JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 20 (2021) 183–188
Received: 08 June 2021
Accepted: 20 June 2021

Comparison of web application state management tools

Porównanie narzędzi do zarządzania stanem aplikacji internetowych

Kacper Szymanek*, Beata Pańczyk

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

Abstract
Modern web applications require flow of large amounts of data. To maintain order in code, a state manager was invented. With manager all data can be retrieved from and goes to one place. In this paper, four libraries for state management (NgRx, Ngxs, Redux, Vuex) were analyzed. Five criteria were used for the study: code metrics, solution structure, availability of ready-made implementations, community support, and performance testing. Results showed that there is not the best tool in every criterion, but when comparing the results obtained, the most universal solution is Vuex.

Keywords: management state; FLUX; store

Streszczenie
Nowoczesne aplikacje internetowe wymagają przepływu dużej ilości danych. Do utrzymania porządku w kodzie wykorzystuje się zarządcę stanu. Dzięki menadżerowi wszystkie dane mogą być pobrane z oraz trafiają w jedno miejsce. W niniejszym artykule analizie poddano cztery biblioteki do zarządzania stanem – NgRx, Ngxs, Redux, Vuex. Do badania wykorzystano pięć kryteriów: metryki kodu, strukturę rozwiązania, dostępność gotowych implementacji, wsparcie społeczności oraz przetestowano wydajnościowo. Wyniki wykazały, że nie ma narzędzia najlepszego w każdym kryterium, jednak zestawiając ze sobą otrzymane rezultaty najbardziej uniwersalnym rozwiązaniem jest Vuex.

Słowa kluczowe: zarządzanie stanem; FLUX; store

*Corresponding author
Email address: **kacper.szymanek1@pollub.edu.pl** (K. Szymanek)

1. Wstęp

Nowoczesne systemy informatyczne wymagają przepływu coraz większej ilości danych, nad którymi ciężko zapanować tak, aby nie tworzyć nadmiernych ilości kodu oraz aby aplikacja była łatwa w rozbudowie. Coraz bardziej na popularności zyskują wzorce projektowe, które pomagają zapobiec takiemu zjawisku. Wzorcem, który porusza problem przetrzymywania i przepływu danych jest FLUX.

Na rynku narzędzi programistycznych jest dużo gotowych implementacji tego wzorca. W tej pracy porównane zostały cztery najpopularniejsze biblioteki tj. NgRx, Ngxs, Redux, Vuex. Każde narzędzie przystosowane jest do innego szkieletu programistycznego.

2. Przegląd literatury

W artykule [1] autorzy przedstawili model i zasady rozwoju "inteligentnego" systemu mieszkaniowego z wykorzystaniem struktury modułowej czujników i wzoru architektonicznego strumienia danych Redux. Wykazali, że wzorzec ten może być wykorzystywany nie tylko jak zakładali autorzy do aplikacji internetowych, ale również w tworzeniu inteligentnych domów.

Artykuł [2] próbuje odpowiedzieć na pytanie jak poradzić sobie z dużymi systemami informatycznymi. Biblioteka Redux została tam wykorzystana do zarządzania stanem aplikacji internetowej. Autorzy udowodnili, że w obecnych czasach zastosowanie menadżera to standard w przypadku rozbudowanych aplikacji internetowych.

Autorzy artykułu "Nowoczesne technologie tworzenia graficznego interfejsu użytkownika w aplikacjach internetowych" [3] badali trzy szkielety programistyczne tj. Angular, React, Vue. Skupili się na szybkości ładowania strony oraz obciążeniu pamięci. Wyniki wskazują, że najgorszym szkieletem jest Angular. Nie wykorzystano tam jednak zarządcy stanu, który mógłby znacznie zmienić wyniki badania.

Artykuły [4,5] skupiają się wokół implementacji wzorca menadżera stanu dla dwóch bibliotek tj. MobX oraz Redux. Oba artykuły pobieżnie sprawdzają możliwość wykorzystania rozwiązań do aplikacji. Lepsza technologia została wybrana na podstawie subiektywnych odczuciach autorów i jest to Redux.

Podsumowując przeprowadzoną analizę nie istnieje zbyt dużo badań prowadzonych na narzędziach zarządzania stanem. MobX oraz Redux są głównymi badanymi technologiami, gdzie ostatecznie lepsza okazuje się druga biblioteka. Wybór ten opierany jest głównie ze względu na lepszą dokumentację, możliwość większego dostosowania implementacji wzorca. Wyniki wskazują również, iż MobX naruszą zasadę "pojedynczego źródła prawdy" [6] i pozwala na deklarowanie wielu źródeł z danymi.

3. Cel badania

Celem badania w niniejszym artykule jest porównanie bibliotek do zarządzania stanem w aplikacjach internetowych oraz wybranie najlepszej możliwej biblioteki. Analizę przeprowadzono dla czterech menadżerów stanu w trzech różnych szkieletach programistycznych

tj. NgRx oraz Ngxs dla Angular, Redux dla React oraz Vuex dla Vue. Redux to biblioteka, która jest przystosowaną do wielu narzędzi, natomiast pozostałe implementacje są dedykowanymi rozwiązaniami do przypisanych im szkieletów.

4. Wzorzec projektowy Flux

Wzorzec projektowy to schemat rozwiązania powtarzających się problemów programistycznych. Wzorzec, który dotyczy zarządzaniem danych w aplikacjach internetowych to Flux. Komponenty widoku zostają uzupełnione o jednokierunkowy przepływ danych oraz jedno źródło z którego dane mogą być pobierane oraz zapisywane. Można wyróżnić trzy główne elementy wzorca:

- magazyn danych – miejsce w którym przechowywany jest stan;
- dyspozytor (ang. dispatcher) – pełni rolę centralnego węzła w aplikacji;
- akcje – elementy pomocnicze przy komunikacji dyspozytor – magazyn danych.

Wzorzec ten powstał aby rozwiązać główny problem wzorca MVC (ang. Model, View, Controller), gdzie przekazywanie danych między widokami jest bardzo problematyczne. Główne elementy MVC to model, widok oraz kontroler. Kontroler odpowiada za manipulację modelem, który aktualizuje widok [6-8]. Oba wzorce Flux i MVC umożliwiają tworzenie aplikacji internetowych, ale Flux to nowa architektura aplikacji, która koncentruje się na **jednokierunkowym przepływie danych.** Przepływ ten rozpoczyna akcja, na którą reaguje dyspozytor, który zmienia wartości w magazynie z danymi. Pojedynczy kontener pozwala na jedną zmianę bez konieczności uaktualniania tych samych danych rozproszonych po całym systemie [9]. Flux jako architektura może być implementowany dowolnie, co powoduje różnorodne nazewnictwo w dostępnych bibliotekach z menadżerem stanu np. dla narzędzia Redux oraz Ngrx za zmiany w magazynie odpowiada reduktor (ang. reducer)[10], natomiast w Vue mutacje (ang. mutation). Oba mechanizmy są sterowane przy pomocy wbudowanego w narzędzie dyspozytora.

5. Aplikacje testowe

W celu zbadania postawionego problemu zaprogramowano cztery aplikacje w odpowiednich szkieletach programistycznych. Każda aplikacja wykorzystuje odpowiednio przystosowaną do narzędzia bibliotekę do zarządzania stanem. Ma to na celu zbadanie przepływu dużych i małych danych (obiekt JSON zawierający zagnieżdżenia, oraz listę 100 obiektów) przez zarządcę stanu oraz jego wpływ na wydajność systemu. Aplikacje realizują funkcjonalności zarządzania zajezdnią autobusową i posiadają tylko wycinek całego rozwiązania. Skupiono się na realizacji przepływu danych pomiędzy komponentami oraz serwerem.

Pierwszą aplikacje zaimplementowano w szkielecie Angular w wersji 10, w której zastosowano rozwiązanie sterowania stanem przygotowane przez bibliotekę NgRx w wersji 9.2.0.

Druga aplikacja również została zaimplementowana w szkielecie Angular w wersji 10. Zastosowano tam narzędzie do zarządzania stanem o nazwie Ngxs w wersji 3.6.2.

Trzecia aplikacja wykorzystuje szkielet programistyczny React w wersji 16.13.1 wraz z biblioteką Redux w wersji 7.2.1. Asynchroniczne zapytania do serwera realizowane są za pomocą biblioteki Redux-Thunk w wersji 2.3.0. Pozwoli ona na porównanie pełnego przepływu danych jakie narzucają pozostałe narzędzia.

Ostatnią aplikację zaimplementowano w Vue w obecnie najnowszej wersji 3 z wykorzystaniem kompozycji API (ang. Composition API), które pozwala na szybszą implementacje komponentów. Menadżer stanu jaki został wykorzystany to dedykowane narzędzie o nazwie Vuex w wersji 4.

Po za aplikacjami klienckimi zaimplementowano prostą aplikację serwerową, z którą komunikują się wszystkie aplikacje klienckie. W tym celu wykorzystano technologię ASP .Net Core w wersji 3.1. Serwer ma za zadanie zwracać obiekty w formacie JSON oraz listy z obiektami. Dane nie są zwracane z bazy danych lecz są wpisane w kod aplikacji. Spowodowane jest to tym, aby czasy odpowiedzi serwera były możliwie najkrótsze.

6. Metody przeprowadzenia badań

Badanie ma na celu znaleźć odpowiedź na pytanie: ***Które narzędzie pozwala na najprostsze, najszybsze i najwydajniejsze zaimplementowanie rozwiązania informatycznego przy użyciu mechanizmu zarządzania stanem?***

W tym celu wybrano pięć **kryteriów** do przeprowadzenia analizy porównawczej.

1. Pierwszym kryterium jest metryka kodu. W celu zbadania, zestawiono magazyn danych, dyspozytor, akcje oraz reduktor czyli moduł mechanizmu zarządzania stanem. Porównaniu poddano liczbę linii jaka jest potrzebna do zaimplementowania poszczególnych elementów wzorca. Następnie pliki zostały zestawione pod względem ich wielkości. Wykorzystano w tym celu jednostkę bajt.
2. Drugim kryterium jest architektura rozwiązania. W tym fragmencie porównano możliwości rozmieszczenia poszczególnych fragmentów kodu, możliwość ich rozdzielenia na odrębne pliki oraz dołożenia dodatkowej abstrakcji w formie fasady.
3. Kolejnym etapem jest badanie dostępności gotowych implementacji. Ma to na celu zbadanie dostępności istniejących już w bibliotekach rozwiązań oraz metod i możliwość pełnego dostosowania ich do potrzeb własnych. Do zbadania zostało zaimplementowane rozwiązanie wykorzystujące to samo żądanie do API przy maksymalnym wykorzystaniu gotowych metod i funkcji bibliotek, bez konieczności tworzenia własnych.
4. Następnym etapem badania jest przeanalizowanie wsparcia społeczności. Porównaniu zostały poddane słowa kluczowe – tagi dla badanych technologii oraz wybrana została najbardziej wyszukiwane narzędzie

w serwisie StackOverflow. Jest to obecnie największy portal zrzeszający ludzi ze świata IT na świecie. [11].

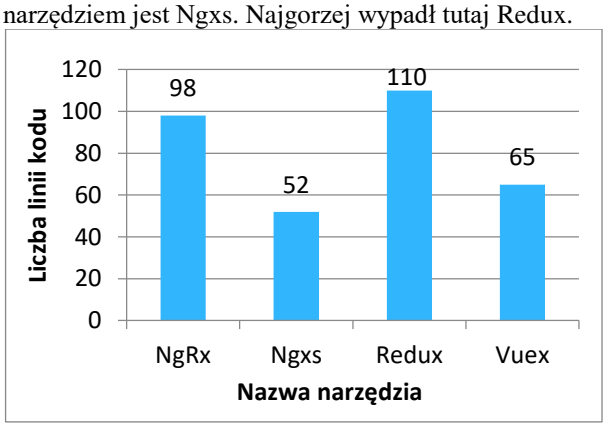
5. Ostatnim kryterium są testy wydajnościowe. Wykorzystano tutaj moduły z zaimplementowanych aplikacji i wykonano trzy scenariusze. Pierwszy to pobranie obiektu JSON z API, zapisanie go oraz pobranie z menadżera stanu. Drugi to pobranie z API stu elementowej listy oraz zapisanie i pobranie do komponentu z magazynu. Ostatni to dodatkowo przekazanie danych z informacją o liście 100 elementowej z magazynu do komponentu dziecka. Od wszystkich czasów potrzebnych na przetworzenie danych odjęto czas żądania API. Wykonano 15 prób dla każdego ze scenariuszy.

7. Wyniki badań

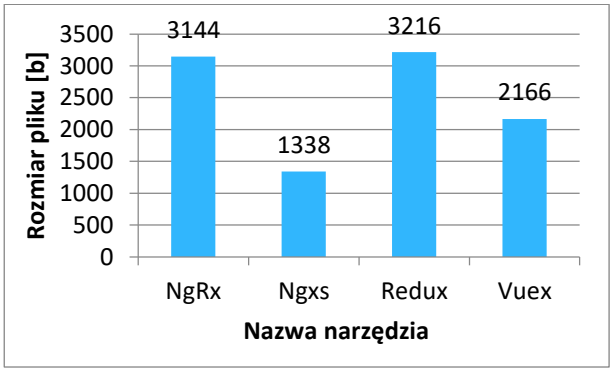
7.1. Metryka kodu

Aby zaprezentować wyniki porównania liczby linii kodu oraz wielkości plików jakie zostały zaimplementowane w badanych narzędziach, wykorzystano moduł menadżera stanu który pobiera obiekt JSON z aplikacji serwerowej. Następnie zapisuje go w magazynie, z którego zostaje pobrany do komponentu.

Na Rysunku 1 zaprezentowano sumaryczne zestawienie linii kodu wszystkich narzędzi, natomiast rysunek 2 prezentuje sumaryczne zestawienie rozmiarów wszystkich plików badanych narzędzi. Najlepszym narzędziem jest Ngxs. Najgorzej wypadł tutaj Redux.

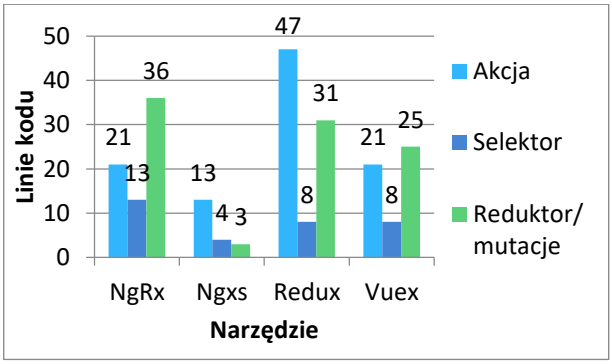


Rysunek 1: Sumaryczna liczba linii kodu wszystkich elementów zarządcy stanu.



Rysunek 2: Rozmiar wszystkich plików zarządcy stanu.

Porównując ze sobą elementy które odpowiadają za tą samą czynność w badanych narzędziach, odczytać można, że najmniej kodu jest w plikach Ngxs. Największy fragment menadżera to akcje zaimplementowane w Redux – 47 linii, natomiast najmniejszy to selektor z narzędzia Ngxs – 3 linie. Obsługa magazynu w narzędziu Vuex nosi nazwę mutacji i jest tożsama z reduktorami w pozostałych bibliotekach. Pełne zestawienie danych przedstawia Rysunek 3.



Rysunek 3: Sumaryczne zestawienie liczby linii kodu dla takich samych elementów wszystkich badanych narzędzi.

7.2. Architektura rozwiązania

W tabeli 1 zestawiono wszystkie narzędzia i możliwości podziału na pliki. Ngxs jako jedyne z badanych narzędzi nie pozwala na dowolne manipulowanie kodem. Redux jako jedyne narzędzie można zaimplementować w dowolnym szkielecie programistycznym.

Tabela 1. Porównanie dowolności architektury w badanych narzędziach

Opis / Narzędzie	Ngrx	Ngxs	Redux	Vuex
Możliwość podziału na pliki	Tak	Tak	Tak	Tak
Możliwość wydzielenia fragmentów kodu odpowiadających elementom wzorca flux	Tak	Nie	Tak	Tak
Możliwość podziału na moduły	Tak	Tak	Tak	Tak
Możliwość dodania dodatkowej warstwy dostępu – fasady	Tak	Tak	Tak	Tak
Możliwość implementacji w dowolnym szkielecie programistycznym bez większych ingerencji w bibliotekę	Nie	Nie	Tak	Nie

7.3. Dostępność gotowych implementacji

Tabela 2 prezentuje porównanie narzędzi pod względem zgodności z wzorcem projektowym Flux.

Wszystkie założenia wzorca tj. pojedyncze źródło prawdy oraz jednokierunkowy przepływ danych zostały spełnione w badanych narzędziach. Posiadają one także niezbędne elementy tj. akcje, dyspozytor, magazyn danych. Tylko Ngxs wymaga klasy jako akcji, w pozostałych badanych narzędziach akcja jest implementowana jako metoda.

Tabela 2. Porównanie prawidłowego dostosowania do wzorca Flux

Opis / Narzędzie	Ngrx	Ngxs	Redux	Vuex
Prawidłowe przygotowanie narzędzia do wzorca Flux	Tak	Tak	Tak	Tak
Akcje	Tak	Tak	Tak	Tak
Dyspozytor	Tak	Tak	Tak	Tak
Magazyn danych	Tak	Tak	Tak	Tak
Jednokierunkowy przepływ danych	Tak	Tak	Tak	Tak
Sposób implementacji Akcji	Funkcja	Klasa	Funkcja	Funkcja

7.4. Wsparcie społeczności

Najbardziej znanym oraz popularnym narzędziem zrzeszającym programistów oraz osoby związane ze światem Informatyki jest obecnie StackOverFlow. Jest to publiczna platforma do zadawania pytań programistycznych oraz do dzielenia się wiedzą [9]. Platforma pozwala na analizę wsparcia społeczności narzędzi. Udostępnia statystykę związaną ze słowami kluczowymi czyli tagami. Tagi są związane z technologią, której dotyczy pytanie. Do sprawdzenia największego wsparcia społeczności należy zliczyć liczbę tagów występujących przy pytaniach. Oprócz tego strona pozwala sprawdzić w jakich miesiącach oraz latach dany tag jest najczęściej wyszukiwaną frazą.

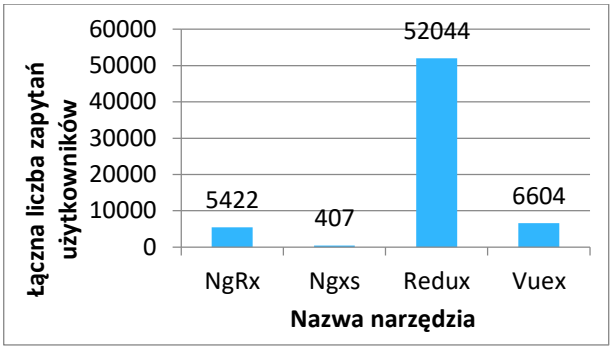
Najpopularniejszy sposób na zbudowanie słowa kluczowego to nazwa taga taka sama jak narzędzia. Często można także spotkać dodanie specjalistycznej nazwy problemu do nazwy biblioteki w ten sposób tworząc słowo kluczowe.

Sumując liczbę dostępnych tagów po jakich można wyszukiwać zapytania, narzędzie Redux posiada największą ich liczbę spośród wszystkich badanych. Drugi w kolejności jest NgRx, następnie Vuex. Najmniej słów kluczowych z badanych narzędzi posiada Ngxs – 2. Wyniki łącznej liczby słów kluczowych zestawiono w Tabeli 3.

Tabela 3. Liczba słów kluczowych na portalu StackOverFlow

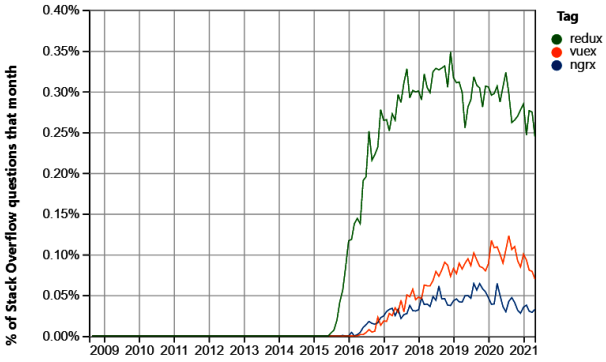
Nazwa narzędzia	Liczba słów kluczowych
NgRx	14
Ngxs	2
Redux	35
Vuex	7

Wykres zestawiający łączną liczbę zapytań użytkowników o badane biblioteki zaprezentowano na Rysunku 4. Wyniki wskazują, że najczęściej zadawane są pytania o technologię Redux. Vuex oraz NgRx plasują się na podobnym poziomie zapytań. Najmniej wyszukiwanym narzędziem jest Ngxs.



Rysunek 4: Liczba zapytań o narzędzie według tagów w serwisie StackOverFlow.

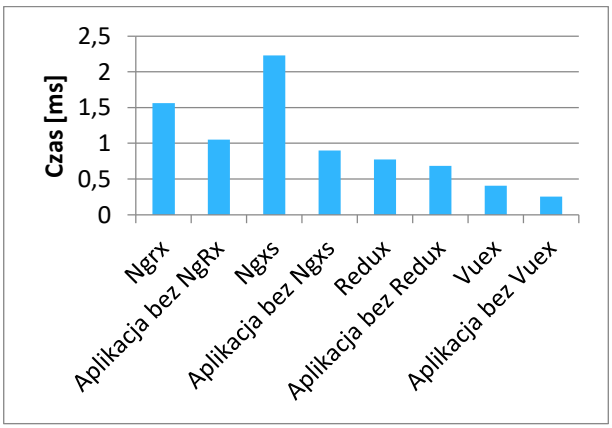
Rysunek 5 przedstawia procent wszystkich zapytań zadanych w całym portalu StackOverFlow, w poszczególnych latach. Redux widocznie przoduje nad pozostałymi narzędziami. Istotną informacją jaką można odczytać z tego rysunku jest to, że wszystkie narzędzia zaczęły być wyszukiwane w podobnym czasie tj. na przełomie 2015 oraz 2016 roku. Ngxs jest tak mało popularny, że nie występuje na wykresie udostępnionym przez portal StackOverFlow. Ngrx jest drugim najmniej popularnym narzędziem.



Rysunek 5: Procent pytań StackOverflow w danym roku [12]

7.5. Testy wydajnościowe

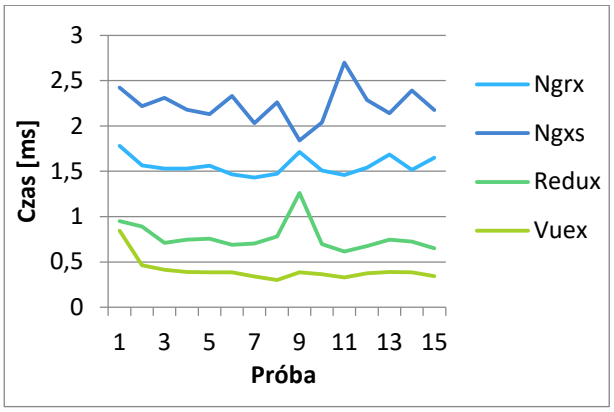
Średni czas potrzebny do przetworzenia obiektu JSON przez narzędzia przedstawiono na Rysunku 6. Najlepszą biblioteką jest Vuex. Najgorszy czas przetworzenia potrzebny jest implementując rozwiązanie w Ngxs.



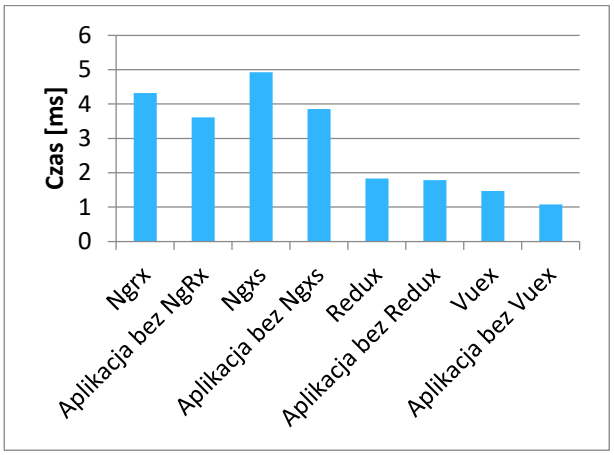
Rysunek 6: Wykres średniego czasu potrzebnego na przetworzenie obiektu JSON przez aplikacje.

Rysunek 7 przedstawia wykres z wynikami pomiarów z 15 prób przetworzenia danych. Vuex posiada najbardziej zbliżone wyniki w każdym przypadku. Najgorzej wypada Ngxs. Poza największym czasem potrzebnym na przetworzenie danych w magazynie, posiada także największe rozbieżności w wynikach co może wskazywać na małą stabilność tego rozwiązania.

Najlepszy czas przetworzenia 100 elementowej listy ma narzędzie Vuex (Rysunek 8). Jest to duża przewaga nad najgorszym wynikiem dla Ngxs Drugim najlepszym narzędziem jest Redux.



Rysunek 7: Wykres 15 prób z czasem potrzebnym na przetworzenie danych.

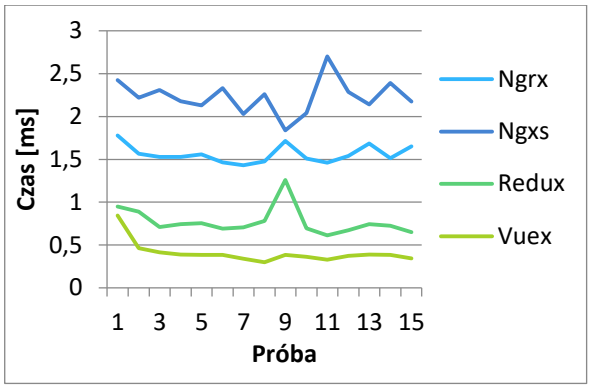


Rysunek 8: Wykres średniego czasu potrzebnego na przetworzenie 100 elementowej listy przez aplikacje.

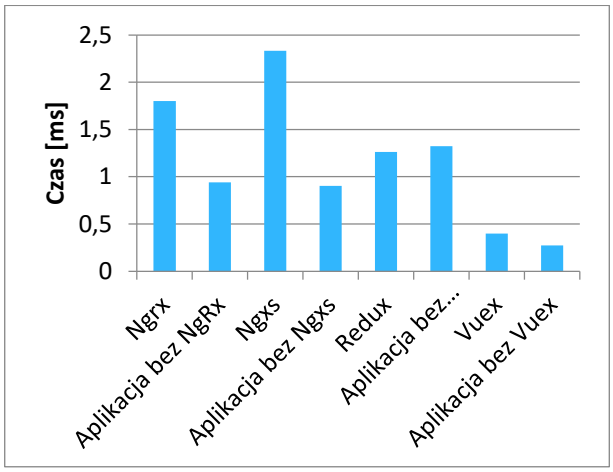
Vuex tak jak w przypadku pojedynczego obiektu, tak i w przypadku listy zachowuje się bardzo liniowo i stabilnie bez żadnych odchyłów. Najgorzej prezentuje się Ngxs (Rysunek 9).

Rysunek 10 prezentuje wykres średniego czasu potrzebnego do przetworzenia oraz przekazania 100 elementowej listu do komponentu dziecka. Wyniki wskazują, iż Ngxs również tutaj wypada najgorzej. Najlepszym narzędziem tak jak w poprzednich scenariuszach jest Vuex.

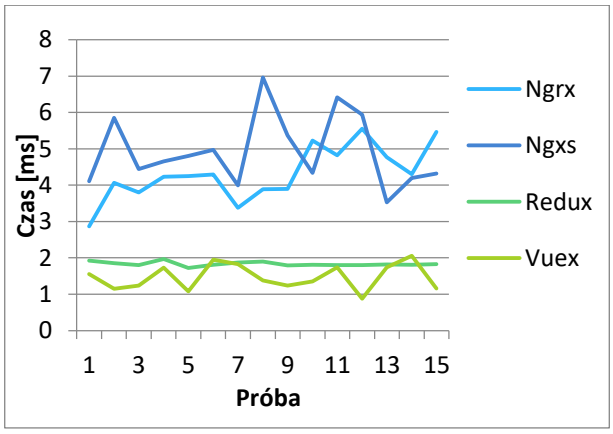
Zestawiając zebrane dane na wykresie liniowym (Rysunek 11) zauważyć można, że tym razem to Redux posiada najbardziej zbliżone do siebie wyniki



Rysunek 9: Wykres 15 prób z czasem potrzebnym na przetworzenie 100 elementowej listy obiektów.

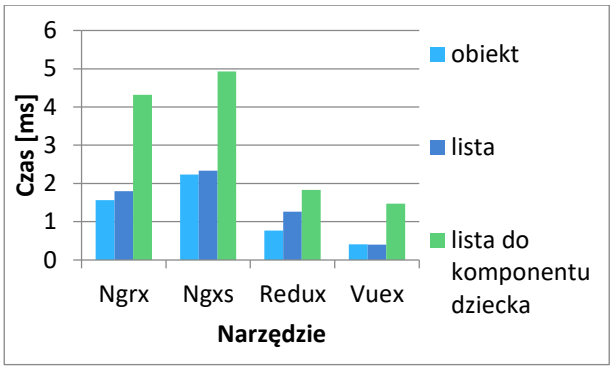


Rysunek 10: Wykres średniego czasu potrzebnego na przetworzenie 100 elementowej listy przez aplikacje i przekazanie jej do komponentu dziecka.



Rysunek 11: Wykres 15 prób z czasem potrzebnym na przetworzenie 100 elementowej listy obiektów i przekazanie jej do komponentu dziecka.

Podsumowując wszystkie scenariusze badania wydajności (Rysunek 12), najwięcej czasu zajmowało przetworzenie i przekazanie 100 elementowej listy do komponentu. Biblioteki przystosowane do szkieletu programistycznego Angular tj. Ngrx oraz Ngxs poradziły sobie najgorzej w tym zadaniu. Najlepiej we wszystkich scenariuszach poradził sobie Vuex.



Rysunek 12: Średnie czasy badanych bibliotek w 3 różnych scenariuszach.

8. Wnioski

Celem artykułu było porównanie bibliotek do zarządzania stanem w aplikacjach internetowych.

Do wskazania najlepszego narzędzia, wykorzystano punktację, która była przyznawana w skali od 1 do 4, gdzie 4 oznacza najlepsze narzędzie a 1 najgorsze. Dodatkowo przyznano 1 punkt gdy narzędzie spełniało zadane kryterium i 0 gdy nie.

Najlepszym narzędziem według tego badania jest Vuex. Na drugim miejscu są biblioteki przystosowane do szkieletu programistycznego Angular tj. NgRx oraz Ngxs. Na ostatnim miejscu znajduje się Redux. Wyniki porównania przedstawiono w Tabeli 4.

Tabela 4: Wyniki punktowe zebrane z kryteriów badanych narzędzi.

Opis / Narzędzie		Ngrx	Ngxs	Redux	Vuex
Metryka kodu	Najmniejszy rozmiar plików	2	4	1	3
	Najmniej linii kodu do implementacji	2	4	1	3
Architektura rozwiązania	Możliwość podziału na pliki według wzorca flux	1	0	1	1
	Możliwość implementacji w dowolnym szkielecie programistycznym	0	0	1	0
Dostępność gotowych implementacji	Dostępność największej ilości gotowych metod	4	3	1	2
	Zgodność ze wzorcem FLUX	1	1	1	1
Wsparcie społeczności	Największe zainteresowanie społecznością	2	1	4	3
Testy wydajnościowe	Najszybsze narzędzie	2	1	3	4
Suma punktów		**14**	**14**	**13**	**17**

Literatura

[1] A. Kazarian, V. Teslyuk, I. Tsmots, J. Greguš, Development of a «smart» home system based on the modular structure and architectural data flow pattern Redux, Procedia Computer Science. 155 (2019), 35-42.

[2] S. Mukhiya1, K. Hung. An Architectural Style for Single Page Scalable Modern Web Application, International Journal of Recent Research Aspects, 5(4) (2018), 6-13.

[3] M. Kaproń, B. Pańczyk. Nowoczesne technologie tworzenia graficznego interfejsu użytkownika w aplikacjach internetowych, Journal of Computer Sciences Institute 15 (2020), 139-152.

[4] D. Holmstedt, Analyzing and implementing a third-party state machine library for FriendlyReader and TeCST, Linköping University, Department of Computer and Information Science, Bachelor's thesis (2019).

[5] W. Wenhao, React Native vs Flutter, Cross-platforms mobile application frameworks, Metropolia University of Applied Sciences, Bachelor of Engineering Information technology Thesis (2018).

[6] Porównanie flux oraz mvc https://madasamy.medium.com/flux-vs-mvc-design-pattern-de134dfaa12b [12.04.2021].

[7] Opis czym jest flux https://jerzywickowski.pl/flux/co-to-jest-flux/, [25.03.2021].

[8] Wprowadzenie do wzorca architektonicznego Flux, https://www.freecodecamp.org/news/an-introduction-to-the-flux-architectural-pattern-674ea74775c9/, [25.03.2021].

[9] A. Boduch, Flux architecture, Packt Publishing Ltd (2016).

[10] D. Bugl, Learning Redux, Packt Publishing Ltd (2017).

[11] StackOverFlow – portal dla programistów, https://stackoverflow.com, [12.03.2021].

[12] Trendy zapytań w latach 2009-2021 w serwisie StackOverFlow, https://insights.stackoverflow.com/trends?tags=ngrx%2Cvuex%2Credux, [11.04.2021].