**POLITECHNIKA BIAŁOSTOCKA**

**WYDZIAŁ** **INFORMATYKI**

**KATEDRA INŻYNIERII OPROGRAMOWANIA**

PRACA MAGISTERSKA

TEMAT:

Analiza wydajności wybranych frameworków JavaScriptowych.

WYKONAWCA:..............................................

*Imię i nazwisko*

PODPIS: .................................

PROMOTOR: ...................................................

*Imię i nazwisko*

PODPIS: .................................

**BIAŁYSTOK** ............ **ROK**

karta dyplomowa

zakres pracy:

1. Zapoznanie się z zagadnieniem oraz dostępnymi frameworkami javascriptowymi.

2. Projekt i implementacja aplikacji testowych w kilku wybranych frameworkach.

3. Testy i porównanie przygotowanych implementacji pod kątem wybranych kryteriów.

4. Opis różnic na poziomie mechanizmów, procesu tworzenia oraz działania aplikacji.

Thesis topic:

………………………………………………………………………………………………….…………………………………………………………………………………………………

SUMMARY

………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

Załącznik nr 4 do „Zasad postępowania przy przygotowaniu i obronie pracy dyplomowej w PB”

Białystok, dnia………………………

imię i nazwisko studenta

................................................

nr albumu

...............................................

kierunek i forma studiów

...............................................

promotor pracy dyplomowej

**OŚWIADCZENIE**

Przedkładając w roku akademickim 2017/2018 Promotorowi ………………………………..………... pracę dyplomową pt.: ...............…………………………………………………….……………………….. ………………….………………………..………………………………………..…………………………….. dalej zwaną pracą dyplomową,

**oświadczam, że:**

1. praca dyplomowa stanowi wynik samodzielnej pracy twórczej,
2. wykorzystując w pracy dyplomowej materiały źródłowe, w tym w szczególności: monografie, artykuły naukowe, zestawienia zawierające wyniki badań (opublikowane, jak i nieopublikowane), materiały ze stron internetowych, w przypisach wskazywałem/am ich autora, tytuł, miejsce i rok publikacji oraz stronę, z której pochodzą powoływane fragmenty, ponadto w pracy dyplomowej zamieściłem/am bibliografię,
3. praca dyplomowa nie zawiera żadnych danych, informacji i materiałów, których publikacja nie jest prawnie dozwolona,
4. praca dyplomowa dotychczas nie stanowiła podstawy nadania tytułu zawodowego, stopnia naukowego, tytułu naukowego oraz uzyskania innych kwalifikacji,
5. treść pracy dyplomowej przekazanej do dziekanatu Wydziału Informatyki jest jednakowa w wersji drukowanej oraz w formie elektronicznej,
6. jestem świadomy/a, że naruszenie praw autorskich podlega odpowiedzialności na podstawie przepisów ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t. j.: Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.), jednocześnie na podstawie przepisów ustawy z dnia 27 lipca 2005 r. Prawo   
   o szkolnictwie wyższym (t. j.: Dz. U. z 2012 r. poz. 572 z późn. zm.) stanowi przesłankę wszczęcia postępowania dyscyplinarnego oraz stwierdzenia nieważności postępowania w sprawie nadania tytułu zawodowego,
7. udzielam Politechnice Białostockiej nieodpłatnie licencji na korzystanie z pracy dyplomowej w celu realizacji przeprowadzenia procedury antyplagiatowej przyjętej w Uczelni oraz na przekazanie pracy do Ogólnopolskiego Repozytorium Prac Dyplomowych, jak również udostępnianie i przechowywanie jej w Bibliotece Politechniki Białostockiej przez okres 50 lat od obrony pracy dyplomowej.

….……………………………………….

czytelny podpis studenta

Na podstawie art. 24 ust. 1 ustawy z dnia 29 sierpnia 1997 roku o ochronie danych osobowych (j.t. z 2014 r., poz. 1182 z późn. zm.) informuję, że administratorem danych jest Politechnika Białostocka, ul. Wiejska 45A, 15-351 Białystok. Dane będą przetwarzane w celach realizacji procedury antyplagiatowej przyjętej w Politechnice Białostockiej i nie będą udostępniane odbiorcom danych w rozumieniu art. 7 pkt 6 ustawy o ochronie danych osobowych. Osobie, której dane dotyczą, przysługuje prawo dostępu do treści swoich danych oraz ich poprawiania. Podanie danych jest obowiązkowe (art. 167b ustawy z dnia 27 lipca 2005 roku Prawo o szkolnictwie wyższym j.t. Dz.U. z 2012 r., poz. 572 z późn. zm.).

Spis treści

[1. Wstęp 6](#_Toc489699925)

[1.1. Cel pracy 6](#_Toc489699926)

[1.2. określenie tezy- brak u mnie 6](#_Toc489699927)

[1.3 przyczyny, dla których autor się interesuje tematyką 6](#_Toc489699928)

[1.4 określenie zastosowanych metod i technologii 6](#_Toc489699929)

[1.5 opis podziału pracy na rozdziały 6](#_Toc489699930)

[2. Wprowadzenie 7](#_Toc489699931)

[3. Metoda badawcza 10](#_Toc489699932)

[3.1 Aplikacja manipulująca elementami. 10](#_Toc489699933)

[3. 2 Funkcjonalności aplikacji. 11](#_Toc489699934)

[3.3 Rodzaje przeprowadzanych testów 14](#_Toc489699935)

[4.x aplikacja do testów autoamatycznych, opis technologii 15](#_Toc489699936)

[4.4 Implementacje 15](#_Toc489699937)

[4. Wyniki i analiza. 16](#_Toc489699938)

[7. Podsumowanie 20](#_Toc489699939)

[Literatura 21](#_Toc489699940)

[Spis rysunków 22](#_Toc489699941)

[Instruckja obslugi programu 23](#_Toc489699942)

# 1. Wstęp

.....

## 1.1. Cel pracy

Celem pracy ...., okreslenie tezy - u mnie brak

## 1.2. określenie tezy- brak u mnie

## 1.3 przyczyny, dla których autor się interesuje tematyką

## 1.4 określenie zastosowanych metod i technologii

## 1.5 opis podziału pracy na rozdziały

# 2. Wprowadzenie

<------- brudnopis ------ >

wprowadzenie czytelnika w stan wiedzy w tej dziedzinie; przegląd literatury przed przystąpieniem do pracy; wykorzystywane narzędzia przy implementacji systemu;

struktura pracy

1. wstep: cel pracy, (teza - u mnie brak), przyczyny zainteresowania sie tematem, zastosowane metody i technologie, podzial pracy

badanie wydajnosci a szersze spojrzenie na to, co sklania programistow do wyboru okreslonego frameworka

https://hackernoon.com/5-best-javascript-frameworks-in-2017-7a63b3870282

2. wprowadzenie: wprowadzenie w stan wiedzy w danej dziedzinie, przeglad literatury:

java script - opis, historia, silnik js, framework-co to, po co, research popularnosci frameworkow, dobór frameworkow, opis wybranych frameworkow - architektura (angular, angular2, react), dlaczego te frameworki;

research testow wydajnosci frameworkow, ogolnie jak bada sie wydajnosc,

https://hackernoon.com/5-best-javascript-frameworks-in-2017-7a63b3870282

(byc moze od tego miejsca osobny rozdzial) przegladarka internetowa - mechanizm dzialania, fazy renderingu strony , model rail, przeglad profilerow - wybor chrome; webdriver, chrome timeline, testowanie wydajnosci frameworkow - czyjes podejscia

3. metodologia - opis metody badawczej, wybór narzędzi

metodologia - dobrana na podstawie wczesniejszego researchu

srodowisko badawcze, metoda - badanie wydajnosci (testy automatyczne, aplikacja do badania, wersje keyed i non-keyed, opis implementacji i optymalizacje, rodzaje testow, jak przeprowadzane, jak uzyskiwane wyniki, jakie dane otrzymuje, jak je prezentuję), chrome timeline, webdriver, inne metody analizy (liczba linii kodu, krzywa uczenia)

4. wyniki i analiza,

licza linii kodu (odniesienie do implemntacji); wyniki testow wydajnosci; porównanie implementacji (opis różnic na poziomie mechanizmów, procesu tworzenia oraz działania aplikacji)

5. wnioski (dyskusja)

6. podsumowanie: wnioski, czy cel zostal osiagniety, dalsze sposoby badania

ze jest mozliwosci rozszerzeania o kolejne frameworki

uwagi:

- co to framework:  
http://poznajprogramowanie.pl/czym-jest-framework-i-po-co-go-uzywac/

- metody badania:  
warmupy, wydluzanie czasu trwania testu - pokazac tez probke wynikow bez tych ulepszen; test count-srednia, warmup iterations, dopiswanie do pliku-kilka dni (rozne dni), save best, reject worst, do nothing  
- metody badawcze naukowe  
obliczanie sredniej arytmetycznej

-opis: rendering: recalc, update, layout - czyli dzialanie przegladarki, dzialanie chrome, chrome tracing

- sposob pomiaru wydajnosci

- na razie nie rozdrabiac sie tylko zrobic ogolny obraz i potem go uszczegolowiac - bo wsiakne

- przy opisie metod - jako baza - aplikacja vanillajs - bo tam sa bazowe pojecia

- ustalic kolejnosc metod testowych - np add\_500f\_2k, add500f\_1k - mam je odwrotnie

- usunac przypadek add500przy4k

- troche okroic przypadki testowe - jutro zrobic badania wszystkie - na razie z wynikiem dla angulara2 tak jak jest - ewnetualnie szybko zerknac czy jest roznica w implementacji

- opis implementacji, optymalizacje, keyed i non-keyed - wyjasnienie; pokazanie fragmentow kodu - troche tylko - np wywolanie add we wszystkich frameworkach - ze updateView to jak dzialanie frameworka

- roznice w implementacjach, sposoby podejscia

-! zrobic przyjazne nazwy benchmarkow

-! wyniki beda chyba za obszerne i w pracy moge omowic tylko poniektore przypadki, a resztę tylko podsumowac krotko - a wiec w pracy taka okrojona wersja - tym lepiej bo calosc opracowac to duzo roboty

- wstep: javascript, framework javscrtip; research frameworkow, sposoby testowania wydajnosci; dobor sposobu, opis terminologii uzywanej podczas testowania (chrome timeline, rail, fazy renderingu strony, dzialanie przegladarki)

- opis pracy studenta: aplikacje do testowania - napisane przy pomocy okreslonych bibliotek (frameworkow); przypadki testowe - opis; implementacja aplikacji; porównanie implementacji (liczba linii kodu, ocena krzywej uczenia się); aplikacja do testow automatycznych, parser i wyswietlanie (parser - mozna dac krotki opis parsowania ramek); wyniki - analiza i synteza

ale poza samym badaniem wydajności, mimo wszystko warto będzie zawrzeć też szersze spojrzenie

nawet jeśli to wydajność będzie głównym przedmiotem badania

choć by coś takiego jak rozmiar kodu, ile jest w nim boilerplate - to często interesuje wybierających te czy tamtą technologię

albo jak złożone jest zrobienie tej samej rzeczy w róźnych frameworkach

nawet learning curve, choć to raczej dość subiektywne

2.7 Testy wydajności

ewentulane testy samego czasu ramek na innych przegladarkach - juz reczne nawet

micro, macro  
definicja benchmarku: \*m. in. wybór zagadnień do porównania (to na inny wypadek)  
Benchmark, testowanie wzorcowe - test wydajności systemu komputerowego: sprzętu lub oprogramowania.

Istnieje wiele programów, które testują różne charakterystyki sprzętu komputerowego i oprogramowanie - moc pojedynczej maszyny, interakcje w systemie klient-serwer (z pojedynczym lub wieloma klientami) czy liczbę transakcji na sekundę w systemie przetwarzania transakcyjnego.

W miarę jak pojawiają się nowe wersje oprogramowania i sprzętu, zmieniają się składowe testy benchmarków i ich wagi w obliczaniu wyniku benchmarku - dlatego jednym z warunków uzyskania wiarygodnej oceny w testach porównawczych jest konieczność zastosowania tej samej wersji benchmarku.  
https://en.wikipedia.org/wiki/Benchmark\_(computing)

benchmark - kilka definicji (benchmark pojecie oglne)  
benchmark w naukach komputerowych

testy wydajnosci, benchmarki - ogolnie

2.7 Wydajność oprogramowania.  
potem wydajnosc aplikacji www

If it is fast and ugly, they will use it and curse you;

if it is slow, they will not use it.

— David Cheriton, computer science professor & billionaire entrepreneur

Wydajnosc aplikacji, wydajnosc oprogramowania  
IEEE, IEC, ISO -> dostarczają definicje  
Performance (definicja) - stopień w jakim system spełnia jego wyznaczone funkcje w ramach danych ograniczeń, takich jak szybkość, zużycie zasobów, dokładność.

response time - jaki czas dokladnie jest mierzony, od kiedy do kiedy

3.1 Metoda badawcza   
3 fazy metody: 1) projekt benchmarków 2) wykonanie benchmarków 3) prezentacja i analiza wyników; pojęcie benchmark engineering  
wymagania stawiane benchmarkom: reprezentatywnosc (trafnosc)

--------------------------------

Ten projekt dotyczy porównania frameworków JavaScript przy użyciu testów wydajności (tzw. benchmarków). W związku z tym niniejszy rozdział zawiera krótkie wprowadzenie do metod testowania wydajności. Następnie omówione są frameworki JavaScript. Jako że pojęcie MVC dotyczy omawianych frameworkow, bedzie równiez przedstawione.

Benchmarking  
Ta sekcja zawiera ogólne wprowadzenie do benchmarkingu w dziedzinie inżynierii oprogramowania. Bedą przedstawione populane definicje, omówione zostanie znaczenie testowania wydajności oraz przedstawiony zostanie podział metod testowania wydajności takich jak mikro i makro-benchmarking.

Definicje pojęcia "benchmark"  
ISO - miedzynarodowa organizacja normalizacyjna, IEC - miedzynarodowa komisja elektrotechniczna, IEEE - instytut inżynierów elektryków i elektroników  
ISO oraz IEC defniuja benchmark jako: "Norma na podstawie której można zmierzyć lub oszacować wyniki". Podobnie IEEE opisuje benchmark jako: "Norma na podstawie której można dokonać pomiarów lub porównań. Procedura lub test pozwalający na porównanie systemów lub komponentów względem siebie lub do wzorca". SPEC - standard performance evaluation corporation dostarcza węższej definicji, skoncentrowanej na wydajności systemów komputerowych: "Test lub zbiór testów zaprojektowanych do porównania wydajności jednego systemu komputerowego względem drugiego". Wikipedia: "Test wydajności systemu komputerowego: sprzętu lub oprogramowania (...) Można badać np: liczbę operacji zmiennoprzecinkowych procesora, wydajność kompilatorów lub systemów zarządzania bazami danych".

Computer benchmarking  
Benchmarki przeprowadza się dla porównania różnych platform, narzędzi lub technik wykonując pewne eksperymenty mające na celu znalezienie szukanych różnic. Benchmarking to zwykle program luz zestaw programów które oceniają wydajnośc aplikacji lub konkretnego rozwiązania pod pewnymi warunkami, w odniesniu do wydajności innej aplikacji czy rozwiązania. Benchmarking to uczciwe porównanie pomiędzy różnymi rozwiązaniami (Bouckaert). Benchmarking pomaga być bardziej otwartym na inne podejścia niż być zaślepionym jednym rozwiązaniem które może się wydawać najlepszym rozwiązaniem danego problemu. Benchmarking jest także użawany jako sposób standaryzacji pomiarów oraz dostarczenia powtarzalnych obiektwynych danych. W dziedzinie nauk komputerowych : porównanie CPU, DB, algorytmy. Benchmarki także są używane do oceny wydajności javascriptu.

Charakter (rodzaje) computer benchmarków  
Następujące metryki to główne metryki w benchmarkingu:   
- czas - ile czasu trwa wykonanie pewnej operacji  
- wskaźnik (rate) - szybkość z jaką system może wykonać pewną pracę

# 3. Metoda badawcza

zrobic instrukcję tak zeby promotor mogl pobrac zrodla i przetestowac u siebie oraz opis przeprowadzania przykladoweg testu (przetestowac na innym laptopie, załączyc odpowiedni chromedriver)

krótki wstep do rozdzialu.... ze testowanie bedzie automatyczne na stowrzonych aplikacjach manipulujacych i testy poprzez webdriver

uwagi do pisania tego rozdzialu:  
- uzasadnienie doboru takiej aplikacji   
- w jaki sposob dobierałem przypadki testowe - dlaczego takie a nie inne (czyli opis co mozna robic z elementami modelu) - moze analiza jakichs przykladowych aplikacji i co robią one z modelem, co zazwyczaj dzieje sie z modelem w aplikacji  
- (w przypadku nie zmieniania brzmienia tematu) ma to byc analiza porównawcza, nie wydajnosci, wiec niech rozdzial skupia sie na ogólnej metodzie badania, nie tylko wydajnosci, ale tez porównania implementacji pod wzgleem architektury, ilosc oferowanych przez framework mechanizmow, ilosci linii kodu, krzywa uczenia itd.

ZACZĄC OD PRZECZYTANIE ROZDZIALU 2 PERFORMANCE COMPARISION i na nim sie wzorowac (napisac ze wzoruje sie na nim i na SK, moge je opisac - ten opis bedzie dla framework research, a tutaj odniesienie i bazowanie sie na tym)

spis tresci rozdziału:  
najpierw jescze odniesienie sie do pracy na ktorej sie wzoruję, tj SK (rozdzial wczesniej moze traktowac ogolniej o wydajnosci i robic jakis research, albo nawet opisywac po krótce badanie SK - moge tym zaczac rozdzial, zeby bylo kompletniej)  
1. Aplikacja manipulująca - opis ogólny.  
uzasadnienie powstania takiej a nie innej aplkacji testowej, podanie że wzorowałem się na sposobie Stefan Krause? sposoby testowania frameworkow - i wlasnie sposob SK  
2. Funkcjonalnosci aplikacji manipulującej.  
3. Rodzaje przeprowadzanych testów.  
dlaczego takie a nie inne  
sposob przeprowadzania testow - testy automatyczne  
Środowsiko testowe - takze opis sprzetu do testowania  
aplikacja do testowania  
app2 - tutaj uzywam funkcji matematycznych do wyliczania wynikow - dowiedziec sie jakie dokladnie metody sa stosownane i opisac je  
app3  
implementacje aplikacji manipulującej -  
skad wiadomo ze dobre (optymalne) - porownanie z pewnymi wzorcowymi implementacjami  
 zarządzanie modelem, optymalizacje, jak kod byl pisany, zeby byl optymalny, ze byly przeprowadzane testy aby aplikacje byly optymalne  
ze operujemy na modelu ktory nie ma powizania z baza - bo badamy framework - normlanie jeszcze do tego dochodzi czas komunikacji z baza  
sposob kompilacji  
spojrzenie na operacje testowe pod kątem implementacji  
wersje keyed i non-keyed

https://sites.google.com/a/chromium.org/chromedriver/

opis jak mozna testowac recznie (zacząc od tego wyjasnienia)

, i ze robi to aplikacja do testowania - warmupy, test wlasciwy, test count, ileś podejść (czyli test count \* podejscia) - z tego wyliczanie sredniej ( 3 podejscia: best, reject, nothing), czas testu (2 sek - optymalny, zeby testy nie zanizaly wynikow - jak za szybkie - napisac ze wczesniejsze badanie wykazalo to) - tutaj przydalaby sie jakas metoda badawcza naukowa do opisania; co dalej - kazdy pojedynczy przypadek testowy w ten sposob, duzo przypadkow - podsumowania kategorii i podsumowanie ogolne; że badam czas (ms); jak wyglada pojedyncza próbka testowa: javascript, recalc, layout, update itd; badanie pamieci - after;

testy automatyczne - webdriver, chrome webdriver, webdriver protocol, wybór chrome, chrome tracing; - aplikacja do testow automatycznych- krotki opis

wyniki generowane prez aplikację, aplikacja 2 ktora wylicza srednie wyniki

pokaz aplikacji wyswietlacjacej wyniki - jakis screen - i potem moge juz zamieszczac screeny wynikow i bedzie wiaodmo skad są - ze to aplikacja

## Implementacje aplikacji testowych we frameworkach

wersje keyed, non-keyed - reazlizacja operacji

## 3.1 Różne podejścia do badania wydajności frameworków.

chmielewski email

comparative eval : https://github.com/kamicane/slickspeed/tree/master

comparing backbone, jquery and javascript

runeberg joakim - fragment o badaniu

stefan krause: http://www.stefankrause.net/wp/?p=392 - ze to zadna praca naukowa ale to aktualnie najbardziej rozbudowany test frameowrkow js jaki mozna znalesc w sieci - motywacja do tego by samemu zrobic podobne testy, ze kod jest open source

https://github.com/krausest/js-framework-benchmark

benchmarkin javascript - praca   
http://dbmonster.sourceforge.net/

https://www.romexsoft.com/blog/js-frameworks-comparison/  
https://medium.com/javascript-scene/angular-2-vs-react-the-ultimate-dance-off-60e7dfbc379c  
https://evancz.github.io/react-angular-ember-elm-performance-comparison/  
https://www.cleveroad.com/blog/react-vs-angular-ultimate-performance-research-2017

ogolnie benchmark konkretnego frameworka

hasła pod jakimi szukałem:

javascript frameworks performance comparison

javascript framewokrs benchmarks

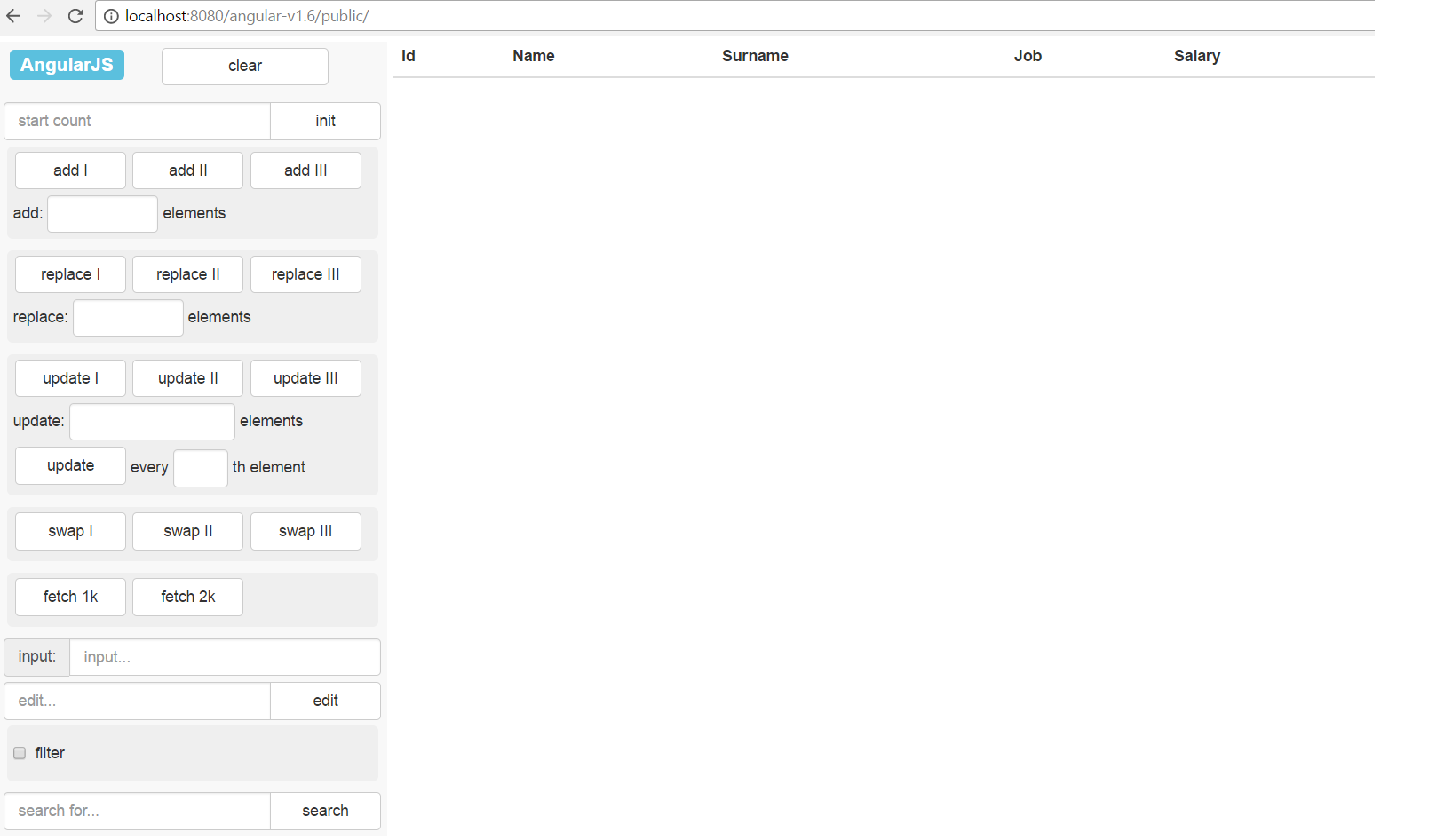
angular 2 benchmark

## 3.1 Aplikacja manipulująca elementami.

tutaj napisac jak powstawaly te aplikacje - chodzi o przyjętą metodę badawczą - dlaczego te a nie inne funkcjonalnosci

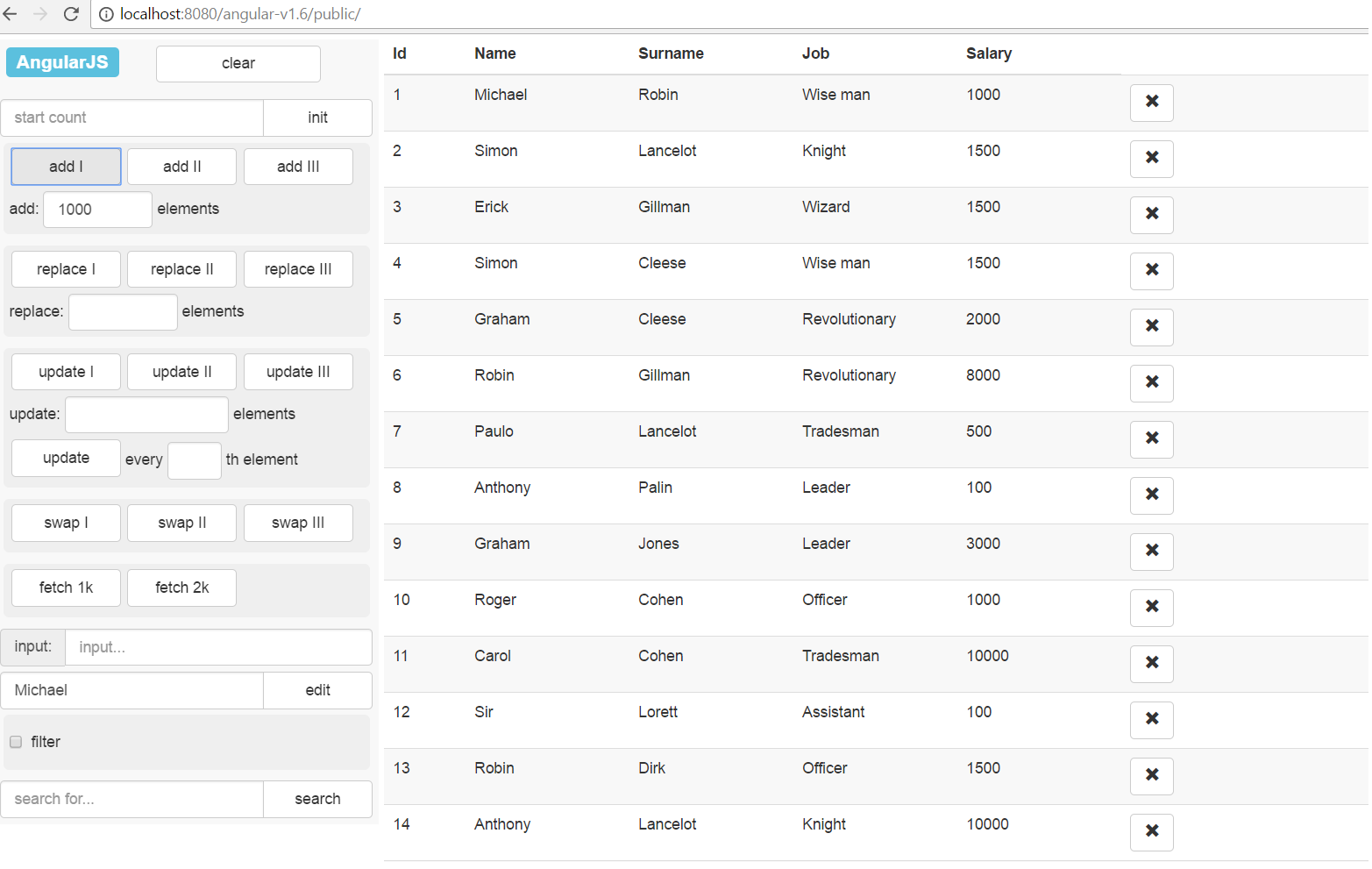
W celu testowania wydajności należało zaimpelemntować aplikacje testowe.

W celu testowania wydajności, zostały zaprojektowane aplikacje napisane przy użyciu testowanych bibliotek (angular, angular2, react, czysty javascript). Każda aplikacja posiada dwie wersje, tzw. keyed oraz non-keyed. Wersja keyed polega na tym, że w aplikacji każdemu obiektowi modelu danych jest przyporządkowany konkretny obiekt DOM (obiekt widoku), wersja non-keyed natomiast nie definiuje takiego powiązania i traktuje obiekt DOM jako kontener, który może przechowywać dowolny obiekt modelu. W sumie powstało więc 8 różnych aplikacji. Każda z nich wygląda tak samo i oferuje takie same funkcjonalności, różnią się natomiast wewnętrzną implementacją, w tym bibliotekami używanymi do osiągnięcia tych samych rezultatów. Widok przykładowej aplikacji po załadowaniu prezentuje Rys. 4.1.



Rys. 4.1 Aplikacja do testowania AngularJS po załadowaniu

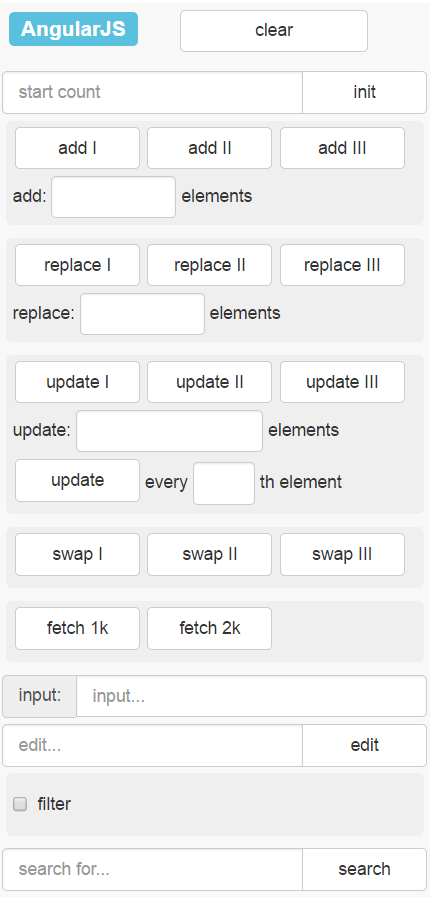
Interfejs aplikacji jest prosty, bez dodatkowych efektów wizualnych (bogatsze ostylowanie wydłuża czas renderingu), aby uwydatnić wpływ działania danego frameworka (czyli czas działania kodu javascript) na wydajność mierzonych operacji. W skład interfejsu wchodzi boczny panel menu (po lewej) gdzie mamy dostęp do wszystkich funkcjonalności. Po prawej w głównej części znajduje się tabela, w której będą wyświetlane dane. Pojedynczy obiekt danych to zwykły obiekt JavaScript posiadający właściwości: {id, imię, nazwisko, praca, zarobki}. Obiekty danych są odwzorowane na wiersze tabeli. Działania wykonują się na dużym zbiorze danych (1-3 tys. elementów w modelu danych). Rys. 4.2 prezentuje przykładowy widok aplikacji po wykonaniu operacji "dodaj 1 tys. elementów".



Rysunek 4.2 Aplikacja do testowania AngularJS - dodanie 1 tys. elementów

## 3. 2 Funkcjonalności aplikacji.

Dostęp do funkcjonalności aplikacji jest oferowany poprzez menu boczne, pokazane na Rys 4.3



Rysunek 4.3 Menu aplikacji manipulującej

Aplikacja posiada następujące funkcjonalności (analizując kolejno pola na pasku menu):

1. clear - usuwanie wszystkich uprzednio utworzonych elementów (tabela z danymi będzie pusta)
2. init - tworzenie nowych losowych elementów w liczbie podanej jako parametr w polu tekstowym obok przycisku (dodawanie na początek). Każdy element posiada unikalne id.
3. add I, II, III - dodawanie nowych losowych elementów w liczbie podanej jako parametr w polu tekstowym przy etykiecie "add". Cyfra rzymska I, II, III oznacza, że dodawanie nowych elementów odbywa się kolejno: na początek, w środek lub na koniec (jeśli istnieją już jakieś elementy). Każdy element posiada unikalne id.
4. replace I, II, III - zamiana pewnej grupy już istniejących elementów, na nowe (stare obiekty modelu są usuwane a w ich miejsce trafiają nowe obiekty). Liczność grupy jest podawana jako parametr w polu tekstowym przy etykiecie "replace". Cyfra rzymska I, II, III oznacza, że nowa grupa zamieni grupę składającą się z określonej liczby, kolejno pierwszych, środkowych bądź ostatnich elementów. Nowe obiekty posiadają nowe id.
5. update I, II, III - aktualizacja pewnej grupy już istniejących elementów (pola obiektów są modyfikowane). Liczność grupy jest podawana jako parametr w polu tekstowym przy etykiecie "update". Cyfra rzymska I, II, III oznacza, że będzie zaktualizowana grupa składającą się z określonej liczby, kolejno pierwszych, środkowych bądź ostatnich elementów. Obiekty nie zmieniają id.
6. update every - aktualizacja istniejących obiektów jak wyżej, z tym że obiekty są aktualizowane co n-ty element (liczba n przekazana jako parametr w polu tekstowym)
7. swap I, II, III - zmiana miejscami dwóch elementów modelu (czyli dwóch wierszy w tabeli). Cyfra rzymska I, II, III oznacza zamianę kolejno między dwoma pierwszymi, środkowymi lub ostatnimi wierszami w tabeli.
8. fetch 1k,2k - pobieranie obiektów danych z serwera. Obiekty są predefiniowane, w ilości 1 tys. bądź 2 tys.
9. input - pole tekstowe, służące badaniu reakcji aplikacji na wpisywanie nowych danych przy jednoczesnym istnieniu wielu wierszy danych (uprzednio dodanych np. za pomocą metody "init")
10. edit - wpisywanie danych w to pole tekstowe powoduje edycję pierwszej komórki w pierwszym wierszu tabeli. Operacja jest przeprowadzana gdy dodano wiele wierszy danych (np. za pomocą metody "init")
11. filter - filtrowanie wierszy tabeli w ten sposób by wyświetlić jedynie elementy o id podzielnym przez 10
12. search - wyszukiwanie w wierszach tabeli, komórek zawierających określony ciąg znaków. Komórka ze znalezionych tekstem jest podświetlana na żółto. Wyszukiwanie następuje w reakcji na wpisanie ciągu znaków w pole tekstowe.
13. kliknięcie na wierszu w tabeli - wiersz zmienia kolor na niebieski
14. przycisk "usuń wiersz" - kliknięcie powoduje usunięcie wiersza z tabeli

## 3.3 Rodzaje przeprowadzanych testów

Aplikacja manipulująca pozwala wykonywać rozmaite operacje na tabeli z danymi (modelu danych). Do przeprowadzenia testów zostały zdefiniowane następujące przypadki testowe (nazwa w nawiasie odnosi się do wykorzystywanej funkcji aplikacji manipulującej):

1. dodawanie (add I, II, III) - 500, 1 tys. lub 2 tys. elementów do pustej tabeli (przypadki A. 1-3) ; dodawanie 500 elementów na początek, środek i koniec przy istniejącym 1 tys. lub 3 tys. elementów (przypadki A. 4-9); dodawanie 1 bądź 2 elementów na początek lub koniec przy istniejącym 1 tys. danych (przypadki A. 10-13).
2. zamiana (replace I, II, III) - 500, 1 tys. lub 2 tys. elementów, przy istnieniu takiej samej ich liczby (przyp. B. 1-3); zamiana grupy 500 początkowych, środkowych lub końcowych elementów przy istnieniu 1,5 tys. bądź 3,5 tys. (przyp. B. 4-9); zamiana 1 lub 2 elementów na początku, lub 1 na końcu przy istnieniu 1 tys. elementów (przyp. B. 10-12);
3. aktualizacja (update I, II, III) - 500, 1 tys. lub 2 tys. elementów, przy istnieniu takiej samej ich liczby (przyp. C. 1-3); aktualizacja grupy 500 początkowych, środkowych bądź końcowych elementów przy istnieniu 1,5 tys. bądź 3,5 tys. (przyp. C. 4-9); aktualizacja 1 lub 2 elementów na początku, lub 1 na końcu przy istnieniu 1 tys. elementów (przyp. C. 10-12);
4. aktualizacja częściowa (update every) - aktualizacja co drugi, trzeci bądź czwarty element przy istnieniu kolejno 1 tys. , 1,5 tys. bądź 2 tys. elementów (przyp. D. 1-3);
5. ...

Za pomocą tych konkretnych operacji ( przypadków testowych ) zostanie zbadana każda z 8 aplikacji manipulujących.

## 3.4 Metoda badawcza

# 4. Wyniki i analiza.

email do stefan krause w sprawie aplikacji angular2?

testy cząstkowe i podsumowania - moge pokazac kilka czastkowych a potem tylko podsumowania zamiescic - i dac opis liczenia statystyk

Niektore przypdki testowe maja charakter pogladowy i nie beda wliczane do obliczania srednich z testow poniewaz moga wprowadzac duze dyspropozrcje w wynikach (jak dzialania typu dodaj 1 element przy 1 tys danych - wtedy rozstep danych jest zbyt duzy) Tzn przeprowadzam testy i je analizuje, a czesc z testwo wliczam do liczenia srednich, a czesc pomijam (jak dzialnia na 1 elemencie).

Porównanie testów wydajnosci moich z przeprowadzonymi przez kogos innego (stefan krause).

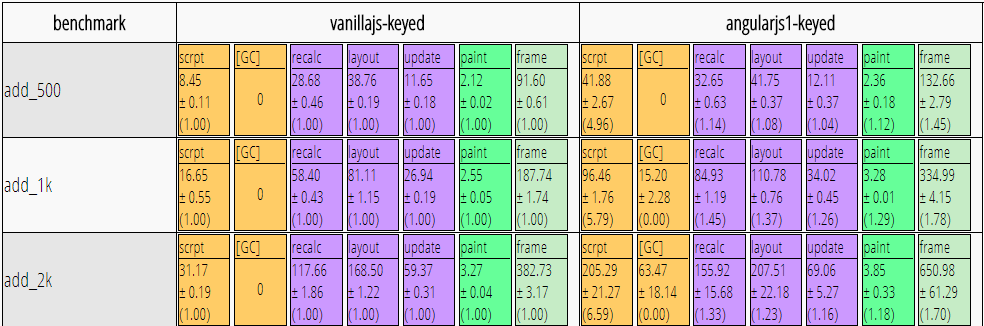
a) Dodawanie elementów

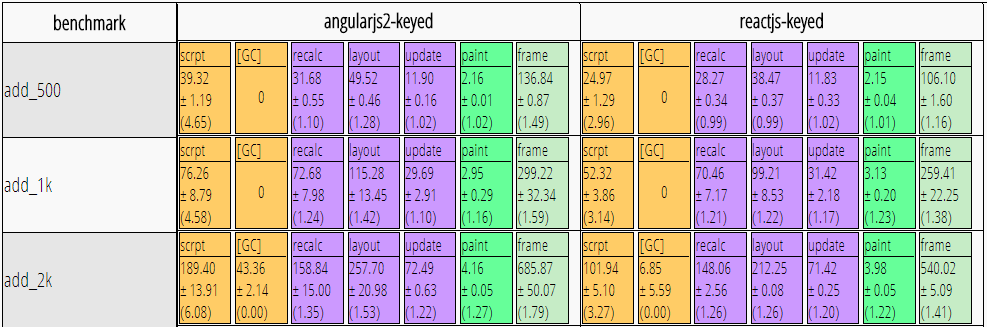
Dodawanie elementów polega na tworzeniu nowych losowych elementów w tabeli z danymi.

Operacja dodawania składa się z:  
- utworzenia losowych danych (funkcja randomObjects)  
- aktualizacji modelu danych, tj. dodaniu losowych danych utworzonych w kroku wyżej, do aktualnego modelu danych  
- odzwierciedleniu modelu danych na widok aplikacji (czyli utworzenie elementów DOM w sposób specyficzny dla frameworka; w aplikacji vanillajs odpowiada za to funkcja updateView)

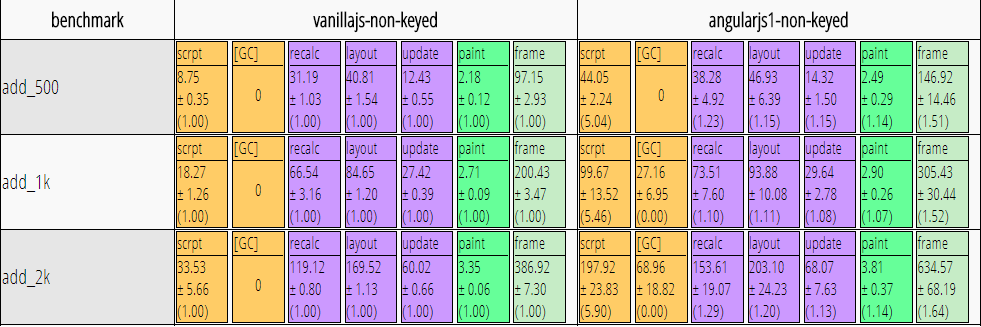
dodawanie na pusto

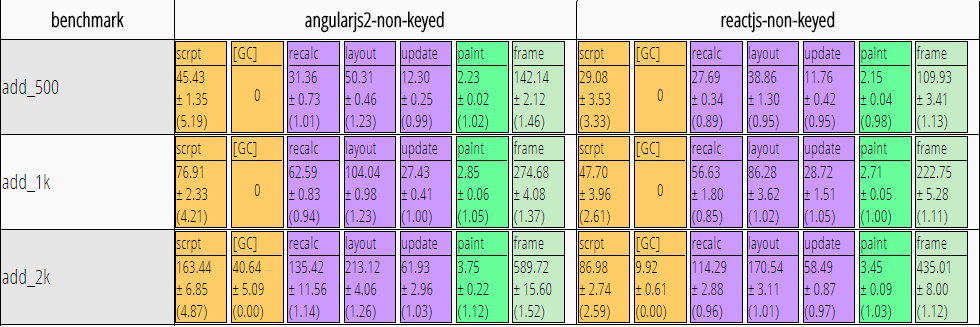
keyed



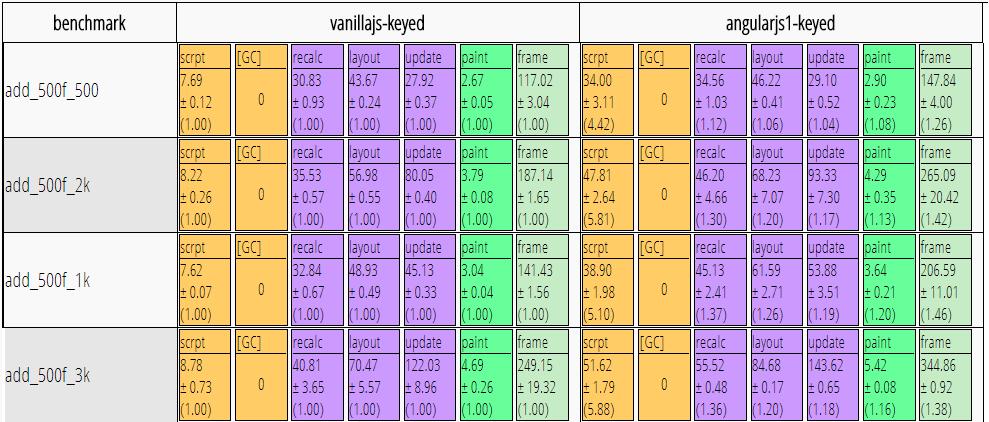


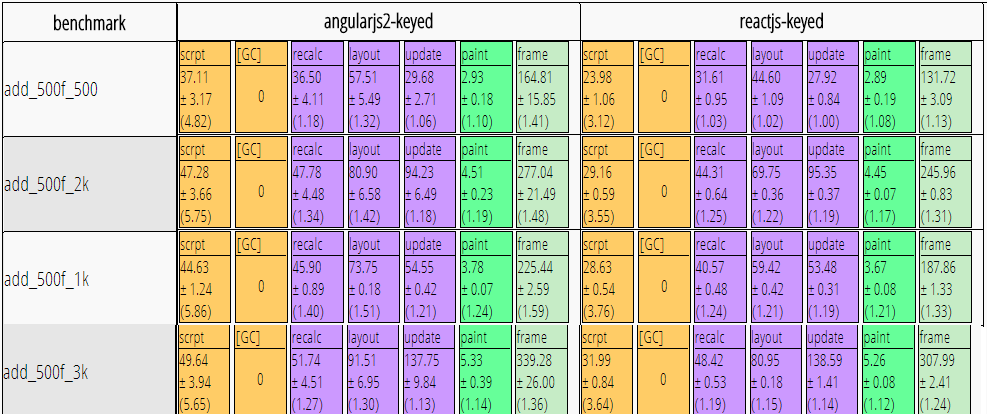
non-keyed



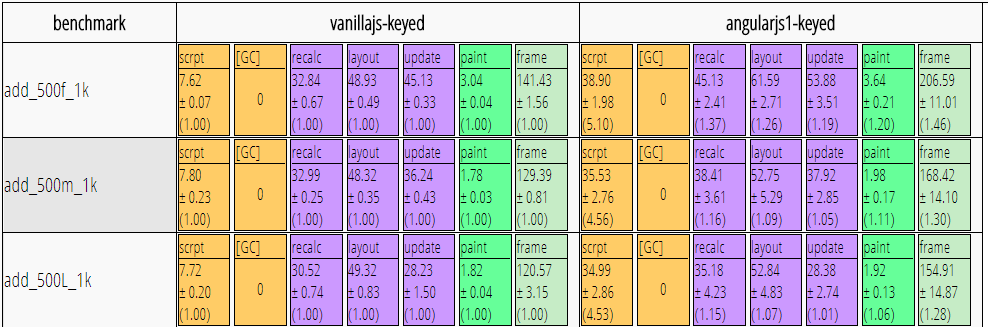


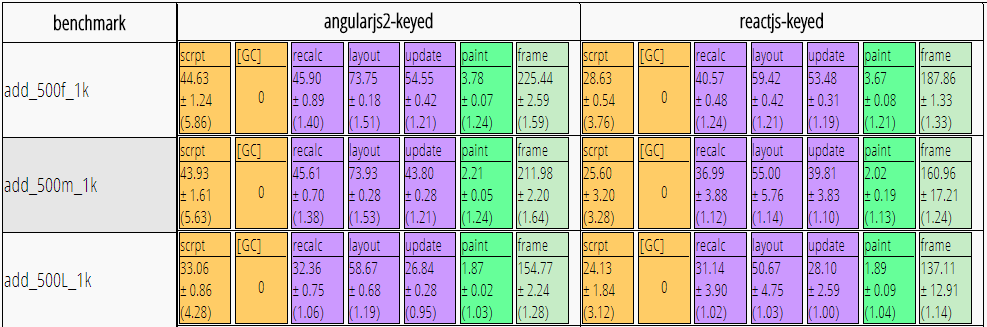
dodawanie przy elementach istniejacych:



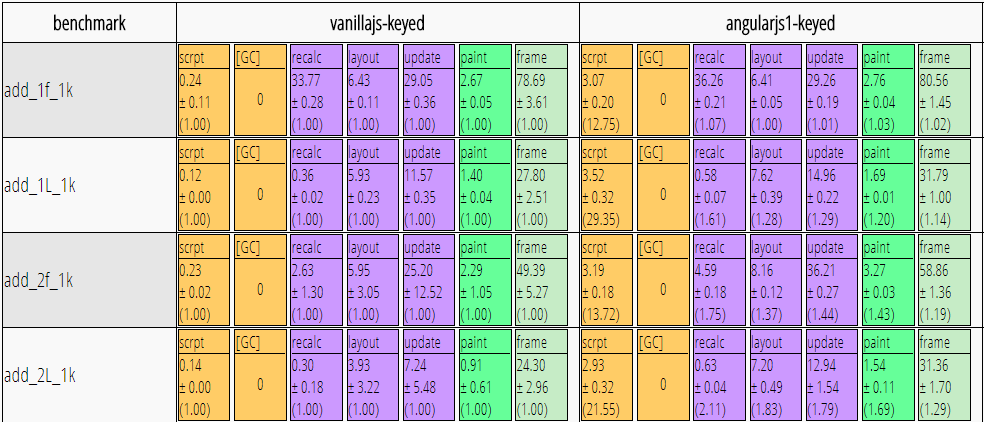


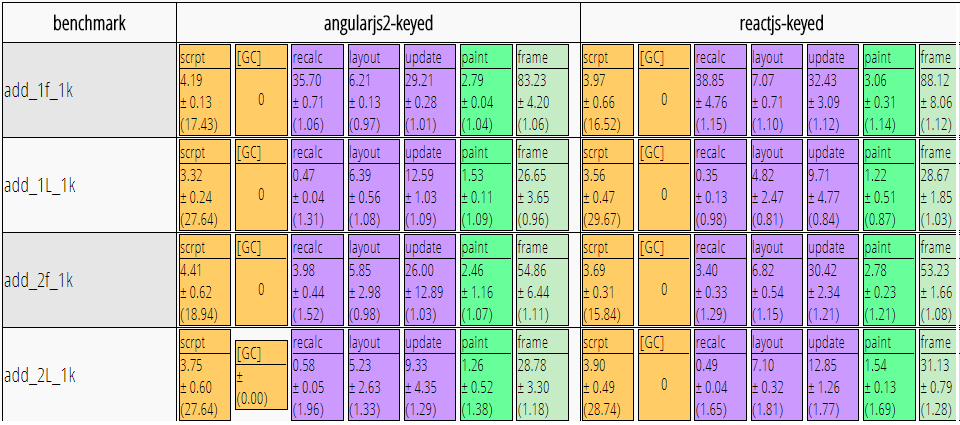
dodawanie - sprawdzanie metody:





dodawanie - pojedyncze elementy





# 6. Podsumowanie

Wnioski szczegółowe i uogólnione. Czy i w jakim stopniu cel pracy został osiągnięty. Kierunki ewentualnej kontynuacji pracy, sposób przyszłego wykorzystania. Wnioski mają wynikać z treści pracy.

angular2 nie jest wiele lepszy niz angular1 (pod wzgledem wydajnosci), ale pod wzgledem projektownaia to duzy postep (przy czym wydajnosciowo jest jednak troche lepszy); React jest najszybszy, ale jest bardziej skomplikowany; vanillajs jest najszybszy ogolnie; kedyed i non keyed ; porownanie kiedy jaki framework jest szybszy przy jakich operacjach

# Literatura

1. ala ...

2. ola...

# Spis rysunków

# Instruckja obslugi programu