JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 10 (2019) 71-76
Wysłane: 2018-11-29
Przyjęte: 2018-12-07

Porównanie nowych możliwości tworzenia aplikacji PHP na przykładzie Laravel i CodeIgniter

Daniel Drabik*
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Artykuł ten przestawia wyniki porównania możliwości tworzenia aplikacji internetowych korzystając z języka PHP przy użyciu szkieletów budowy aplikacji – Laravel i CodeIgniter. Tekst wykazuje różnice dwóch implementacji między innymi w sposobie wytworzenia, aspektach architektury i wydajności.

Słowa kluczowe: laravel; codeigniter

*Autor do korespondencji.

Adres e-mail: daniel.drabik@outlook.com

Comparison of new ways of creating PHP applications using Laravel and CodeIgniter example

Daniel Drabik*
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Abstract. This article presents a comparision of different ways of creating PHP web applications using Laravel and CodeIgniter frameworks. The text shows differences of two implementations including ways of creation, architecture aspects and performance.

Keywords: laravel; codeigniter

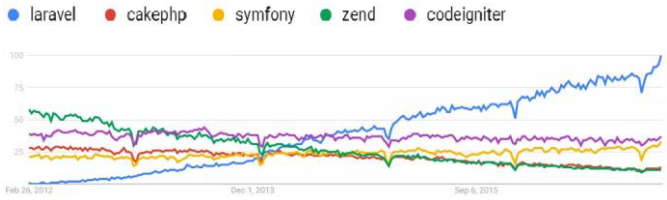
*Corresponding author.

E-mail address: daniel.drabik@outlook.com

1. Wstęp

Ideą tego artykułu jest porównanie dwóch niezależnych frameworków do tworzenia aplikacji internetowych w języku PHP – Laravel i CodeIgniter. Wybór został uwarunkowany popularnością zapytań w wyszukiwarce Google [1] (Rys. 1) oraz brakiem podobnych prac zawierających szczegółowe zestawienie tych narzędzi.

Istnieją artykuły, które starają się wskazać różnice pomiędzy obydwoma frameworkami. Nie uwzględniają one jednak badań nad wydajnością lub testów symulujących zwiększony ruch użytkowników [2-4].



Rys. 1. Wyniki zapytań poszczególnych frameworków PHP w wyszukiwarce Google na przestrzeni pięciu lat

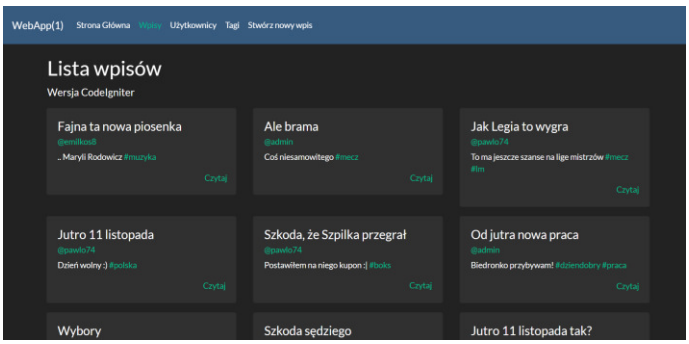
2. Cel i zakres badania

Celem badania jest porównanie wydajności oraz sposobów wytwarzania aplikacji przy użyciu frameworka Laravel i CodeIgniter. Do wykonania testów zostały utworzone dwie aplikacje testowe posiadające taką samą szatę graficzną oraz funkcjonalności. Różnicą jest szkielet budowy aplikacji. Projekty korzystają ze wspólnej bazy danych oraz środowiska testowego.

3. Aplikacje testowe

Aplikacja wykonana przy użyciu obydwu frameworków jest prostym dziennikiem internetowym – mikro blogiem. Podstawową funkcją jest tworzenie wpisów składających się z tytułu, treści oraz tagów tematycznych. Dodatkowo można wyświetlić wszystkie posty napisane przez danego autora.

Możliwe jest również pokazanie wszystkich wpisów, użytkowników oraz tagów znajdujących się w bazie danych. W każdej z podstron jest możliwość przejścia do pojedynczego elementu. Szata graficzna strony została maksymalnie uproszczona. W tym celu wykorzystano darmowy szablon Bootstrap (Rys. 2).



Rys. 2. Widok prezentujący listę wszystkich wpisów

4. Analiza porównawcza

Najważniejsze cechy Laravel i CodeIgniter zostały zaprezentowane w tabeli 1.

Tabela 1. Właściwości frameworków

	Laravel	**CodeIgniter**
Rok wprowadzenia	9 czerwca 2011	28 lutego 2006
Aktualna wersja i rok wydania	5.7 (04.09.2018)	3.1.9 (12.06.2018)
Instalacja i konfiguracja	Wymagane użycie Composera do pobrania instalatora. Konfiguracja polegająca na edycji pliku php.	Rozpakowanie archiwum. Konfiguracja polegająca na edycji pliku php.
Waga frameworka [MB]	30 MB	11 MB
Obsługiwane bazy danych	MySQL, SQLite, PostgreSQL, SqlSrv	MySQL, PostgreSQL, MS SQL, SQLite, CUBRID, Oracle, Interbase/Firebird, ODBC
Stosowany ORM	Eloquent	ActiveRecord
Wzorzec projektowy	Model-View-Controller	Model-View-Controller
Zabezpieczenia	Zabezpieczenie przed SQL Injection, filtrowanie XSS, ochrona przed atakiem CSRF, system autentykacji	Zabezpieczenia URI, filtrowanie XSS, ochrona przed atakiem CSRF
Silniki szablonów (widok)	Blade	Brak dedykowanego silnika (czysty kod PHP + HTML)
Licencja	X11 (MIT)	X11 (MIT)

4.1. Instalacja i konfiguracja

Proces instalacyjny CodeIgniter jest prosty i intuicyjny. Polega on na pobraniu najnowszej wersji ze strony twórców [5]. Następnie należy rozpakować zawartość otrzymanego archiwum do głównego katalogu projektu. Kolejnym krokiem jest zdefiniowanie adresu URL aplikacji. Odbywa się to w pliku *application/config/config.php*. Należy zmodyfikować wartość elementu tablicy *$config*:

```
$config['base_url'] = 'http://localhost/ci';
```

Kolejną czynnością jest zdefiniowanie połączenia z bazą danych. W tym celu należy uzupełnić tablicę *$db* znajdującą się w pliku *application/config/database.php* o dane takie jak host, nazwa bazy, login i hasło użytkownika oraz port. Tablica *$db* posiada domyślnie klucz *default*. Jest to identyfikator połączenia z bazą danych. Możliwe jest utworzenie wielu różnych połączeń pod innymi kluczami tablicy. Edytując wartość zmiennej *active_group* o odpowiedni identyfikator zostaje wskazane połączenie, z którego aplikacja będzie obecnie korzystać.

Laravel do instalacji wykorzystuje narzędzie Composer, czyli system zarządzania pakietami dedykowany dla języka PHP. Jest to jeden z wymogów Laravel. W przypadku braku narzędzia Composer, należy go zainstalować i skonfigurować na docelowej maszynie. Zwiększa to złożoność całego procesu instalacji w porównaniu do CodeIgniter. Posiadając skonfigurowany Composer należy pobrać instalator dla Laravel wykorzystując komendę:

```
composer global require "laravel/installer"
```

Następnie w docelowym katalogu projektu wykonać polecenie:

```
laravel new app1
```

Instalator utworzy nowy projekt, który zostanie nazwany *app1*. Laravel pozwala uruchomić serwer lokalny pod adresem *http://localhost:8000*, korzystając z interfejsu Artisan za pomocą komendy:

```
php artisan serve
```

Do obsługi aplikacji testowej nalezy ustawić połączenie połączenie z bazą danych oraz zdefiniowania adresu URL aplikacji. Konfiguracja ta przebiega poprzez edycję pliku *.env*. W odróżnieniu od CodeIgniter obie te czynności wykonywane są w tym samym miejscu.

4.2. Struktura plików

Struktura plików i katalogów obu projektów została ukazana na rysunku 3.

W CodeIgniter folder *application* jest odpowiedzialny za przechowywanie kodu logiki aplikacji. Wszystkie funkcjonalności oraz metody powinny znaleźć się właśnie w tym folderze. Katalog ten zawiera w sobie kilka podkatalogów:

- config – posiada pliki konfiguracyjne aplikacji np. ustawienia połączenia bazy danych;
- controllers – przechowuje kontrolery. Jest to podstawowa część aplikacji;
- hooks – służy do umieszczania kodu rozbudowującego oraz modyfikującego podstawowe funkcjonalności frameworka;
- libraries – przechowuje biblioteki;
- models –zawiera modeli aplikacji;

- views – odpowiada za przechowywanie widoków projektu.

Katalog *system* zawiera pliki kluczowe do działania aplikacji. Jest to rdzeń całego szkieletu frameworka.

Rys. 3. Struktura projektów

Każdy katalog w projekcie Laravel jest odpowiedzialny za inny aspekt infrastruktury aplikacji, a jego nazwa jest logicznie dopasowana do zadania:

- app – miejsce przechowywania aplikacji. Zawiera pliki obsługujące komendy konsolowe, zdarzenia, wyjątki, kontrolery, usług, modele oraz filtry i klasy żądań;
- bootstrap – skrypty uruchomieniowe frameworka;
- config – pliki konfiguracyjne aplikacji oraz klasy obsługujące bazy danych i ich migracje;
- public – katalog rdzenia projektu. Zawiera plik konfiguracji .htaccess, favicon strony, pliki JavaScript, kaskadowe arkusze stylów oraz *index.php*, który uruchamia aplikację;
- resource – katalog przechowujący pliki językowe oraz widoki projektu;
- routes – zawiera zasady routingu aplikacji;
- storage – miejsce magazynowania pamięci podręcznej, automatycznie generowanych logów oraz plików przesyłanych od strony użytkownika;
- tests – katalog zawierający przypadki testowe;
- vendor – zewnętrzny kod pobierany przy użyciu Composera.

Ważnymi elementami struktury projektu Laravel są niektóre pliki znajdujące się w głównym folderze projektu:

- .env – konfiguracja zmiennych środowiskowych projektu;
- artisan – interfejs konsolowy Artisan;
- composer.json – manifest zależności projektu;
- phpunit.xml – konfiguracja PHPUnit do obsługi testów jednostkowych;
- server.php – lekki serwer do lokalnego rozbudowywania aplikacji.

4.3. Routing

Edycja routingu w CodeIgniter polega na modyfikacji pliku znajdującego się w *application/config/routes.php*. Umiejscowiona jest w nim tablica *$route*, w której można dostosowywać zasady routingu. Klucz tablicy odpowiada ścieżce URL, która zostanie przechwycona, natomiast wartość tablicy decyduje o wywołaniu konkretnej metody kontrolera.

W Laravel, routing jest definiowany w czterech różnych plikach – *web.php*, *api.php*, *channels.php* oraz *console.php*. Znajdują się one w katalogu *routes* i są automatycznie inicjalizowane przez framework. Każdy z plików odpowiada za routing innego interfejsu aplikacji. Do obsługi aplikacji testowej został wykorzystany jedynie interfejs webowy.

Ustalanie zasad routingu polega na korzystaniu z metod obiektu klasy *Route*. Dostępne podstawowe funkcje to *get()*, *post()*, *patch()*, *put()*, *delete()* i *options()*. Każda z metod odpowiada innemu żądaniu przeglądarki. Wszystkie z nich przyjmują natomiast dwa parametry – ścieżkę *uri* i *callback* czyli wywołanie funkcji zwrotnej.

4.4. Kontroler

W CodeIgniter utworzenie kontrolera wiąże się z wygenerowaniem nowej klasy, która musi dziedziczyć po bazowej klasie *CI_Controller* i umieszczeniem jej w katalogu *application/controllers*. Ważnym krokiem jest dodanie w konstruktorze wywołania konstruktora rodzica klasy. Dodatkowo można wprowadzić w nim inicjalizację potrzebnych bibliotek i modeli.

Definiowanie kontrolerów w Laravel odbywa się w katalogu *app/Http/Controllers*. Podobnie jak w CodeIgniter, dobrą praktyką jest utworzenie klasy, która dziedziczy po wbudowanej klasie o nazwie *Controller*. Nie jest to co prawda wymagane, ale zyskuje się dostęp do różnych ciekawych

metod, takich jak na przykład *middleware()*, *validate()*, *dispatch()*.

4.5. Model

W CodeIgniter tworzenie modelu polega na wygenerowaniu nowej klasy, która dziedziczy po bazowej klasie *CI_Model*. Jest to analogiczne z procesem wytwórczym kontrolera w tym frameworku. Klasę należy umieścić w katalogu *application/models*. Modele w CodeIgniter posiadają rozbudowaną listę metod ułatwiającą otrzymywanie i wprowadzanie danych.

Istnieje możliwość nadania statusu globalnego modelowi. Oznacza to, iż będzie on inicjalizowany dla wszystkich kontrolerów podczas uruchamiania się silnika frameworka. W celu ustawienia takiej opcji należy dodać nazwę modelu do tablicy automatycznego włączenia, która znajduje się w *application/config/autoload.php*.

Model w Laravel wykorzystuje dziedziczenie po klasie *Model*. Ma to związek z dostarczanymi funkcjami zapewniającymi swobodne pobieranie informacji z bazy danych. Przykładem takiej metody jest *all()*. Funkcja ta wywołana na klasie modelu *Post* stworzonej do obsługi wpisów, pobiera wszystkie wiersze z bazy danych w tabeli *posts*. Laravel nie prosi o podanie tabeli, z której mają zostać uzyskane informacje. Przyjęto zasadę, iż nazwa tabeli w bazie danych powinna składać się z nazwy klasy modelu oraz dołączonej na końcu litery „s".

Miejsce magazynowania modeli to katalog *app*. Operacje na bazie danych mogą być wykonywane bezpośrednio przy użyciu klasy *DB*, która pozwala w łatwy sposób zarządzać danymi.

4.6. Widok

W przeciwieństwie do innych popularnych frameworków w CodeIgniter nie wykorzystano żadnego znanego systemu szablonów, jak na przykład Twig czy Smarty. Wyświetlanie danych, tworzenie pętli raz instrukcji warunkowych przebiega przy użyciu wbudowanej składni języka PHP jaka jest dostępna w aplikacjach nie zawierających frameworków lub zewnętrznych bibliotek.

Laravel magazynuje swoje widoki w katalogu *resources/views*. Do obsługi wykorzystuje dedykowany silnik szablonów – Blade. W odróżnieniu od rozwiązań innych frameworków (na przykład Twig), pomimo posiadania własnej składni - nie istnieje zakaz wykorzystywania kodu napisanego w PHP [6]. Ponadto wszystkie widoki są kompilowane do postaci czystego kodu PHP. Dla swoich plików Blade korzysta z rozszerzenia *.blade.php*.

4.7. Porównanie wybranych metryk kodu

W tabeli 2 zostały porównane długości i wagi przykładowych komponentów obu aplikacji testowych.

Tabela 2. Metryki kodu

		Laravel	**CodeIgniter**
Kontroler wpisów	Liczba linii kodu	83	93
	Liczba znaków	2162	3330
	Rozmiar w KB	2.11	3.32
Model wpisów	Liczba linii kodu	49	111
	Liczba znaków	1210	3435
	Rozmiar w KB	1.18	3.74
Rozmiar aplikacji w KB		20.3	170
Rozmiar projektu w MB		37.9	14.6

5. Przebieg badania

Aplikacje testowe zostały uzupełnione o metody, które mierzą czas wykonywania danej operacji:

- uruchamiania strony powitalnej;
- wprowadzania dużej liczby wpisów;
- generowanie strony internetowej składającej się z dużej liczby elementów (wpisów).

Wszystkie pomiary czasu przeprowadzone zostały na dwóch niezależnych maszynach – komputerze stacjonarnym (PC) i laptopie, które posiadają różne wersje oprogramowania zestawione w tabeli 3.

Tabela 3. Wersje zastosowanego oprogramowania

	PC	**Laptop**
Serwer Apache	2.4.29	2.4.34
PHP	7.1.15	7.2.10
Serwer MySQL	5.6.36-83.0	5.6.39-83.1

Porównanie specyfikacji obu maszyn zostało zaprezentowane w tabeli 4.

Tabela 4. Specyfikacja sprzętowa

	PC	**Laptop**
Procesor	Intel Pentium G4560 2 x 3,50 GHz	Intel Core i5-3210M 2 x 2,50 GHz
Pamięć RAM	8 GB	8 GB
Dysk	550 MB/s dla odczytu 540 MB/s dla zapisu	555 MB/s dla odczytu 510 MB/s dla zapisu

Do mierzenia czasu została wykorzystana wbudowana funkcja PHP *microtime()*, która zwraca obecną godzinę w formie mikrosekund. Odejmując wartość przed i po wykonaniu operacji otrzymuje się liczbę sekund jaką potrzebował skrypt na jej realizację. Każdy test został powtórzony 5 razy, natomiast wynik końcowy to średnia z wszystkich prób.

5.1. Czas generowania strony

Strona powitalna obu aplikacji nie wykorzystuje żadnych informacji pochodzących z bazy danych. W tabeli 5 pokazano pomiary czasu generowania się dokumentów HTML.

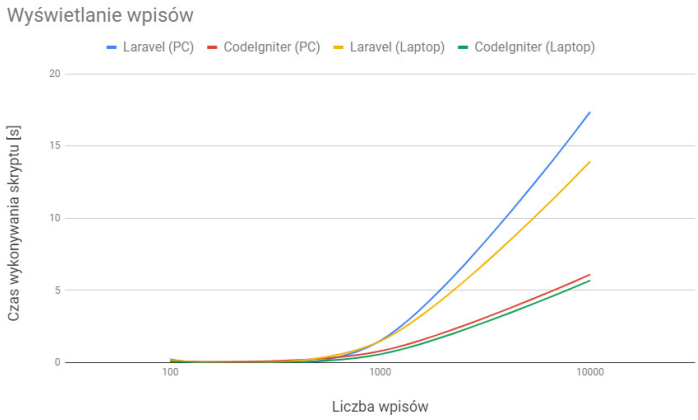
Tabela 5. Czasy generowania strony powitalnej

	CodeIgniter	Laravel
PC	0.00133 s	0.00679 s
Laptop	0.00188 s	0.01116 s

W tabeli 6 i na rysunku 4 porównano czas generowania się podstrony, która pobiera informacje z bazy danych. Przetestowano warianty wyświetlania listy wpisów w liczbie 100, 1 000 i 10 000. Obie aplikacje korzystają z tej samej bazy danych, na tym samym serwerze lokalnym.

Tabela 6. Czasy wyświetlania wpisów z bazy danych

	Liczba wpisów	CodeIgniter	Laravel
PC	100	0.09252 s	0.22297 s
	1000	0.79095 s	1.50472 s
	10000	6.0967 s	17.36832 s
Laptop	100	0.06506 s	0.16905 s
	1000	0.57473 s	1.47966 s
	10000	5.68999 s	13.93202 s



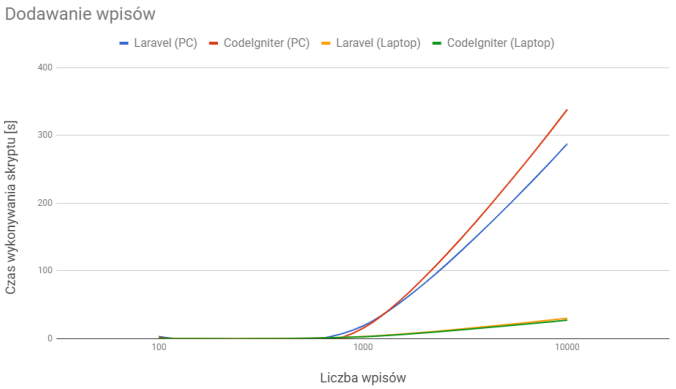
Rys. 4. Wykres zależności czasowych na podstawie danych z tabeli 6

5.2. Operacje wprowadzania wpisów

Kolejnym badaniem było sprawdzenie czasu wykonywania się metod wprowadzających nowe wpisy do bazy danych. Przed każdą operacją, tabela *posts* w bazie danych była opróżniana. Na potrzeby badania zostały stworzone odpowiednie metody, których zadaniem było generowanie wpisów w obu frameworkach. Funkcje te przyjmują jeden argument – liczbę iteracji dodawania postów jaką ma wykonać metoda. Badanie przeprowadzono w trzech konfiguracjach - 100, 1000 i 10 000 postów. Wyniki przedstawia tabela 7 oraz rysunek 5.

Tabela 7. Czasy wprowadzania wpisów do bazy danych

	Liczba wpisów	CodeIgniter	Laravel
PC	100	2.05762 s	3.10915 s
	1000	15.25737 s	18.66885 s
	10000	338.18025 s	287.47388 s
Laptop	100	0.33715 s	0.33725 s
	1000	2.74994 s	2.95962 s
	10000	27.22635 s	29.91931 s



Rys. 5. Wykres liniowy sporządzonych na podstawie danych z tabeli 7

5.3. Działanie pod obciążeniem

Do wykonania tzw. *stress testu* wykorzystano Apache JMeter [7]. Test polega na imitowaniu wejść użytkowników na stronę. JMeter pozwala wysyłać zapytania HTTP do serwera oraz kreować scenariusze zachowania użytkowników. Program jest udostępniany przy użyciu licencji Apache w wersji 2.0. Na potrzeby badania stworzono dwa scenariusze testowe. Parametry scenariuszy prezentuje tabela 8. Pierwszy test symuluje ruch 30 użytkowników, którzy w ciągu 50 sekund wchodzą na stronę i po około 30-60 s przechodzą na inne podstrony. Test został powtórzony 20 razy. Drugi test symuluje wejście 250 osób na stronę w ciągu 15 sekund. Otrzymane wyniki prezentuje tabela 9.

Tabela 8. Parametry scenariusza do testów obciążeniowych

	Test #1	Test #2
Liczba wątków (użytkowników)	30	250
Czas rozpoczęcia wszystkich wątków	50	15
Ilość powtórzeń	20	20

Tabela 9. Wyniki *stress testu*

	Test	CodeIgniter		Laravel	
		PC	Laptop	PC	Laptop
Średni czas odpowiedzi [ms]	#1	151	337	387	389
	#2	1903	9545	376	9620
Min. czas odpowiedzi [ms]	#1	94	155	166	246
	#2	93	152	11	232
Maks. czas odpowiedzi [ms]	#1	3134	2478	8661	8453
	#2	7481	44050	8068	39458
Odchylenie standardowe	#1	158.76	222.31	325.98	400.78
	#2	1443.39	13292.89	514.43	12843.94
Liczba błędów	#1	0%	0%	0.5%	0.5 %
	#2	0%	23.65%	11.33%	24.91%
Średnia przepustowość (zapytania na sekundę)	#1	0.6	0.6	0.6	0.04
	#2	4.7	4.0	4.8	4.1

5.4. Ocena frameworków

Na podstawie wyników przeprowadzonego porównania, dokonano autorskiej oceny obu frameworków (tabela 10).

Każdy element został oceniony w skali 1-5, gdzie 5 jest oceną najwyższą.

Tabela 10. Ocena frameworków

	CodeIgnitier	**Laravel**
Instalacja i konfiguracja	5	3
Struktura aplikacji	4	3
Praca na bazach danych	3	5
Testy wydajnościowe	4	2
Działanie pod obciążeniem	4	3
Metryki kodu	3	5
Stopień skomplikowania	4	3
Licencja	5	5
Sumaryczna ocena	**32**	**29**

6. Wnioski

Na podstawie przeprowadzonych badań można zauważyć, iż ważnym aspektem poprawnego i optymalnego działania każdego frameworka jest wykorzystanie najnowszej wersji języka PHP. W przypadku obu aplikacji zauważalny jest skok wydajnościowy i redukcja czasu wykonywania się operacji. Zwłaszcza w przypadku operacji dodawania nowych wpisów do bazy danych, gdzie widać nawet dziesięciokrotne zmniejszenie czasu potrzebnego do wykonania zadania.

Biorąc pod uwagę testy związane z generowaniem statycznej strony oraz wyświetlaniem listy wpisów, można zauważyć dominację CodeIgniter nad Laravel. Laravel zaczyna pokazywać swoją przewagę dopiero przy dodawaniu wielotysięcznej liczby rekordów do bazy danych. Przewaga jednak dotyczy tylko konfiguracji ze starszą wersją języka PHP (na PC). Korzystając z najnowszej wersji - Laravel nieznacznie przegrywa z lekkim CodeIgniter.

Badanie symulujące ruch użytkowników przy użyciu oprogramowania JMeter - również wskazuje na wyższość CodeIgniter. Dopiero na słabszej maszynie przy zwiększonym obciążeniu, aplikacja zaczęła pokazywać błędy. Laravel zwracał błędy nawet przy małym ruchu na każdej z maszyn.

Na podstawie informacji z tworzenia aplikacji testowych oraz przeprowadzonych badań można jednoznacznie polecić framework CodeIgniter. Jest on bardzo lekki i przyjemny. W połączeniu z najnowszą wersją PHP, również wydajny. Ponadto posiada niski próg wejścia dla programistów, którzy nie mieli wcześniej styczności z żadnym innym frameworkiem.

Literatura

[1] https://trends.google.pl/trends/explore?date=today%205-y&q=laravel,cakephp,symfony,zend,codeigniter [26.10.2017]

[2] https://www.quora.com/What-is-the-differences-between-Codeigniter-and-Laravel [27.11.2018]

[3] https://www.elsner.com/laravel-vs-codeigniter-php-framework/ [27.11.2018]

[4] https://www.codementor.io/chirilovadrian360/laravel-or-codeigniter-ilos0bfw9 [27.11.2018]

[5] https://www.codeigniter.com/download [17.11.2018]

[6] https://symfony.com/doc/current/templating.html [10.10.2018]

[7] https://jmeter.apache.org/ [27.11.2018]

[8] https://laravel.com/docs/5.7 [13.04.2018]

[9] M. Bean, Laravel 5 Essentials, Packt Publishing 2015

[10] T. Matula, Laravel, Tworzenie aplikacji, Receptury, Helion, Gliwice 2015

[11] https://www.codeigniter.com/user_guide/ [17.11.2018]

[12] R. Foster, CodeIgniter Web Application Blueprints, Packt Publishing 2015

[13] K. Suzuki, M. Whitney, CodeIgniter Testing Guide, LeanPub 2016