JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 20 (2021) 175–182
Received: 05 June 2021
Accepted: 21 June 2021

Comparison of selected view creation technologies in applications using the Laravel framework

Porównanie wybranych technologii tworzenia widoków w aplikacjach wykorzystujących szkielet programistyczny Laravel

Albert Wiktor Woś*, Beata Pańczyk

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract
This article presents the result of a comparison of selected view creation technologies in applications using the Laravel framework. Using the multi-criteria analytic hierarchy process method the most efficient template system was selected from Blade, Twig, and Smarty. Presented template systems were used with Laravel framework to create a landing page on the basis of which the experiment was conducted.

Keywords: template engines; Laravel; comparative analysis; AHP method

Streszczenie
W niniejszym artykule przedstawiono rezultat porównania wybranych technologii widoków w aplikacjach wykorzystujących szkielet programistyczny Laravel. Za pomocą wielokryterialnej metody analitycznego procesu hierarchicznego wyłoniony został najbardziej wydajny system szablonów spośród Blade, Twig oraz Smarty. Przedstawione silniki szablonów wykorzystane zostały razem z platformą programistyczną Laravel w celu utworzenia stron typu "landing page", na podstawie których przeprowadzono eksperyment.

Słowa kluczowe: systemy szablonów; Laravel; analiza porównawcza; metoda AHP

*Corresponding author
Email address: **albert.wiktor.wos@gmail.com** (A. W. Woś)

1. Wstęp

Istnieje wiele metod tworzenia aplikacji internetowych. Z każdym dniem powstają nowe technologie ułatwiające ten proces. Innowacje te są odpowiedzią na coraz większe zapotrzebowanie, by zaistnieć w sieci i poszerzyć swój zasięg na nowe grupy odbiorców. Z tego powodu coraz większą popularnością cieszą się szkielety programistyczne, które pozwalają programistom tworzyć rozbudowane strony szybciej oraz w bezpieczniejszy sposób, tym samym dewaluują praktykę pisania stron internetowych jedynie za pomocą własnego kodu.

Jednym ze szkieletów programistycznych, ułatwiających pracę z tworzeniem aplikacji internetowych jest Laravel. Framework ten napisany został w języku PHP, który posłużył między innymi do napisania platform programistycznych Symfony czy CakePHP, systemu zarządzania treścią Wordpress, a także najbardziej popularnego serwisu społecznościowego, jakim jest Facebook. Laravel jako jeden z najbardziej popularnych szkieletów programistycznych, bazujących na architekturze MVC, dostarcza autorski system szablonów Blade. Pozwala on, podobnie jak inne znane silniki szablonów Twig oraz Smarty, na separację logiki aplikacji od warstwy widoku i na ograniczenie redundancji kodu dzięki ponownemu wykorzystaniu go w innych widokach, w których wygląd posiada elementy wspólne.

1.1. Przegląd literatury

W celu uzasadnienia wyboru technologii, kryteriów oraz metodyki badań dokonano przeglądu literatury. Wiele prac dotyczy wyodrębnienia odpowiedniej technologii na podstawie różnych kryteriów. Najczęściej są to: popularność, wydajność oraz uniwersalność. Przykładem takiej pracy jest artykuł „Analysis and Practical Application of PHP Frameworks in Development of Web Information Systems" [1]. Zamieszczono w nim zestawienie języków programowania działających po stronie serwera, gdzie najwyższy wynik osiągnął język PHP, który jest wykorzystywany w przypadku 87.5% aplikacji internetowych, co argumentuje wybór tego języka do badań.

Kolejne zestawienie popularności użycia technologii przedstawiono w artykule na stronie „morioh.com" [2]. Na podstawie trendów wyszukiwań haseł, tagów oraz wzmianek na innych stronach wskazano platformę programistyczną Laravel jako tę najbardziej popularną spośród innych platform programistycznych wykorzystujących język PHP.

Natomiast w publikacji „Pro PHP MVC" [3] opisano zalety architektury Model-Widok-Kontroler, a także jako przykład najpopularniejszych systemów szablonów wskazano Twig oraz Smarty. Systemy te również pojawiają się najczęściej w zestawieniu zaprezentowanym na stronie „socialcompare.com" [4], gdzie przedstawiono dane dotyczące najbardziej popularnych szkieletów programistycznych napisanych w języku PHP.

Potwierdza to popularność tych silników szablonów oraz ich uniwersalność, a także motywuje zasadność porównania ich z systemem Blade dostarczanym przez Laravel.

W celu porównania technologii niezbędny jest wybór kryteriów, które staną się wyznacznikiem danego procesu. Ich wyodrębnienia dokonano po przeanalizowaniu wyników badań i argumentacji zaprezentowanej w literaturze przedmiotu.

Przykładem pracy porównującej technologie jest artykuł „RWD jako narzędzie optymalizacji stron internetowych" [5]. Zaprezentowano w nim badania czynników mających wpływ na wydajność projektów responsywnych. W tym przypadku jednym z kryteriów był czas ładowania oraz rozmiar strony.

W artykule „Analiza porównawcza wydajności frameworków Angular oraz Vue.js" [6] porównano szkielety programistyczne, wykorzystujące Javascript Angular i Vue.js za pomocą kryteriów dotyczących czasów renderowania, wymiany danych, poszczególnych elementów działania badanych aplikacji, a także rozmiaru plików końcowych.

Oprócz ustalenia kryteriów niezbędny jest także wybór metody zastosowanej do porównania. Przykład tejże znajduje się w publikacjach „Analiza wydajnościowa i możliwościowa narzędzia Laravel do tworzenia nowoczesnych aplikacji webowych" [7], a także „Zbadanie wydajności aplikacji internetowych utworzonych z wykorzystaniem Spring MVC oraz JavaServer Faces" [8]. W obu pracach badane platformy programistyczne oraz technologie wykorzystane zostały do utworzenia aplikacji o analogicznych funkcjonalnościach, które następnie posłużyły do testów wydajności. Dodatkowo w pracy porównującej technologie Spring MVC oraz JavaServerFaces testy wykonane zostały dzięki narzędziu JMeter, który pozwolił na przetestowanie aplikacji na serwerze lokalnym oraz eksport wyników.

Kolejnym etapem po przeprowadzeniu testów jest zestawienie wyników oraz ich porównanie, a następnie przedstawienie z wykorzystaniem odpowiedniej metody. Ponadto w przypadku wielu kryteriów niezbędna jest ich optymalizacja. W artykule „Comparison the processing speed between PHP and ASP.NET" [9] porównano wydajność szybkości przetwarzania dla języka PHP oraz ASP.NET. Każdej z technologii przyznawano punkty dla poszczególnych kryteriów. Dodatkowo wyniki badań przedstawiono w postaci wykresów i w tabelach, gdzie zestawiono punkty oraz technologię, która je uzyskała. Następnie punkty dla każdego z języków zsumowano i wyłoniono ten najlepszy.

Podobny w założeniu, choć bardziej miarodajny sposób porównania danych został przedstawiony w pozycji „Analiza jakości wybranych systemów e-learningowych za pomocą wielokryterialnej metody analitycznego procesu decyzyjnego AHP" [10]. W publikacji tej zaprezentowano badania dotyczące jakości systemów MOODLE oraz LAMS. Do analizy tych systemów wykorzystano metodę analitycznego procesu hierarchicznego – AHP. Metoda ta pozwoliła autorowi ustalić zależności między poszczególnymi kryteriami, które zostały zestawione, a wynikami dla porównywanych systemów. Dodatkowo w publikacji tej obliczono współczynnik niezgodności CR, który w tej metodzie pozwala określić spójność wag każdego z kryterium względem pozostałych. Wyznaczone na podstawie dziewięciostopniowej skali, przedstawionej w tej pozycji, wagi oraz eksperyment pozwoliły na obliczenie wyników dla poszczególnych systemów. Wyniki te dodano dla każdego z przedmiotów badań i wskazano system e-learningowy o największej sumie punktów.

Przedstawione w przeglądzie literatury pozycje uzasadniają wybór technologii, jakimi są Laravel oraz systemy szablonów Blade, Twig oraz Smarty, a także wybór metodyki badań i sposobu analizy wyników metodą AHP.

1.2. Cel badań

Celem badań było porównanie wybranych systemów szablonów umożliwiających tworzenie widoków w aplikacjach internetowych wykorzystujących szkielet programistyczny Laravel. W tym celu na podstawie przeglądu literatury zostały wybrane trzy systemy szablonów: Blade, Twig oraz Smarty. Spośród tych technologii, z wykorzystaniem trzech kryteriów: „objętość kodu wynikowego", „objętość kodu, który twórca musi napisać, by uzyskać dany efekt" oraz „szybkość ładowania strony", wyznaczony został najbardziej wydajny silnik szablonów. W ten sposób uzyskano odpowiedź na pytanie badawcze: „Blade, Twig czy Smarty – który z wymienionych systemów szablonów jest najbardziej wydajny do tworzenia widoków w aplikacjach typu »strona lądowania«, wykorzystujących platformę programistyczną Laravel?".

1.3. Zakres badań

Zakres badań objął utworzenie trzech aplikacji za pomocą szkieletu programistycznego Laravel. Każda z aplikacji wykorzystywała inny z systemów szablonów pozwalających na wygenerowanie widoku i odpowiednich im kontrolerów umożliwiających zwracanie widoku wraz z danymi. Wygląd aplikacji w każdym przypadku był identyczny i obejmował tę samą funkcjonalność. W ramach badań przeprowadzono testy każdej z aplikacji, a także ankietę pozwalającą na wyznaczenie wag kryteriów.

2. Obszar badań

Przed rozpoczęciem badań rozpoznano badane technologie i odpowiednio zaplanowano eksperyment.

2.1. Badane technologie

Przedmiot badań stanowią systemy szablonów Blade, Twig oraz Smarty, które, podobnie jak Laravel, zostały napisane w języku PHP. Systemy te pozwalają użytkownikowi na oddzielenie warstwy logiki biznesowej od logiki prezentacji. Umożliwia to łatwiejsze zarządzanie aplikacją oraz organizację kodu. Dodatkowo odgraniczenie poszczególnych warstw projektu pozwala na ponowne wykorzystanie kodu, który odpowiada za widok aplikacji dzięki powtórnemu użyciu tego samego wyglądu razem z inną treścią, dostarczaną z innej warstwy. Systemy szablonów umożliwiają ten proces poprzez wykorzystanie warunków oraz pętli, a także dzięki możliwości rozszerzania danego widoku przez dołą-

czenie wcześniej utworzonych bloków widoków czy całych plików. Kolejną zaletą silników szablonów jest umożliwienie użytkownikowi wyświetlania różnego rodzaju dostarczanych przez model danych, które dodatkowo mogą być wykorzystane lub nie na podstawie zaimplementowanych wewnątrz widoku warunków. Zasada działania systemów szablonów opiera się na tokenizacji danych źródłowych, które następnie segmentowane, zostają uporządkowane. Następstwem tego jest przekonwertowanie tych danych do kodu wynikowego dzięki parserom – programom analizującym składnię oraz ją interpretującym na podstawie wcześniej określonej gramatyki.

System szablonów Blade jest silnikiem dostarczonym przez twórców szkieletu programistycznego Laravel. Wykorzystuje on dyrektywy oraz tagi, których użycie przyśpiesza pracę z widokiem aplikacji. Blade kompiluje szablony do kodu PHP, który przechowywany jest w pamięci podręcznej do momentu, gdy zostanie ponownie zmodyfikowany [11]. W badaniach wykorzystano system Blade w wersji 7.0.

Kolejnym badanym silnikiem szablonów jest Twig, który jest domyślnym systemem szablonów dla Symfony, jednakże możliwe jest użycie go w innych platformach programistycznych. Autorzy tego silnika wymienili szybkość jako jedną z cech tej technologii, ponieważ szablon kompilowany jest do zoptymalizowanego kodu PHP. Kolejną cechą wymienioną przez twórców silnika Twig jest bezpieczeństwo, zapewnione użytkownikom poprzez tryb piaskownicy, a także elastyczność dzięki możliwości wykorzystania parserów oraz lekserów, które pozwalają użytkownikowi tworzyć własne filtry oraz flagi. W badaniach użyty został Twig w wersji 3.2.1 [12].

Ostatnim z badanych systemów szablonów jest Smarty, który wydany został w 2002 roku i dalej jest rozwijany przez twórców. Podobnie jak pozostałe systemy kompiluje kopię szablonów do skryptu PHP. Smarty został opisany przez twórców jako szablon cechujący się szybkością, łatwością użycia a także elastycznością i bezpieczeństwem. W eksperymencie wykorzystany został Smarty w wersji 3.1.36 [13].

Wszystkie wymienione systemy szablonów zaimplementowane zostały w szkielecie programistycznym Laravel. Framework ten napisany został w języku PHP i wydany na licencji MIT. Bazuje on na wzorcu architektonicznym Model-Widok-Kontroler (MVC), co pozwala użytkownikom na tworzenie aplikacji internetowych w bardziej zorganizowany sposób dzięki oddzieleniu warstw aplikacji.

Dodatkowo szkielet programistyczny Laravel ułatwia tworzenie bardziej rozbudowanych aplikacji internetowych dzięki wbudowanym komponentom, takim jak system autentykacji, linia komend Artisan, a także Eloquent Model, który wspomaga projektowanie bazy danych wykorzystywanej w aplikacji. W badaniu wykorzystano Laravel w wersji 7.0 [11].

Opisane systemy szablonów charakteryzują się podobnymi cechami oraz podobną metodą działania, jednakże zauważalnie różnią się pod względem składni, której przykłady dla poszczególnych systemów przedstawiono w Tabela 1.

Tabela 1: Przykładowe zestawienie elementów składni systemów szablonów Blade, Twig i Smarty

	Blade	Twig	Smarty
Dołączenie pliku o rozszerzeniu html:	@include('lorem')	{{include('lorem') }}	{include file = "lorem.html"}
Pętla for	@for ($i=0; $i<10;$i++) current value: {{ $i }} @endfor	{% for i in 0..10 %} current value: {{ i }} {% endfor %}	{for $i=1 to 10} current value: {$i} {/for}
Warunek if	@if (1< 18) @endif	{% if 1 < 18 %} {%endif%}	{if (1< 18)} {/if}
Komentarz	{{-- comment --}}	{# comment #}	{* comment *}
Rozszerzenie pliku layout na przykładzie sekcji o nazwie „body”	@yield ('body')	{% block body %} {% endblock %}	{block name = "body"} {/block}
Oznaczenie, że dany plik rozszerza plik o nazwie „layout”	@extends ('layout')	{%extends "layout"%}	{extends 'layout.tpl'}
Oznaczenie sekcji o nazwie „body”	@section ('body') @endsection	{% block body %} {% endblock %}	{block name = "body"} {/block}
Użycie zmiennej o nazwie „var”	{{$var}}	{{ var }}	{$ var }

Na podstawie Tabela 1. widać, że składnia użycia systemu Blade charakteryzuje się wykorzystaniem symbolu „@”. W przypadku systemu Smarty najczęściej wykorzystanym znacznikiem są nawiasy klamrowe, natomiast dla Twig jest to dodatkowo znak „%”. Kolejną widoczną różnicą są rozszerzenia widoków dla poszczególnych systemów. Dla Blade jest to „blade.php”, dla Twig - „twig”, natomiast dla Smarty - „tpl”.

2.2. Kryteria badań

Wybór kryteriów został uargumentowany w przeglądzie literatury i przedstawia się następująco:

- najkrótszy czas ładowania;
- najmniejszy rozmiar kodu, który został napisany w celu wygenerowania ustalonej treści;
- najmniejszy rozmiar plików końcowych, które użytkownik pobiera w celu wyświetlenia treści.

Czas ładowania aplikacji określa czas w sekundach, jaki upływa od momentu zainicjowania połączenia do

otrzymania odpowiedzi i pełnego obsłużenia żądania przez serwer.

Rozmiar kodu jest parametrem przedstawiającym liczbę użytych do napisania kodu znaków, które wykorzystują dany system szablonów oraz związane z nim elementy, takie jak dołączenie wymaganych bibliotek. Podczas zliczania znaków wzięto pod uwagę kontroler zwracający widok, a także pliki „layout" oraz „welcome", które zawierały składnię wykorzystywaną przez dany silnik w widokach oraz kod HTML użyty do utworzenia struktury strony.

Rozmiar plików określa liczbę bajtów pobranych przez użytkownika w celu wyświetlenia widoku dostarczonego przez aplikację. Parametr ten wskazuje, jak dobrze zoptymalizowany został kod wykorzystany do utworzenia widoku dla poszczególnych systemów szablonów. Dodatkowo wartość ta przekłada się na czas renderowania treści przez przeglądarkę, ale także jakość konwersji plików szablonów użytych przez badaną technologię.

2.3. Procedura przygotowania badań i ich realizacji

Przed przystąpieniem do eksperymentu przygotowane zostało środowisko badawcze. W celu utworzenia aplikacji użyto PhpStorm w wersji 2019.3.2 – zintegrowanego środowiska programistycznego wydanego przez firmę JetBrains, natomiast do zasymulowania działania serwera wykorzystano pakiet serwera XAMPP w wersji 2.4.46.

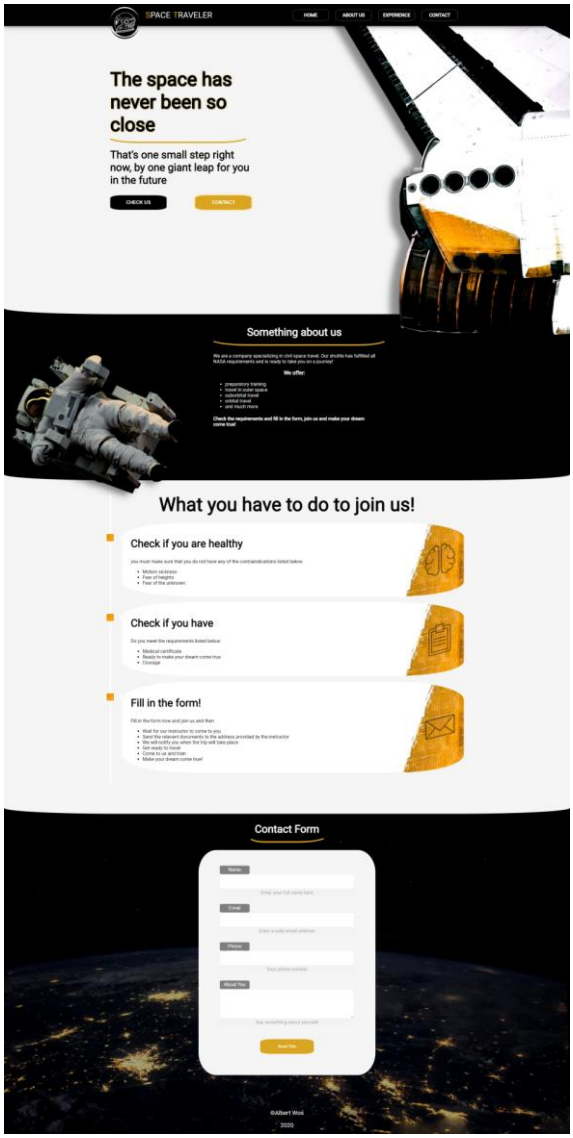
W celu instalacji szkieletu programistycznego Laravel oraz dodatkowo systemów szablonów Twig oraz Smarty wykorzystano system zarządzania bibliotekami - Composer w wersji 2.0, dzięki któremu możliwe było wydanie następujących komend:

- utworzenie projektu wykorzystującego framework Laravel - "composer create-project laravel/laravel nazwa-projektu 7.0.0";
- zainstalowanie systemu szablonów Twig – "composer require "twig/twig:^3.0";
- zainstalowanie systemu szablonów Smarty – „composer require smarty/smarty".

Dodatkowo zwiększony został limit pamięci w pliku php.ini do 2048 MB.

Do badań przygotowano także widok aplikacji typu „landing page" (strona lądowania). Widok ten został następnie wykorzystany w testowanych aplikacjach. Funkcją tego typu strony jest przedstawienie informacji na temat danego produktu czy usługi, a także przechwycenie uwagi użytkownika odwiedzającego stronę [14]. Dodatkowo strona lądowania charakteryzuje się węższym zakresem funkcjonalności niż strona główna, by zminimalizować możliwe rozproszenie uwagi użytkownika innymi opcjami dostępnymi na stronie [15]. Kolejnym charakterystycznym aspektem strony docelowej jest zwięzły tekst oraz treści strony uzupełnione obiektami graficznymi skompresowanymi i zapisanymi w odpowiednich formatach, a także odpowiednie wykorzystanie znaczników HTML.

Elementy charakterystyczne dla strony typu „landing page" zostały wykorzystane w utworzonych do badań aplikacjach, posiadających ten sam wygląd i te same funkcjonalności (Rysunek 1).



Rysunek 1: Wygenerowany widok aplikacji.

Do utworzenia aplikacji i skonfigurowania środowiska badawczego oraz testowania wykorzystano komputer z zainstalowanym systemem operacyjnym Windows 10 Home Edition o następującej specyfikacji:

- procesor: Intel Core i5-9300H, 2.40 GHz;
- pamięć RAM: 16 GB;
- karta graficzna: NVIDIA GeForce GTX 1650;
- dysk SSD: The Western Digital PC SN730.

Testy natomiast przygotowano za pomocą aplikacji Apache JMeter w wersji 5.3. Aplikacja ta napisana została w języku JAVA i wydana przez firmę Apache na licencji otwartego oprogramowania [16]. JMeter służy w głównej mierze do testowania aplikacji korzystających z serwerów HTTP i FTP oraz mierzenia parametrów, takich jak:

- przepustowość,
- czas odpowiedzi,
- czas przeprowadzanego testu,
- liczba żądań oraz bajtów, które serwer obsługuje.

Aplikacja ta pozwala również tworzyć testy, które ustawiają grupy wątków. Grupy te mogą zawierać opcje uwzględniające metodę obsługi wyjątku w razie niepowodzenia testu, liczbę wątków - liczbę przeprowadzonych testów w jednym momencie, a także, ile razy należy powtórzyć dany test. Ponadto Apache JMeter umożliwia skonfigurowanie żądań, które określają cel testów. Aplikacja ta dzięki słuchaczom może również przetworzyć wysyłane żądania oraz wyświetlić wyniki testów, które następnie można wyeksportować.

Do badań w aplikacji utworzono test, w którym następnie dodano grupy wątków dla każdego z badanych systemów szablonów. Dla każdej grupy wykonano 100 powtórzeń przy pojedynczym wątku. Dodatkowo do każdej z grup dodano próbnik „HTTP Request" z wartością dla pola o tej samej nazwie równą „GET" i nazwą serwera w polu „Server Name or IP" – „localhost". Dla każdego z próbników pola „Path" różniły się i zostały uzupełnione wartościami przedstawionymi w Tabela 2.

Tabela 2: Grupy wątków wraz z konfiguracją próbnika

	Path
Blade	comparison/comparison-blade/public/
Twig	comparison/comparison-twig/public//
Smarty	comparison/comparison-smarty/public/

Do testu dodany został także słuchacz „View Results Tree", który umożliwił obserwację wyników oraz ich eksport do pliku CSV. Ostatnim krokiem konfiguracji było zapisanie planu testu do pliku o rozszerzeniu jmx. Plik taki można następnie wczytać do programu, by wykonać tak skonfigurowane testy.

2.4. Scenariusz badań

Konfiguracja środowiska badawczego umożliwiła przeprowadzenie badań, których poszczególne etapy wykonane zostały w następującej kolejności:

1. Uruchomienie pakietu serwera WWW XAMP razem z modułem Apache;
2. Konfiguracja wirtualnych adresów badanych aplikacji;
3. Uruchomienie narzędzia JMeter;
4. Wczytanie skonfigurowanych wcześniej testów w narzędziu JMeter;
5. Podział wyników ze względu na badany system szablonów.

3. Metodyka badań

Podczas porównania przedmiotów badawczych wykorzystując więcej niż jedno kryterium, niezbędna jest ich optymalizacja. W tym celu wykorzystuje się metody optymalizacji wielokryterialnej. Umożliwiają one merytorycznie poprawne wyłonienie przedmiotu badań, który osiągnął najlepszy wynik. Jedną z tego typu metod jest metoda analitycznego procesu hierarchicznego opracowana przez Thomasa L. Saaty'ego [17]. Metoda ta uwzględnia następujące punkty krytyczne:

1. Przedstawienie problemu w postaci struktury hierarchicznej;
2. Ustalenie decydentów, którzy pomogą w ustaleniu wag kryteriów (do ustalenia wag kryteriów, które będą porównywane, przeprowadzone zostały badania ankietowe, opisane w rozdziale 3.1);
3. Porównanie parami w celu wyznaczenia priorytetów;
4. Obliczenie współczynnika niezgodności.

Następnie, wykorzystując metodę trywialną, wyznaczone zostało kryterium łączne na podstawie wyliczonych wcześniej wag.

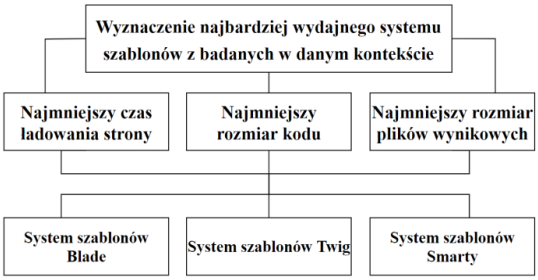
3.1. Ankieta

W celu określenia wag kryteriów wykorzystana została ankieta przygotowana na platformie Google Forms. Ankieta zatytułowana „Waga kryterium wyboru systemu szablonów do tworzenia widoków aplikacji internetowych" zawierała pytania dotyczące ankietowanego oraz porównań parami wag kryteriów. W celu przygotowania ankiety poprawnej merytorycznie wykorzystano wskazówki z artykułu „Design a questionnaire" [18]. Pytania zostały sformułowane w sposób jednoznaczny, a odpowiedzi wzajemnie się wykluczały. Pierwsze sześć pytań dotyczyło ankietowanego, jego wieku, płci, informacji, czy respondent jest studentem, czy też pracuje, najczęściej używanego przez ankietowanego języka programowania, posiadanego doświadczenia w branży IT oraz tego, czy respondent kiedykolwiek używał system szablonów. Ostatnie trzy pytania dotyczyły porównania parami dwóch kryteriów oraz przewagi jednego nad drugim w skali od -4 (przewaga kryterium Y nad X) do 4 (przewaga kryterium X nad Y).

Grupą docelową badania ankietowego byli programiści - pracownicy i studenci posiadający doświadczenie w branży IT oraz w korzystaniu z systemów szablonów. Z tego powodu w badaniach wzięte pod uwagę zostały jedynie te odpowiedzi, które zawierały odpowiedź twierdzącą na pytanie dotyczące doświadczenia w branży IT.

3.2. Struktura hierarchiczna

W badaniach wykorzystano trzy wcześniej opisane kryteria, które zostały usytuowane w strukturze hierarchicznej wraz z przedmiotami badań oraz celem badań (Rysunek 2).



Rysunek 2: Struktura hierarchiczna problemu badawczego.

3.3. Wyznaczenie priorytetów

Porównane zostały kryteria zestawione parami. Pozwoliło to uzyskać współczynnik wagowy, który określa priorytety wykorzystane do zoptymalizowania kryteriów. Do analizy tej użyta została dziewięciostopniowa skala zaproponowana przez Thomasa L. Saaty'ego [17],

której wartości zostały zestawione ze skalą użytą w ankiecie. Wyniki pochodzące z ankiety i przekształcone odpowiednio dzięki tej skali pozwoliły utworzyć macierz porównań o liczbie wierszy oraz kolumn równej liczbie badanych kryteriów. Pola w tej macierzy powyżej przekątnej uzupełnione zostały danymi dotyczącymi porównania kryteriów parami. Przekątną macierzy stanowią jedynki, które oznaczają zestawienie kryteriów odnoszących się do samych siebie. Dodatkową własnością tej macierzy jest to, że jest ona odwrotnie symetryczna. Oznacza to, że poniżej przekątnej pola uzupełnione zostały odwrotnościami wag priorytetów.

Tabela 3: Utworzona macierz porównań

	A	B	C
A	1	3	2
B	0,33	1	1
C	0,5	1	1

Następnie na podstawie macierzy porównań obliczone zostały priorytety wykorzystując wzory opisane w kolejnych trzech krokach:

1. Obliczono średnie geometryczne r_x dla każdego z trzech wierszy według wzoru:

$$r_x = \sqrt[n]{p_{x,y_1} * p_{x,y_2} * p_{x,y_3}} \quad (1)$$

gdzie *p* jest elementem macierzy porównań, natomiast indeksy *x* oraz *y* odpowiednio oznaczają numer wiersza i kolumny, a *n* liczbę porównywanych kryteriów;

2. Zsumowano obliczone w poprzednim kroku średnie geometryczne;
3. Wyznaczono priorytety w_x za pomocą następującego wzoru:

$$w_x = \frac{r_x}{\sum_{i=1}^{n} r_i}. \quad (2)$$

3.4. Współczynnik niezgodności

Spójność wyliczonych wag względem siebie przedstawiona została za pomocą współczynnika niezgodności *CR*. Thomas L. Saaty zasugerował, że, aby wagi były poprawne, wartość współczynnika *CR* nie może przekroczyć 10% [17].

W badaniach próg ten wyliczony został następująco:

1. Najpierw obliczono wartość własna λ_{max} według wzoru:

$$\lambda_{max} = \sum_{x=1}^{n} \sum_{y=1}^{n} p_{x,y} * w_x \quad (3)$$

gdzie *n* oznacza liczbę kryteriów, które są porównywane. Znaczenia symboli *p*, *w*, *x* oraz *y* wyjaśnione zostały w rozdziale 3.33.);

2. Następnie obliczony został indeks niezgodności *IC* za pomocą wzoru:

$$IC = \frac{\lambda_{max} - n}{n - 1} \quad (4)$$

3. Wykorzystując obliczony indeks niezgodności *IC*, a także wartość stablicowanego indeksu losowanego *RI*, podanego przez twórcę metody AHP (W przypadku 3 kryteriów *RI* wynosi 0,52), obliczono współczynnik niezgodności *CR* za pomocą wzoru:

$$CR = \frac{IC}{RI} \quad (5)$$

3.5. Kryterium łączne

Ostatnim krokiem niezbędnym do wyznaczenia wartości zoptymalizowanych i możliwych do porównania jest wykorzystanie metody trywialnej wyznaczania kryterium łącznego. W tym celu do obliczenia zastosowano następujący wzór:

$$K_i = \sum_{j=1}^{k} w_j \left(\frac{K_{min}}{K_{i,j}} \right) \quad (6)$$

gdzie symbolem K_i oznaczono wynik dla kryterium. Symbol w_j przedstawia wagę j-tego kryterium, K_{min} – najmniejszą wyznaczoną wartość dla danego kryterium, natomiast $K_{i,j}$ wyznaczoną wartość dla danej technologii i obliczanego kryterium.

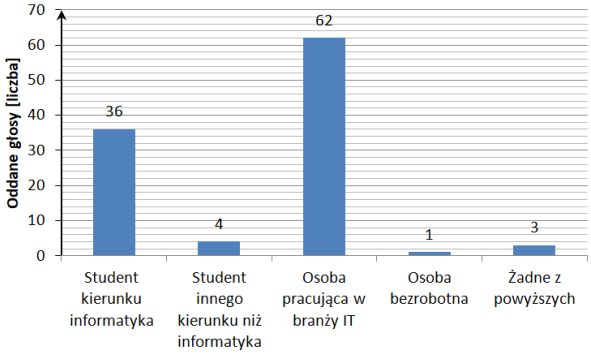
4. Wyniki

Rezultaty badań obejmują dane liczbowe wynikające z przeprowadzanego eksperymentu, a także odpowiedzi respondentów uzyskane z ankiety przeprowadzonej w dniach 22.02.2021 - 22.03.2021 r.

4.1. Ankieta

W ankiecie opisanej w rozdziale 3.11. uzyskano 113 odpowiedzi, z czego 85 zakwalifikowało się do badań ze względu na twierdzącą odpowiedź na pytanie dotyczące posiadania doświadczenia w branży IT. Większość ankietowanych - 84.71% zadeklarowała odpowiedź „20 – 40 lat”. Wiek pomiędzy 41 a 60 lat wskazało 12.94% respondentów. Jedynie 2.35% ankietowanych zaznaczyło odpowiedź „poniżej 20 lat”. Żaden z respondentów nie wybrał odpowiedzi „powyżej 60 lat”.

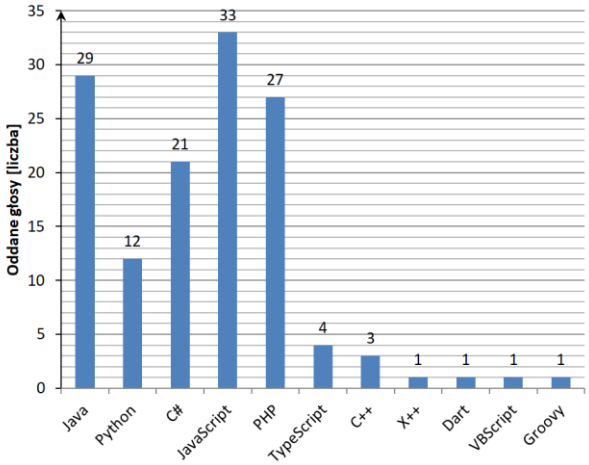
Odpowiedzi ankietowanych na pytanie „Wybierz opcję/opcje najlepiej Ciebie opisującą/opisujące.” przedstawione zostały na wykresie (Rysunek 3).



Rysunek 3: Rozkład odpowiedzi dotyczących związku ankietowanego z informatyką.

Odpowiedzi na pytanie dotyczące najczęściej wykorzystywanego języka przez respondenta przedstawiono na wykresie (Rysunek 4). Widać, że najczęściej wybie-

ranymi językami są JavaScript z 33, Java z 29 oraz PHP z 27 odpowiedziami.



Rysunek 4: Wybór języka najczęściej używanego przez respondenta.

Głównym celem ankiety było zebranie odpowiedzi dotyczących porównań parami wag kryteriów. Kryterium „A" oznaczało „najkrótszy czas ładowania strony", „B" – „najmniejszy rozmiar kodu, który został napisany w celu wygenerowania ustalonej treści", natomiast kryterium „C" – „najmniejszy rozmiar plików końcowych, które użytkownik pobiera w celu wyświetlenia treści". W pytaniu dotyczącym porównania kryterium „A" z „B" średnia z odpowiedzi wyniosła -0.96, co oznacza niewielką przewagę kryterium „A" nad „B". W porównaniu kryterium „A" z „C" średnia ta wyniosła -0.5. Oznacza to przewagę kryterium „A" nad „C". W przypadku zestawienia kryterium „B" z „C" średnia z odpowiedzi wyniosła 0,05 i oznacza to minimalną przewagę kryterium „C" nad „B".

4.2. Wagi kryteriów

Na podstawie wyników przeprowadzonej ankiety możliwe było wyznaczenie docelowych priorytetów dla każdego z kryteriów. Dodatkowo w tym celu przeprowadzono konwersję odpowiedzi respondentów do dziewięciostopniowej skali zaproponowanej przez twórcę metody AHP. Wynikiem tej operacji są dane w Tabela 4., przedstawiającej wagi kryteriów zestawionych parami. Następnie, na podstawie metodyki opisanej w rozdziale 3.3. oraz wag przedstawionych w tejże tabeli, obliczone zostały priorytety dla poszczególnych kryteriów. Wynik tego działania został przedstawiony w Tabela 5.

Tabela 4: Tabela wag kryteriów zestawionych parami

Para	Wynik
Przewaga ważności kryterium A nad kryterium B	3
Przewaga ważności kryterium A nad kryterium C	2
Przewaga ważności kryterium B nad kryterium C	1

Wyliczone priorytety umożliwiły obliczenie współczynnika niezgodności *CR*. W procesie tym wykorzystano metodykę przedstawioną w rozdziale 3.4. Obliczony *CR* wyniósł 1,34 %. Wynik ten nie przekroczył zaproponowanego przez twórcę progu - zatem obliczone wagi są poprawne.

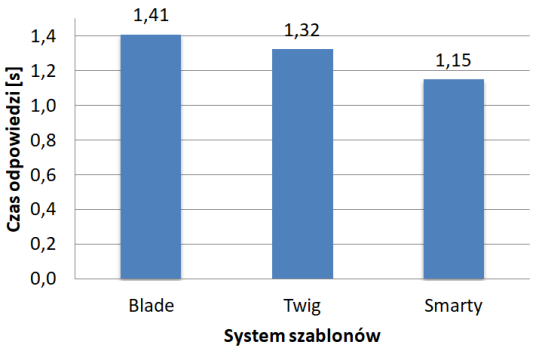
Tabela 5: Zestawienie kryteriów z ich priorytetami

Kryterium	Priorytet
Najkrótszy czas ładowania strony	0,56
Najmniejszy rozmiar kodu, który został napisany w celu wygenerowania ustalonej treści	0,20
Najmniejszy rozmiar plików końcowych, które użytkownik pobiera w celu wyświetlenia treści	0,24

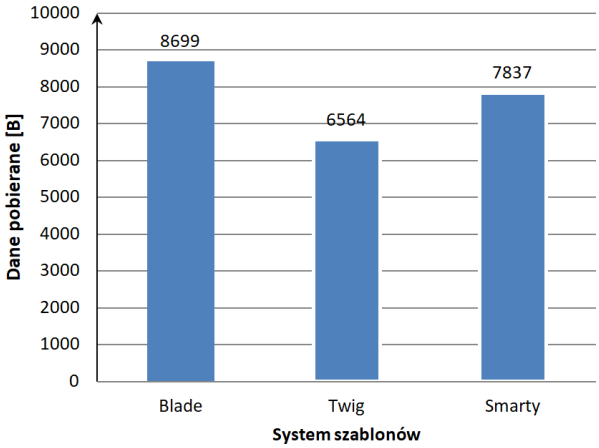
4.3. Wynik eksperymentu

Wynikiem opisanego w rozdziałach 2.3-2.4 eksperymentu jest 300 wierszy danych – po 100 dla każdej z badanych technologii. Dane te dotyczą między innymi czasu odpowiedzi aplikacji oraz odebranych przez użytkownika bajtów. Podczas eksperymentu 10 testów (po 4 w przypadku systemów Twig i Blade oraz 2 w przypadku Smarty) zakończyło się błędem spowodowanym zbyt dużym obciążeniem serwera liczbą zapytań. Dane wynikłe z testów zakończonych niepowodzeniem nie zostały wzięte pod uwagę w celu uniknięcia zaburzeń rezultatu badań.

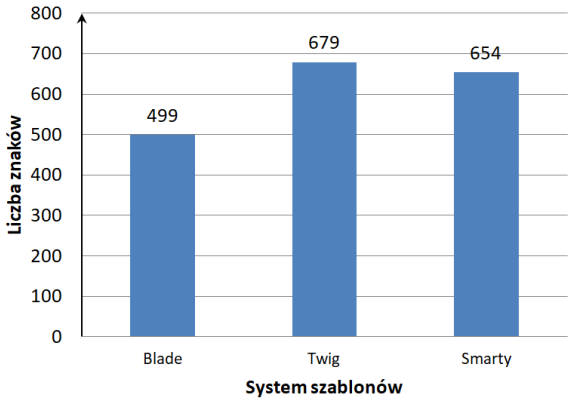
Wyniki eksperymentu przedstawione zostały na wykresie (Rysunek 5). Wyniki dotyczące liczby odebranych bajtów ukazuje następny wykres (Rysunek 6), natomiast ostatni z wykresów (Rysunek 7) prezentuje rozkład liczb użytych podczas korzystania z poszczególnych silników szablonów.



Rysunek 5: Rozkład czasu odpowiedzi poszczególnych aplikacji.



Rysunek 6: Rozkład liczby bajtów pobieranych przez użytkownika.



Rysunek 7: Rozkład liczby znaków użytych podczas korzystania z poszczególnych silników szablonów.

4.4. Wynik końcowy

Posiadając wyniki eksperymentu oraz obliczone wagi kryteriów, wyznaczono wartości punktowe zdobyte przez badane technologie w poszczególnych kryteriach. Dane przedstawione zostały w Tabela .

Tabela 6: Ocena punktowa

	Blade	Twig	Smarty
Najkrótszy czas ładowania strony	0,45	0,48	0,55
Najmniejszy rozmiar kodu, który został napisany w celu wygenerowania ustalonej treści	0,21	0,15	0,16
Najmniejszy rozmiar plików końcowych, które użytkownik pobiera w celu wyświetlenia treści	0,18	0,24	0,20
Suma punktów	0,84	0,87	0,91
Miejsce	3	2	1

5. Wnioski

Odpowiedzią na postawione w rozdziale 1.2. pytanie o to, który z badanych silników szablonów jest najbardziej wydajny w określonych warunkach - jest silnik Smarty. System ten okazał się również silnikiem, który pozwolił na najszybsze załadowanie strony. Jednakże pozostałe systemy przodowały w innych kryteriach. W przypadku najmniejszego rozmiaru plików końcowych, które użytkownik pobiera w celu wyświetlenia treści - Twig okazał się najlepszą technologią. W przypadku aplikacji wykorzystującej Blade potrzeba było z kolei najmniejszej liczby znaków w celu użycia tego silnika szablonów.

Literatura

[1] N. Prokofyeva, V. Boltunova, Analysis and practical application of PHP frameworks in development of web information systems, Procedia Computer Science 104 (2017) 51–56, http://dx.doi.org/10.1016/j.procs.2017.01.059.

[2] A. Jain, Najlepsze platformy programistyczne 2019, https://morioh.com/p/69f2267777df, [22.11.2020].

[3] C. Pitt, Pro PHP MVC, Apress, Berkeley, CA, (2012) 1–7, http://dx.doi.org/10.1007/978-1-4302-4165-2_1.

[4] Porównanie platform programistycznych wykorzystujących PHP, https://socialcompare.com/en/comparison/php-frameworks-comparison, [22.11.2020].

[5] H. Ochim, B. Pańczyk, RWD jako narzędzie optymalizacji stron internetowych, Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska 4 (2016) 81–86, http://dx.doi.org/10.5604/01.3001.0009.5196.

[6] R. Baida, M. Andriienko, M. Plechawska-Wójcik, Analiza porównawcza wydajności frameworków Angular oraz Vue.js, Journal of Computer Sciences Institute 14 (2020) 59–64, http://dx.doi.org/10.35784/jcsi.1577.

[7] P. Mincewicz, M. Plechawska-Wójcik, Analiza wydajnościowa i możliwościowa narzędzia Laravel do tworzenia nowoczesnych aplikacji webowych, Journal of Computer Sciences Institute 7 (2018) 108–115.

[8] N. Kovalchuk, E. Łukasik, Zbadanie wydajności aplikacji internetowych utworzonych z wykorzystaniem Spring MVC oraz JavaServer Faces, Journal of Computer Sciences Institute 1 (2016) 34–37.

[9] K. Bounnady, K. Phanthavong, S. Pathoumvanh, K. Sihalath, Comparison the processing speed between PHP and ASP. NET, 2016 13th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON) (2016) 1–5, http://dx.doi.org/10.1109/ECTICON.2016.7561484.

[10] A. Stecyk, Analiza jakości wybranych systemów e-learningowych za pomocą wielokryterialnej metody analitycznego procesu decyzyjnego AHP, Informatyka Ekonomiczna 49(3) (2018) 78–88, http://dx.doi.org/10.15611/ie.2018.3.07.

[11] Dokumentacja platformy programistycznej Laravel oraz systemu szablonów Blade, https://laravel.com/docs/7.x, [27.10.2020].

[12] Dokumentacja systemu szablonów Twig, https://twig.symfony.com/doc/3.x/, [27.10.2020].

[13] Dokumentacja systemu szablonów Smarty, https://www.smarty.net/docs/en/, [27.10.2020].

[14] I. Teodorescu, V. Vasile, Landing Pages Features to Attract Customers, Ovidius University Annals, Economic Sciences Series 15(2) (2015) 360–363.

[15] Edytor służący do tworzenia stron docelowych, https://landingi.com/pl/projektuj/landing-page/, [04.01.2021].

[16] Dokumentacja Apache JMeter, https://jmeter.apache.org/, [04.01.2021].

[17] A. Prusak, J. Strojny, P. Stefanow, Analityczny Proces Hierarchiczny (AHP) na skróty - kluczowe pojęcia i literatura, Humanities and Social Sciences 4(19.21) (2014) 179-192, http://dx.doi.org/10.7862/rz.2014.hss.66.

[18] D. H. Stone, Design a questionnaire, British Medical Journal 307(6914) (1993) 1264–1266.