WYŻSZA SZKOŁA BANKOWA w POZNANIU

Wydział Finansów i Bankowości

inż. Jakub Eichner

Porównanie platform programistycznych pod względem tworzenia stron internetowych oraz ich pozycjonowania

Praca Magisterska

Promotor

dr inż. Grzegorz Nowak

Poznań, 2022

Spis treści

Rozdział 1

Wprowadzenie

1.1 Opis dziedziny problemu

Gwałtowny wzrost popularności, dostępności i możliwych zastosowań sieci WWW (ang. World Wide Web) spowodował ogólnospołeczny rozwój w tworzeniu serwisów internetowych w celach promocji i sprzedaży różnego rodzaju usług lub produktów.

W badaniach przeprowadzonych w 2001 roku [1] udało się zindeksować ponad 550 miliardów dokumentów znajdujących się w internecie, znaczna większość z nich była dostępna w tzw. ”widocznym WWW” (ang. *visible Web* lub *surface Web*), czyli zindeksowanych przez wyszukiwarki internetowe. W 2002 r. przebadano zawartość 2,024 miliarda stron, natomiast w styczniu 2005 roku znaleziono ponad 11,5 miliarda publicznie indeksowanych stron internetowych. w marcu 2009 indeksowanych stron było już co najmniej 25,21 miliarda. [2]

W raporcie przedstawionym przez inżynierów Google - Jesse Alpert i Nissan Hajaj w 2008 roku można znaleźć pierwszą informację, iż wyszukiwarka odkryła ponad bilion unikalnych adresów URL. Natomiast w maju 2009 roku spośród 109,5 miliona domen znaczna większość z nich, aż 74% działało w formie

komercyjnej, reklamując różnego rodzaju produkty lub świadczone usługi.

Wraz z rosnącym zapotrzebowaniem na tworzenie dedykowanych witryn internetowych wzrasta ilość dostępnych rozwiązań technologicznych wspierających tworzenie i zarządzanie stronami internetowymi. Takie technologie są nazywane Systemami zarządzania treścią (ang. *content management system* – CMS) i stoją u podstaw większości serwisów znajdujących się obecnie w internecie.

Na dzień dzisiejszy najpopularniejszą technologią CMS jest Wordpress [3] - stworzony w oparciu o język PHP oraz system bazodanowy MySql system zaprojektowany pierwotnie do łatwego tworzenia i zagrządzania stronami blogowymi, następnie rozbudowany o szereg kolejnych funkcjonalności oraz setki wtyczek dodających dodatkowe zastosowania. Obecnie przy pomocy Wordpress'a na świecie funkcjonuje ponad 455 milionów stron, których tematyki i rodzaje zaczynają się na personalnych blogach hobbystycznych, a kończą na sklepach internetowych przynoszących niebotyczne zyski ich właścicielom.

W ramach niniejszej pracy dyplomowej zaprojektowano i zaimplementowano szereg aplikacji internetowych przy użyciu najpopularniejszych języków programowania. Następnie przy użyciu specjalistycznych narzędzi deweloperskich zostały one porównane pod kątem jakościowych wskaźników internetowych oraz możliwości, które oferują w zakresie optymalizacji stron względem silników wyszukiwania (ang. search engine optimization).

1.2 Cel i zakres pracy

Celem pracy jest implementacja szeregu zaprojektowanych serwisów internetowych o podobnej tematyce przewodniej. Kluczowe różnice pomiędzy

badanymi witrynami będą zawierać się w obszarze wykorzystanych języków programowania oraz systemów programistycznych odpowiedzialnych za przetwarzanie żądań wydawanych przez użytkownika, komunikację, obsługę zapytań na poziomie serwera oraz kompilowanie odpowiedzi dla klienta.

Następnie na opublikowanych serwisach przeprowadzenie szeregu testów i badań mających na celu zbadanie jakości połączenia oraz obsługi zapytań internetowych przez użyte systemy programistyczne. w pierwszej kolejności przeprowadzone zostaną badania przy pomocy narzędzi dostarczanych przez Google w ramach pakietu Web Developer Tools [4]:

- Page Speed Insights [5]
- Lighthouse [6]
- Google Analytics [7]
- Google Search Console [8]
- Google Data Studio

Oraz następnie przy pomocy narzędzi firm zewnętrznych:

- websiteoptimization [9]
- pingdom tools [10]
- webpagetest [11]

Struktura pracy jest następująca. W rozdziałach X i Y przedstawiono opis najważniejszych pojęć teoretycznych oraz zastosowanych technologii wykorzystywanych do obsługi kolejnych aplikacji. Rozdział X opisuje projekt bazowej strony internetowej oraz proces konstrukcji wszystkich aplikacji uwzględnionych w porównaniu.

Rozdział X przedstawia testy przeprowadzone na kolejnych technologiach wraz z ich wynikami, natomiast rozdział X zawiera analizę i podsumowanie zebranych wyników.

Rozdział 2

Wprowadzenie teoretyczne

2.1 Internet

Internet jest globalną siecią zapewniającą możliwość dostępu do zasobów danych znajdujących się na całym świecie. w celu uzyskania dostępu do danych należy przesłać zapytanie na indywidualny adres IP przypisany do publicznej, prywatnej, rządowej lub akademickiej sieci.

Dla poprawnego funkcjonowania tak dużej infrastruktury w jednoznaczny i bezkonfliktowy sposób potrzebne są jasno określone i koordynowane standardy. Obecnie wyznaczaniem kluczowych elementów, które zapewniają sprawne działanie internetu zajmuje się organizacja Internet Assigned Numbers Authority [12], której najważniejszymi obowiązkami jest przydzielanie oraz utrzymywanie unikalnych kodów, adresów i systemów numeracji wykorzystywanych w protokołach, na których opiera się działanie sieci internetowych. Najważniejszymi obszarami działań organizacjami są:

- Przydzielanie nazewnictwa domen - zarządzanie Root'em systemu DNS oraz przydzielanie domen .int i .arpa,
- Zarządzanie zasobami numerycznymi - koordynacja globalnej puli

numerów IP oraz AS, których spis jest udostępniany Regionalnym Rejestrom Internetowym,

- Przydział protokołów internetowych, poprzez współpracę z międzynarodowymi organizacjami normalizacji.

Jest to jedna z najstarszych organizacji w historii Internetu, została założona w latach 70 ubiegłego wieku. Obecnie usługi społecznościowe są przekazane na rzecz Public Technical Identifiers będącego oddziałem ICANN [14] - międzynarodowej organizacji społecznościowej non-profit.

2.2 World Wide Website

Najważniejszą różnicą pomiędzy Internetem a World Wide Web (WWW lub sieć WWW) jest fakt, że Internet to globalne połączenie sieci, natomiast sieć World Wide Web to zbiór informacji, do których można uzyskać dostęp za pośrednictwem Internetu. Innymi słowy, Internet to infrastruktura, a World Wide Web to usługa stanowiąca pewnego rodzaju interfejs dla użytkowników.

Sieć WWW jest najczęściej używaną częścią Internetu. Jej najważniejszą cechą jest hipertekst, czyli metoda pozwalająca na tworzenie odnośników bezpośrednio w pisanym tekście. w większości witryn WWW odnośniki są zawarte jako bezpośrednie odniesienie do adresu witryny w tekście lub jako atrybut *href* tagu <*a*>, dzięki czemu dla końcowego użytkownika są one zawarte jako klikalny atrybut tekstu. Dzięki takim słowom lub zwrotom użytkownik może przemieścić się do powiązanej odnośnikami witryny lub strony. Jako hiperłącza używane są również przyciski, obrazy lub fragmenty obrazów.

2.3 Wyszukiwarki internetowe

Wyszukiwarka to oprogramowanie dostępne w Internecie, które przeszukuje bazę danych informacji zgodnie z zapytaniem użytkownika. Wyszukiwarka dostarcza listę wyników, które stanowią najlepszą odpowiedź na zapytanie informacyjne używtkownika. Obecnie w Internecie dostępnych jest wiele różnych wyszukiwarek, z których każda ma własne możliwości i funkcje. Pierwsza udokumentowana wyszukiwarka nosiła nazwę Archie i służyła do wyszukiwania plików FTP, a pierwsza wyszukiwarka tekstowa - Veronica. Obecnie najpopularniejszą i najczęściej wykorzystywaną wyszukiwarką jest Google. Warto jesdnak pamiętać, że nie jest on monopolistą i pewną część rynku posiadają również konkurenci, wśród których warto wymienić:

- AOL,
- Ask.com,
- Baidu,
- Bing,
- DuckDuckGo,
- Yahoo.

2.3.1 Jak działa wyszukiwarka?

Za pomocą największych wyszukiwarek można znaleźć miliardy różnych stron, z których w zależności od zapytania wyszukiwarka wybiera kilkadziesiąt najlepiej odpowiadających domniemanej potrzebie informacyjnej. Wydedukowanie najlepiej pasujących odpowiedzi na zapytanie użytkownika jest możliwe dzięki szeregowi różnych algorytmów.

Każda wyszukiwarka w pierwszej kolejności zbiera bazę danych obejmującą jak największą ilość stron znajdującej się w sieci WWW. w procesie *indeksowania* roboty przeszukujące (ang. crawler) skanują kolejne witryny i zapisują najważniejsze, znalezione tam treści. Dla prawidłowego zindeksowania treści strony w pierwszych krokach algorytmu wykonywane są operacje pozwalające uzyskać jak najbardziej uniwersalną miarę jakości. Są to kolejno:

1. Usunięcie słów znajdujących się w stop liście (ang. stop words) - pozbycie się słów o minimalnym znaczeniu merytorycznym. Najczęściej są to spójniki oraz popularne słowa znajdujące uniwersalne zastosowanie w niespokrewnionych tematykach (np. *mp3*), które nie wpływają na poprawną identyfikację treści dokumentu. Listy takich słów są tworzone indywidualnie dla każdego języka,

2. zapisanie pozostałych wyrazów oraz częstotliwości ich występowania jako słownika danej podstrony,

3. udokumentowanie znajdujących się na stronie odnośników do innych witryn,

4. zapamiętanie umieszczonych na stronie multimediów.

Zebrane w ten sposób dane służą do tworzenia rankingów dopasowanych do zapytania użytkownika. Rankingi tworzone są za pomocą specjalnej grupy algorytmów, które bazując na utworzonych słownikach badają zbieżność z frazami użytymi w zapytaniu. Jednym z podstawowych funkcji rankingowych jest Okapi BM25 [**?**].

2.4 Search Engine Optimalization

Kierując się oficjalną definicją przedstawianą przez Google [15]:

> ”SEO – optymalizacja witryn pod kątem wyszukiwarek, czyli proces ulepszania witryn na potrzeby wyszukiwarek. To także nazwa stanowiska osoby, która zajmuje się tym zawodowo: zatrudniliśmy specjalistę SEO, aby zwiększyć naszą obecność w internecie.”

Optymalizacja pod kątem wyszukiwarek internetowych (ang. Search Engine Optimalization) polega na tworzeniu i optymalizacji treści na w sposób, który pomaga użytkownikom (i robotom wyszukiwarek) znaleźć odpowiedzi na kluczowe zapytania. Celem działań SEO na danej stronie internetowej jest poprawa jakości treści, w celu poprawy pozycji w wynikach wyszukiwarek. Co przełoży się na większy ruch organiczny, który dana witryna pozyskuje jako odwiedzających z wyników wyszukiwania.

2.4.1 Wpływ technologii na pozycjonowanie

Wśród ważnych czynników wpływających na ocenianie strony przez bardziej zaawansowane algorytmy jest szybkość ładowania strony. Wyszukiwarki w celu zapewnienia użytkownikom bezproblemowego doświadczenia wyszukiwania, nagradza strony, które ładują się szybko. Dlatego też, bardzo ważna jest odpowiednia architektura odbierania zapytań użytkownika, szybkość ich przetwarzania i zwracania odpowiedzi. Głównym celem badań jest porównanie tego aspektu zwiększania potencjału pozycjonowania w przekroju różnych technologii oraz systemów programistycznych.

Rozdział 3

Zastosowane technologie

3.1 HTML

HTML (ang. Hyper Text Markup Language) jest standardowym językiem znaczników dla dokumentów przeznaczonych do wyświetlania w przeglądarce internetowej. Może być wspomagany przez różne technologie technologie, jak kaskadowe arkusze stylów (CSS) i języki skryptowe, takie jak JavaScript. Pierwszy raz zastosowany w 1980 roku przez fizyka Tim Berners-Lee pod prototypową nazwą ENQUIRE. Pierwotnie miał służyć jako system dla naukowców CERN do formatowania i udostępniania prac naukowych. w 1989 roku zaproponowane zostało użycie systemu hipertekstowego w sieci internetowej, rok później została stworzona formalna dokumentacja oraz oprogramowanie do wyświetlania plików pośrednio z serwera.

Pierwszym ogólnodostępnym opisem struktury języka HTML było opracowanie *HTML Tags* [16], wydanym przez jego twórcę w 1991 roku. HTML składa się z szeregu elementów, które służą do obudowywania lub zawijania różnych części treści, tak aby wyglądały w określony sposób lub zachowywały się w określony sposób.

3.2 JavaScript

JavaScript jest skryptowym językiem programowania, pozwalającym na implementację złożonych funkcji na stronach internetowych - które umożliwiają dynamiczne generowanie i przedstawianie treści w przeglądarce klienta - wyświetlając aktualizacje treści, interaktywne mapy lub animowane grafiki 2D/3D. Bardzo powszechnym zastosowaniem JavaScriptu jest dynamiczna modyfikacja HTML i CSS w celu aktualizacji interfejsu użytkownika, za pomocą interfejsu API.

3.2.1 NodeJS

Node.js jest środowiskiem programistycznym przeznaczonym do tworzenia aplikacji wykorzystujących język JavaScript uruchamianych poza środowiskiem przeglądarki. Środowisko to jest oparte na licencji *open source*, co umożliwia jego dynamiczny rozwój oraz między platformową dostępność. Umożliwia uruchamianie skryptów po stronie serwera, co pozwala na przetwarzanie dynamicznych struktur aplikacji internetowych przed dostarczeniem ich użytkownikowi. Takie podejście jest częścią paradygmatu *„JavaScript everywhere”*, który zakłada unifikacje skryptów serwerowych z tymi przetwarzanymi po stronie użytkownika.

3.2.2 React

React to stworzona przez Facebook i utrzymywana na licencji MIT biblioteka JavaScript służąca do budowania interfejsów użytkownika. U jej podstaw stoją 3 założenia:

- Deklaratywność: React pozwala na tworzenie interfejsu użytkownika

za pomocą projektowania widoków, które będą renderowane przez interpreter na podstawie stanu aplikacji i dynamicznie dopasowywane do żądań użytkownika. Deklaratywne podejście pozwala na bezkonfliktowe przewidywanie działania kodu i zmniejsza głębokość zagnieżdżenia kodu,

- Component-Based: pozwala tworzyć zamknięte komponenty, które są samowystarczalne w obrębie mikro-systemów i następnie kompilować je w bardziej złożone interfejsy. Umożliwia to również utrzymywanie niezależności stanu aplikacji od Obiektowego Modelu Dokumentu (ang. DOM),

- Learn Once, Write Anywhere: biblioteka nie ma ograniczeń w doborze stosu technologicznego, pozwala to na transmutacje czystego kodu JavaScript do interfejsów budowanych w tym oprogramowaniu. React umożliwia również renderowanie kodu po stronie serwera (ang. SSR - Server Side Rendering), dzięki czemu można usprawnić działanie aplikacji po stronie klienta.

3.3 PHP

PHP (ang. *Hypertext Preprocessor*) [17] jest szeroko stosowanym językiem skryptowym ogólnego przeznaczenia o otwartym kodzie źródłowym, który znajduje najszersze zastosowanie we wspomaganiu tworzenia dynamicznych stron internetowych, wzbogacone dodatkowo o możliwość osadzania skryptów PHP wewnątrz struktur języka HTML. W przeciwieństwie do statycznego wypisywania danych w klasycznych stronach opartych na języku HTML, strony zbudowane przy pomocy skryptów PHP opierają się na tożsamych tagach HTML z osadzonym kodem, który pozwala na dynamiczne generowanie zawartości w serwisie internetowych z bazodanowej warstwy aplikacji internetowej.

Znaczącą cechą języka PHP jest fakt, iż wszystkie części kodu są renderowane po stronie serwera, generując gotowy plik HTML, który jest wysyłany do klienta, dzięki czemu obciążenie kompilacją kodu spoczywa całkowicie na serwerze pozwalając na bezusterkowe doświadczenie użytkownika w interakcji z aplikacją. Jednocześnie stanowi to ogromną korzyść dla utrzymywania bezpieczeństwa aplikacji, dzięki przesyłaniu klientowi jedynie pliku wynikowego ograniczamy do minimum jego ingerencję w działanie serwisu odbierając jednocześnie możliwość negatywnych interakcji z funkcjonalnością witryny.

3.4 Python

Python [18] jest językiem programowania ogólnego przeznaczenia wysokiego poziomu. Wykorzystuje podejście wieloparadygmatowe, co oznacza, że obsługuje proceduralne, obiektowe i niektóre funkcyjne konstrukcje programistyczne. Został stworzony przez Guido van Rossuma jako następca innego języka (o nazwie ABC) w latach 1985-1990 i jest obecnie używany w wielu dziedzinach, takich jak tworzenie stron internetowych, data science, DevOps oraz automatyzacja/produktywność. Wbudowane wysokopoziomowe struktury danych, w połączeniu z dynamicznym typowaniem i wiązaniem danych, czynią go najpopularniejszym narzędziem do szybkiego tworzenia aplikacji, a także do wykorzystania jako język skryptowy. Python obsługuje moduły i pakiety, co sprzyja modularności programów i wielokrotnemu wykorzystaniu kodu (ang. *'Code once, run everywhere'*). Interpreter Pythona i obszerna biblioteka standardowa są dostępne w postaci źródłowej lub binarnej w ramach licencji open source.

3.4.1 Django

Django jest wysokopoziomowym frameworkiem dedykowanym do tworzenia aplikacji internetowych przy użyciu języka Python, który umożliwia szybkie tworzenie bezpiecznych i łatwych w utrzymaniu stron internetowych. Django był początkowo rozwijany w latach 2003-2005 przez zespół zajmujący się tworzeniem i utrzymaniem stron internetowych gazet. Po stworzeniu wielu witryn zespół zaczął wyodrębniać i ponownie wykorzystywać wiele wspólnych kodów i wzorców projektowych. Ten wspólny kod przekształcił się w ogólny system do tworzenia stron internetowych, który w lipcu 2005 roku został udostępniony jako projekt ”Django”. Django stale się rozwijał i był ulepszany, począwszy od pierwszego wydania (1.0) we wrześniu 2008 roku, aż do niedawno wydanej wersji 4.0 (2022). w każdym wydaniu dodano nowe funkcje i poprawki błędów, począwszy od obsługi nowych typów baz danych, silników szablonów i buforowania, a skończywszy na dodaniu generycznych funkcji i klas widoków. Django może być używany do budowania niemal każdego rodzaju stron internetowych - od systemów zarządzania treścią i encyklopedii, po sieci społecznościowe i serwisy informacyjne. Może współpracować z dowolnym frameworkiem po stronie klienta i dostarczać treści w niemal każdym formacie (w tym HTML, RSS, JSON, XML).

3.4.2 Laravel

Laravel [19] to framework PHP typu open-source przeznaczony do tworzenia aplikacji internetowych z ekspresyjną i elegancką składnią. Dzięki gotowej bibliotece funkcji znacząco ułatwia wiele procesów związanych z tworzeniem serwisów internetowych, takich jak uwierzytelnianie oraz sesje użytkowników i buforowanie danych. Laravel dostarcza również potężne narzędzia

bazodanowe, w tym ORM (ang. *Object Relational Mapper*) o nazwie Eloquent oraz wbudowane mechanizmy do tworzenia migracji baz danych.

Laravel wykorzystuje wzorzec projektowy MVC (ang. *Model View Controller*) [20]. Model reprezentuje kształt danych, na których operuje aplikacja. Takie rozwiązanie pozwala na rzeczywiste odwzorowanie struktury bazy danych w projekcie. Kontroler wchodzi w interakcję modelem. w momencie otrzymania żądania od użytkownika kontroler łączy się z modelem i pobiera potrzebne informacje. Jeżeli użytkownik wyśle żądanie zmiany danych w systemie, kontroler jest odpowiedzialny za aktualizację modelu. w kontrolerze znajduje się większość logiki aplikacji. Finalnie kontroler używa tych informacji do tworzenia widoków. Widok jest szablonem, który reprezentuje dane poprane przez kontroler z modelu. w widokach znajduje się znaczna większość komponentów HTML zawartych w aplikacji.

Rozdział 4

Projekt i konstrukcja stron

4.1 Bazowa strona - HTML

Pierwszą zaimplementowaną aplikacją jest bazowa strona z wykorzystaniem języka HTML, arkuszy stylów CSS i prostych skryptów stworzonych w języku JavaScript w celu nadania aplikacji wyglądu i funkcjonalności korespondujących ze standardami tworzenia nowoczesnych platform internetowych [21]:

1. Prostota - ważne jest, aby pamiętać że tworzona aplikacja internetowa ma przede wszystkim spełniać określone założenia funkcjonalne. w związku z tym wizualna strona produktu powinna wyglądać profesjonalnie i zachęcająco, jednocześnie nie utrudniając użytkownikowi wykorzystanie funkcjonalności witryny.

2. Hierarchia Wizualna - podążając w generalnym kierunku wyznaczonym w pierwszym dobrą praktyką jest, aby najważniejsze elementy były widoczne dla użytkującego w pierwszej kolejności. Z tego podejścia wynika częsta decyzja o umieszczaniu dużego banera informującego o głównej charakterystyce na samej górze strony, często umieszczając w jego obszarze odnośnik do akcji, która ułatwi użytkownikowi przejście do kluczowej funkcjonalności.

3. Intuicyjna Nawigacja - zaplanowanie i stworzenie struktury nawigacyjnej, która pozwoli użytkownikowi w instynktowny sposób poruszać się w obrębie domeny, umożliwiając w ten sposób bezproblemowe skorzystanie z całości funkcjonalności aplikacji.

4. Spójność projektu - Poza utrzymaniem spójnej nawigacji, ogólny wygląd witryny powinien być podobny na wszystkich jej stronach. Tła, schematy kolorów, kroje pisma, a nawet ton tekstów to obszary, w których spójność ma pozytywny wpływ na użyteczność.

5. Responsywność - Projektowanie responsywnej aplikacji internetowej oznacza stworzenie elastycznej struktury strony, która pozwoli na dopasowanie treści wyświetlanej na stronie do wymiarów urządzenia, z korzysta użytkownik. Można to osiągnąć za pomocą przyjaznych dla urządzeń mobilnych szablonów HTML lub poprzez stworzenie specjalnej konwersji strony dla różnych szerokości wyświetlanej treści.

6. Konwencjonalność - Bardzo ważnym elementem w pierwszym odbiorze strony przez użytkowników jest zastosowanie pewnych konwencji, które wraz z rozwojem internetu stały się nieodłączną częścią interakcji z serwisami internetowymi. Najczęstszymi zastosowania takich rozwiązań jest:

    - umieszczanie menu nawigacyjnego u góry strony,
    - pozycja logo strony w lewym górnym rogu ekranu i umieszczenie pod nim odnośnika do głównej strony,
    - zmiana kolorów interaktywnych elementów strony po najechaniu na nie kursorem,

- dodawanie strzałek nawigacji do wszelkiego rodzaju ruchomych galerii i suwaków z obrazami.

4.2 React

Struktura aplikacji zaprojektowanej z użyciem biblioteki React składa się zawsze z 3 kluczowych komponentów:

1. Główna aplikacja, która łączy wszystkie elementy programistyczne aplikacji. Jest ona odpowiedzialna za generowanie odpowiedzi dla zapytań użytkownika, to tutaj odbywa się routing oraz obsługa przesyłanych widoków do klienta.

2. Widoki stron są modułami, które odpowiadają za łączenie pomniejszych komponentów aplikacji w gotowe widoki głównych elementów strony. Najważniejszymi widokami stron stworzonymi na potrzeby tej aplikacji są *Home* oraz *Posts*, które łączą odpowiednio elementy widoku głównej strony i podstron poświęconych elementom tekstowym witryny.

3. Komponenty są najmniejszymi elementami składowymi widoku aplikacji, dostarczają one gotowe statyczne lub dynamiczne elementy strony oparte na elementach HTML. Najważniejszymi komponentami stworzonej aplikacji są:

   - *NavBar* przechowuje kod paska nawigacyjnego, który jest wyświetlany u góry jako pierwszy element w widoku każdej ze stron.
   - *Footer* przechowuje kod stopki, która jest zawarta u dołu widoku każdej strony i podaje między innymi informacje o autorze i prawach autorskich projektu.

- *Header* jest funkcją zwracającą nagłówek strony głównej, pomaga on odpowiednio formatować zdjęcia znajdujące się w tle oraz dopasowywać elementy stylów tekstowych nagłówka.
- *BlogCarusel* to komponent zwracający dynamiczny element zawierający suwak z obrazkami i tytułami kolejnych podstron znajdujących się w obrębie domeny. Jeżeli ich liczba jest większa niż 3 formatuje je w animowaną galerię, która jest interaktywna z perspektywy użytkownika.
- *Content* jest elementem odpowiedzialnym za renderowanie opisu tekstowego oraz nakładanie funkcji stylów na pole z treścią na głównej stronie.
- *Blog Article* służy do generowania treści artykułów oraz obrazów znajdujących się w treści wyświetlanej na niższych podstronach aplikacji.
- *Blog Links* pozwala na dołączanie odnośników do 3 innych wpisów poniżej artykułu wyświetlanego na podstronie.

Wszystkie komponenty są następnie kompilowane przy użyciu *Node Packet Manager*. Do przechowywania plików wykorzystany został system kontroli wersji *GitHub*, skąd repozytorium zostało umieszczone na platformie hostingowej *Heroku*.

4.3 NodeJS

Aplikacja stworzona przy pomocy oprogramowania NodeJS jest oparta o środowisko programistyczne *Express*, które pozwala na obsługę zapytań sieciowych i generowanie odpowiedzi z poziomu programu. Dodatkowo do

przeprowadzania odwołań wewnątrz struktury plików wykorzystana została biblioteka *path* oraz w celu obsługi zapytań biblioteka *http*.

W pierwszej kolejności program ustala port wyjściowy, z którego będą nasłuchiwane zapytania użytkownika. Kolejnym krokiem jest załadowanie i dołączenie statycznych plików, które są niezmienne podczas działania aplikacji. Ostatnim krokiem przygotowania jest rozpoczęcie nasłuchiwania i następnie po otrzymaniu zapytania od użytkownika kompilowanie i zwrot odpowiedzi w postaci szablonu odpowiedniej strony.

Program jest kompilowany przy użyciu programu *Node*. Natomiast wszystkie pliki aplikacji są przechowywane przy pomocy systemu kontroli wersji *GitHub*, skąd repozytorium zostało umieszczone na platformie hostingowej *Heroku*.

4.4 Django

Kluczowym elementem tworzenia aplikacji przy pomocy środowiska programistycznego Django jest plik *manage.py*, który pozwala na wysokopoziomowe kontrolowanie struktury projektu. Wszystkie opcje oraz trasy prowadzące do katalogów i kluczowych komponentów są przechowywane w pliku *settings.py*, definiuje on również ścieżkę do katalogu głównego (ang. *root*).

Pierwszym krokiem po odebraniu żądania od użytkownika jest wyznaczenie trasy do odpowiedniego widoku, procedura ta odbywa się wewnątrz pliku *urls.py*, który przechowuje pewnego rodzaju tablice trasowania, w której są zawarte połączania adresów z odpowiadającymi im widokami.

Wewnątrz pliku *views.py* znajdują się funkcje pozwalające na kompilowanie elementów strony zawartych w poszczególnych widokach w sposób podobny do funkcjonowania komponentów w bibliotece React.

Finalnym krokiem tworzenia aplikacji jest użycie biblioteki *gunicorn* służącej jako serwer HTTP interfejsu bramy sieciowej (ang. *Web Server Gateway*) Python oraz biblioteki *whitenoise*, której zadaniem jest poprawne ładowanie plików statycznych podczas kompilowania projektu.

Po stworzeniu plików

- *requirements.txt* który zawiera spis bibliotek potrzebnych do kompilacji,
- *runtime.txt* określającego wersję środowiska obsługującą projekt

projekt został umieszczony na platformie *Heroku* oraz w systemie kontroli wersji *GitHub*.

4.5 Laravel

Podstawowym komponentem projektu jest plik *index.php*, który przyjmuje i obsługuje pierwsze żądanie użytkownika. Następnie serwer wywołuje renderowanie odpowiedniego pliku widoku *.blade.php* - w tych plikach przechowywane są szablony HTML oraz kod PHP niezbędne do wyświetlenia pełnego widoku strony.

Silnik szablonów Blade w Laravelu umożliwia bezpośrednie działanie na danych tekstowych w aplikacji internetowej. w tym celu obsługuje on wiele operacji, takich jak łączenie modeli danych, przetwarzanie kodu aplikacji w szablonach źródłowych, kierowanie danych wyjściowych do określonego pliku tekstowego lub strumienia.

Co ważne szablony Blade [22] pozwalają na przetwarzanie układów widoków bez wpływu na wydajność i szybkość działania aplikacji.

Rezultat programu jest renderowany bezpośrednio przez kompilator na serwerze, co pozwala na zminimalizowanie wpływu środowiska na czas przetwarzania zapytań.

Rozdział 5

Badania

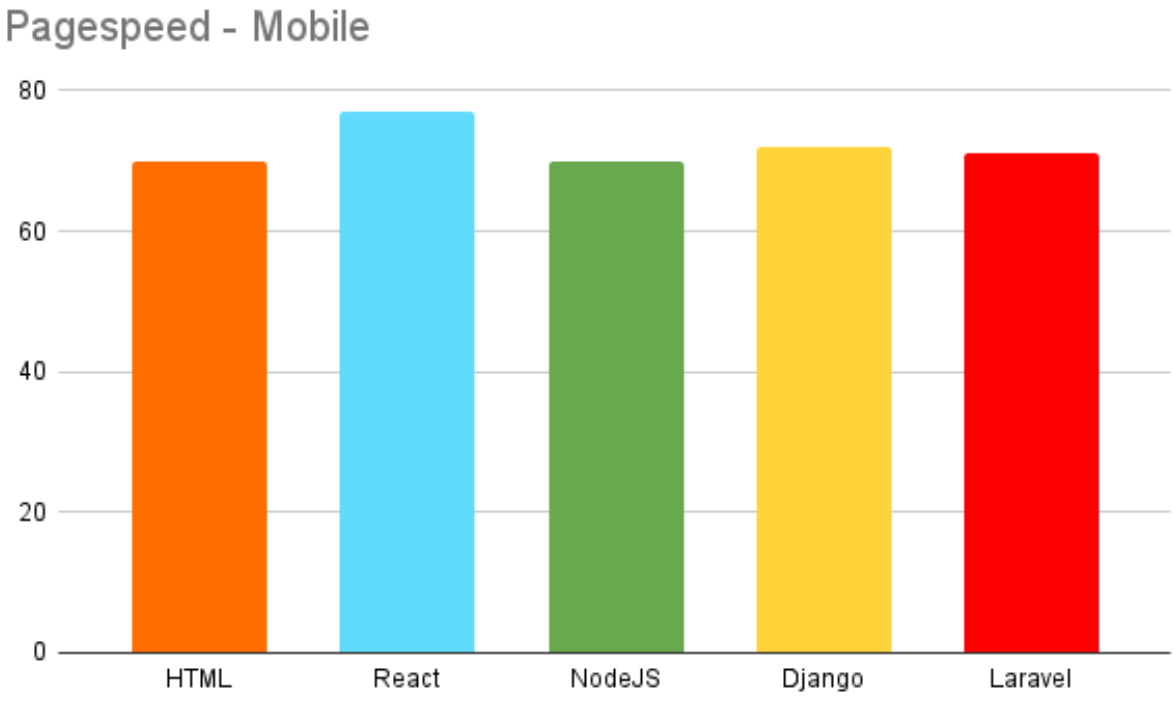
5.1 Narzędzia Google

5.1.1 PageSpeed Insights - Mobile

Testy przeprowadzone przy pomocy narzędzia Page Speed Insights [23] wykazały następujące rezultaty dla renderowania badanych aplikacji w kontekście szybkości i jakości wyświetlania na urządzeniach mobilnych:

	HTML	React	NodeJS	Django	Laravel
Pagespeed - Mobile	70	77	70	72	71
First Contentful Paint [s]	1,5	1,8	2,2	2,2	1,5
Speed Index [s]	4,4	2,1	3	2,2	3,8
Largest Contentful Paint [s]	7,4	5,6	8,4	8,2	7,3
Time to interactive [s]	5,4	4,4	5,1	3,9	5,5
Total Blocking time [ms]	70	90	40	0	30
Cumulative Layout Shift	0,002	0,002	0	0	0

Tablica 5.1: Wyniki badań przeprowadzonych za pomocą narzędzia PageSpeed Insights dla urządzeń mobilnyych



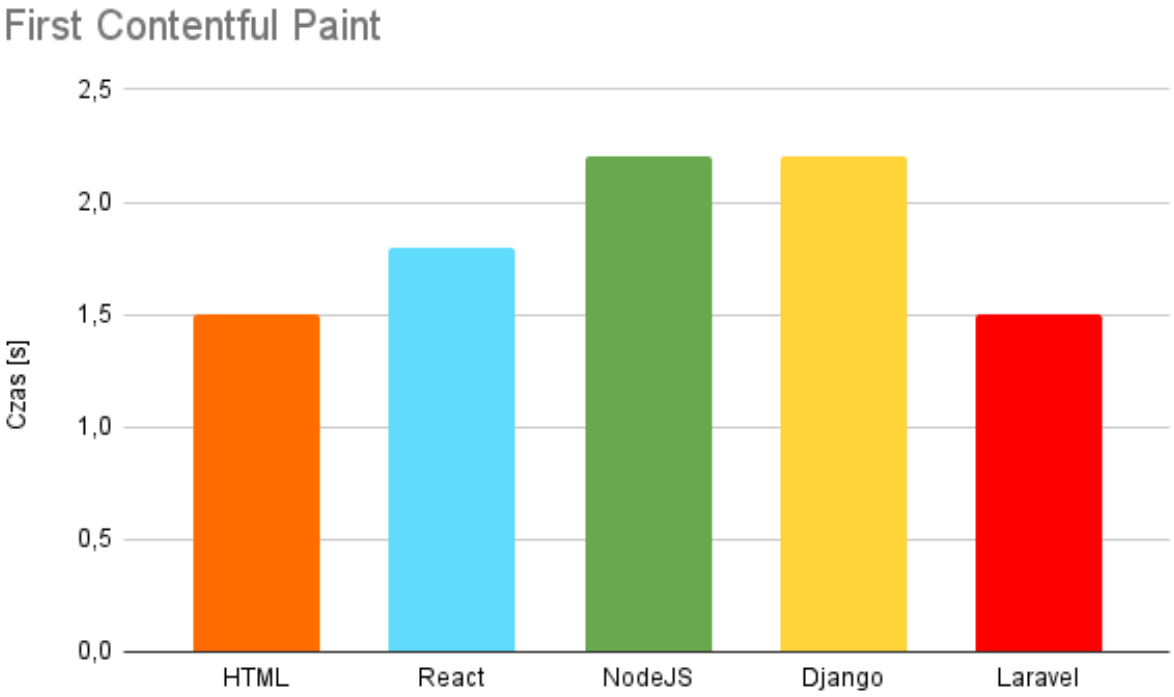
Rysunek 5.1: Wyniki badań narzędziem PageSpeed Insights Mobile. Źródło własne

PageSpeed Insights Mobile jest zbiorczą metryką oceniającą ogólną wydajność stron internetowych w środowiskach mobilnych. na wynik tego badania składa się szereg pomniejszych czynników omówionych w późniejszych podpunktach tej sekcji. Jednocześnie już te wyniki pozwalają wysunąć pierwsze spostrzeżenia.

Spośród dedykowanych środowisk programistycznych żadne nie uzyskało wyniku niższego od bazowej strony wyświetlanej jedynie za pomocą statycznych dokumentów HTML, co potwierdza zasadność stosowania ich, jako podstawę do tworzenia aplikacji internetowych.

Na tle innych programów znacząco wyróżnia się biblioteka React, która

osiągnęła wynik o 10% wyższy niż podstawowa strona. U podstaw tej przewagi leży przede wszystkim bardzo dobry paramets *Speed Index* oraz bezkonkurencyjny czas ładowania największego obrazu strony (ang. *Largest Contentful Paint*).



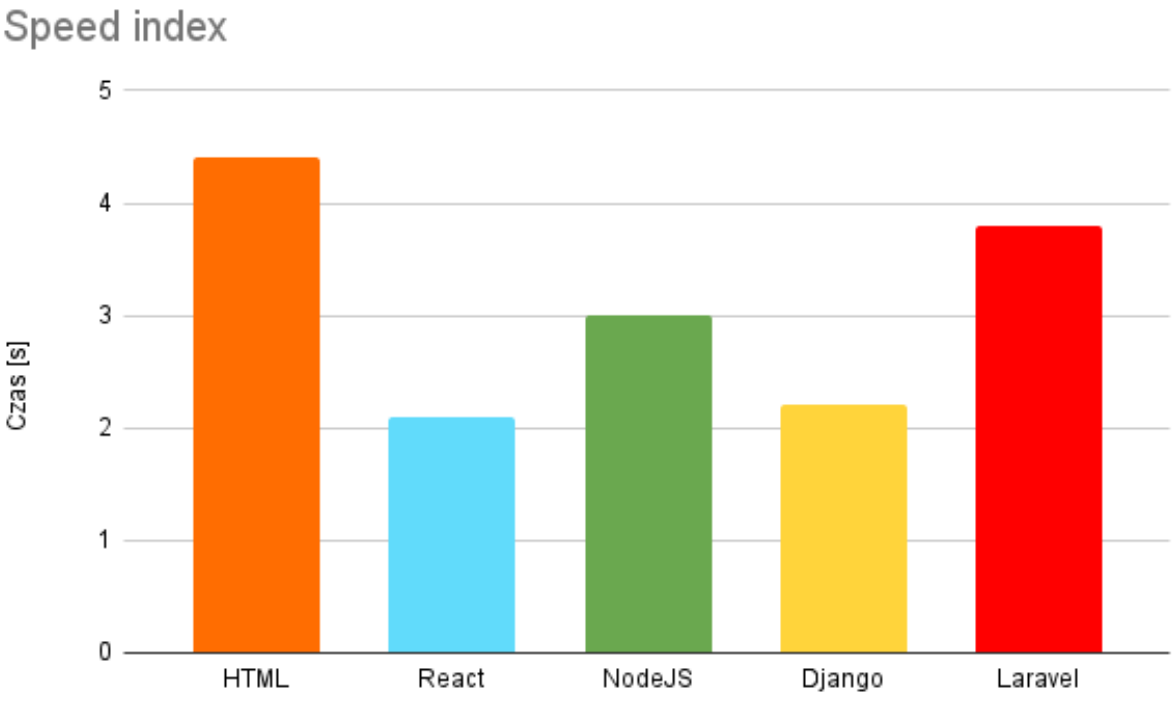
Rysunek 5.2: Wyniki badań narzędziem PageSpeed Insights Mobile dla metryki *First Contentful Paint*. Źródło własne

First Contentfun Paint [24] jest metryką, która mierzy czas od chwili wysłania zapytania, które zostaje odebrane przez aplikację, do momentu wyświetlenia na stronie wyświetlanie na ekranie użytkownika pierwszej zawartości finalnej strony, która jest wartościowa dla użytkownika. w skład takich treści wchodzi przede wszystkim tekst wyświetlany na stronie, obrazy (również obrazy tła) oraz elementów <*svg*> lub <*canvas*> o kolorach innych niż

biały.

Zgodnie z instrukcjami dostarczanymi przez Google, aby zapewnić użytkownikowi dobre doświadczenie użytkowania aplikacji należy minimalizować wartość tego pomiaru. Jako złoty standard dla otrzymania pierwszych elementów strony został przyjęty czas 1.8 sekundy, który jak podaje autor narzędzia powinien być co najwyżej trzecim kwartylem czasów osiąganych przez użytkowników witryny. Strony które nie spełniają tego wymagania, jednak ich czas wciąż mieści się w przedziale poniżej 3 sekund są wciąż uznawane jako wykonane w jakościowy sposób.

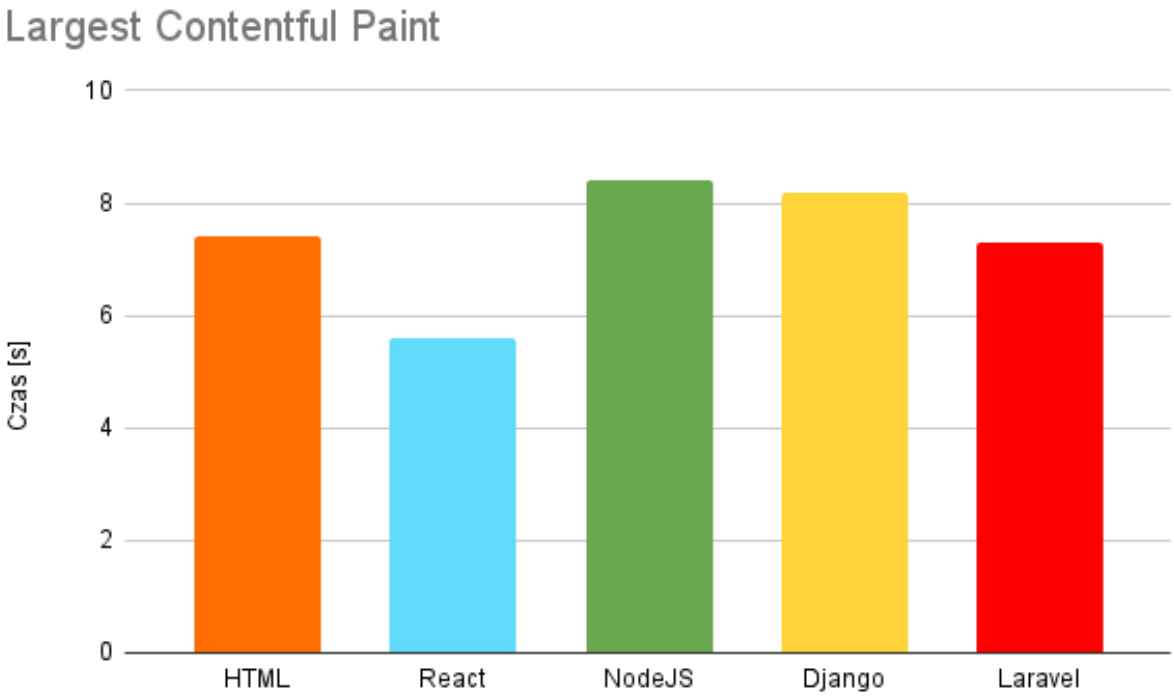
Spośród zebranych technologii jedynie oprogramowanie zbudowane na języku PHP uzyskało wynik tak dobry, jak bezpośrednie przekazywanie przez serwer gotowych plików HTML. Tak dobry rezultat jest zasługą zastosowania widoków *.blade.php*, które mogą być kompilowane w locie bez potrzeby zaciągania dodatkowych zasobów, co widocznie spowolniło pozostałe rozwiązania technologiczne.



Rysunek 5.3: Wyniki badań narzędziem PageSpeed Insights Mobile dla metryki *Speed Index*. Źródło własne

Speed Index [25] jest wskaźnikiem szybkości mierzącym szybkość wizualnego wyświetlania treści podczas ładowania strony. Narzędzie w pierwszej kolejności przechwytuje kolejne klatki obrazów w procesie ładowania strony w przeglądarce i następnie oblicza postęp wizualny następujący wraz z kolejnymi klatkami. Do wykonywania tych obliczeń służy specjalna implementacja modułu Speedline środowiska NodeJS.

Najlepsze wyniki w pomiarze tej wartości zostały osiągnięte przez bibliotekę React oraz środowisko programistyczne Django. Jest to z dużą pewnością zasługa strumienia dynamicznie renderowanych modułów widoku, które znajdują podobne zastosowania w każdej z tych technologii.



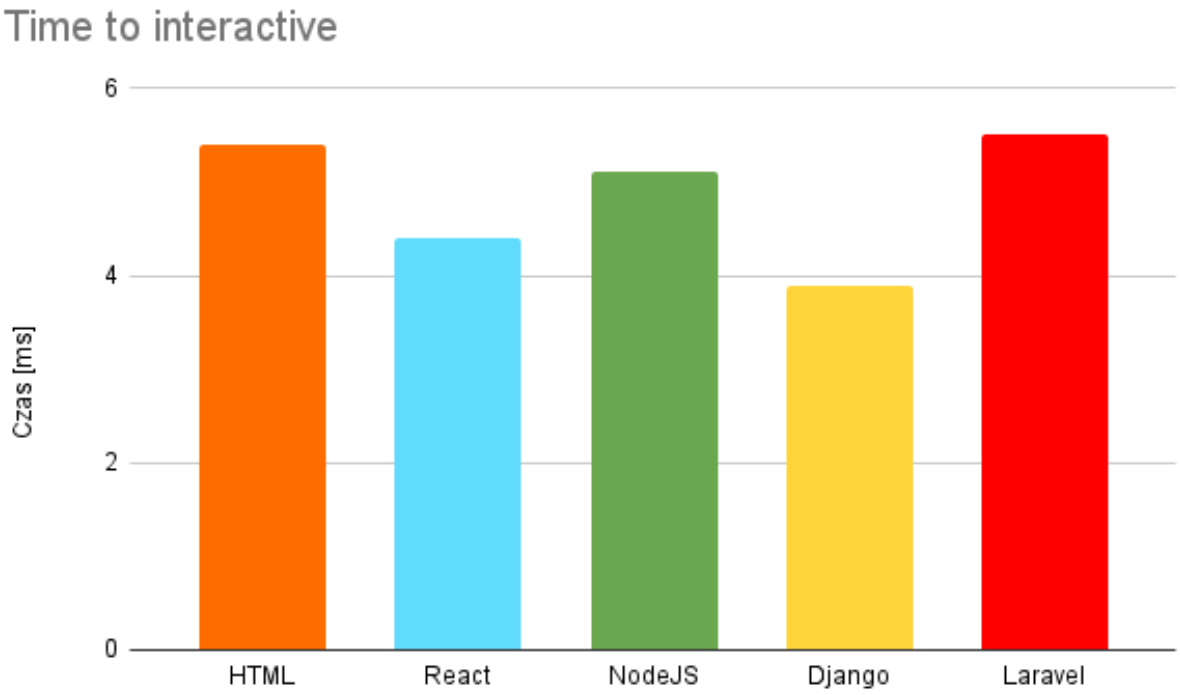
Rysunek 5.4: Wyniki badań narzędziem PageSpeed Insights Mobile dla metryki *Largest Contentful Paint*. Źródło własne

Metryka *Largest Contentful Paint* [26] odzwierciedla czas, który jest potrzebny aplikacji na wyświetlenie użytkownikowi największego elementu, najczęściej tekstowego lub obrazowego, który docelowo znajduje się na ekranie załadowanej strony w oknie użytkownika.

Ten sposób pomiaru jest sukcesorem stosowanych wcześniej technologii takich jak *Windows load event* czy *DOM Content Loaded*, które sprawdzały całkowity czas ładowania wszystkich elementów strony. Jednak z biegiem czasu witryny internetowy zawierają coraz więcej drobnych elementów takich jak widżety, które dodają znaczącą ilość czasu potrzebnego do pełnego załadowania strony, jednocześnie nie wpływając istotnie na doświadczenie użytkownika

podczas użytkowania witryny. Dlatego obecnie ważniejszy stał się czas, w którym aplikacja przedstawia użytkownikowi najważniejszą treść, która interesuje go na stronie.

Liderem w tej kategorii jest aplikacja bazująca na bibliotece React, która zawdzięcza swój sukces podziałowi strony na moduły, spośród których priorytetowo są renderowane te o największej istotności.



Rysunek 5.5: Wyniki badań narzędziem PageSpeed Insights Mobile dla metryki *Time to interactive*. Źródło własne

Time to interactive [27] sprawdza czas, który jest potrzebny stronie na osiągnięcie pełnej interaktywności z poziomu okna użytkownika. Do uznania strony za w pełni interaktywną konieczne jest spełnienie trzech kryteriów:

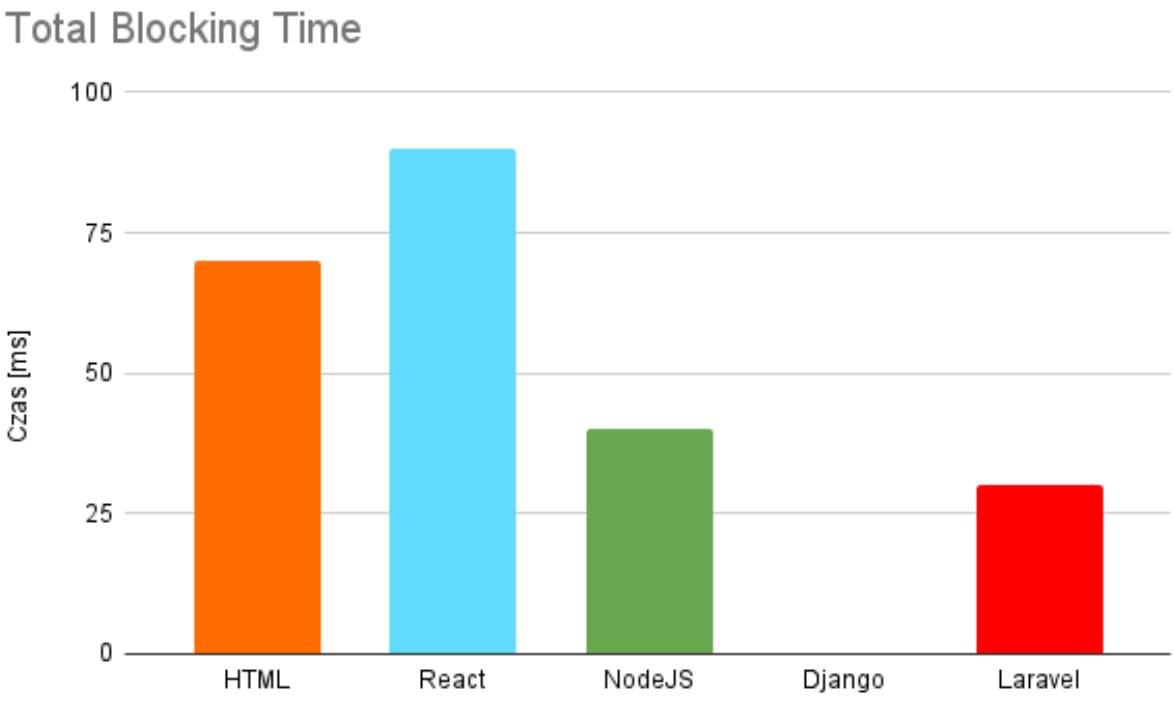
1. Strona wyświetla użyteczną zawartość, co jest mierzone za pomocą

wskaźnika First Contentful Paint,

2. Obsługa zdarzeń jest zarejestrowana dla większości widocznych elementów strony,

3. Strona reaguje na interakcje użytkownika w ciągu 50 milisekund.

Najlepszy rezultat w tym badaniu został osiągnięty za pomocą środowiska programistycznego Django, którego sukces jest ściśle powiązany z posiadaniem jednoznacznej tablicy odniesień adresów do widoku, która pozwala na szybkie odnajdywanie komponentów potrzebnych do responsywności.



Rysunek 5.6: Wyniki badań narzędziem PageSpeed Insights Mobile dla metryki *Total Blocking Time*. Źródło własne

Całkowity czas blokowania wątku (ang. *Total Blocking Time*) [28] mierzy

całkowity czas między wyrenderowaniem *First Contentful Paint* a czasem potrzebnym do odblokowania interakcji (ang. *Time to interactive*), w którym główny wątek był zablokowany na tyle długo, że uniemożliwiał reakcję na dane wejściowe.

Główny wątek jest uważany za żablokowanyźa każdym razem, gdy występuje zadanie, które działa na głównym wątku przez ponad 50 milisekund (ms). Określenie żablokowanyóznacza, że przeglądarka nie może przerwać zadania, które jest w toku. Jeśli więc użytkownik wejdzie w interakcję ze stroną w trakcie wykonywania długiego zadania, przeglądarka musi poczekać na zakończenie zadania, zanim będzie mogła na nie odpowiedzieć. Jeżeli czas blokowania wątku przekracza 50ms użytkownicy często uznają stronę za wolną lub nieresponsywną, co negatywne wpływa na ich percepcję aplikacji.

Bezpośrednim następstwem bardzo dobrego wyniku w metryce *Time to interactive* jest bardzo dobry rezultat osiągnięty przez aplikację opartą o Django. Największy czas blokowania głównego wątku wykazany przez renderowanie za pomocą React'a może być spowodowane sekwencyjnym kompilowaniem modeli, które w ten sposób mogą niepotrzebnie blokować proces przeglądarki.

5.1.2 PageSpeed Insights - Desktop

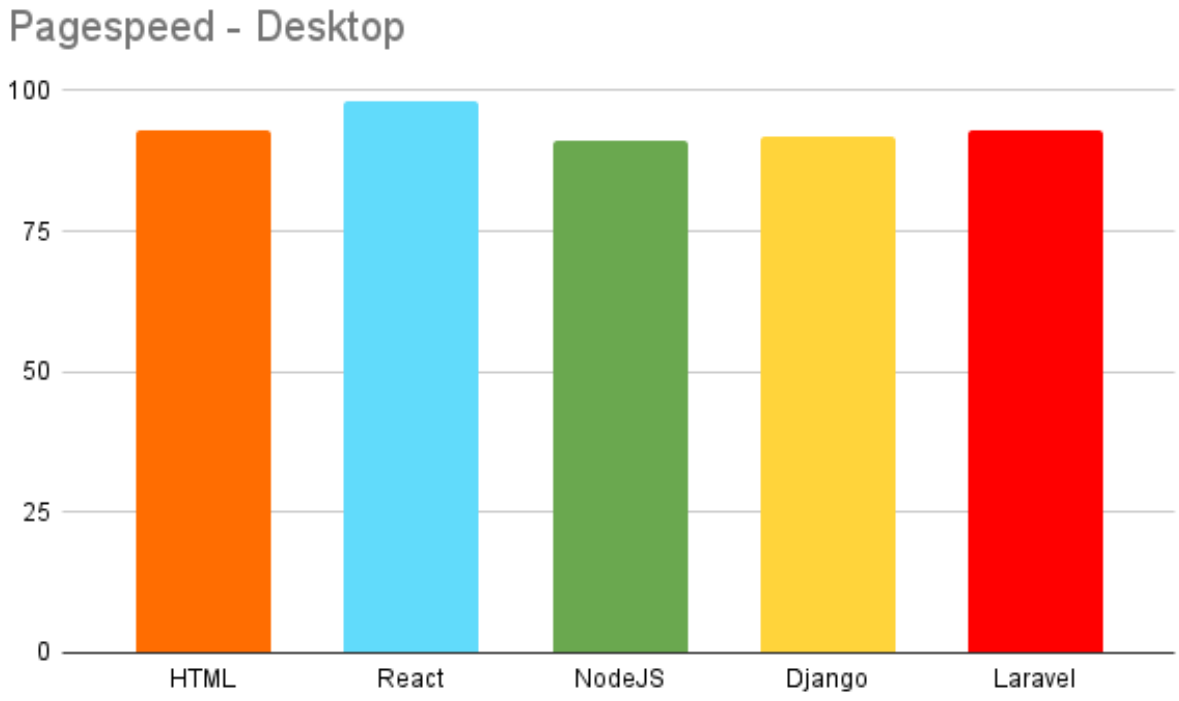
Badanie narzędziem PageSpeed Insights dla renderowania aplikacji w trybie przeznaczonym na urządzenia stacjonarne wykazało następujące rezultaty:

	HTML	React	NodeJS	Django	Laravel
Pagespeed - Desktop	93	98	91	92	93
First Contentful Paint [s]	0,5	0,5	0,7	0,7	0,5
Speed Index	1,1	0,6	1,1	0,7	1
Largest Contentful Paint [s]	1,7	1,2	1,8	1,8	1,7
Time to interactive [s]	1,2	0,7	1,3	1,1	1
Total Blocking time [ms]	0	0	0	0	0
Cumulative Layout Shift	0,006	0,01	0,006	0,006	0,006

Tablica 5.2: Wyniki badań przeprowadzonych za pomocą narzędzia PageSpeed Insights dla urządzeń stacjonarnych

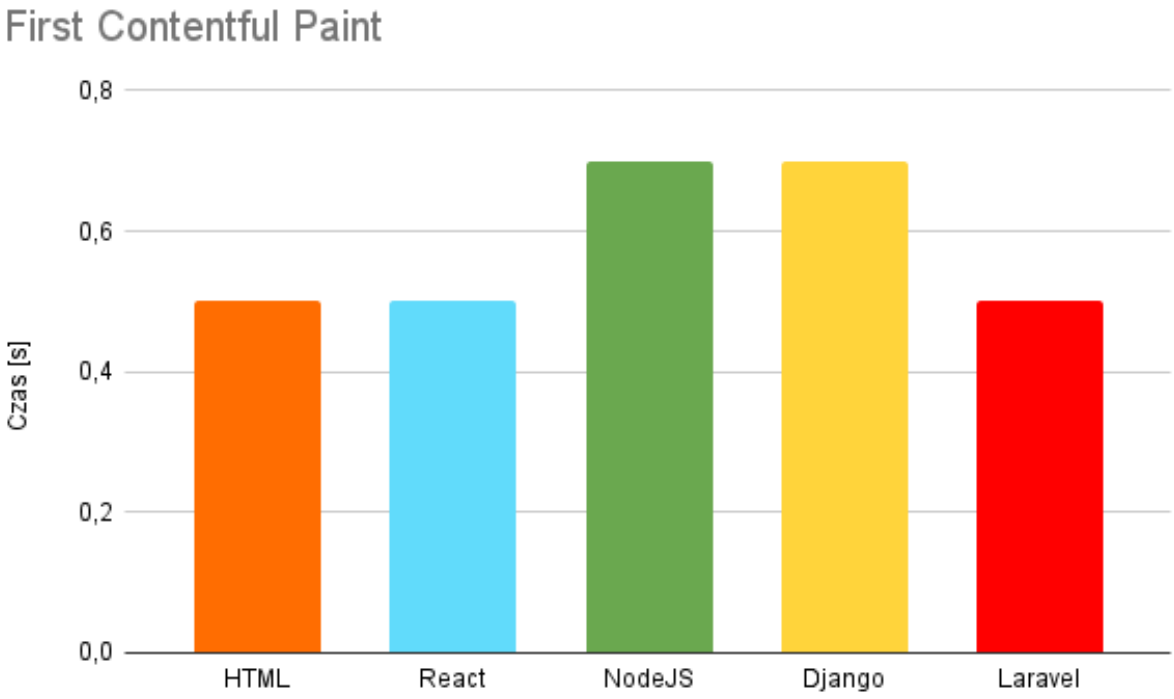
Wyniki przedstawione w tej sekcji są odpowiednikiem badań wykazanych w sekcji *PageSpeed Insights - Mobile*. Cała metodologia badań oraz interpretacja rezultatów pozostają niezmienione. Najistotniejszą różnicą jest przeprowadzenie tych pomiarów przy pomocy środowiska o wyższej mocy obliczeniowej oraz szybszym połączeniu internetowym, które mają odzwierciedlać przewagę platform stacjonarnych nad mobilnymi w tych aspektach.

Wprowadzenie nowych zmiennych pozwoli ze znacznie większą pewnością określić, które spośród wybranych technologii są predysponowane do tworzenia aplikacji z myślą o szerokiej skali urządzeń i oferują rozwiązania pozwalające czerpać korzyści z większego wachlarza środowisk klienckich.



Rysunek 5.7: Wyniki badań narzędziem PageSpeed Insights Desktop. Źródło własne

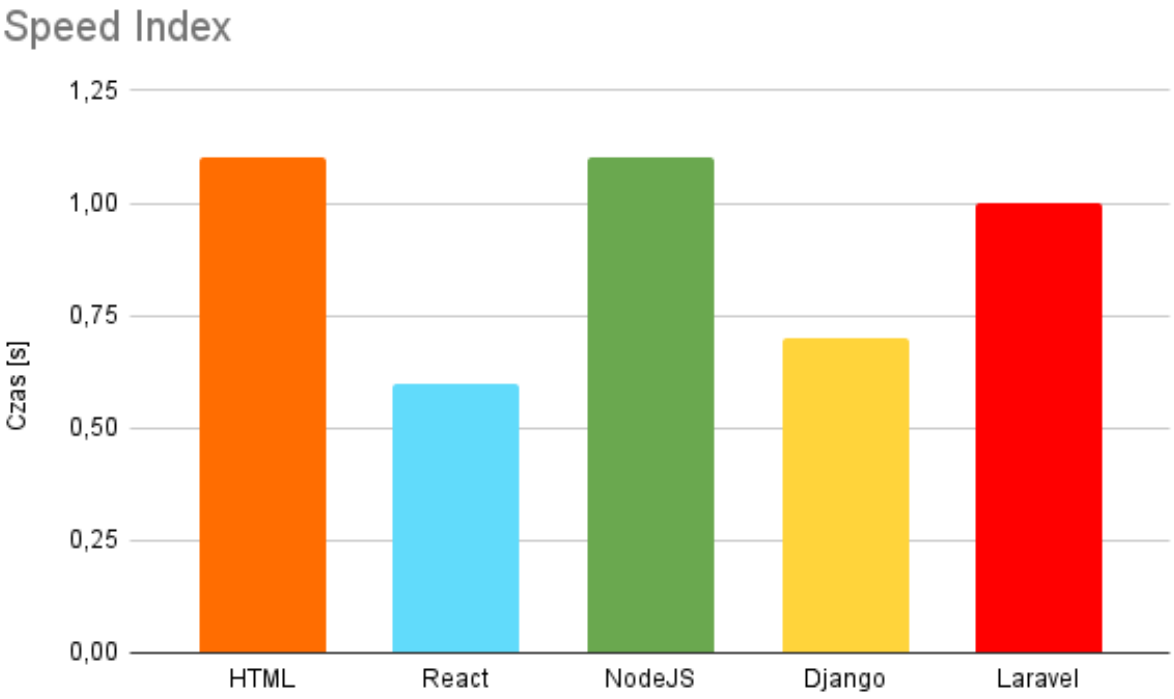
Dla zastosowań stacjonarnych można zaobserwować utrzymanie trendu dominacji aplikacji opartej o bibliotekę React, jednocześnie przy wzroście wyników wszystkich technologii skala tej przewagi znacząco spadła czyniąc rezultat końcowy znacznie bardziej wyrównanym.



Rysunek 5.8: Wyniki badań narzędziem PageSpeed Insights Desktop dla metryki *First Contentful Paint*. Źródło własne

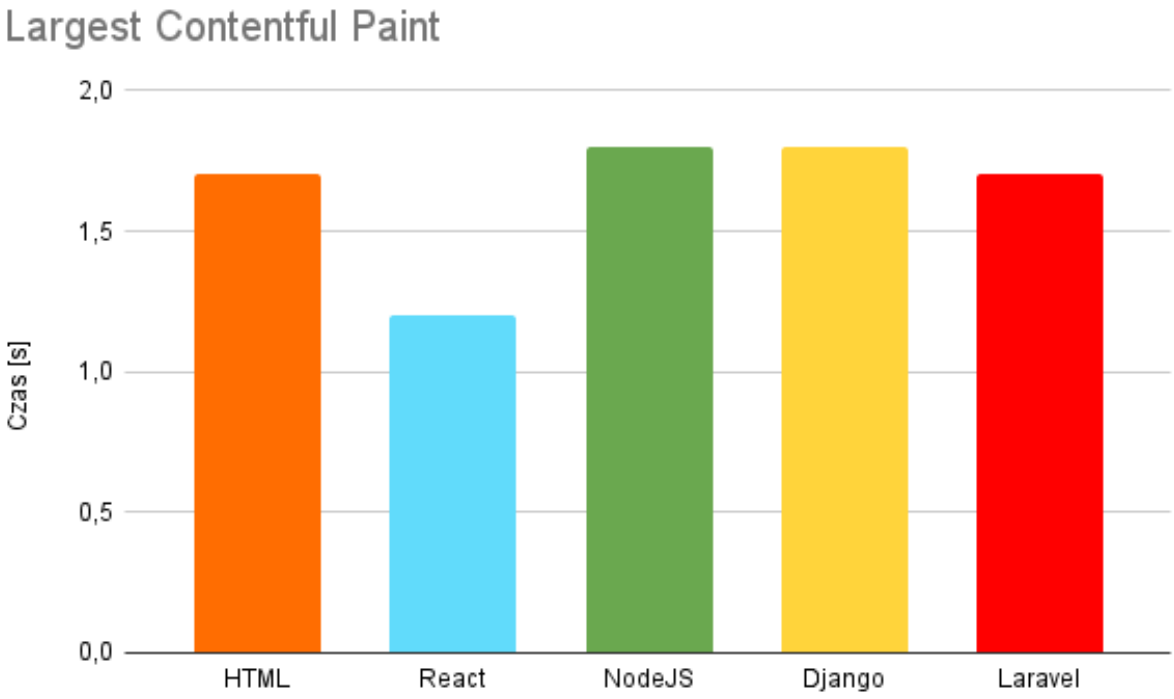
Przy zwiększonej mocy obliczeniowej obserwujemy znaczące przyspieszenie w renderowaniu pierwszych znaczących elementów na ekranie użytkownika. w większości przypadków nie wpłynęło to na relacje pomiędzy wynikami uzyskanymi przez badane technologie.

Największą poprawę można zaobserwować w zastosowaniu technologii React, która dzięki środowiku o większej mocy obliczeniowej jest w stanie zrównać się wydajnością z technologiami wiodącymi w tym zestawieniu.



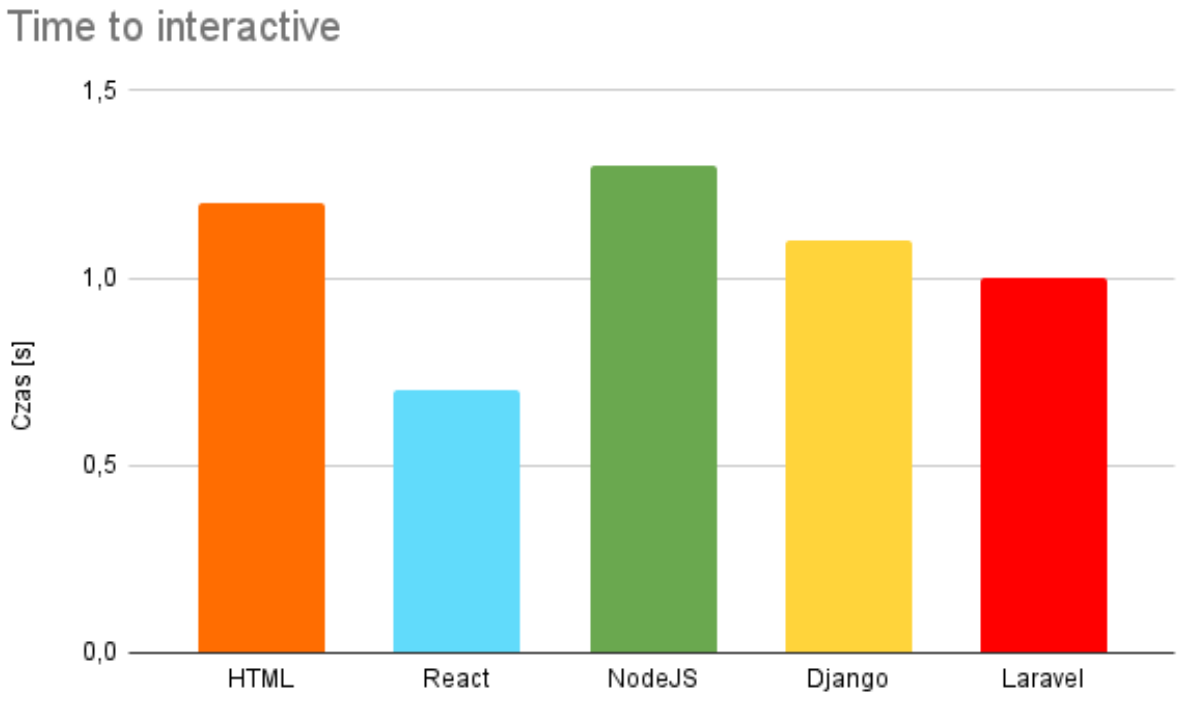
Rysunek 5.9: Wyniki badań narzędziem PageSpeed Insights Desktop dla metryki *Speed Index*. Źródło własne

Najwidoczniejszym odstępstwem przy renderowaniu aplikacji na stacjonarnych urządzeniach jest spadek jakości wykonywania procesów w aplikacji zaprojektowanej przy pomocy NodeJS, która w odniesieniu do innych technologii dokonuje mniejszego przyspieszenia procesu renderowania strony wyjściowej.



Rysunek 5.10: Wyniki badań narzędziem PageSpeed Insights Desktop dla metryki *Largest Contentful Paint*. Źródło własne

W przypadku renderowania największej części wartościowych informacji na stronie zwiększona moc obliczeniowa znajduje liniowe odbicie w wydajności wszystkich aplikacji objętych testem.



Rysunek 5.11: Wyniki badań narzędziem PageSpeed Insights Desktop dla metryki *Time to interactive*. Źródło własne

Wpływ jakości sprzętu na czas potrzebny aplikacjom do uzyskania pełnej interaktywności jest największy dla strony opartej o technologię Laravel, która dzięki dodatkowym zasobom jest w stanie przyjąć i obsłużyć niezbędne w tym celu żądania użytkownika w czasie jednej sekundy.

5.1.3 Lighthouse - Mobile

Przeprowadzone testy przy użyciu narzędzia Lighthouse wykazały następujące rezultaty dla generowania treści przeznaczonej na urządzenia mobilne:

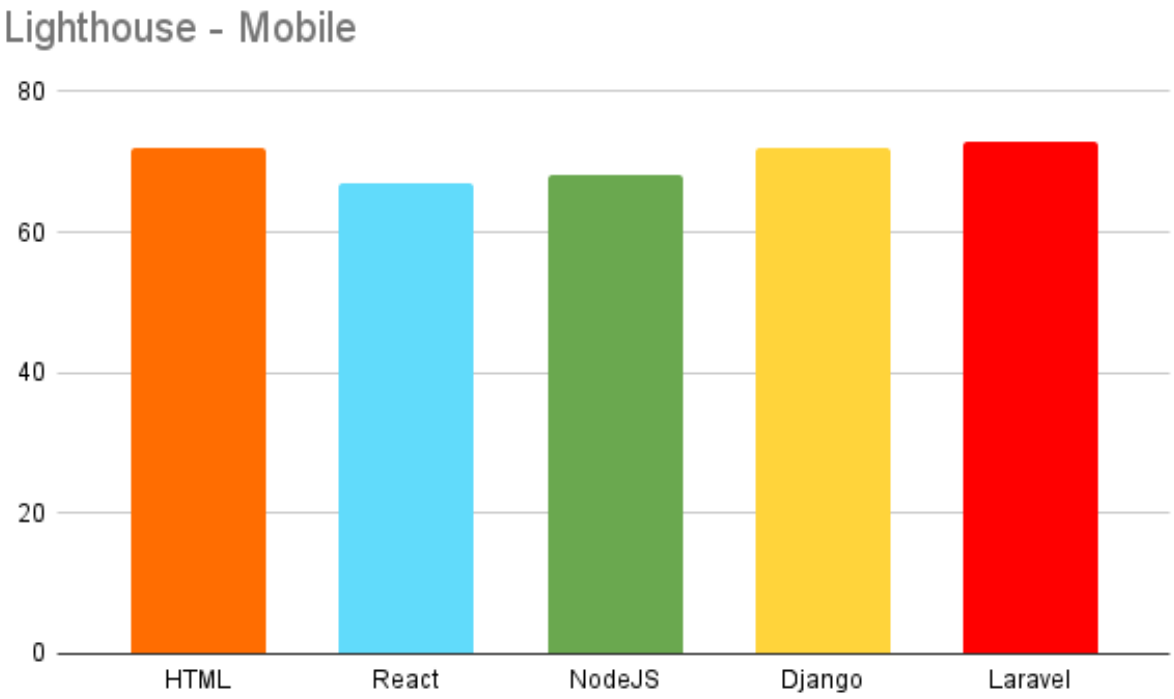
	HTML	React	NodeJS	Django	Laravel
Lighthouse - Mobile	72	67	68	72	73
Ułatwienia Dostępu	78	78	78	78	78
Sprawdzone Metody	83	92	92	92	83
SEO	100	97	100	100	100

Tablica 5.3: Wyniki badań przeprowadzonych za pomocą narzędzia Lighthouse dla urządzeń mobilnych

Podczas przeprowadzania audytu Google Lighthouse narzędzie kilkakrotnie wczytuje witrynę docelową i zbiera metryki dotyczące struktury, znaczników i wydajności witryny. Audyt dostarcza raport wydajności w kilku kategoriach w postaci oceny liczbowej w skali od 0 do 100. Każda z tych kategorii jest oceniana na podstawie następujących kryteriów:

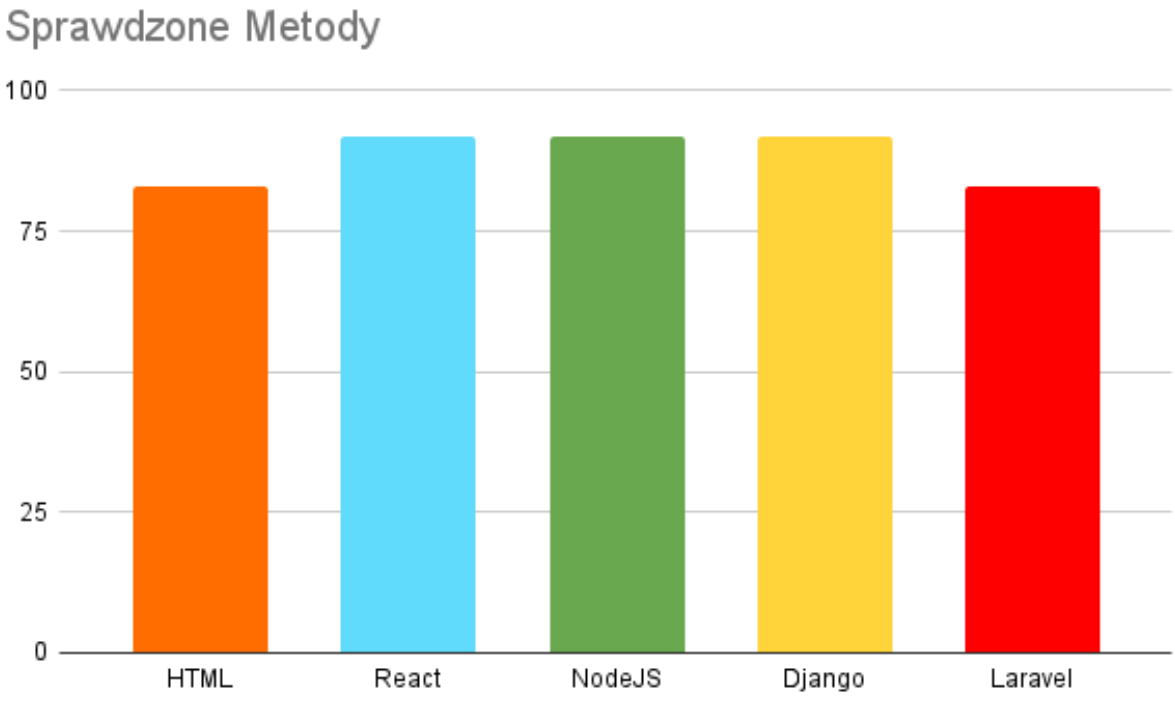
- Ocena główna jest oceniana na podstawie szybkości ładowania strony podobnych do metryk PageSpeed Insight,
- Ułatwienia Dostępu są metryką określającą przyjazność strony dla osób korzystających z technologii takich jak czytniki ekranu lub mających problemy z prawidłowym rozpoznawaniem barw,
- Sprawdzone Metody określają czy projekt strony wpasowuje się w nowoczesne standardy projektowania strony internetowych,
- SEO (ang. *Search Engine Optimisation* sprawdza czy strona jest dopasowana pod kątem optymalizacji witryn dla pozycjonowania

w wynikach wyszukiwania. Jest to duży obszar projektowania stron internetowych, w którym znajduje się prawidłowe wdrożenie elementów takich jak nazwy nagłówków, uzupełnienie słów kluczowych czy stworzenie alternatywnych opisów obrazów koniecznych do prawidłowej interpretacji przez silnik wyszukiwania.



Rysunek 5.12: Wyniki badań narzędziem Lighthouse Mobile. Źródło własne

W ogólnej ocenie stron od najwyższego standardu nieznacznie odstają aplikacje oparte o język programowania JavaScript. Może być to spowodowane brakiem implementacji wysokopoziomowego modułu umożliwiającego asynchroniczne przetwarzanie kodu źródłowego (ang. *top level await functions*) w starszych wersjach języka. Brak takich operacji może znacząco zwiększać obciążenie głównego wątku, co negatywnie wpływa na ocenę jakości.



Rysunek 5.13: Wyniki badań narzędziem Lighthouse Mobile dla metryki *Sprawdzone Metody*. Źródło własne

Niższy poziom oceny w zakresie zastosowania sprawdzonych metod projektowania aplikacji internetowych widoczny w wyniku technologii Laravel jest najprawdopodobniej spowodowany priorytetowym traktowaniem szybkiej kompilacji, która utrudnia odwoływanie się do najnowszych standardów w implementacji bibliotek i modułów dostępnych w pozostałych technologiach.

5.1.4 Lighthouse - Desktop

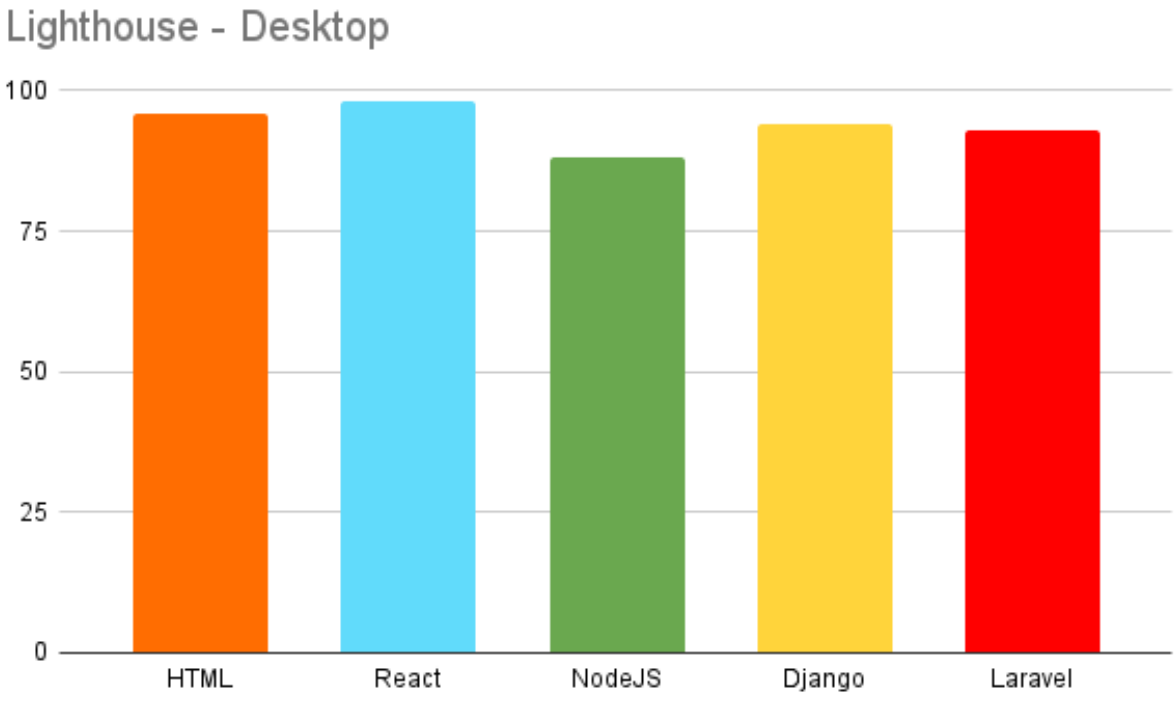
Rezultaty badań przeprowadzonych przy pomocy narzędzia Lighthouse sprawdzającego jakość generowanych treści wyświetlanej na urządzeniach stacjonarnych prezentują się w następujący sposób:

	HTML	React	NodeJS	Django	Laravel
Lighthouse - Desktop	96	98	88	94	93
Ułatwienia Dostępu	78	78	78	78	78
Sprawdzone Metody	83	92	92	92	83
SEO	100	100	100	100	100

Tablica 5.4: Wyniki badań przeprowadzonych za pomocą narzędzia Lighthouse dla urządzeń stacjonarnych

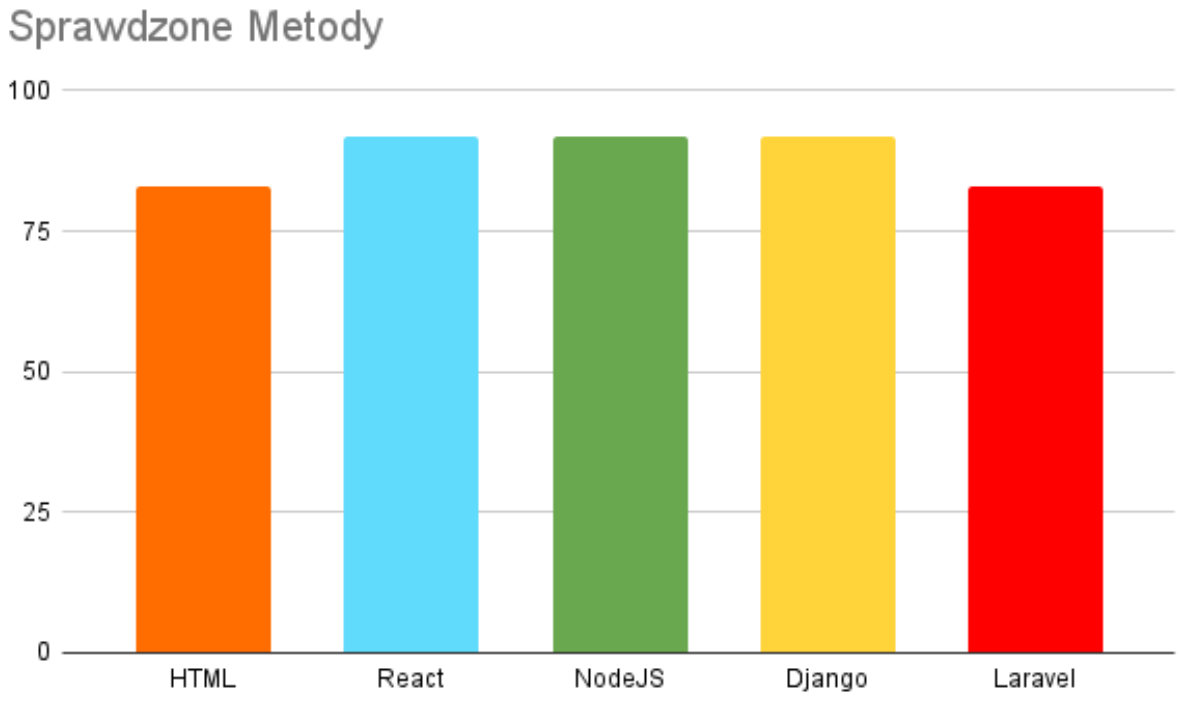
Wyniki przedstawione w tej sekcji są odpowiednikiem badań wykazanych w sekcji *Lighthouse - Mobile*. Cała metodologia badań oraz interpretacja rezultatów pozostają niezmienione. Najistotniejszą różnicą jest przeprowadzenie tych pomiarów przy pomocy środowiska o wyższej mocy obliczeniowej.

Wprowadzenie nowych zmiennych pozwoli ze znacznie większą pewnością określić, które spośród wybranych technologii są predysponowane do tworzenia aplikacji z myślą o szerokiej skali urządzeń i oferują rozwiązania pozwalające czerpać korzyści z większego wachlarza środowisk klienckich.



Rysunek 5.14: Wyniki badań narzędziem Lighthouse Desktop. Źródło własne

Przy większej mocy obliczeniowej przeprowadzonych badań od razu należy zauważyć znaczny wzrost oceny wydajności aplikacji opartych na JavaScipt. Dla biblioteki React szybszy wątek główny oznacza wzrost metryki ogólnej oceny o aż 46%, podczas gdy osiągi bazowej strony wzrosły jedynie o 33%. Jest to jasny sygnał, że dla urządzeń o większej wydajności aplikacje stworzone z użyciem środowisk operujących na języku JavaScript są bardzo dobrym wyborem.



Rysunek 5.15: Wyniki badań narzędziem Lighthouse Desktop dla metryki *Sprawdzone Metody*. Źródło własne

Jak należało się spodziewać poziom zaimplementowanych na poziomie projektowania aplikacji rozwiązań ocenianych przez metrykę jest zupełnie niezależny od jakości i mocy obliczeniowej używanego sprzętu, więc również wyniki nie różnią się na przestrzeni badań wykonywanych na różnych środowiskach.

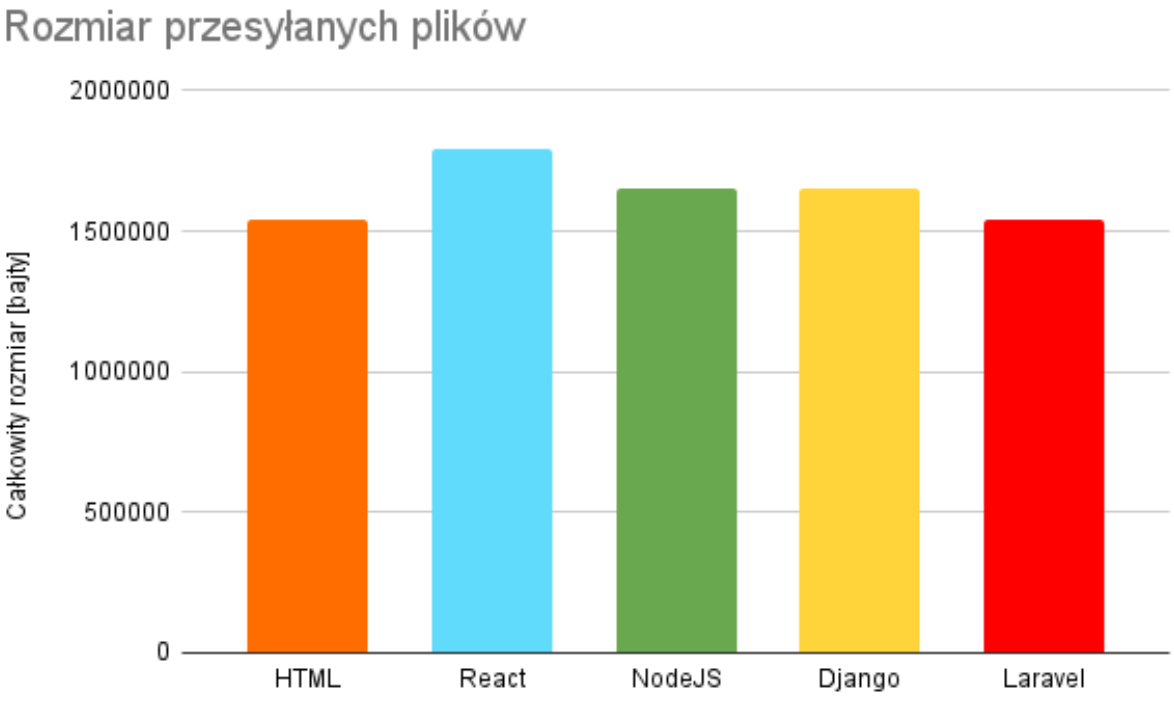
5.2 Narzędzia niezależne

5.2.1 websiteoptimization.com

Ilość zapytań oraz sumaryczna wielkość przesyłanych pakietów zbadane przy pomocy narzędzi dostępnych na stornie *websiteoptimization.com*:

	HTML	React	NodeJS	Django	Laravel
Total HTTP Requests	10	10	10	10	10
Total Size [bajty]	1543481	1789000	1652908	1652957	1543481

Tablica 5.5: Wyniki badań przeprowadzonych za pomocą narzędzia dostępnego na stronie *websiteoptimization.com*



Rysunek 5.16: Wyniki badań narzędziem Website Optimization dla metryki *Rozmiar przesyłanych plików*. Źródło własne

Narzędzie dostępne na stronie internetowej *websiteoptimization.com* pozwala na dokładny pomiar ilości danych przesyłanych przez aplikacje w celu wyrenderowania widoku całej strony całej strony.

Bardzo ciekawą obserwację stanowi identyczna ilość przesłanych danych przez aplikację stworzoną w środowisku Laravel do tych dostarczanych przez bazową stronę operującą jedynie na plikach HTML. Jest to rezultat kompilowania całości kodu PHP po stronie serwera i przesyłanie do przeglądarki jedynie gotowego pliku, który w tym przypadku jest identyczny.

Pozostałe środowiska potrzebują przesłać dodatkowe dane zawierające kod niezbędny do renderowania widoków koniecznych do prawidłowego wyświetlania

końcowego rezultatu.

Największa ilość danych przesyłanych przez aplikację opartą o React'a wynika bezpośrednio ze struktury tego projektu. React nie jest to pełnoprawnym środowiskiem programistycznym, a jedynie biblioteką, w związku z tym konieczne jest przesłanie z serwera definicji wszystkich funkcji znajdujących się w wykorzystywanych pakietach, pomimo iż większość z nich jest nieprzydatna w końcowym procesie tworzenia strony.

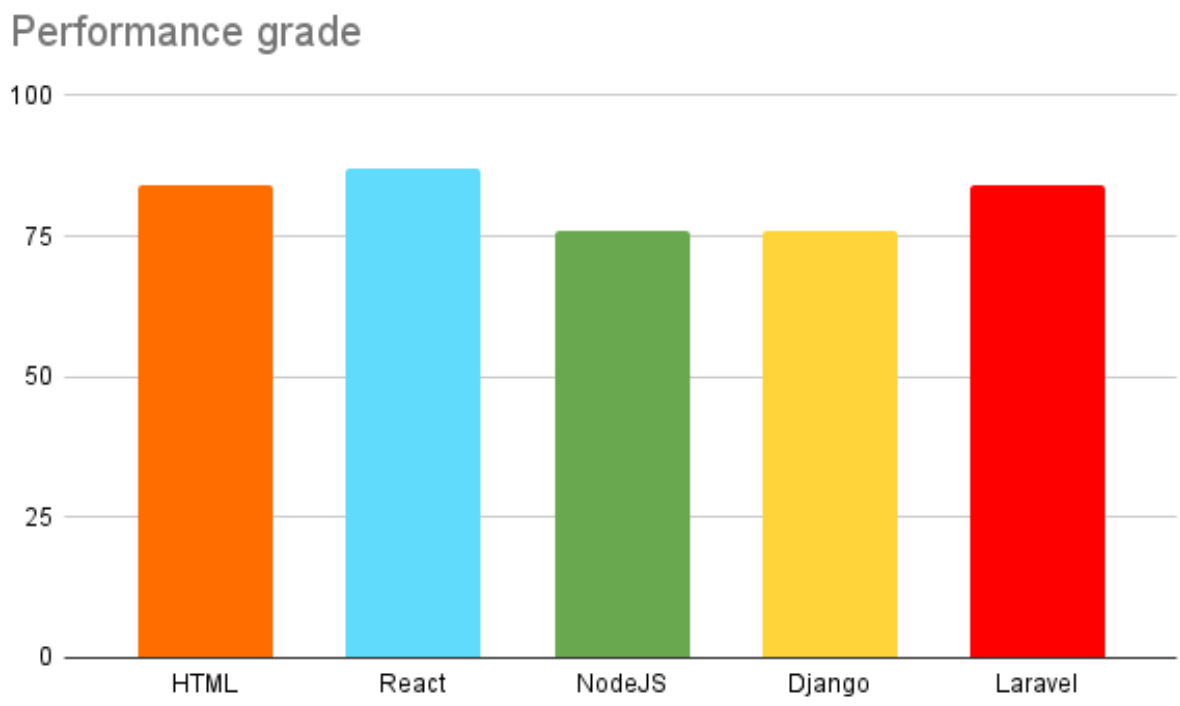
5.2.2 tools.pingdom.com

Ocena wydajności aplikacji uzyskana przy pomocy narzędzi badawczych dostępnych na stronie *tools.pingdom.com*:

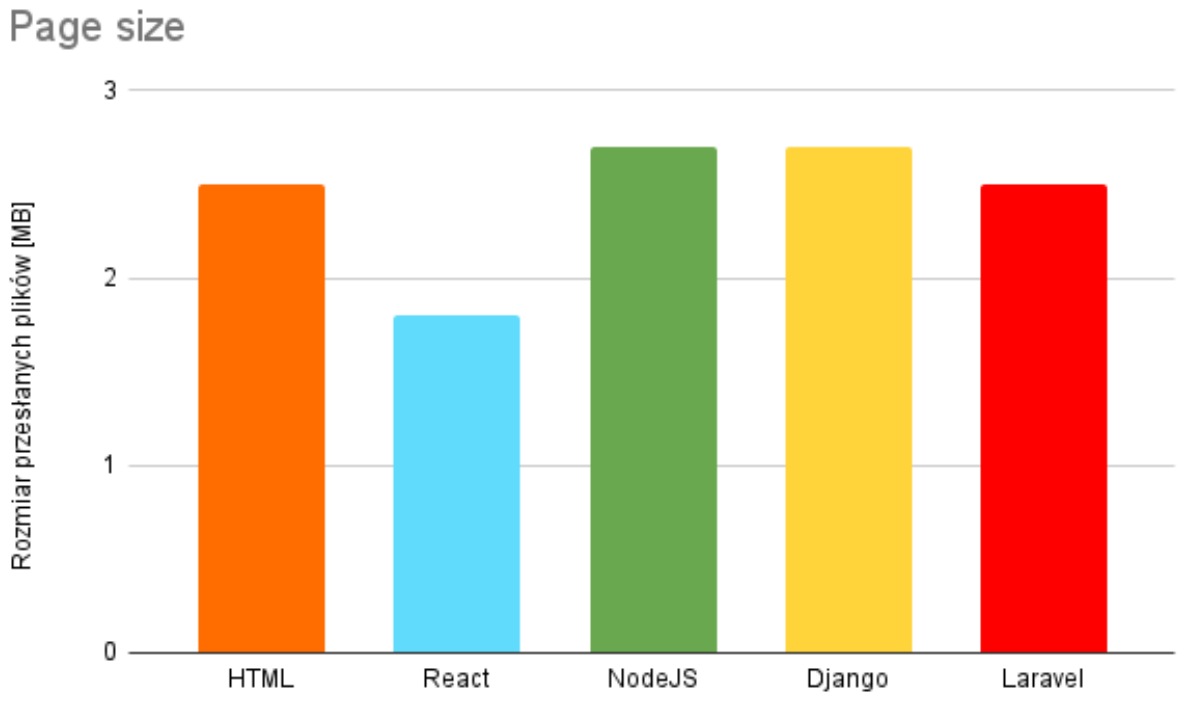
	HTML	React	NodeJS	Django	Laravel
Performance grade	84	87	76	76	84
Page size	2,5	1,8	2,7	2,7	2,5
Load time	550	228	580	434	550
Requests	23	15	23	23	23

Tablica 5.6: Wyniki badań przeprowadzonych za pomocą narzędzia dostępnego na stronie *tools.pingdom.com*

Narzędzia dostępne w witrynie *tools.pingdom.com* są ogólnodostępnymi testami szybkości działania aplikacji internetowych.



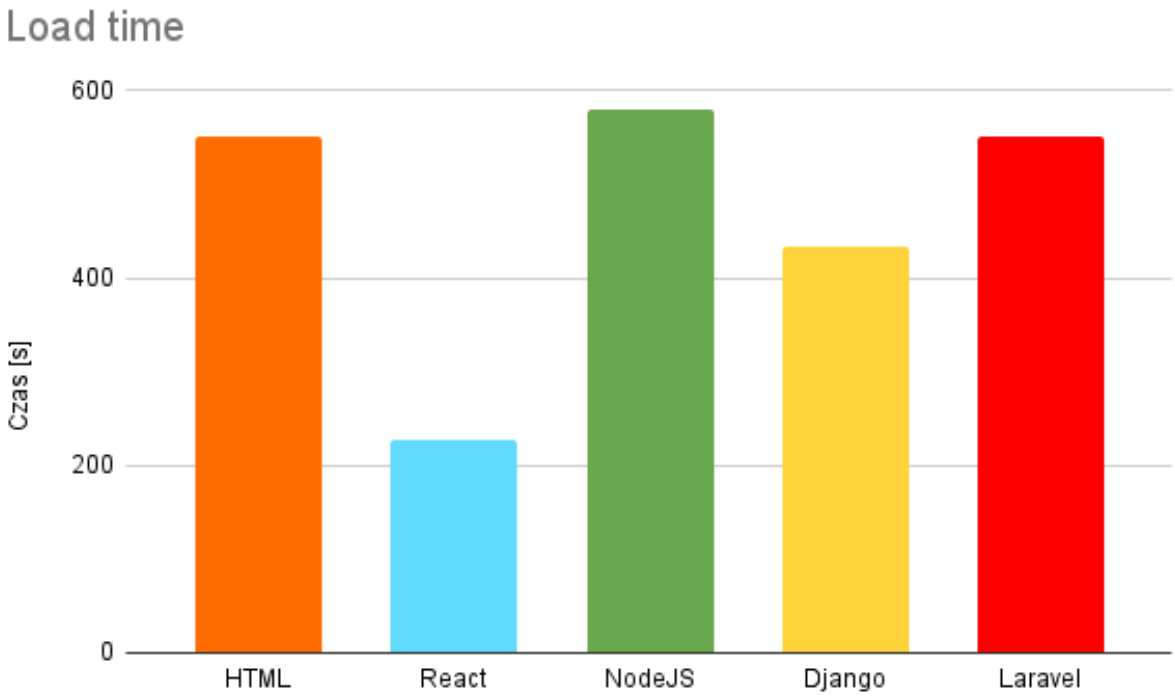
Rysunek 5.17: Wyniki badań narzędziem Tools Pingdom dla metryki *Performance grade*. Źródło własne



Rysunek 5.18: Wyniki badań narzędziem Tools Pingdom dla metryki *Page size*. Źródło własne

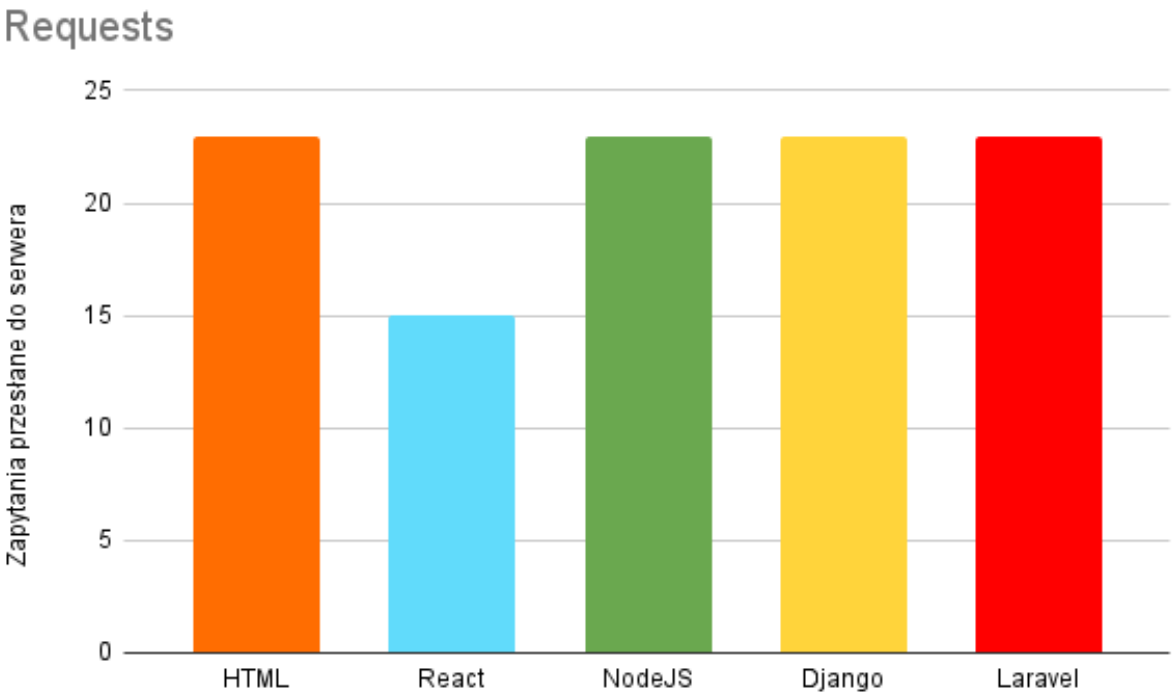
Finalny rozmiar strony jest metryką pozwalającą zobaczyć końcowy rozmiar przeglądanej przez użytkownika strony, w momencie jej pełnego załadowania.

Pomimo widocznej wcześniej największej ilości przesłanych plików, biblioteka React bardzo sprawnie minimalizuje kod pozostawiany w przeglądarce użytkownika. Jest to zapobiegawcze działanie, które ma jednocześnie zapobiec przetrzymywaniu w pamięci zbyt dużej liczby nieużywanych modułów. Dzięki bardzo rozbudowanemu systemowi czyszczenia pamięci z niepotrzebnych w przyszłości modułów końcowy rozmiar aplikacji używającej React'a jest znacząco, bo o ponad 25% mniejszy od kolejnej w zestawieniu technologii.



Rysunek 5.19: Wyniki badań narzędziem Tools Pingdom dla metryki *Load time*. Źródło własne

Czas ładowania wszystkich aplikacji zmierzony niezależną metryką jest bliźniaczo podobny do badania *Speed Index* przeprowadzonego przy pomocy narzędzia *PageSpeed Insight*. Taki rezultat jedynie potwierdza wniosek o dużej użyteczności modułowości aplikacji opartych o React oraz Django przy tworzeniu szybki aplikacji internetowych.



Rysunek 5.20: Wyniki badań narzędziem Tools Pingdom dla metryki *Requests*. Źródło własne

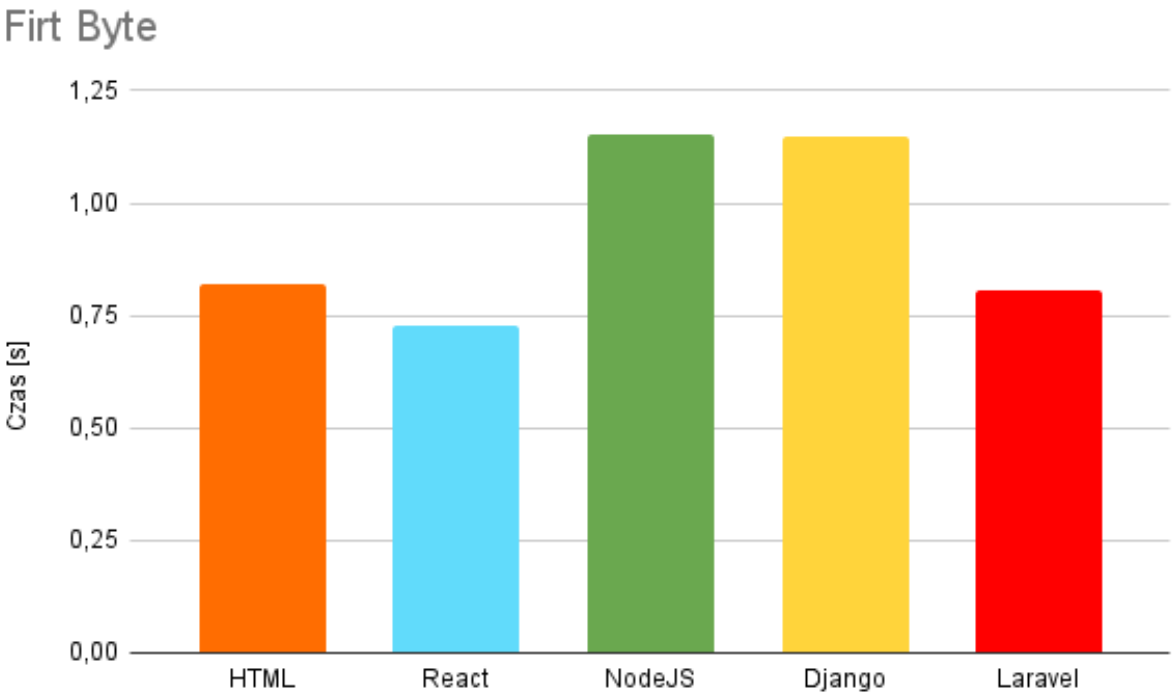
Ilość zapytań do serwera potrzebnych do uzyskania kompletnego widoku strony wskazuje na stałą liczę interpelacja potrzebnych do uzyskanie końcowego produktu dla wszystkich technologii z wyjątkiem React'a, który minimalizuje ilość potrzebnych interakcji klient-serwer dzięki rozpoznawaniu na poziomie kompilatora, które moduły aplikacji są skorelowane i potrzebne do przesłania, wyprzedzając tym samym ilość zapytań od użytkownika.

5.2.3 webpagetest.org

Dane na temat jakości i szybkości ładowania witryny uzyskane za pomocą narzędzia *webpagetest.org*:

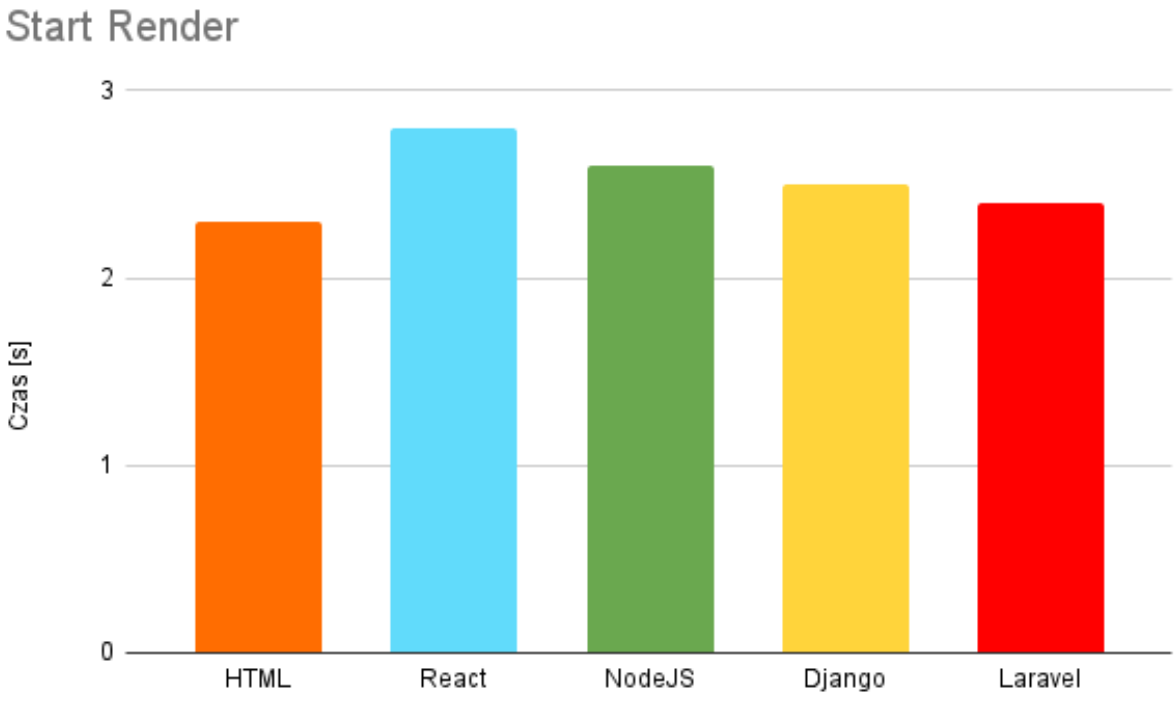
	HTML	React	NodeJS	Django	Laravel
First Byte [s]	0,82	0,727	1,153	1,148	0,805
Start Render [s]	2,3	2,8	2,6	2,5	2,4
First Contentful Paint [s]	2,096	2,673	2,452	2,459	2,264
Speed Index [s]	3,489	3,655	4,46	4,379	3,432
Largest Contentful Paint [s]	5,16	4,05	5,762	5,079	4,601
Cumulative Layout Shift	0,032	0,046	0,03	0,03	0,031
Total Blocking Time [s]	0,22	0	0,209	0,236	0,32
Total Byte [KB]	2,433	1,789	2,634	2,632	2,433

Tablica 5.7: Wyniki badań przeprowadzonych za pomocą narzędzia dostępnego na stronie *webpagetest.org*



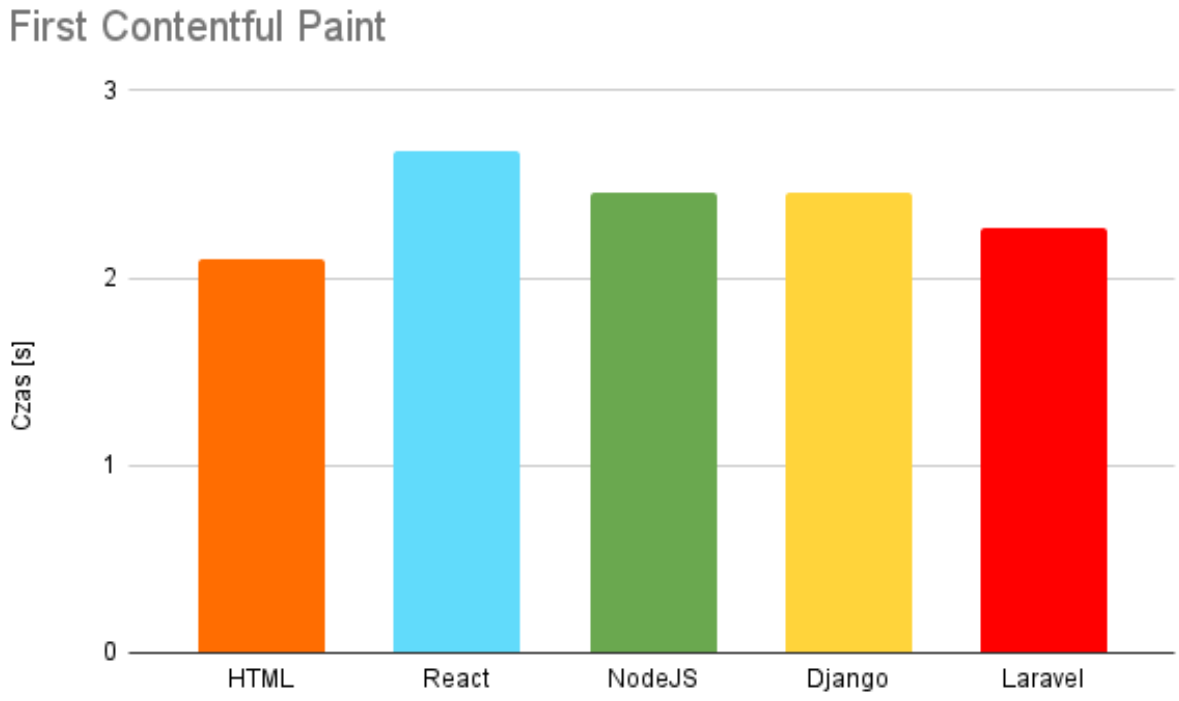
Rysunek 5.21: Wyniki badań narzędziem WebPageTest dla metryki *Firt Byte*. Źródło własne

Czas potrzebny na przesłanie pierwszego bita jest metryką związaną znacznie bliżej z jakością serwera, na którym funkcjonuje aplikacja i jego zmienność nie jest wartościowa dla badania skupiającego się na różnicach technologicznych.



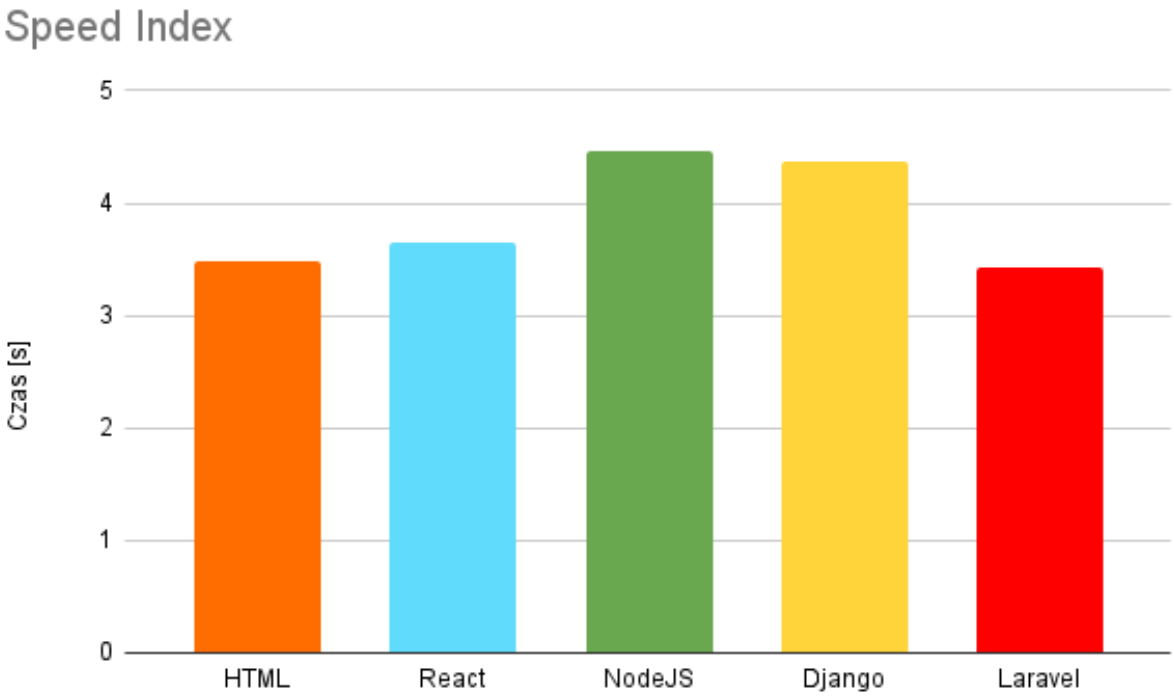
Rysunek 5.22: Wyniki badań narzędziem WebPageTest dla metryki *Start Render*. Źródło własne

Start Render jest badaniem określającą moment otrzymania przez użytkownika pierwszej informacji zwrotnej pozwalającej na wygenerowanie wizualnych elementów strony. Jest to metryka podobna do *First Contentful Paint*. Możemy zauważyć jednak, że wszystkie aplikacje poddane temu badaniu uzyskują bardzo podobne rezultaty w tych dwóch pomiarach, co pozwala nam stwierdzić, że w pierwszej kolejności kompilowane są elementy stron, które pozwalają na wyświetlenie treści użytkownikowi.



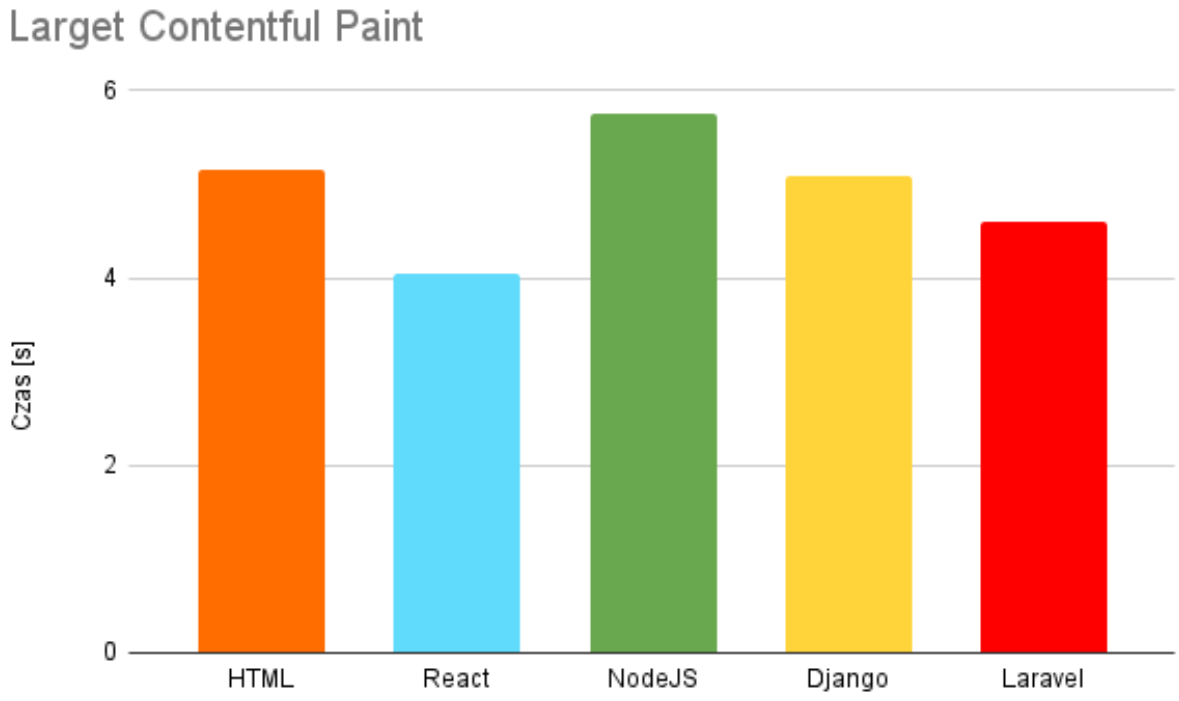
Rysunek 5.23: Wyniki badań narzędziem WebPageTest dla metryki *First Contentful Paint*. Źródło własne

First Contentful Paint wyznaczony przy pomocy innej metodologii potwierdza większość rezultatów osiągniętych podobnym badaniem używając aplikacji *PageSpeed Insight* z wyjątkiem bardzo dobrego wyniku uzyskanego przez aplikację opartą na React. Biorąc pod uwagę zarówno rezultaty badań *PageSpeed Insight* w wersji na urządzenia mobilne, jak i stacjonarne, możemy przypuszczać, że symulowany w tym badaniu sprzęt posiadał jeszcze niższą moc obliczeniową, która zgodnie z poprzednimi przypuszczeniami ma największy wpływ na działanie aplikacji opartych o bibliotekę React.



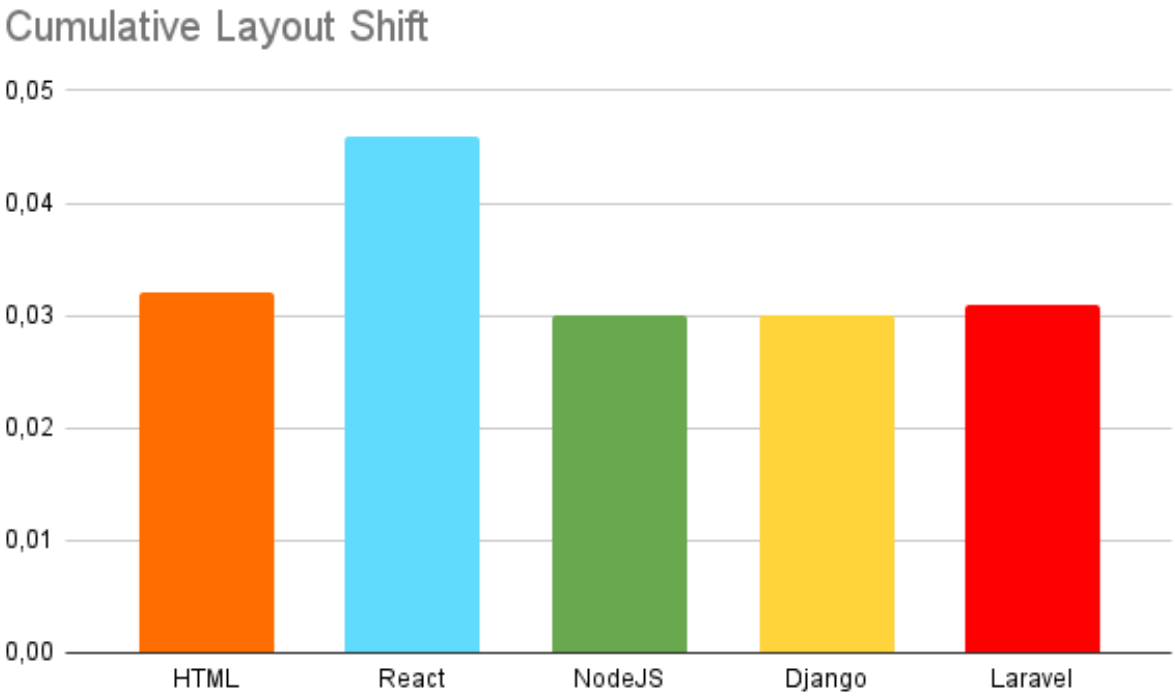
Rysunek 5.24: Wyniki badań narzędziem WebPageTest dla metryki *Speed Index*. Źródło własne

Różnice w wynikach badania z podobnymi testami przestawionymi w badaniach *Tools Pingdom Load Time* oraz *PageSpeed Speed Index* prawdopodobnie wynikają z różnic w metodologii tych testów. Poprzednie próby liczyły wartość szybkości ładowania strony od momentu uzyskania pierwszej odpowiedzi od serwera. Narzędzie *webpagetest.org* oblicza tę metrykę od momentu wysłania zapytania, co faworyzuje aplikacje o szybszym czasie obsłużenia pierwszego żądania, co nie jest w pełni zależne od zastosowanej technologii.



Rysunek 5.25: Wyniki badań narzędziem WebPageTest dla metryki *Largest Contentful Paint*. Źródło własne

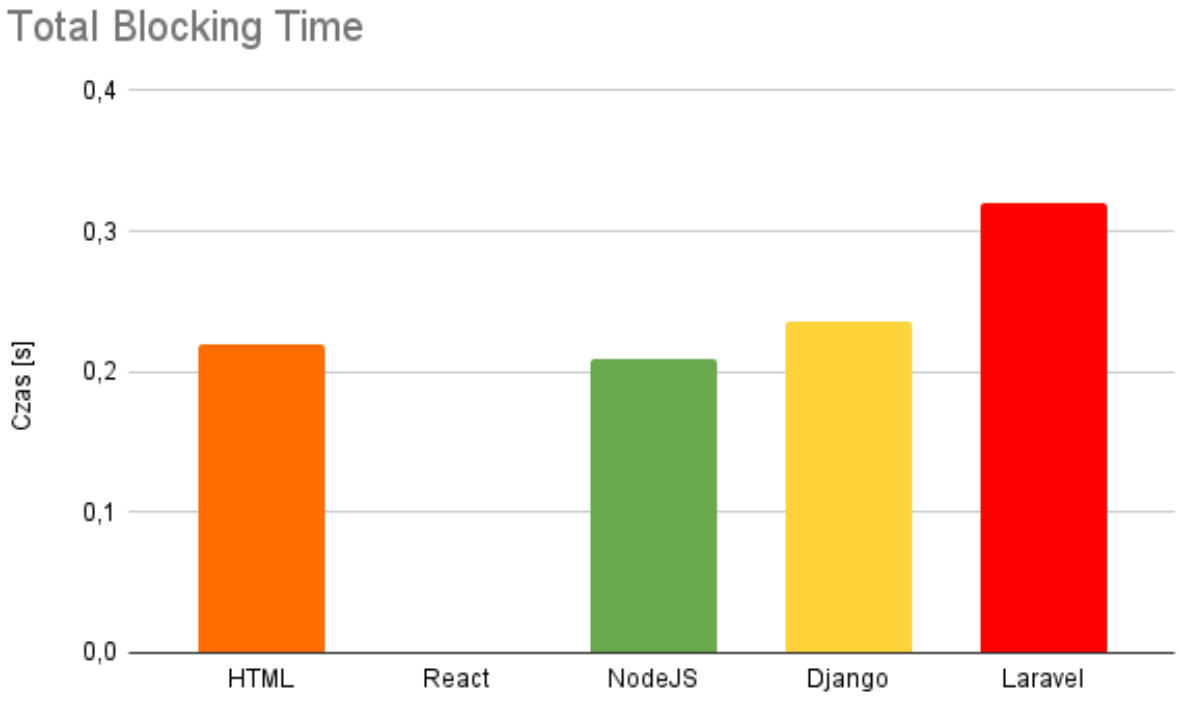
Uzyskane za pomocą tego badania wartości czasu potrzebnego do wyrenderowania największej części znaczących dla użytkownika treści potwierdzają przypuszczenia wysunięte w analizie badania *PageSpeed Insigth - Largest Contentful Paint*.



Rysunek 5.26: Wyniki badań narzędziem WebPageTest dla metryki *Cumulative Layout Shift*. Źródło własne

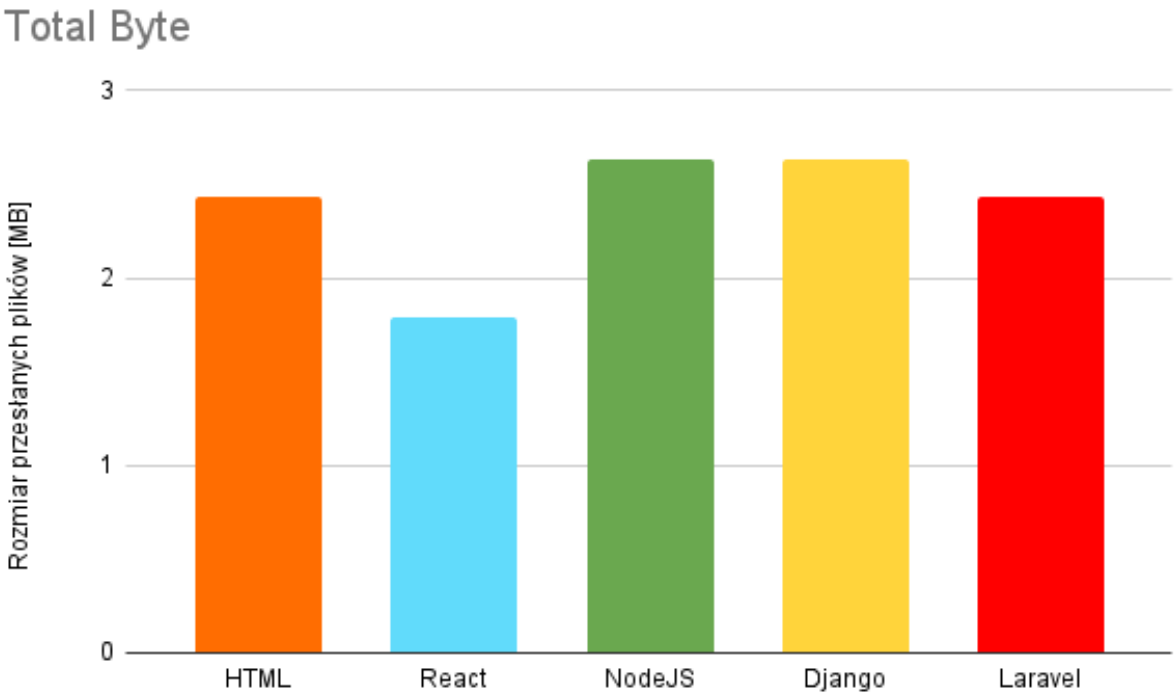
Cumulative Layout Shift [29] jest metryką odzwierciedlającą ilość zawartości, która podczas renderowania strony została przesunięta w sposób nieoczekiwany dla użytkownika. Przyczyną takiego zachowania elementów jest najczęściej zła obsługa obrazów o nieznanych wymiarach, błędne renderowanie czcionki lub treści dynamicznie wyświetlane na stronie.

Technologią, która wywołała największe przesunięcie w strukturze strony był React, jednak uzyskane przez wszystkie z aplikacji wyniki są tak małe, że ich wpływ na wygląd i funkcjonalność strony jest niezauważalny dla normalnego użytkownika.



Rysunek 5.27: Wyniki badań narzędziem WebPageTest dla metryki *Total Blocking Time*. Źródło własne

Uzyskane w tym badaniu wartości są niewiele większe od błędu pomiarowego, co w połączeniu ze skrajnie różnymi rezultatami względem podobnego badania przeprowadzonego w punkcie *PageSpeed Insights Mobile - Total blocking time* wskazuje na dużą zależność tej metryki od czynników niezależnych, takich jak pojemność pamięci podręcznej urządzenia testowego lub wielkość szyny pamięci procesora w tym urządzeniu.



Rysunek 5.28: Wyniki badań narzędziem WebPageTest dla metryki *Total Byte*. Źródło własne

Uzyskane za pomocą tego badania wartości rozmiaru pamięci zajmowanego przez końcową postać wyświetlanych użytkownikowi stron potwierdzają przypuszczenia wysunięte w analizie badania *Tools Pingdom - Page Size*.

Rozdział 6

Podsumowanie

Celem niniejszej pracy było zbadanie wpływu platformy programistycznej na jakość tworzonej aplikacji, pod względem wartości wskaźników internetowych oraz potencjału pozycjonowania. Część teoretyczna zawiera historię i podstawowe założenia globalnej sieci internetowej oraz wyszukiwarek internetowych, jako podstawowej ścieżki poruszania się po sieci większości użytkowników. Podkreślono również wysoką wartość optymalizowania stron internetowych pod względem wskaźników jakości internetowej wyznaczanych przez twórców silników wyszukiwarek.

Praca zawiera dokładne przedstawienie najpopularniejszych rozwiązań stosowanych w nowoczesnym projektowaniu, modelowaniu i tworzeniu aplikacji internetowych. Bazując na najstarszych technologiach wciąż można tworzyć strony spełniające wizualne standardy oczekiwane przez użytkowników. Wiele mniej rozbudowanych stron można utworzyć jedynie przy użyciu języka HTML, arkuszy stylów CSS i kompilowanych w przeglądarce kodów JavaScript. Jednak możliwości stwarzane przez nowoczesne środowiska programistyczne pozwalają na uproszczenie procesu tworzenia nowych aplikacji internetowych oraz usprawnienie ich funkcjonowania zarówno po stronie użytkownika, jak i obsługi ze strony serwera.

Wyniki przeprowadzonych badań pozwoliły na zweryfikowanie pozytywnego wpływu stosowania nowoczesnych środowisk programistycznych do konstrukcji aplikacji internetowych. W wielu aspektach zastosowanie *frameworków* pozwala na zmniejszenie nakładu pracy potrzebnego do uzyskania określonego zbioru funkcjonalności. Dodatkowo dobrze zaprojektowana aplikacja sieciowa może zredukować liczbę zapytań do serwera potrzebnych do uzyskania pełnego działania programu, a także zmniejszyć czas potrzebny na dostarczenie użytkownikowi żądanej usługi. Wybór najlepszego środowiska jest uzależniony od wielu czynników, w skład których wchodzą: docelowa platforma, grupa użytkowników, złożoność końcowej funkcjonalności czy potrzeby skalowania poziomego aplikacji w przyszłości. Jednocześnie korzyści płynące z zastosowania wysokopoziomowych rozwiązań programistycznych jednoznacznie pokazują na lepszą jakość zastosowania tych rozwiązań nad metodami statycznymi.

Metody tworzenia aplikacji internetowych są stale rozwijane ze względu na ciągle rosnące zainteresowanie przenoszeniem funkcjonalności życia codziennego do sieci. Przy wyborze technologii, definiującej ścieżki rozwoju aplikacji internetowych, istotne jest rozważenie mocnych i słabych stron każdej z dostępnych technologii, a także dostosowanie ich wyboru do kluczowych potrzeb funkcjonalności projektu.

Bibliografia

[1] *Google Time Machine: the Web in 2001*, http://googlesystem.blogspot.com/2008/09/google-time-machine-web-in-2001.html.

[2] *Indexing The World Wide Web*, https://static.googleusercontent.com/media/research.google.com/pl//pub

[3] *WordPress Market Share In 2022*, https://www.envisagedigital.co.uk/wordpress-market-share/.

[4] *Tools for Web Developers*, https://developers.google.com/web/tools.

[5] *PageSpeed Insights*, https://pagespeed.web.dev/.

[6] *LightHouse*, https://developers.google.com/web/tools/lighthouse.

[7] *Google Analytics*, https://marketingplatform.google.com/about/analytics/.

[8] *Google Search Console*, https://search.google.com/search-console/.

[9] *websiteoptimization*, http://www.websiteoptimization.com/services/analyze/.

[10] *Pingdom Tools*, https://tools.pingdom.com/.

[11] *Web Page Test*, https://www.webpagetest.org/.

[12] *Internet Assigned Numbers Authority*, https://www.iana.org/about.

[13] *ICANN*, https://www.icann.org/.

[14] *BM25*, https://nlp.stanford.edu/IR-book/html/htmledition/okapi-bm25-a-non-binary-model-1.html.

[15] *Search Engine Optimization (SEO) Starter Guide*, https://developers.google.com/search/docs/beginner/seo-starter-guide.

[16] *Hyper Text Markup Language Tags*, https://www.w3.org/History/19921103-hypertext/hyperte

[17] *PHP*, https://www.php.net/manual/en/intro-whatis.php.

[18] *Python*, https://www.python.org/doc/essays/blurb/.

[19] *Laravel*, https://laravel.com/.

[20] *Laravel Model View Controller*, https://www.howtogeek.com/devops/what-is-laravel-and-how

[21] *Guidelines for website design and usability*, https://blog.hubspot.com/blog/tabid/6307/bid/30557/6-guidelines-for-exceptional-website-desi

[22] *Laravel Blade templates*, https://scand.com/company/blog/what-is-laravel-used-for/.

[23] *PageSpeed Insights Mobile testing methology*, https://developers.google.com/speed/docs/insights/v4/about.

[24] *First Contentful Paint*, https://web.dev/fcp/.

[25] *Speed Index*, https://web.dev/speed-index/.

[26] *Largest Contentful Paint*, https://web.dev/lcp/.

[27] *Time to interactive*, https://web.dev/interactive/.

[28] *Total blocking time*, https://web.dev/tbt/.

[29] *Cumulative layout shift*, https://web.dev/cls/.

Załączniki

1. *Kod źródłowy aplikacji* – plik source_code.zip

Spis tablic

Spis rysunków