JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 29 (2023) 341–345
Received: 23 June 2023
Accepted: 4 July 2023

Performance analysis of React v. 18.1.0 and Angular v. 11.0.2 development frameworks

Analiza wydajności szkieletów programistycznych React v. 18.1.0 i Angular v. 11.0.2

Albert Poniedziałek*, Beata Pańczyk

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

Abstract
This paper presents the results of the comparative analysis of the two JavaScript frameworks. The research was carried out using test applications with the same functionalities, implemented in both environments. The number of bytes occupied by both projects, the number of lines of code, the average RAM consumption and the time efficiency were used as criteria for comparison. Automatic load tests and technical analysis of both environments were performed. The results showed that React is able to better handle a large amount of data (around 100000 records), while for small data sets oscillating around 1000 records, no significant differences in both application performance were demonstrated. For a medium number of records (10000), React proved to be more efficient.

Keywords: React; Angular; performance analysis

Streszczenie
W artykule porównano dwa szkielety programistyczne we wskazanych wersjach. Badania przeprowadzono za pomocą aplikacji testowych o takich samych funkcjonalnościach, zaimplementowanych w obu środowiskach. Za kryterium porównawcze przyjęto liczbę bajtów zajmowanych przez oba projekty, liczbę wierszy z kodem, koniecznych do implementacji, średnie zużycie pamięci RAM oraz wydajność czasową. Wykonano automatyczne testy obciążeniowe oraz analizę techniczną obu środowisk. W wyniku badań stwierdzono, że React lepiej radzi sobie z obsługą dużej liczby danych (około 100000 rekordów), natomiast w przypadku małych zbiorów danych oscylujących w okolicach 1000 rekordów nie wykazano istotnych różnic w działaniu obu aplikacji. W przypadku umiarkowanej liczby rekordów (10000) bardziej wydajny okazał się React.

Słowa kluczowe: React; Angular; analiza wydajnościowa

*Corresponding author
Email address: **albert.poniedzialek@pollub.edu.pl** (A. Poniedziałek)

1. Wstęp

Obecnym standardem w tworzeniu aplikacji internetowych są aplikacje typu SPA (ang. Single Page Application). Dostępnych jest wiele różnych szkieletów programistycznych, które przyspieszają proces generowania kodu JavaScript. Najbardziej popularne z nich to Angular, React.js, Vue.js, Ember.js czy Backbone.js [1]. Są to rozwiązania, które oferują bardzo zaawansowane funkcje i gotowe do użycia komponenty. Problem stanowi fakt, że każde środowisko oferuje nieco inny zestaw funkcjonalności i nierzadko początkujący programista ma duży problem z wybraniem odpowiedniego narzędzia do pracy.

2. Cel i zakres badań

W artykule porównano dwa najbardziej popularne szkielety programistyczne (ReactJS v.18.1.0 oraz AngularJS v11.02) w celu wskazania, który z nich jest wydajniejszy. W momencie podjęcia badań (rok 2022) wskazane wersje były najnowsze.

W pracy zwrócono uwagę na podobieństwa oraz różnice pomiędzy obydwoma narzędziami, zbadano efektywność czasową aplikacji przy wykonywaniu operacji na różnej ilości danych.

Postawiono dwie tezy:

T1. React jest bardziej wydajny czasowo przy obsłudze dużej ilości danych w porównaniu do środowiska Angular.

T2. Angular jest bardziej wydajny czasowo przy pracy z mniejszą ilością danych.

3. Analiza literatury

Na temat tworzenia aplikacji internetowych istnieje wiele publikacji, od artykułów poświęconych analizie składni poszczególnych szkieletów programistycznych, po analizę cech danych środowisk i porównania wydajności w pracy z danymi.

Stanowią one dobre źródła wiedzy do nauki i są także punktem wyjścia do badań, prowadzonych dla kolejnych wersji szkieletów programistycznych, które wprowadzają coraz to nowe udogodnienia i jednocześnie zwiększają efektywność działania aplikacji internetowych.

W artykule R. Nowackiego z roku 2016 [2] myślą przewodnią jest analiza porównawcza trzech szkieletów programistycznych języka JavaScript (React, Angular i EmberJS). Podano tam cztery kryteria porównawcze:

trudność nauki biblioteki, dodatkowe narzędzia charakterystyczne dla danej platformy, wydajność na urządzeniach stacjonarnych oraz wydajność na urządzeniach mobilnych. Konkluzją badań było wskazanie środowiska React jako najlepszego narzędzia, między innymi z powodu dużej społeczności zbudowanej wokół tego środowiska.

W pracy W. Stępniaka i Z. Nowaka z roku 2017 [3] przeanalizowano skuteczność metod przyspieszających proces ładowania aplikacji typu SPA, w tym mechanizmów oferowanych przez protokół HTTP/2. Zbadano wydajność gotowych komponentów dostarczanych ze szkieletem programistycznym Angular.

Za pomocą niewielkich aplikacji implementujących tę samą funkcjonalność w środowisku Angular w wersji 2.0, porównano również wydajność różnych konstrukcji oferowanych przez Angular. W badaniu wykorzystano serwer Apache, przeglądarkę Chrome dla systemu Linux i Android jak również narzędzia Browser-perf i h2load. Wnioskami było wskazanie środowiska jako dobre oraz wydajne podczas tworzenia aplikacji typu SPA.

W artykule A. Kumara i R. K. Singha z roku 2016 [4] analizie poddano także platformy React i Angular. Cechy obu szkieletów przedstawiono w postaci tabeli. Oceniano m.in. takie subiektywne cechy jak krzywą uczenia. We wnioskach stwierdzono, że oba środowiska są porównywalne a podstawowym kryterium wyboru powinny być potrzeby samego programisty i jego poziom doświadczenia.

Artykuł M. Kaluża i K. Troskot z roku 2018 [5] wyjaśnia różnice w modelu aplikacji MPA (ang. Multi Page Application) i SPA. Zdefiniowano tu zagadnienia wpływające na rozwój aplikacji MPA i SPA. Autorzy poddali analizie również inne szkielety programistyczne, takie jak Angular, Vue.js i React-js, poprzez oceny jakościowe każdego z zagadnień wpływających na rozwój aplikacji MPA i SPA. W podsumowaniu wskazano VueJS, jako najlepszy wybór.

Artykuł K. Kowalczyka z roku 2022 [6] zawiera porównanie wydajności bibliotek Angular oraz React. Badanie zostało zrealizowane poprzez implementację aplikacji w obu środowiskach i zbadaniu ich wydajności. W artykule zestawiono jedynie zalety i wady omawianych szkieletów.

Reasumując, literatura dotycząca porównania biblioteki React i Angular nie daje jednoznacznej odpowiedzi na pytanie, jak również często opisuje starsze wersje obu środowisk, przez co nie możliwe jest wskazanie, które ze środowisk jest lepszym wyborem dla programisty. Studiując dostępną literaturę czytelnik jest w stanie wywnioskować jedynie kilka faktów:

React posiada dużą społeczność zebraną wokół siebie jak również dobrze radzi sobie z dużymi zbiorami danych, zaś Angular jest wydajny przy małych zbiorach danych oraz posiada wiele funkcjonalności wspomagających tworzenie aplikacji. Dodatkowo czytelnik może odnieść słuszne wrażenie, że wybór pomiędzy środowiskami React czy Angular wcale nie jest trywialny i wymagane są dalsze badania w tym kontekście.

W niniejszym artykule podjęto kolejną próbę porównania obu szkieletów w wersjach (rok 2022), React 18.1.0 i Angular 11.0.1.

4. Charakterystyka badanych technologii

W tym rozdziale zestawiono najważniejsze cechy wyróżniające oba badane narzędzia.

4.1. React

React jest biblioteką utworzoną dla języka JavaScript, jego głównym przeznaczeniem jest tworzenie interfejsu graficznego w aplikacjach internetowych. Jest prosty w instalacji i użyciu, pozwala na korzystanie z języka JavaScript, jak również z TypeScript.

Cechą najbardziej wyróżniającą React jest wirtualny element DOM (ang. Document Object Model). Umożliwia on realnemu elementowi DOM wykonanie jak najmniejszej liczby operacji podczas ponownego renderowania treści strony, aktualizując jedynie te fragmenty, które uległy zmianie. Proces ten działa na zasadzie porównania wirtualnego elementu DOM do wyświetlonego DOM. Dzięki takiemu podejściu polepsza się wydajność działania strony.

Kolejną cechą szczególną React jest język JSX utworzony specjalnie na potrzeby tej biblioteki. Umożliwia on stosowanie kodu HTML i XML oraz samych komponentów React wewnątrz funkcji JavaScript. JSX umożliwia również przekazywania tak przygotowanych elementów do kolejnych funkcji bez potrzeby tworzenia ciągów znakowych, jak w przypadku innych bibliotek.

Ponadto React bazuje na komponentach posiadających własne stany, dzięki czemu nie wpływają nawzajem na siebie i nie wymuszają aktualizacji innego komponentu, gdy same są aktualizowane (co zapewnia wirtualny DOM). Dodatkowo rozdzielenie stanów bazujące na *React Hooks* pozwala na śledzenie zmian w aplikacji i wykonanie zdefiniowanych czynności w momencie aktualizacji stanu komponentu.

React wspiera również renderowanie po stronie serwera, dając możliwość wysyłania gotowych już komponentów z serwera do aplikacji klienckiej, tym samym zwiększając płynność działania i poprawiając ogólne wrażenie podczas korzystania z witryny.

Dużym atutem biblioteki React jest możliwość konwersji projektu w React Native, który jest wieloplatformowy w przeciwieństwie do podstawowej wersji biblioteki i jest wykorzystywany również do tworzenia aplikacji dla systemów Android, Android TV, iOS, macOS, tvOS i Windows.

4.2. Angular

Angular jest szkieletem programistycznym o otwartym kodzie, jest platformą m.in. do tworzenia aplikacji typu SPA. Został napisany w języku TypeScript, właścicielem licencji i głównym deweloperem jest firma Google.

Angular, podobnie jak React, wspiera renderowanie po stronie serwera, posiada moduły i jest zorientowany na komponenty. Ma dobre wsparcie dla przeglądarek mobilnych w przeciwieństwie do React, który musi być specjalnie do tego celu przestawiony do React Native.

Ponadto Angular posiada narzędzia konsolowe wspomagające wytwarzanie aplikacji, na przykład łatwe debugowanie. Posiada on również wsparcie dla wersjonowania semantycznego (ang. Semantic Versioning).

Domyślnym językiem dla Angular jest TypeScript, który jest bardziej skomplikowanym językiem od JavaScript. Angular daje możliwość zmienienia natywnego języka na JavaScript, może to jednak generować wiele problemów dla niedoświadczonego programisty.

Podstawowe cechy obu rozwiązań zostały przedstawione w tabeli 1.

Tabela 1: Podstawowe cechy React i Angular

Atrybut	React	Angular
Autor	Jordan Walke (pracownik Facebook)	Google
Data powstania	marzec 2013	maj 2016
Aktualnie rozwijane przez	Facebook	Google
Typ licencji	licencja MIT	licencja MIT
Możliwość tworzenia aplikacji mobilnych	React Native	Ionic Framework
Struktura projektu	Elastyczna, oparta na komponentach	Stała i złożona struktura platformy, oparta na komponentach
Domyślny używany język	JavaScript z możliwością użycia składni JSX	TypeScript
struktura DOM	wirtualny DOM	rzeczywisty DOM
Krzywa uczenia	relatywnie niska	Może być stroma dla początkujących
Dynamiczne wiązanie interfejsu użytkownika	Bezpośrednie połączenie stanów z interfejsem użytkownika	Wiązanie interfejsu użytkownika na poziomie zwykłego obiektu lub właściwości

5. Metoda badań

Na potrzeby badań opracowane zostały dwie proste aplikacje typu SPA. Aplikacje, oferują te same funkcjonalności i umożliwiają generowanie i wysłanie danych do bazy, pobrania i wyświetlenie danych, sortowanie i usuwanie elementów struktury DOM przechowujących dane. Aplikacje wykonują operacje na relacyjnej bazie danych MySQL, przechowującej rekordy z informacjami o użytkownikach.

Oba projekty zostały utworzone na tym samym urządzeniu i poddane jedynie drobnej modyfikacji plików package.json, tak aby uzyskać odpowiednie wersje środowiska. Po instalacji niezbędnych modułów, dodano dodatkowe elementy, które umożliwiają wykonywanie testów obciążeniowych.

Wygląd aplikacji testowych jest identyczny, dzięki czemu całkowicie został wyeliminowany wpływ kodu CSS na działanie aplikacji testowych.

Obsługiwane przez aplikacje dane mają postać tablicy z ośmioma polami zawierającymi dane wygenerowane za pomocą biblioteki fakerjs [7], są nimi m.in. imię, nazwisko, płeć, kraj zamieszkania. Z aplikacji klienckich dane przesyłane są do aplikacji serwerowej, zaimplementowanej na platformie NodeJS i Express.

Porównanie wydajności aplikacji polegało na wyznaczaniu czasu wykonywania operacji na danych (pobieranie, wstawianie, sortowania i usuwanie różnej liczby rekordów z bazy danych).

Programy uruchomione zostały na lokalnym sprzęcie i były wystawione na lokalnych portach.

Analiza porównawcza została przeprowadzona z zastosowaniem testów manualnych i automatycznych. Na potrzeby badań zdefiniowano 5 różnych scenariuszy, w 3 wariantach (liczba rekordów: 1000, 10000 i 100000). Każdy scenariusz został wykonany po 20 razy. W skrajnych przypadkach jeden test trwał nawet 30 minut. Parametry sprzętu, na którym zostały wykonane badania zostały przedstawione w tabeli 2.

Tabela 2: Parametry sprzętu do testów

Parametr	Wartość
Procesor	AMD Ryzen 7 5800H
Ram	16 GB
Dysk	1000 GB SSD
Karta graficzna	NVIDIA GeForce RTX 3070
System operacyjny	Windows 10 Home

Dla testów automatycznych zrealizowano 5 scenariuszy badawczych:
S1. Generowanie danych testowych,
S2. Wysłanie i zapis w bazie wygenerowanych danych.
S3. Pobranie danych z bazy i ich wyświetlenie,
S4. Sortowanie wyświetlonych danych,
S5. Usunięcie elementów przechowujących dane.

Po każdym teście przeglądarka była resetowana co wyeliminowało wzajemną interferencję między kolejnymi testami.

6. Analiza wyników badań

Wyniki badań zestawiono w kolejnych podrozdziałach.

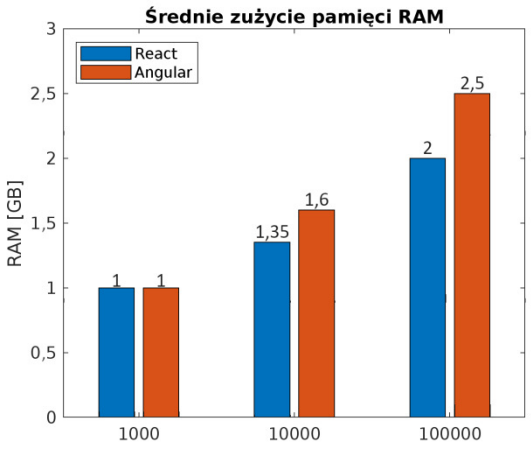
6.1. Analiza testów manualnych

Dane z testów dotyczących wykorzystania sprzętu przedstawione zostały w postaci wykresów słupkowych na Rysunkach 1 i 2.

W przypadku pobierania oraz wysyłania 1000 rekordów (Rysunek 1) miało miejsce niewielkie zużycie pamięci, około 1GB w trakcje całej akcji. Operacja wysyłania danych trwała relatywnie krótko, średnio około 15.5s dla React oraz 17s dla Angular. Operacja pobierania danych z bazy trwała średnio około 6s dla obu środowisk.

Przy umiarkowanym obciążeniu dla 10000 rekordów (Rys.1) można zauważyć większą różnicę w wykorzystaniu pamięci RAM przez oba środowiska. W przypadku biblioteki React jest to wartość powyżej 1GB a momentami nawet 1.5GB. Dla Angular, zużycie pamięci RAM ciągle rosło aż do około 1.7GB bez żadnego zwolnienia pamięci.

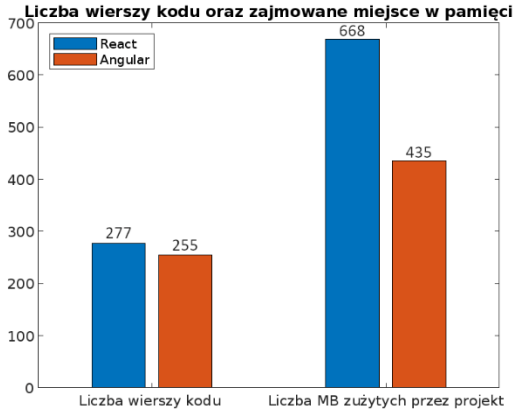
Najciekawsze rezultaty zaobserwowane dla wariantu 100000 rekordów (Rysunek 1). Podczas tego testu zużycie RAM osiągało nawet 5GB dla React pod koniec wykonywania testu. W przypadku Angular, zużycie RAM dochodziło nawet do 6GB.



Rysunek 1: Średnie zużycie pamięci RAM dla różnej liczby rekordów.

Podsumowując system zwalniania pamięci działał cały czas i zmniejszał na bieżąco zużycie jednak odbywało się to zbyt wolno, wskutek czego zużycie pamięci potrafiło nadal narastać.

Z tego wynika, że odbieranie oraz wysyłanie rekordów w sposób rekurencyjny nie jest najlepszym rozwiązaniem, gdyż środowiska oraz przeglądarki nie mają wbudowanych dostatecznie dobrych systemów zarządzania pamięcią RAM. Przy większej liczbie rekordów zużycie nadal będzie rosło.



Rysunek 2: Liczba linii kodu programistycznego i miejsce w pamięci zajmowane przez projekty.

Dla obu projektów sprawdzono ile miejsca zajmują w pamięci (Rysunek 2). Biblioteki oraz pliki utworzone przez środowisko React zajmują 668 MB pamięci, podczas gdy Angular potrzebuje tylko 435MB, co stanowi około 65% tego, co potrzebuje React.

W przypadku liczby linii kodu (Rysunek 2) w Angular należało napisać o 22 linie kodu mniej (8% mniej) niż w React.

6.2. Testy automatyczne

Dane z testów automatycznych przedstawione zostały w postaci wykresów pudełkowych.

Wykresy tego typu przedstawiają dane na temat mediany oznaczonej jako czerwona linia w obrębie pudełka. Prezentują też dane na temat 25-tego oraz 75-tego centyla (oznaczone jako dolne i górne części pudełka). „Wąsy" pokazują najmniejsze i największe wartości, które nie zostały oznaczone jako odbiegające od reszty. Wartości znacznie odbiegające od pozostałych są oznaczone symbolem czerwonego plusa.

Rysunki 3-5 prezentują wyniki testów w zależności od liczby rekordów.

6.2.1. Wariant obsługi 1000 rekordów

Dla 1000 rekordów (Rysunek 3) oba środowiska uzyskały zbliżone wyniki. React radzi sobie lepiej dla scenariuszy wysyłania (12% szybciej), pobierania (11% szybciej) i usuwania (13% szybciej). Angular szybciej wykonuje operacje sortowania (5% szybciej) i generowania danych (4% szybciej).

Z testów wynika zatem, teza o lepszej obsłudze małej liczby danych nie może zostać ani potwierdzona, ani zaprzeczona, ponieważ wyniki pomiarów nie wskazują jednoznacznie na żadne ze środowisk.

6.2.2. Wariant obsługi 10000 rekordów

Dla 10000 rekordów (Rysunek 4) bardziej jednoznacznie widać różnice w szybkości wykonywania operacji na danych. React szybciej wykonał cztery operacje: sortowanie (15% szybciej), usuwanie (25% szybciej), wysyłanie (12% szybciej) i pobieranie (31% szybciej).

Jedynie w przypadku generowania 10000 rekordów Angular wypadł lepiej o około 3%.

6.2.3. Wariant obsługi 100000 rekordów

W przypadku obciążenia największą badaną liczbą rekordów (Rysunek 5) React także wykonał zdecydowanie szybciej operacje sortowania (55% szybciej), usuwania (15% szybciej) i pobierania (5% szybciej). W przypadku wysyłania średnie czasy różnią się o około 3% na korzyść środowiska React, jednak z analizy wykresów (Rysunek 5) wynika, że dla React wyniki są bardziej „rozrzucone", zaś w przypadku Angular wyniki są bardziej zwarte. Tylko dla przypadku generowania danych nie można jednoznacznie wskazać żadnego ze środowisk, ponieważ na wykresie pudełkowym (Rysunek 5) są one niemal takie same a średnie czasy są prawie identyczne.

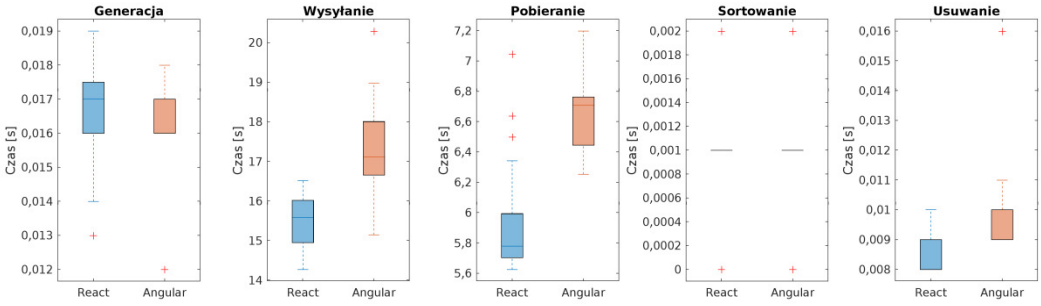
W wyniku otrzymanych rezultatów można potwierdzić tezę, że React szybciej wykonuje operacje na dużej liczbie danych.

7. Wnioski

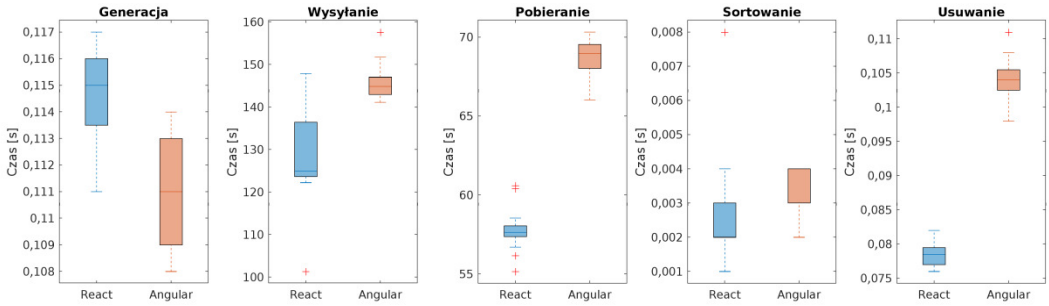
Z analizy wyników przedstawionej w rozdziale 6 można potwierdzić tezę T1 dotyczącą lepszej wydajności środowiska React podczas obsługi dużej liczby danych (100000 rekordów). Wykonane badania wskazały większą szybkość operacji wykonywanych na danych przez aplikację React na poziomie od 15% do nawet 55% w scenariuszu z sortowaniem 100000 rekordów.

Teza T2 dotycząca lepszej wydajności środowiska Angular podczas obsługi małej liczby danych (1000 rekordów) nie może zostać ani potwierdzona, ani obalona, ponieważ uzyskane rezultaty nie wskazują jednoznacznie na żadne ze środowisk. W tym wariancie

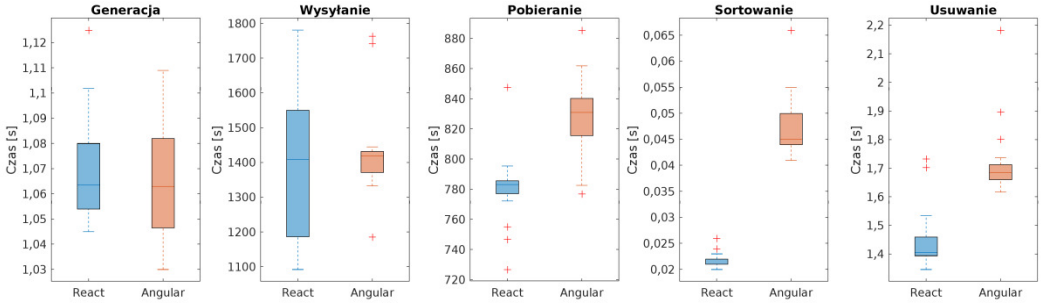
3 scenariusze wykazały lepsze działanie biblioteki React zaś 2 pozostałe wskazały, że szybszy jest Angular.



Rysunek 3: Wyniki testów dla obu środowisk dla wariantu obsługi 1000 rekordów.



Rysunek 4: Wyniki testów dla obu środowisk dla wariantu obsługi 10000 rekordów.



Rysunek 5: Wyniki testów dla obu środowisk dla wariantu obsługi 100000 rekordów.

Podsumowując oba szkielety programistyczne poradziły sobie dobrze z zadanym obciążeniem i często różnice czasowe były bardzo małe.

Literatura

[1] Statystyki popularności JavaScript 2022, https://2022.stateofjs.com/en-US/, [20.05.2023].

[2] R. Nowacki, M. Plechawska-Wójcik, Analiza porównawcza narzędzi do budowania aplikacji Single Page Application – AngularJS, ReactJS, Ember.js, Journal of Computer Sciences Institute 2 (2016) 98-103, https://doi.org/10.35784/jcsi.122.

[3] W. Stępniak, Z. Nowak, Performance analysis of SPA web systems, Advances in Intelligent Systems and Computing 521 (2017) 235-247, https://doi.org/10.1007/978-3-319-46583-8_19.

[4] A. Kumar, R. K. Singh, Comparative Analysis of AngularJS and ReactJS, International Journal of Latest Trends in Engineering and Technology 7 (2016) 225-227, https://doi.org/10.21172/1.74.030.

[5] M. Kaluža, K. Troskot, B. Vukelić, Comparison of front-end frameworks for web applications development, Zbornik Veleučilišta u Rijeci 6 (2018) 261-282, https://doi.org/10.31784/zvr.6.1.19.

[6] R. Ollila, N. Mäkitalo, T. Mikkonen, Modern Web Frameworks: A Comparison of Rendering Performance, Journal of Web Engineering 21(03) (2022) 789–814, https://doi.org/10.13052/jwe1540-9589.21311.

[7] Strona główna biblioteki fakerjs, https://fakerjs.dev/, [08.06.2023].