**POLITECHNIKA BIAŁOSTOCKA**

**WYDZIAŁ** **INFORMATYKI**

**KATEDRA INŻYNIERII OPROGRAMOWANIA**

PRACA MAGISTERSKA

TEMAT:

Analiza porównawcza 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

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;

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

3. metodologia - opis metody badawczej, wybór narzędzi

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

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

hoć 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

# 3. Metoda badawcza

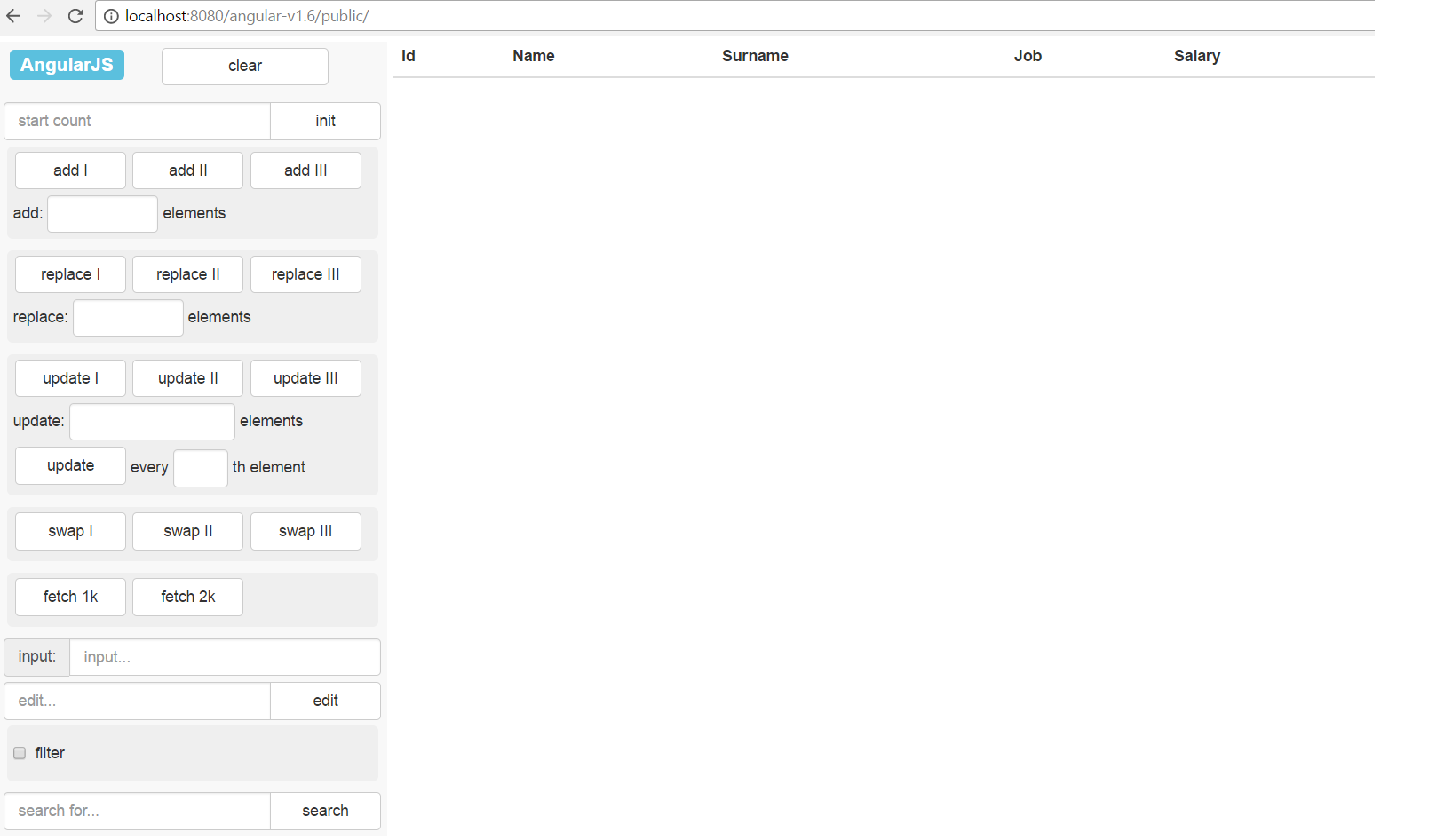
krótki wstep do rozdzialu....

uwagi do pisania tego rozdzialu:

- uzasadnienie doboru takiej aplikacji

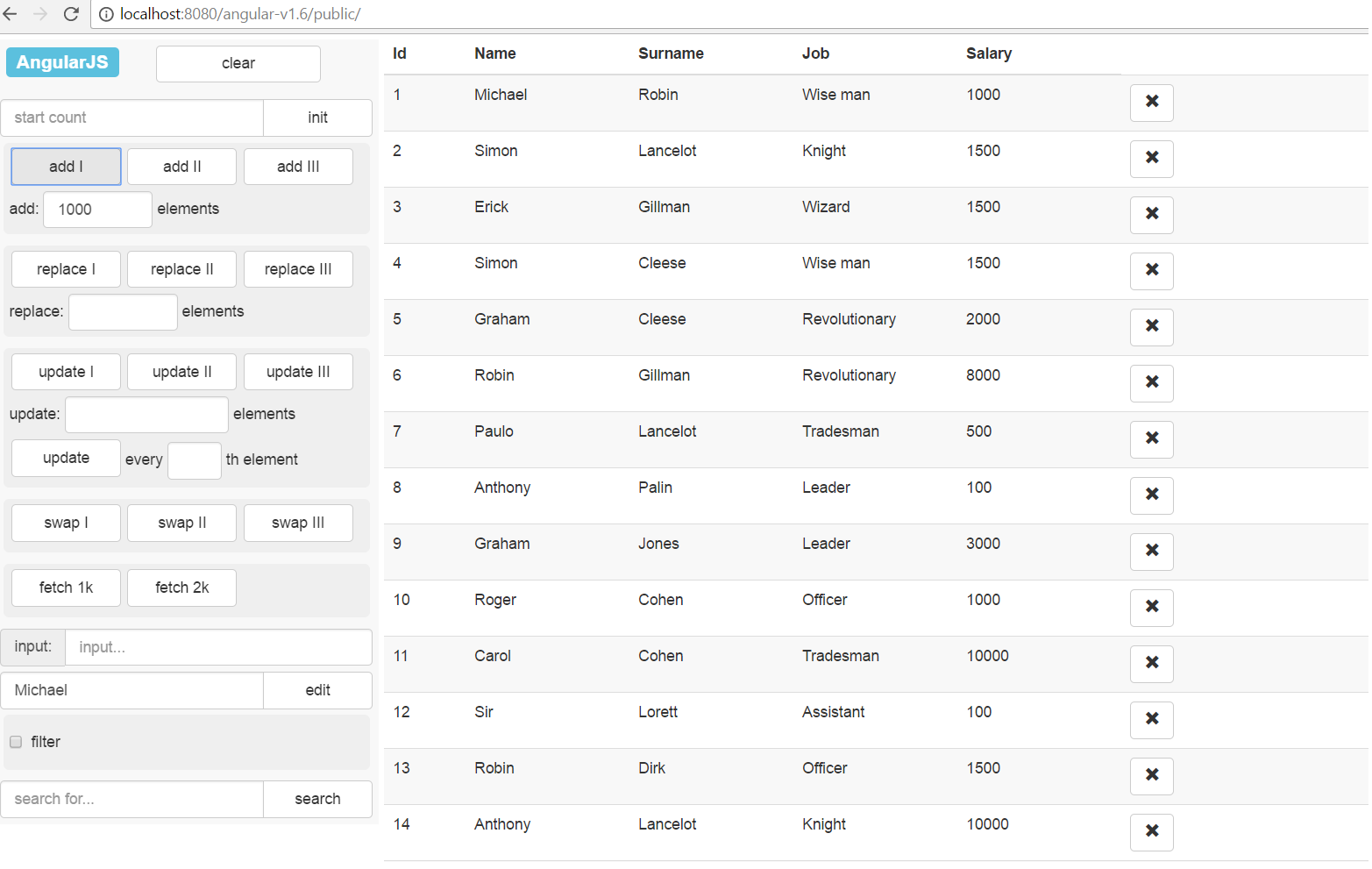
## 3.1 Aplikacja manipulująca elementami.

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

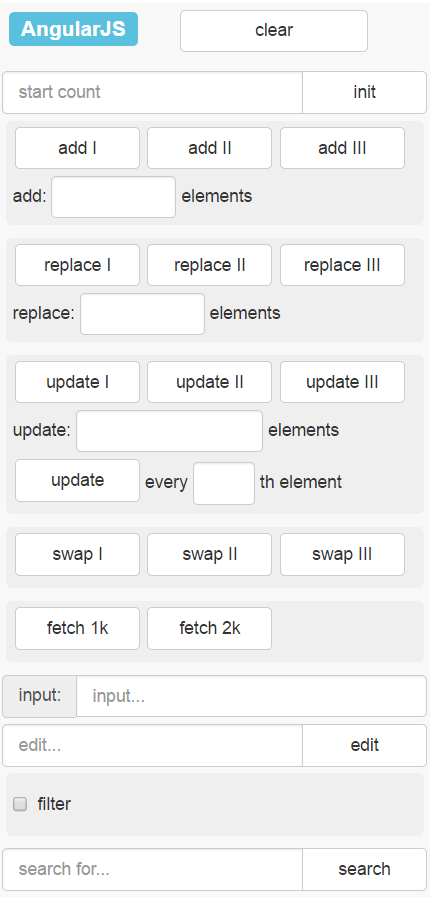
Interfejs aplikacji jest prosty, bez dodatkowych efektów wizualnych (bogatsze ostylowanie wydłuża czas renderingu), aby uwydatnić wpływ działania danego frameworka (czyli czas działania kodu javascript) na wydajność mierzonych operacji. W skład interfejsu wchodzi boczny panel menu (po lewej) gdzie mamy 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

Badania przeprowadzam w przeglądarce chrome, gdyż posiada ona rozbudowany zestaw narzędzi do analizy wydajności aplikacji webowych (timeline, tracing). Uruchamiam serwer poprzez który będę łączył się z aplikacjami testowymi, wpisuję w przeglądarkę adres dowolnej, np http://localhost:8080/vanillajs/public (aplikacja w czystym js). Uruchamian konsolę chrome (ctr+shift+c), zakładka performance, ctrl+e (rozpoczęcie rejestrowania wydajności). Przykładowo chcę zbadać przypadek dodawania 1tys elementów do strony. W tym celu wpisuje w odpowiednie pole ilość dodawanych elementów (1000), klikam odpowiedni przycisk (add). Następuje dodanie elementów w aplikacji. Nastepnie wciskam przycisk clear - aby pozbyc sie dodanych elementów. Znowu wciskam kombinacje add-clear, aby dodawać i usuwać dodawane elementy (aby doadwanie zawsze było w tym samym stanie aplikacji). Kilkukrotne powtarzanie wykonuję aby z otrzymanych próbek obliczyć średnie czasy. Przechodzę znowu do zakładaki performance i wciskam ctrl+e aby zakończyć test. Z wykresów odczytuję takie informacje jak czas wykonywania skryptu, czasy renderingu oraz calkowity czas dla kazdego dodawania jakie wykonalem. Obliczając średnie czasy moge wybrać kilka najlepszych próbek i je uśrednić albo odrzucić kilka najgorszych a usrednic pozostałe. Analogicznie posteuje dla innych przypadków (usuwanie, edycja, filtrowanie itp). W efekcie otrzymam czasy trwania poszczegolnych operacji (zwanych benchmarkami) dla poszczególnych frameworków.

dokladny spsob badania, ile probek itd

testy automatyczne

Opis wszystkich testów i podsumowania. Czyli jakie testy przeprowadzę przy uzyciu aplikacji, ze testy pamieciowe, ze podsumowania kategorii i ogolne podsumowanie.

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, aczesc pomijam (jak dzialnia na 1 elemencie).

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

Testy można wyklikac recznie bądź autoamtycznie.

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

4.y metoda badawcza - czyli opis jak badam, jak uzyskuje probki i wyniki

## 4.x aplikacja do testów autoamatycznych, opis technologii

wersje keyed, non-keyed - reazlizacja operacji

## 4.4 Implementacje

# 4. Wyniki i analiza.

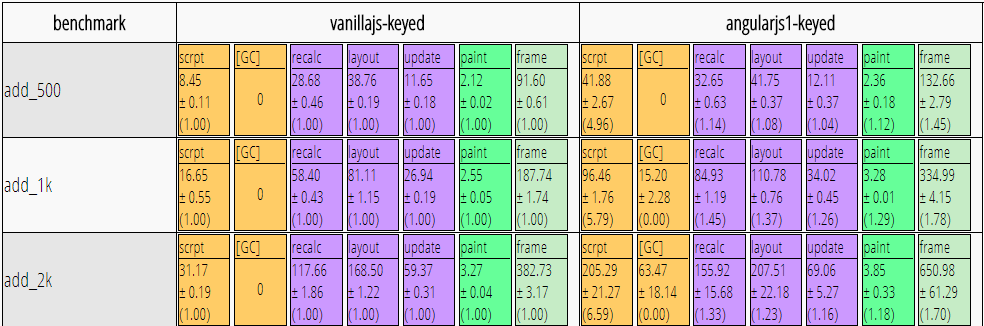
a) Dodawanie elementów

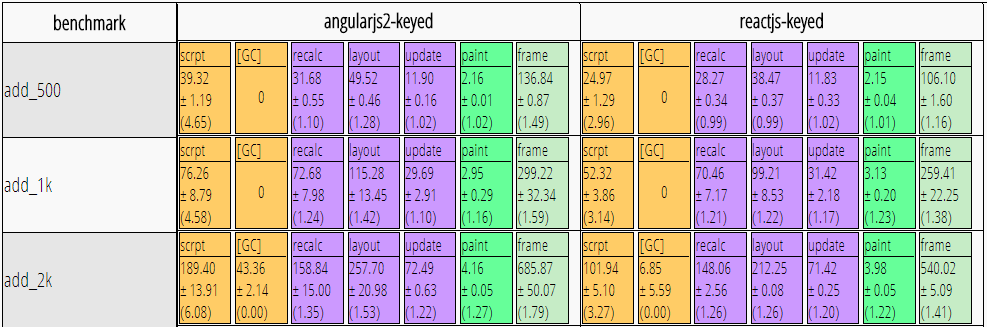
Dodawanie elementów polega na tworzeniu nowych losowych elementów w tabeli z danymi.

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)

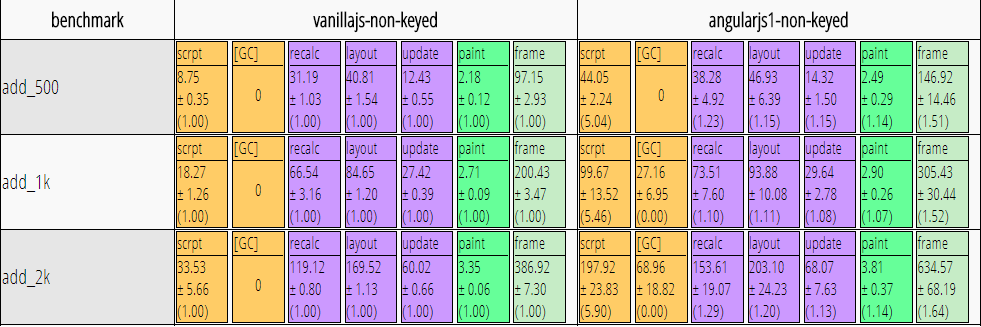
dodawanie na pusto

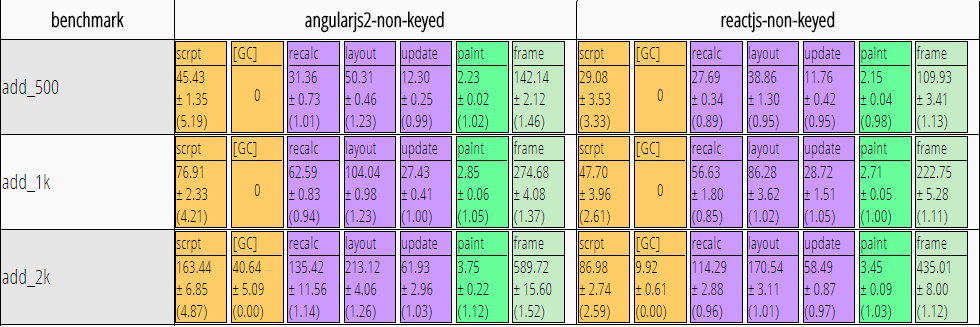
keyed



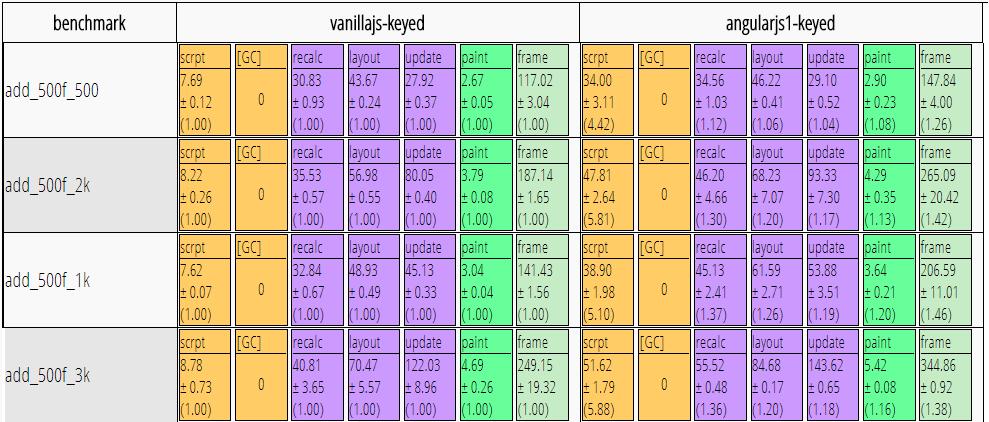


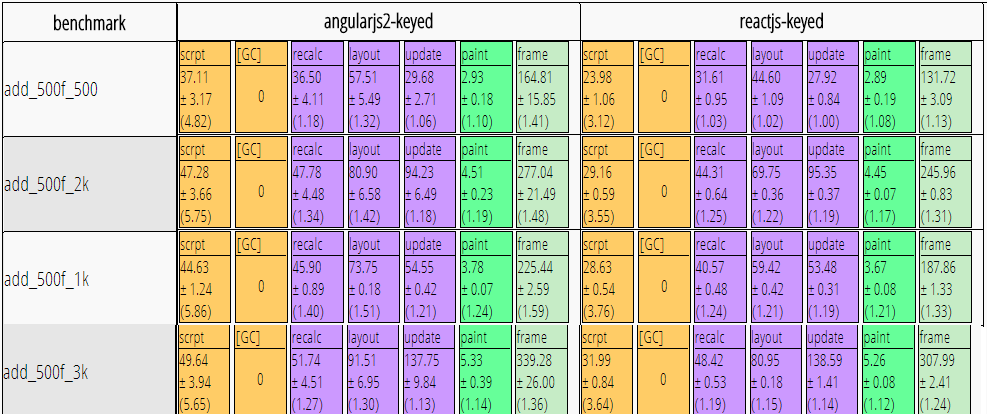
non-keyed



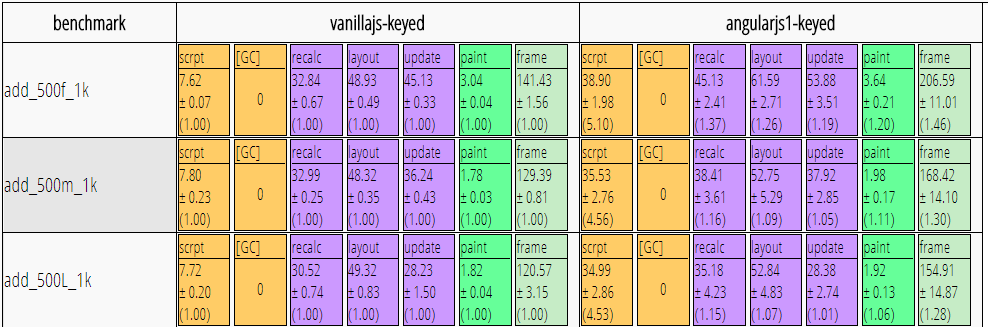


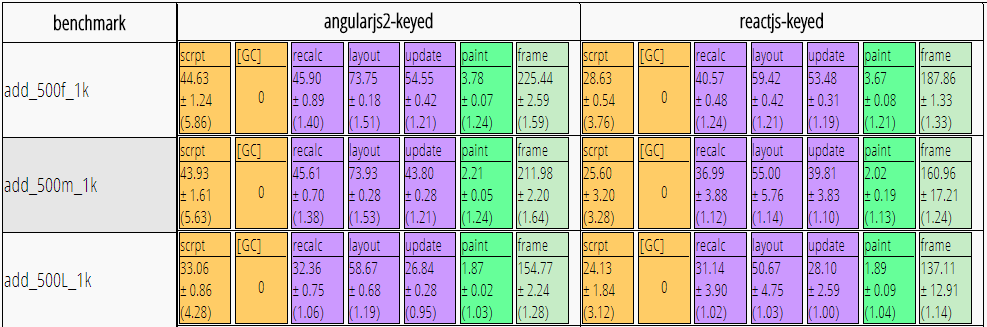
dodawanie przy elementach istniejacych:



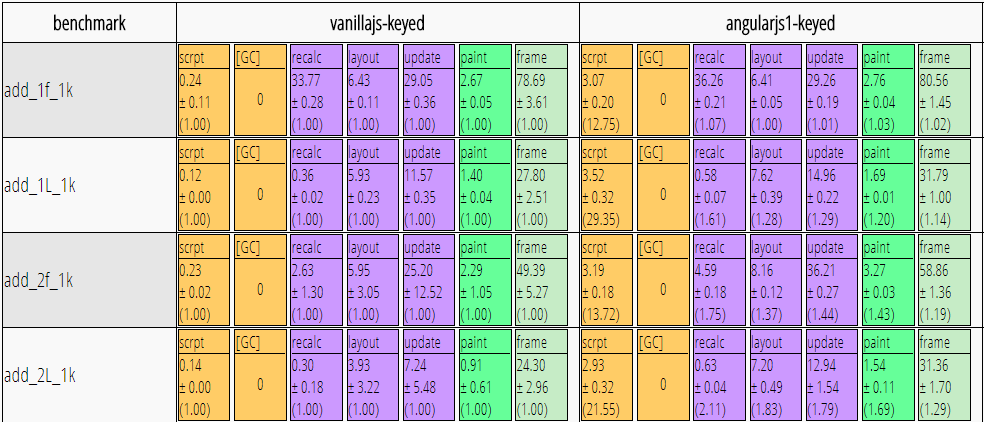


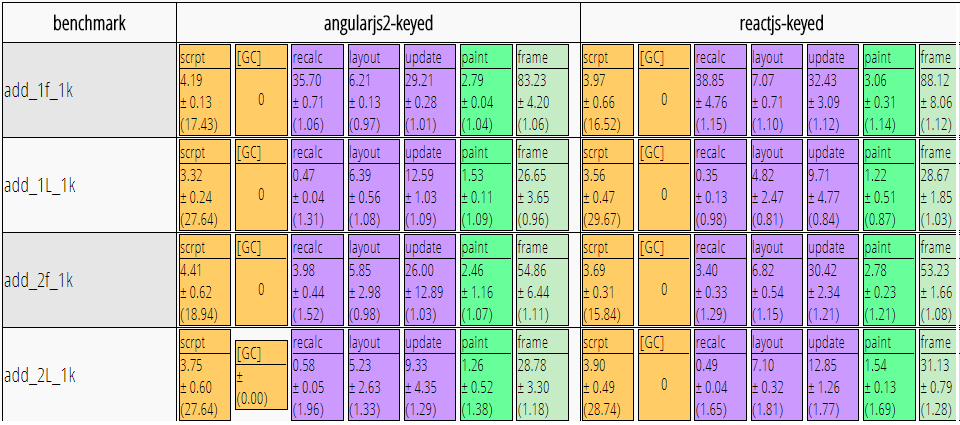
dodawanie - sprawdzanie metody:





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