JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 32 (2024) 223–230
Received: 10 June 2024
Accepted: 3 July 2024

Performance analysis of web application client layer development tools using Angular, React and Vue as examples

Analiza wydajności narzędzi do tworzenia warstwy klienta aplikacji internetowych na przykładzie Angular, React i Vue

Damian Bogusz*, Paweł Ciszewski, Beata Pańczyk

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

Abstract
The article presents a comparative analysis of tools for developing the client layer of web applications. The tools chosen for it were Angular 17.1.0, React 18.2.0 and Vue 3.3.4. The research process was carried out using the *"Benchmark"* test application, implemented with each technology. It was prepared as a performance tester, implementing tests consisting of operations that manipulate a table of data, in order to measure their execution time. The results of the performed tests became the basis for further analysis. As a result of the study, Angular became the most efficient tool in the largest number of tests. Slightly lower performance was observed with Vue, and the least efficient application turned out to be the one prepared with React.

Keywords: Angular; React; Vue; comparative analysis

Streszczenie
W artykule przedstawiono analizę porównawczą narzędzi do tworzenia warstwy klienta aplikacji internetowych. Wybrano do niej narzędzia Angular 17.1.0, React 18.2.0 i Vue 3.3.4. Proces badawczy przeprowadzono z wykorzystaniem aplikacji testowej *"Benchmark"*, zaimplementowanej z użyciem każdej z technologii. Przygotowana została ona jako tester wydajności, realizujący testy złożone z operacji manipulujących tabelą danych, celem pomiaru czasu ich wykonania. Wyniki przeprowadzonych testów stały się podstawą do analizy porównawczej. W efekcie badania, najwydajniejszym narzędziem w największej liczbie testów został Angular. Nieznacznie mniejsza wydajność zaobserwowana została przy Vue, a najmniej wydajną aplikacją okazała się ta, przygotowana z użyciem React.

Słowa kluczowe: Angular; React; Vue; analiza porównawcza

*Corresponding author
Email address: damian.bogusz@pollub.edu.pl (D. Bogusz)

1. Wstęp

Rynek technologii do wytwarzania aplikacji internetowych ulega nieustannym zmianom. Liczne narzędzia wspomagające programistów poddawane są regularnym aktualizacjom usprawniającym oraz rozwijającym ich dotychczasowe możliwości. Wraz z szybkim rozwojem rynku część narzędzi traci na znaczeniu wśród deweloperów, podczas gdy inne je zyskują.

W związku z tym, istotnym wydaje się regularne przeprowadzanie badań porównawczych, pozwalających określić aktualny stan techniczny i wydajność najpopularniejszych z nich. Wyniki takich badań mogą być niezwykle pomocne zarówno dla indywidualnych programistów, jak i też dużych firm, wytwarzających oprogramowanie komercyjnie. Wybór najlepszego dla danego zastosowania narzędzia, najczęściej pod kątem wydajności gotowej aplikacji, okazuje się bowiem często kluczową decyzją początkowego etapu jej projektowania.

Celem niniejszego artykułu jest zaspokojenie tejże potrzeby, poprzez przeprowadzanie wydajnościowej analizy porównawczej narzędzi Angular 17.1.0, React 18.2.0 i Vue 3.3.4, zajmujących czołowe miejsca w rankingach popularności w ostatnich latach, na przykład w corocznej ankiecie dla programistów *Stack Overflow Developer Survey* [1]. Celem analizy jest wyłonienie lidera wśród rozważanych technologii.

Do realizacji celu posłużyła autorska aplikacja testowa *"Benchmark"*. Pozwoliła ona na porównanie wydajności badanych narzędzi z zastosowaniem elementów testowania wzorcowego (ang. *benchmark*). Analiza wyników badania pozwoliła na wytypowanie narzędzia cechującego się największą wydajnością.

2. Przegląd literatury

Przed przystąpieniem do badań przeanalizowano wybrane artykuły naukowe z branżowych konferencji i czasopism, podejmujące tematykę narzędzi React, Angular i Vue. Skupiono się tutaj na technicznych i wydajnościowych analizach szkieletów programistycznych i bibliotek służących do tworzenia warstwy klienckiej aplikacji internetowych.

W artykule *"Modern Web Frameworks: A Comparison of Rendering Performance"* [2], opublikowanym w marcu 2022 roku, rozważane są techniczne aspekty narzędzi React 17.0.1, Svelte 3.35.0, Angular 11.2.3, Blazor 5.0.3 i Vue 3.0.7, związane z różnicami w podejściach do renderowania komponentów aplikacji i towarzyszącym mu odświeżaniem treści statycznych i dynamicznych. Omówione zostają tu również rozbudowane

testy wzorcowe wykonane w ramach badania, których wyniki odzwierciedlają wcześniejsze rozważania autorów. Zgodnie z nimi, najlepsze wyniki osiągnął Svelte, znacząco wyprzedzając pozostałe narzędzia. Ostatnie miejsce w zestawieniu przypadło dla Blazor. Spośród pozostałych narzędzi pierwsze miejsce zajął Vue, a ostanie React. Autorzy sugerują, iż wyniki te w znacznej mierze spowodowane są przez różnice w mechanizmach związanych z renderowaniem treści. React i Blazor bowiem przetwarzają treści statyczne i dynamiczne podczas każdego przebiegu renderowania, w przeciwieństwie do pozostałych narzędzi, gdzie dla treści statycznych komponentu ma to miejsce jedynie podczas jego wstępnego renderowania.

Artykuł konferencyjny z 2020 roku, *"Comparison: Angular vs. React vs. Vue. Which framework is the best choice?"* [3] zawiera analizę porównawczą tytułowych narzędzi. Rozważane aspekty ukierunkowane były na zebranie danych pozwalających na wytypowanie rozwiązania najbardziej przyjaznego i wydajnego dla programisty. Poza wymienionymi aspektami, autorzy poruszyli tutaj także kwestię popularności narzędzi, poziom progu wejścia dla osób bez wcześniejszego doświadczenia z badanymi technologiami oraz elastyczność technologii. Przeprowadzona została także ankieta wśród programistów. Podsumowując przeprowadzoną analizę, autorzy przygotowali zestawienie technologii, w którym pod względem wydajności, elastyczności kodu i rozmiaru projektu, ostatnie miejsce zajął Angular. Jako rozwiązanie optymalne dla profesjonalnych programistów wytypowany został React. Z powodu niskiego progu wejścia, Vue określony został jako najbardziej przyjazny dla początkujących twórców aplikacji internetowych.

Innym artykułem, odnoszącym się do badanych technologii, jest *"Analiza obciążeniowa aplikacji internetowych z użyciem szkieletów Angular, React i Vue"* [4], opublikowany w czerwcu 2022 r. Zawarty został w nim techniczny przegląd narzędzi oraz analiza ukierunkowana na aspekt wydajności i szybkości działania. Autorzy swoje badania oparli na przygotowanych aplikacjach testowych o analogicznej funkcjonalności. Posłużyły one do pomiaru czasu wykonania określonych operacji CRUD (ang. *Create, Read, Update, and Delete*) celem zmierzenia wydajności. W wyniku analizy wyników najlepszy okazał się Vue. Następne miejsca zajęły kolejno Angular i React.

Artykuł *"Research and Analysis of the Frontend Frameworks and Libraries in E-Business Development"* [5], z 2019 roku, przedstawia analizę porównawczą technologii programistycznych ukierunkowaną na zastosowanie w sektorze e-biznesu. Autorzy rozpoczynają go od przedstawienia aktualnego stanu rynku tytułowych narzędzi oraz porównują ich wybrane aspekty techniczne jak np. budowa kodu, czy realizacja określonych procesów. W wyniku przeprowadzonej analizy, rozwiązaniami najbardziej wydajnymi okazały się React i Vue. Autorzy jednak zauważają, że są to narzędzia, które, z powodu swojej specyfikacji, najlepiej sprawdzą się przy mniejszych i średnich aplikacjach. Jako rozwiązanie mniej wydajne, lecz o bardzo rozbudowanych możliwościach, dla dużych aplikacji zalecają narzędzie Angular.

Przeanalizowane artykuły dostarczają różnych informacji o badanych technologiach, z różnych okresów czasowych oraz o sposobach ich analizowania i testowania. Widoczne jest, iż narzędzia te stale się zmieniają, z czym wiązać mogą się niekiedy różne wnioski płynące z podobnie przeprowadzonych badań. React, Angular i Vue regularnie zyskują od twórców nowe funkcjonalności oraz ulepszenia tych już istniejących. W związku z tym, cykliczne powtarzanie badań pozwalających określić ich bieżącą wydajność i pozycję na rynku jest uzasadnione.

Wyniki badań z 2019 i 2020 roku wskazywały na przewagę narzędzi React i Vue nad Angular w aspekcie wydajności. W artykułach z 2022 roku pierwsze miejsce w zestawieniach zajął Vue, jednak Angular okazał się być technologią bardziej wydajną od React. W ramach niniejszego artykułu sformułowano następującą hipotezę badawczą: *Aplikacje przygotowane za pomocą narzędzia Vue cechują się większą wydajnością od aplikacji utworzonych z użyciem Angular, które z kolei są bardziej wydajne od tych, przygotowanych z wykorzystaniem React*. Próba potwierdzenia tej hipotezy przeprowadzona zostanie poprzez badanie wydajności.

3. Obiekty badań

W niniejszym rozdziale przedstawiono najbardziej charakterystyczne cechy badanych technologii. Dotyczą one aspektów związanych z podejściem do tworzenia aplikacji, sposobem jej działania oraz sposobami implementacji typowych dla aplikacji internetowych mechanizmów.

3.1. Angular 17.1.0

Angular to platforma programistyczna oferująca szkielet oparty na podejściu komponentowym oraz cechująca się szerokim wyborem zintegrowanych i dedykowanych bibliotek. Dawniej (do stycznia 2022 roku) rozwijana była jego wersja wykorzystująca język JavaScript, ciesząca się wówczas dużym zainteresowaniem, o czym świadczyć mogą na przykład wyniki corocznej ankiety *Stack Overflow Survey*, badającej popularność narzędzi programistycznych, czy też liczba nadal utrzymywanych repozytoriów z projektami wykorzystującymi to narzędzie dostępnych na platformie *Github* [6]. Obecnie wersja ta nie jest już wspierana, gdyż firma Google skupiła swoje zasoby na usprawnianiu i dalszym rozwoju wersji opartej o język *TypeScript*.

Szkielet ten oparty jest na rzeczywistym DOM (ang. *Document Object Model*), zapewniającym bezpośrednie manipulowanie rzeczywistym drzewem obiektów dokumentu (DOM) w przeglądarce [7]. Komponenty składowe aplikacji zwykle przygotowywane są zgodnie z zalecanym sposobem, polegającym na podziale kodu programistycznego na 3 pliki, zawierające kolejno: klasę *TypeScript* z główną logiką komponentu, szablon HTML z dyrektywami szkieletu oraz arkusz stylów.

3.2. React 18.2.0

React to biblioteka JavaScript/TypeScript rozwijana przez firmę Meta (dawniej Facebook) i używana do

tworzenia interfejsu użytkownika aplikacji internetowych. Budowa aplikacji oparta jest na komponentach, definiowanych jako niezależne elementy, które mogą być wykorzystane wielokrotnie. Korzystają one ze składni *JSX* [8], sprawiającej, że tworzenie widoków jest intuicyjne i czytelne w związku z możliwością połączenia logiki i treści poprzez połączenie kodu *JavaScript* i *HTML* w jednym miejscu.

Fundamentalną koncepcją biblioteki React jest virtual DOM (ang. *Virtual Document Object Model*), reprezentujący aktualny stan widoku aplikacji [9]. Dzięki porównywaniu wirtualnego DOM z rzeczywistym, React decyduje jakie elementy należy zaktualizować. Podejście takie minimalizuje liczbę manipulacji rzeczywistym drzewem DOM, co objawia się szybszymi i bardziej płynnymi aktualizacjami interfejsu aplikacji.

3.3. Vue 3.3.4

Vue to progresywny szkielet programistyczny JavaScript/TypeScript autorstwa Evana You. Podejście progresywne oznacza, iż poszczególne części narzędzia Vue mogą być do istniejącej aplikacji wprowadzane stopniowo [10]. Nie ma tutaj potrzeby tworzenia złożonej aplikacji od samego początku jej procesu wytwórczego, a potrzebne funkcjonalności mogą być do niej dodawane wraz z rozwojem.

Podobnie jak React, Vue.js wykorzystuje wirtualny DOM do minimalizowania manipulacji rzeczywistym drzewem DOM i zapewnienia szybkich i wydajnych aktualizacji interfejsu użytkownika. Automatyczne wykrywanie zmian w danych pozwala na aktualizowanie tylko tych jego części, które zostały zmienione. Architektura narzędzia, podobnie jak w przypadku wcześniej omawianych, opiera się na komponentach. Interfejs użytkownika budowany jest z wielu małych, niezależnych komponentów, nierzadko wielokrotnie wykorzystywanych.

3.4. Zestawienie wybranych cech narzędzi

Wybrane cechy narzędzi Angular, React i Vue zestawiono w tabeli 1. Zawarte informacje pokazują podobieństwa i różnice pomiędzy badanymi technologiami. Najwięcej różnic dotyczy kwestii związanych z implementacją funkcjonalności wzbogacających podstawowe możliwości szablonowych aplikacji. W React i Vue często wymaga ona dołączenia do projektu aplikacji dodatkowych bibliotek, podczas gdy Angular wiele funkcjonalności posiada wbudowanych.

4. Metody badań i materiał badawczy

W celu zmierzenia wydajności badanych narzędzi przygotowano materiał badawczy (w postaci aplikacji testowej), za pomocą którego możliwe było dokonanie odpowiednich pomiarów. Zrealizowano to poprzez serię testów, które polegały na wykonaniu odpowiednich zadań (manipulujących tabelą danych) za pomocą trzech wersji aplikacji *"Benchmark"*.

Tabela 1: Wybrane cechy narzędzi Angular, React i Vue

	Angular	**React**	**Vue**
Typ narzędzia	Szkielet programistyczny	Biblioteka	Szkielet programistyczny
Język programowania	TypeScript	JavaScript, TypeScript	JavaScript, TypeScript
Twórcy	Google	Facebook	Evan You
Architektura	Oparta na komponentach	Oparta na komponentach	Oparta na komponentach
Składnia	Oparta na szablonach	JSX	Oparta na szablonach, JSX
Przechowywanie stanu aplikacji	Przechowywany lokalnie, Usługi	Przechowywany lokalnie, React Redux (zewnętrzna biblioteka)	Przechowywany lokalnie, Pinia (oficjalna biblioteka)
Trasowanie (ang. *Routing*)	Angular Router	Za pomocą stanów, React Router (zewnętrzna biblioteka)	Za pomocą stanów, Vue Router (oficjalna biblioteka)
Obsługa formularzy	Angular Forms	Za pomocą stanów, Formik (zewnętrzna biblioteka)	Za pomocą stanów, VeeValidate (zewnętrzna biblioteka)
Obsługą żądań HTTP	HTTP Client	Brak własnego klienta	Brak własnego klienta
Rodzaj DOM	DOM	VDOM	VDOM

4.1. Scenariusze i metody badawcze

Aplikacja *"Benchmark"* to tester wydajności posiadający elementy testowania wzorcowego (ang. *benchmark*). Przy użyciu każdej z badanych technologii utworzono trzy wersje aplikacji pozwalającej na wykonanie kolejnych testów o różnym stopniu złożoności. Głównym ich celem było zmierzenie czasu wykonania określonych operacji na tabeli danych, od momentu rozpoczęcia danego zadania, do momentu odświeżenia widoku aplikacji. Każdy z testów wykonany został dziesięć razy, a wyniki zostały uśrednione. Poszczególne zadania przedstawiono w postaci następujących scenariuszy:

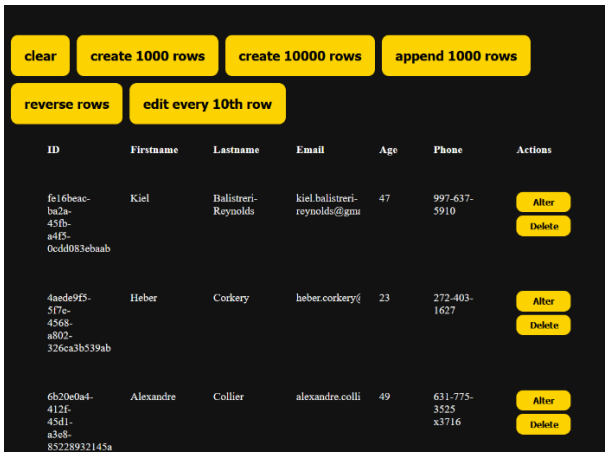
- Scenariusz I - utworzenie tabeli zawierającej 1000 wierszy,
- Scenariusz II - utworzenie tabeli zawierającej 10000 wierszy,

- Scenariusz III - rozszerzenie tabeli zawierającej 1000 wierszy o kolejne 1000 wierszy,
- Scenariusz IV - edycja pojedynczego wiersza tabeli zawierającej 1000 wierszy,
- Scenariusz V - edycja co 10 wiersza w tabeli zawierającej 1000 wierszy,
- Scenariusz VI - usunięcie pojedynczego wiersza z tabeli zawierającej 1000 wierszy,
- Scenariusz VII - usunięcie całej tabeli zawierającej 1000 wierszy,
- Scenariusz VIII - odwrócenie kolejności wierszy w tabeli zawierającej 1000 wierszy.

Do przeprowadzenia badań wykorzystano narzędzia deweloperskie dwóch przeglądarek internetowych [11, 12], pozwalające na nagrywanie profili wydajności witryn internetowych. Przeglądarki te to *Google Chrome* w wersji 123.0.6312.123 (wersja 64-bitowa) i *Mozilla Firefox* w wersji 124.0.2 (wersja 64-bitowa).

4.2. Interfejs aplikacji testowej

Ekran aplikacji *"Benchmark"* (Rysunek 1) zawiera tabelę danych oraz przyciski pozwalające na jej modyfikację, zgodnie z zadaniami przewidzianymi w ramach scenariuszy badawczych. Interfejs graficzny zawiera jedynie elementy wymagane do pełnego działania aplikacji. Utworzenie minimalistycznego rozwiązania pozwolić miało na ograniczenie niepotrzebnego wykorzystania zasobów podczas przeprowadzania testów wydajności.



Rysunek 1: Interfejs graficzny aplikacji *"Benchmark"*.

4.3. Model danych

Każdy z wierszy tabeli danych, zarządzanej w aplikacji reprezentowany jest jako jeden obiekt, dostarczający informacji o fikcyjnych osobach. Każda osoba opisywana jest przez takie cechy jak: imię, nazwisko, adres e-mail, wiek, numer telefonu oraz sztuczny identyfikator. W trzech wersjach aplikacji struktura wspomnianych obiektów zgodna jest ze wspólnym interfejsem, zamieszczonym na Rysunku 2.

```
export interface Person {
  id: string;
  firstname: string;
  lastname: string;
  email: string;
  age: number;
  phone: string;
}
```

Rysunek 2: Interfejs danych aplikacji *"Benchmark"*.

4.4. Platforma testowa

Do wykonania wszystkich zaplanowanych testów przygotowano jedną platformę sprzętową. Jej specyfikacja techniczna, zamieszczona w Tabeli 2, pozwala na przeprowadzenie badań bez zakłóceń i fizycznych ograniczeń, a użycie jednego sprzętu pomaga wyeliminować potencjalne różnice w wynikach, które mogłyby powstać w przypadku testowania utworzonych aplikacji na kilku platformach.

Tabela 2: Specyfikacja sprzętowa platformy testowej

Procesor	Intel Core i5-10600K
Płyta główna	MSI MPG Z490 GAMING PLUS
Pamięć RAM	16 GB
Dysk twardy	512 GB, ADATA SX8200PNP
System operacyjny	Windows 10 Education, 22H2

5. Wyniki badań

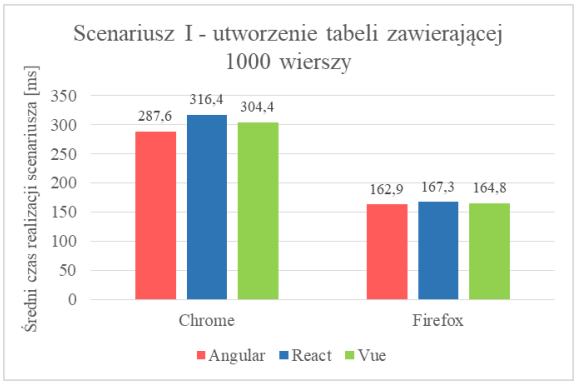
Wyniki badań trzech wariantów aplikacji *"Benchmark"*, wykonanych z użyciem narzędzi Angular, React i Vue, przeprowadzonych w dwóch przeglądarkach internetowych, poddane zostały analizie. Z zebranych w ramach serii pomiarów danych obliczono średnie czasy realizacji scenariuszy badawczych przez poszczególne warianty aplikacji testowej. Następnie zestawiono je w postaci graficznej.

Scenariusz I

Zadanie realizowane w ramach scenariusza polegało na wygenerowaniu tysiąca wierszy danych w pustej tabeli. Średnie czasy realizacji zadania zamieszczono na Rysunku 3. Najlepszy wynik (najkrótszy średni czas) w obu przeglądarkach internetowych odnotowano dla aplikacji przygotowanej w narzędziu Angular, a najgorszy - dla aplikacji utworzonej z użyciem React.

Dla aplikacji wykonanej w React, średni czas realizacji zadania w przeglądarce Google Chrome jest o około 10% dłuższy w stosunku do aplikacji utworzonej w Angular. Dla wariantu przygotowanego w Vue różnica ta wynosi około 5,8%. W przypadku przeglądarki Mozilla Firefox różnice pomiędzy osiągami aplikacji są bardziej

zbliżone. Względem Angular strata wariantu React wynosi tutaj około 2,7%, a Vue - około 1,2%.

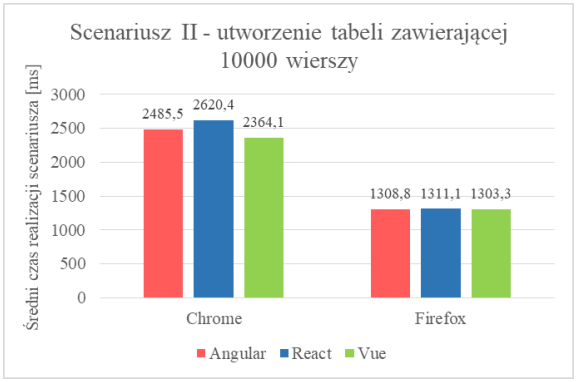


Rysunek 3: Średnie wyniki - Scenariusz I.

Scenariusz II

Zadanie polegało na wygenerowaniu dziesięciu tysięcy wierszy danych w pustej tabeli. Średnie czasy jego wykonania zamieszczono na Rysunku 4. W obu przeglądarkach internetowych najkrótsze średnie czasy realizacji scenariusza osiągnęła aplikacja utworzona za pomocą narzędzia Vue, a najdłuższe - wariant przygotowany z wykorzystaniem React.

W przypadku przeglądarki Firefox wyniki dla trzech aplikacji są niemal identyczne. Wynik aplikacji React, w stosunku do wyniku aplikacji Vue jest wyższy o około 0,6%. Średni czas realizacji zadania dla wariantu przygotowanego z Angular jest wyższy od wariantu wiodącego o około 0,4%. W przypadku Chrome, względem aplikacji najszybszej, dla Angular odnotowano wynik gorszy o około 5,1%, a dla React o około 10,8%.

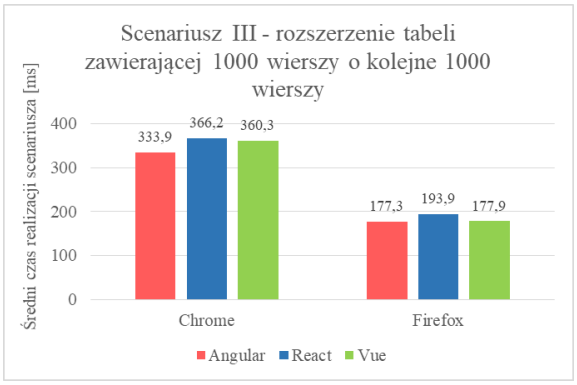


Rysunek 4: Średnie wyniki - Scenariusz II.

Scenariusz III

W niniejszym scenariuszu zadaniem było wygenerowanie tysiąca wierszy danych i rozszerzeniu o nie tabeli wypełnionej uprzednio kolejnym tysiącem wierszy. Średnie czasy realizacji zadania zamieszczono na Rysunku 5. Najniższy średni czas realizacji zadania w przeglądarkach Firefox i Chrome osiągnęła aplikacja przygotowana w Angular, a najwyższy - wariant przygotowany z użyciem React.

W przypadku przeglądarki Chrome, względem lidera (wariant Angular), aplikacja wykonana w Vue poniosła stratę rzędu około 7,9%, a wariant React - aż około 9,4%. Analogiczne testy z wykorzystaniem Firefox pokazały straty wynoszące około 0,3% oraz około 9,4%.

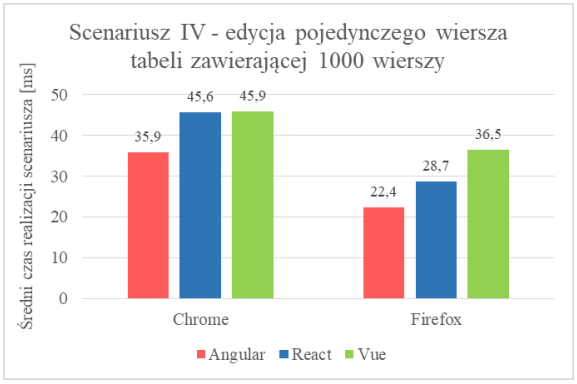


Rysunek 5: Średnie wyniki - Scenariusz III.

Scenariusz IV

Scenariusz ten polegał na edycji pojedynczego wiersza z tabeli zawierającej tysiąc uzupełnionych wierszy. Średnie czasy realizacji zadania zamieszczono na Rysunku 6. W obu przeglądarkach internetowych najkrótsze średnie czasy realizacji zadania uzyskała aplikacja przygotowana z użyciem Angular, a najdłuższe - ta, wykonana za pomocą Vue.

W stosunku do aplikacji wykonanej w Angular, aplikacja utworzona w React w przeglądarce Firefox osiągnęła średni czas realizacji zadania o około 28,1% dłuższy. Dla aplikacji przygotowanej w Vue, strata ta wyniosła aż około 63%. W przypadku przeglądarki Chrome, strata wariantów React i Vue względem lidera to odpowiednio: około 27% i około 27,9%.



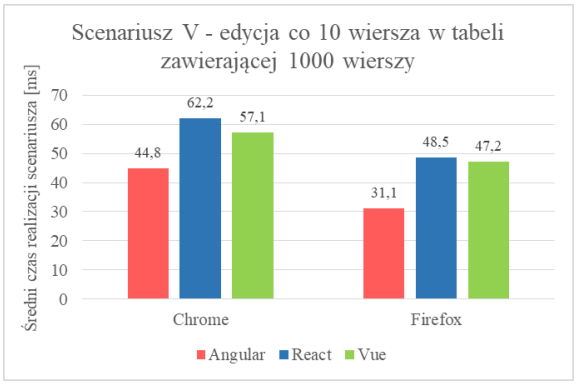
Rysunek 6: Średnie wyniki - Scenariusz IV.

Scenariusz V

Zadanie realizowane w ramach scenariusza V polegało na edycji (wygenerowaniu nowych danych i zastąpieniu poprzednich) każdego co dziesiątego wiersza w tabeli zawierającej tysiąc wierszy. Średnie czasy realizacji scenariusza umieszczono na Rysunku 7. W obu przeglądarkach internetowych najlepsze wyniki uzyskał wariant aplikacji przygotowany z użyciem narzędzia Angular, a najgorsze - przygotowany z wykorzystaniem React.

Dla aplikacji wykonanej w Vue, w stosunku do wykonanej w Angular, średni czas realizacji zadania jest dłuższy jest o około 51,8% w przeglądarce Firefox i o około 27,5% w przeglądarce Chrome. W przypadku

aplikacji utworzonej za pomocą React, straty te wynoszą około 56% w Firefox oraz około 38,8% w Chrome.

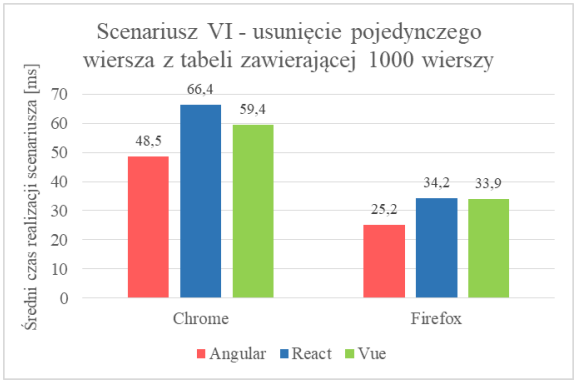


Rysunek 7: Średnie wyniki - Scenariusz V.

Scenariusz VI

Scenariusz VI związany był z realizacją zadania usunięcia pojedynczego wiersza z tabeli zawierającej tysiąc wierszy, uzupełnionych wygenerowanymi wcześniej danymi. Średnie czasy wykonania zadania zamieszczono na Rysunku 8. Najkrótszy średni czas wykonania scenariusza odnotowano dla aplikacji Angular, a najdłuższy dla aplikacji React.

Wariant aplikacji *"Benchmark"* wykonany w Vue, w stosunku do wariantu wiodącego w zestawieniu, uzyskał dłuższy średni czas wykonania zadania o około 22,5% w przypadku przeglądarki Chrome. Dla Mozilla Firefox różnica ta wyniosła aż około 34,5%. Dla wariantu utworzonego w React różnice te wyniosły około 36,9% (Chrome) i około 35,7% (Firefox).

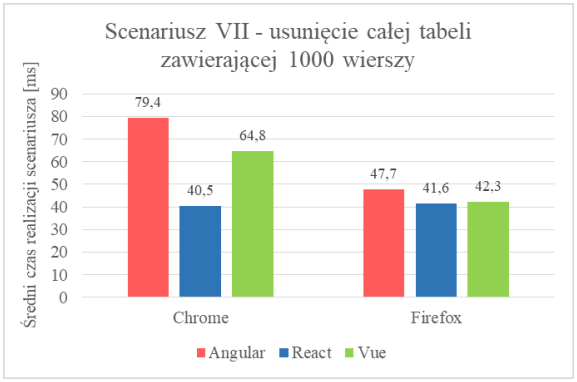


Rysunek 8: Średnie wyniki - Scenariusz VI.

Scenariusz VII

Zadanie realizowane w tym scenariuszu to usunięcie całej tabeli danych zawierającej tysiąc pełnych wierszy. Średnie czasy realizacji scenariusza zamieszczono na Rysunku 9. Najlepszy wynik (najkrótszy czas realizacji) odnotowano dla aplikacji przygotowanej z wykorzystaniem narzędzia React, a najgorszy (najdłuższy czas realizacji) dla aplikacji utworzonej z użyciem Angular.

W przypadku Firefox, względem aplikacji React, średnie wartości czasu uzyskane dla aplikacji Vue są wyższe o około 1,7%. W przypadku Angular, różnica wynosi już 14,7%. Dla przeglądarki Chrome różnice pomiędzy wynikami są bardziej znaczące. W stosunku do aplikacji wiodącej w zestawieniu, aplikacja Vue była wolniejsza o około 60%. Dla aplikacji Angular różnica ta wyniosła aż około 96%.

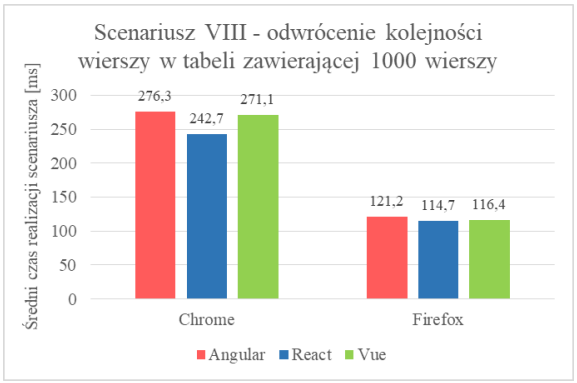


Rysunek 9: Średnie wyniki - Scenariusz VII.

Scenariusz VIII

Zadanie polegało na odwróceniu kolejności wierszy w tabeli zawierającej tysiąc wierszy z danymi. Średnie czasy jego wykonania zamieszczono na Rysunku 10. W obu przeglądarkach internetowych najkrótsze średnie czasy realizacji scenariusza osiągnęła aplikacja React, a najdłuższe - wariant przygotowany z wykorzystaniem Angular.

W przypadku przeglądarki Firefox wyniki dla trzech aplikacji są bardzo zbliżone. Wynik aplikacji Vue, w stosunku do wyniku aplikacji React jest wyższy o około 1,5%. Średni czas realizacji zadania dla Angular jest wyższy od wariantu wiodącego o około 5,7%. W przypadku przeglądarki Chrome, względem aplikacji najszybszej, dla Vue odnotowano wynik gorszy o około 11,7%, a dla Angular o około 13,8%



Rysunek 10: Średnie wyniki - Scenariusz VIII.

Podsumowanie wyników badania

Z zebranych danych wynika, iż w przypadku większości scenariuszy największą wydajnością wykazała się aplikacja przygotowana z użyciem Angular. Największa liczba scenariuszy z najdłuższymi średnimi czasami realizacji przypadła natomiast aplikacji utworzonej za pomocą React. Podsumowanie wyników dla poszczególnych scenariuszy przedstawiono w Tabeli 3.

Tabela 3: Podsumowanie testów wydajności realizacji zadań według scenariuszy (1- najszybciej, 3-najwolniej)

Nr scenariusza	Angular	React	Vue
I	1	3	2
II	2	3	1
III	1	3	2
IV	1	2	3
V	1	3	2
VI	1	3	2
VII	3	1	2
VIII	3	1	2

Aplikacja utworzona przy użyciu narzędzia Angular w najkrótszym czasie wykonała zadania związane z uzupełnieniem pustej lub istniejącej tabeli o tysiąc nowych wierszy. W przypadku scenariusza uzupełniania pustej tabeli o dziesięć tysięcy wierszy, została natomiast wyprzedzona przez aplikację wykonaną z wykorzystaniem Vue, która zadanie to realizowała znacznie szybciej.

Kolejnymi zadaniami, które w średnio najkrótszym czasie wykonywał Angular, były edycja i usunięcie pojedynczego wiersza tabeli oraz edycja jej każdego co dziesiątego wiersza. Wyniki kolejnego zadania - usunięcia zawartości tabeli, przedstawiają znaczną przewagę React nad pozostałymi narzędziami. W przypadku ostatniego zadania - odwrócenia kolejności wierszy w tabeli zawierającej tysiąc wierszy, najszybsze narzędzie jest identyczne - najkrótsze czasy osiąga React.

6. Wnioski

Przeprowadzone testy wydajności wykazały, iż w analizowanych scenariuszach najlepsze wyniki osiągnęła aplikacja przygotowana z użyciem narzędzia Angular. Zrealizowała ona pięć spośród ośmiu scenariuszy w średnio najkrótszym czasie. Następne miejsce w hierarchii wydajności (Tabela 3) zajmuje aplikacja utworzona w Vue. Najgorsze wyniki odnotowano dla aplikacji React - najdłuższe średnie czasy realizacji zadań w pięciu scenariuszach testowych. Wnioski z badań implikują, iż hipoteza badawcza nie została potwierdzona. Pomimo oczekiwań, bazujących na wynikach badań z ostatnich lat, najwydajniejszym narzędziem w niniejszym badaniu okazał się bowiem Angular. W większości przeanalizowanych przypadków testowych przewyższył tym samym zarówno wydajność Vue, jak i React.

Wyniki przeprowadzonego badania nie pozwalają jednoznacznie stwierdzić, iż otrzymana hierarchia wydajności (Angular, Vue, React) powinna być przestrzegana dla każdego rozwiązania. Pozwalają natomiast zauważyć zalety i wady badanych narzędzi, wynikające z ich charakterystyki. Mowa tutaj na przykład o najwyższej wydajności narzędzia Vue przy pracy z dużą ilością danych (scenariusz II), czy też najszybszej realizacji zadań związanych z modyfikacją niewielkich elementów aplikacji przez Angular (scenariusze IV, V, VI). Analiza takich aspektów w procesie projektowania aplikacji, pozwolić może na trafniejszy dobór wykorzystywanej technologii do zastosowania. Do tworzenia rozbudowanych aplikacji najlepszym narzędziem będzie Angular, z dużym zasobem wbudowanych bibliotek i narzędzi lub Vue. Do aplikacji o niskim stopniu złożoności i ograniczonych funkcjonalnościach, lżejszym narzędziem okazać się może natomiast React.

Przeprowadzone badania dodatkowo wykazały bardzo dużą różnicę w czasie realizacji identycznych zadań za pośrednictwem przeglądarek internetowych Google Chrome i Mozilla Firefox, bazujących na różnych silnikach wyświetlania (renderowania). Niektóre zadania wykonywane za pomocą przeglądarki Firefox realizowane były niemal dwukrotnie szybciej niż z użyciem Chrome (scenariusze I, II, III, VI), a przypadku scenariusza VIII - ponad dwukrotnie szybciej. Fakt ten sprawia, iż dla końcowego użytkownika wybór przeglądarki internetowej może mieć niekiedy większy wpływ na wydajność i komfort pracy w aplikacji niż wybór narzędzia programistycznego przez jej twórców. Zjawisko to mogłoby być podstawą do przyszłych badań, rozważających wpływ wyboru przeglądarki internetowej i silnika wyświetlania na wydajność aplikacji.

Badanie dowiodło także celowość regularnego testowania najpopularniejszych na rynku narzędzi wspomagających tworzenie warstwy klienta aplikacji internetowych. Dostępne na rynku technologie wraz z upływem czasu ulegają zmianom, które niekiedy mogą mieć znaczący wpływ na wydajność tworzonych z ich użyciem aplikacji. W momencie przystąpienia do prac nad niniejszym artykułem, poddane badaniom narzędzia dostępne były w następujących wersjach: Angular - 17.1.0 (wydana 17.01.2024 r.), React - 18.2.0 (wydana 17.06.2022 r.), Vue - 3.3.4 (wydana 18.05.2023 r.). Po zakończeniu badań, wiosną 2024 r. uzyskały jednak aktualizacje do nowszych wersji. Wydano bowiem Angular 18.0.1 (29.05.2024 r.), React 18.3.1 (26.04.2024 r.) oraz Vue 3.4.27 (06.05.2024 r.). Niniejszy artykuł pozostawia więc pole do dalszych badań, rozważających zarówno nowe i przyszłe wersje tytułowych rozwiązań, jak i inne dostępne na rynku narzędzia, które w najbliższym czasie również mogą zyskać na znaczeniu wśród twórców aplikacji.

Literatura

[1] Stack Overflow Insights - Stack Overflow Annual Developer Survey, https://insights.stackoverflow.com/survey, [04.03.2024].

[2] R. Ollila, N. Mäkitalo, T. Mikkonen, Modern Web Frameworks: A Comparison of Rendering Performance, Journal of Web Engineering 21 (2022) 789-814, https://doi.org/10.13052/jwe1540-9589.21311.

[3] J. Cincović, M. Punt, Comparison: Angular vs. React vs. Vue. Which framework is the best choice?, ICIST 2020 Proceedings (2020) 250-255.

[4] K. Bielak, B. Borek, M. Plechawska-Wójcik, Web application performance analysis using Angular, React and Vue.js frameworks, Journal of Computer Sciences Institute 23 (2022) 77-83, https://doi.org/10.35784/jcsi.2827.

[5] Y. Xing, J. Huang, Y. Lai, Research and Analysis of the Front-end Frameworks and Libraries in E-Business Development, Proceedings of the 2019 11th International Conference on Computer and Automation Engineering (ICCAE 2019), Association for Computing Machinery (2019) 68-72, https://doi.org/10.1145/3313991.3314021.

[6] Github Website, https://github.com, [12.03.2024].

[7] Angular Documentation, https://angular.io/docs, [12.03.2024].

[8] React Documentation, Writing Markup with JSX, https://react.dev/learn/writing-markup-with-jsx, [12.03.2024].

[9] React API Reference, https://react.dev/reference/react, [12.03.2024].

[10] Vue.js Guide, The Progressive Framework, https://vuejs.org/guide/introduction.html#the-progressive-framework, [12.03.2024].

[11] Chrome DevTools Documentation, Chrome DevTools, https://developer.chrome.com/docs/devtools?hl=pl, [12.03.2024].

[12] Firefox DevTools Documentation, Firefox DevTools Contributor Docs, https://firefox-source-docs.mozilla.org/devtools/, [12.03.2024].