteorJS dobrze współpracuje z bazą danych z frond-end ' em i dlatego napisanie funkcjonalności do tworzenia czy edycji notatek w MeteorJS jest łatwiejsze.

W kolejnym kroku podjęto próbę połączenia frameworków rozdzielając funkcje aplikacji między dwa frameworki. MeteorJS zajmie się pracą z bazą danych i wszystkim co dotyczy logiki biznesowej (back-end), a AngularJS będzie działał z warstwą widoku i prezentacją danych (front-end)).

Przykład 2. Fragment kodu rejestracja / logowanie użytkownika w MeteorJS

```
Template.formRegist.events({ //meteor method
 'click #register': function (event, template) { //register
     var user = {
     login: $(event.currentTarget).children('input#login').val(),
     password: $(event.currentTarget).children(#password').val()
     };
     if(validFormInput(user)){
          //meteor method
          Accounts.createUser(user, function(error, result) {
               if(error){
                    thow error;
               }else if(result){
                    showMessageDisplay('jestes zarejestrowany');
                    initNote(user);
               }
          });
     }else{
          showMessageDisplay('invalid data');
     }
 },
 'click #login': function(event, template){ // login
     var user = {
     login: $(event.currentTarget).children('input#login').val(),
     password:$(event.currentTarget).children('#password').val()
     };
     if(validFormInput(user)){
          // meteor method
          Meteor.loginWithPassword(user.login, user.password, f
               function(error, result) {
                    if(error){
                         thow error;
                    }else if(result){
                         showMessageDisplay(`welcome’);
                         initNote(user);
                    }
               });
     }else{
          showMessageDisplay('invalid data');
     }
 }
});
```

Do zadań realizowanych za pomocą MeteorJS należy:

- logika aplikacji;
- render szablonów;
- routing między szablonami;
- praca z bazą danych;
- RestAPI.

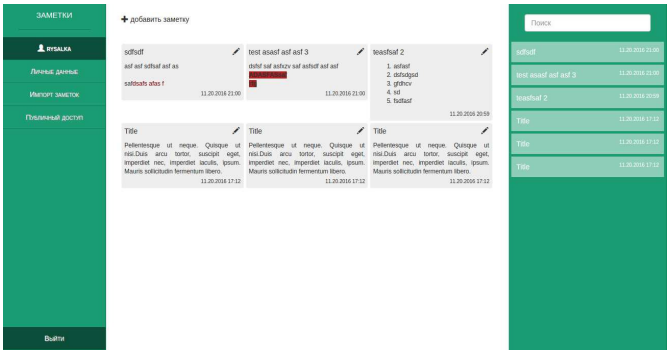
Z kolei AngularJS będzie realizował następujace funkcje:

- data binding $scope;
- traktowanie niektórych zdarzeń (zmiany tekstu w czasie rzeczywistym).

Po eksperymencie przeprowadzono analizę porównawczą stosowanych frameworków, zarówno między sobą, jak i z wynikiem ich połączenia.

Realizacja funkcji rejestracji/logowania i innych funkcji nawigacji strony najbardziej przyjazna dla programisty jest z wykorzystaniem frameworka MeteorJS. Podczas tworzenia strony internetowej w MeteorJS powstało znacznie mniej problemów niż w AngularJS, gdzie bardzo dużo problemów było z wykorzystaniem wbudowanych metod, a dokładniej ze słabą do nich dokumentacją. Do wyszukiwania i usuwania błędów przeprowadzono ponowne debugowanie całego kodu, na co poświęcono dodatkowy czas.

Wygląd końcowy aplikacji powstałej na podstawie projektu i wykonanej w 3 frameworkach przedstawia rys.3.



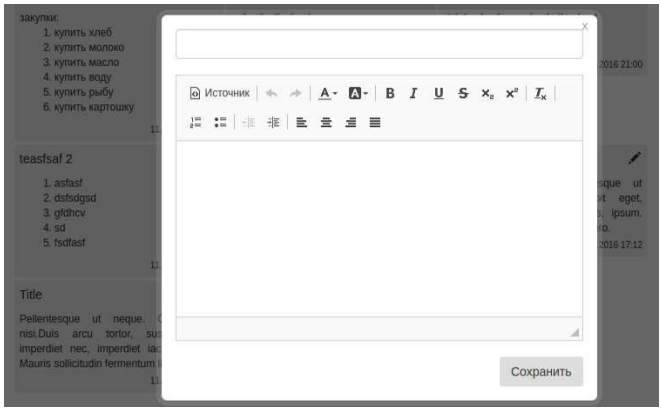
Rys. 3. Wygląd aplikacji dla notatek

Do edycji tekstu notatek był używany edytor tekstu CKEditor v4.6.0, który z kolei był nastawiony na konkretne zadania, konkretne funkcje, posiadał pasek narzędziowy do łatwiejszej edycji tekstu (Rys.4).

6. Rezultaty badań i wnioski

W wyniku badań zostały zidentyfikowane zalety i wady każdego z wybranych frameworków i funkcje, które wykonują bez zarzutu. Właśnie dla tego przeprowadzono połączenie frameworków w jednej aplikacji, aby przy minimalnym nakładzie pracy i czasu móc w pełni zrealizować wszystkie wymagane funkcje.

Hipoteza badawcza pracy: „Możliwe jest połączenie frameworków: Meteor JS i Angular JS w procesie tworzenia aplikacji internetowej" została potwierdzona.



Rys. 4. Formularz tworzenia notatek

Aby uniknąć konfliktów, podjęto decyzje podzielić kod między frameworkami. Kod AngularJS był używany do importu określonej funkcjonalności do głównego kodu MeteorJS, czyli części poszczególnych funkcjonalności znajdują się w różnych plikach, w różnych katalogach (struktura aplikacji jest podzielona między frameworkami), przy tym są całkowicie odizolowane od siebie.

Podczas realizacji scalenia frameworków okazało się dość trudne rozdzielić funkcje frameworków między sobą według łatwości i jakości wykonania. Została przeprowadzona szczegółowa praca nad dokumentacją MeteorJS, ponieważ wystąpił problem z izolacją: MeteorJS nie wymaga podania jaki plik ma w użyciu, automatycznie przetwarza wszystkie dostępne pliki na raz, tworząc z nich "bundle", w którym znajdują się jednocześnie kod MeteorJS i AngularJS. Przy tym powstały problemy z powodu braku możliwości zastosować wybrane moduły funkcjonalne określonego kodu MeteorJS. Problemy rozwiązano w postaci importu i izolacji AngularJS od zewnętrznych manipulacji. Przy prawidłowym pisaniu kodu, frameworki nie wchodziły w konflikt i nie przeszkadzały sobie nawzajem, każdy obsługiwał swoją część kodu. W rezultacie w aplikacji zostały zrealizowane wszystkie projektowane funkcje, przy tym nie ma strat w zakresie projektowania i pozycjonowania strony.

Literatura

[1] Сэмми Пьюривал, Основы разработки веб-приложений, Питер, 2015.

[2] Веб-технологии для разработчиков, https://developer.mozilla.org/ru/docs/Web

[3] Создание веб-сайта. Курс молодого бойца, https://habrahabr.ru/post/273795/

[4] ТОП 10 JavaScript фреймворков и библиотек, https://habrahabr.ru/post/305442

[5] Brad Green & Shyam Seshadri, AngularJS. O'Reilly Media, 2013.

[6] Tom Coleman, Sacha Greif, Discover Meteor: Building Real-Time JavaScript Apps. N/A, 2014.

[7] Kyle Simpson, You Don't Know JS: Up & Going, 2015

[8] David Flanagan, JavaScript: The Definitive Guide, 6th Edition. Activate Your Web Pages, 2011