Propozycja Tematu Pracy Magisterskiej

Jakub Kielaszek
Numer Indeksu: 180757

8 czerwca 2025

dr Tomasz Staś
WSB Merito Chorzów

Spis treści

1 Wprowadzenie

Niniejszy dokument przedstawia propozycję tematu pracy magisterskiej, której celem jest porównanie technologii tworzenia nowoczesnych aplikacji webowych na przykładzie aplikacji wizualizującej algorytmy sortujące. Praca skupi się na analizie wydajności, łatwości rozwoju i efektywności frameworków TypeScript React i Angular w kontekście budowy dynamicznych interfejsów użytkownika. Celem jest empiryczne wykazanie różnic i podobieństw między nimi poprzez stworzenie identycznych funkcjonalnie modułów wizualizacji sortowania, a następnie ich dogłębna analiza pod kątem optymalizacji renderowania, zarządzania stanem i produktywności programistycznej. W dalszych sekcjach przedstawiono zarys tematu, cele, zakres pracy, planowane metody badawcze oraz przewidywane rezultaty.

2 Proponowany Temat Pracy Magisterskiej

Tytuł: Architektura i optymalizacja renderowania w aplikacjach webowych: Analiza React i Angular na przykładzie wizualizatora algorytmów sortowania.

2.1 Uzasadnienie Wyboru Tematu

Wybór tego tematu jest podyktowany nieustannym rozwojem technologii webowych i dynamiczną ewolucją frameworków JavaScript, które stanowią fundament współczesnych aplikacji internetowych. Rosnące zapotrzebowanie na wysoce interaktywne i wydajne aplikacje frontendowe, takie jak wizualizatory danych czy złożone dashboardy, sprawia, że optymalny wybór technologii jest kluczowy dla sukcesu projektu. Temat ten jest szczególnie istotny ze względu na potencjalne zastosowania praktyczne wyników badań dla programistów i architektów systemów, którzy stoją przed decyzją o wyborze frameworka dla nowych projektów. Dodatkowo, analiza porównawcza dwóch wiodących, lecz fundamentalnie różnych pod względem architektury, technologii jak React i Angular, pozwoli na pogłębienie zrozumienia ich mocnych i słabych stron w kontekście specyficznych wymagań wydajnościowych dynamicznych wizualizacji algorytmicznych, co ma bezpośredni wpływ na teorię i praktykę inżynierii oprogramowania.

2.2 Problematyka Badawcza

Główny problem badawczy, na który praca ma odpowiedzieć, to: W jaki sposób architektura i specyfika frameworków React oraz Angular (implementowanych w TypeScript) wpływają na wydajność, łatwość rozwoju i produktywność programistyczną w kontekście budowy dynamicznych aplikacji webowych do wizualizacji algorytmów sortujących? Dodatkowo, praca będzie dążyć do rozwiązania następujących problemów szczegółowych:

- Wydajność renderowania: Który z frameworków (React czy Angular) wykazuje lepszą wydajność w kontekście częstych, dynamicznych aktualizacji interfejsu użytkownika, typowych dla wizualizacji algorytmów sortujących, oraz jakie techniki optymalizacyjne są kluczowe w każdym z nich?
- Zarządzanie stanem i złożoność kodu: Jakie są różnice w podejściu do zarządzania stanem aplikacji w React (np. Hooks, Context API, Redux) i Angular (np. serwisy, RxJS, NgRx) w kontekście utrzymania czystości i skalowalności kodu aplikacji wizualizującej algorytmy, oraz który model przekłada się na mniejszą złożoność implementacji?
- Krzywa uczenia i produktywność developerska: Jakie są różnice w łatwości rozpoczęcia pracy i szybkości tworzenia funkcjonalności w React i Angular dla programisty posiadającego podstawową znajomość języka TypeScript, oraz jakie narzędzia deweloperskie w każdym ekosystemie (np. CLI, DevTools, system typów TypeScripta) wpływają na ogólną produktywność?
- Rozmiar i złożoność bundle: Jakie są różnice w rozmiarze wynikowego bundle aplikacji oraz złożoności zależności w projektach zaimplementowanych w React i Angular, i jaki ma to wpływ na czas ładowania aplikacji?

3 Zakres Pracy

Praca obejmie następujące zagadnienia:

- Teoretyczne podstawy frameworków React i Angular: Dogłębne omówienie architektury, kluczowych koncepcji (np. komponenty, zarządzanie stanem, cykl życia, reaktywność), filozofii działania obu technologii oraz ich integracji z TypeScriptem.
- Projekt i implementacja modułów wizualizujących algorytmy sortujące: Stworzenie dwóch identycznych funkcjonalnie aplikacji (lub modułów w ramach jednej, unifikującej struktury projektu), z których jedna zostanie zaimplementowana w React

(z TypeScriptem), a druga w Angular (z TypeScriptem). Aplikacje te będą wizualizować co najmniej trzy różne algorytmy sortujące (np. Quick Sort, Merge Sort, Bubble Sort), pozwalając na interakcję z danymi (zmiana liczby elementów, prędkości animacji, typu danych).

- Porównawcza analiza wydajności: Przeprowadzenie pomiarów wydajności renderowania (np. czasu aktualizacji DOM, liczby renderów/zmian komponentów) w obu implementacjach przy użyciu narzędzi deweloperskich przeglądarki (np. Chrome DevTools Performance tab, React Profiler) oraz potencjalnie zewnętrznych bibliotek do benchmarkingu. Analiza wpływu optymalizacji specyficznych dla danego frameworka (np. memoizacja w React, strategii detekcji zmian w Angular).

- Ocena łatwości rozwoju i produktywności programistycznej: Subiektywna i obiektywna ocena doświadczeń deweloperskich podczas implementacji, uwzględniająca krzywą uczenia, wsparcie narzędzi (np. CLI, środowisko typowania TypeScripta, jakość dokumentacji), łatwość zarządzania stanem i refaktoryzacji kodu.

4 Planowane Metody Badawcze

Do realizacji celów pracy zostaną zastosowane następujące metody badawcze:

- **Analiza literatury:** Przegląd i synteza istniejących publikacji naukowych, artykułów branżowych, oficjalnej dokumentacji frameworków (React, Angular, TypeScript), standardów EcmaScript oraz źródeł związanych z algorytmami sortującymi i optymalizacją wydajności w aplikacjach webowych. Celem będzie zgromadzenie wiedzy teoretycznej niezbędnej do zrozumienia działania analizowanych technologii oraz podstaw budowy wydajnych interfejsów użytkownika. text

- **Badania empiryczne (projekt i implementacja):**

  - *Metoda studium przypadku (Case Study):* Przeprowadzenie szczegółowej analizy porównawczej na konkretnym przykładzie aplikacji (wizualizatora algorytmów sortujących), zaimplementowanej równolegle w dwóch różnych środowiskach technologicznych (React/TypeScript i Angular/TypeScript). Pozwoli to na praktyczną ocenę różnic i podobieństw między frameworkami w kontrolowanych warunkach.

  - *Eksperyment / Testy wydajnościowe (Benchmarking):* Przeprowadzenie serii pomiarów wydajności dla obu implementacji aplikacji wizualizującej. Obejmować to będzie m.in. pomiar czasu renderowania komponentów, zużycia zasobów procesora i pamięci, oraz płynności animacji podczas wykonywania algorytmów

na różnych zestawach danych i przy różnych obciążeniach. Testy będą powtarzane, aby zapewnić miarodajność wyników.

- *Analiza ilościowa:* Statystyczna analiza zebranych danych dotyczących wydajności (czasów renderowania, liczby operacji DOM, rozmiaru bundle). Porównanie średnich, odchyleń standardowych oraz wykorzystanie podstawowych technik statystycznych do weryfikacji hipotez dotyczących wydajności.
- *Analiza jakościowa:* Ocena produktywności deweloperskiej i łatwości rozwoju, oparta na subiektywnej obserwacji i wnioskach z procesu implementacji w obu technologiach. Uwzględniona zostanie krzywa uczenia, intuicyjność API, jakość dokumentacji, wsparcie narzędzi deweloperskich i łatwość debugowania.

5 Plan Struktury Pracy

Przewidywana struktura pracy magisterskiej przedstawia się następująco:

1. **Wstęp:** Wprowadzenie do tematu, uzasadnienie wyboru tematu, cel i zakres pracy, problematyka badawcza (pytania główne i szczegółowe) oraz ogólna struktura pracy.
2. **Rozdział 1: Podstawy teoretyczne i przegląd technologii frontendowych**
   - 1.1. Ewolucja i znaczenie nowoczesnych aplikacji webowych
   - 1.2. Przegląd paradygmatów i architektury frameworków JavaScript
   - 1.3. Charakterystyka React
     - 1.3.1. Architektura i kluczowe koncepcje (komponenty, Virtual DOM, Hooks)
     - 1.3.2. Zarządzanie stanem w React (np. useState, useContext, Redux)
     - 1.3.3. Optymalizacja wydajności w aplikacjach React
     - 1.3.4. Integracja React z TypeScript
   - 1.4. Charakterystyka Angular
     - 1.4.1. Architektura i kluczowe koncepcje (moduły, komponenty, dyrektywy, wstrzykiwanie zależności)
     - 1.4.2. Zarządzanie stanem w Angular (np. serwisy, RxJS, NgRx)
     - 1.4.3. Optymalizacja wydajności w aplikacjach Angular (Change Detection)
     - 1.4.4. Angular i TypeScript – natywna integracja
   - 1.5. Podstawy algorytmów sortujących i ich wizualizacji

  - 1.5.1. Klasyfikacja i charakterystyka wybranych algorytmów sortujących (np. Quick Sort, Merge Sort, Bubble Sort)
  - 1.5.2. Metody wizualizacji algorytmów (np. animacje, reprezentacje graficzne)

3. **Rozdział 2: Metodyka badań i implementacja aplikacji**
   - 2.1. Założenia projektowe i wymagania funkcjonalne dla wizualizatora algorytmów
   - 2.2. Opis implementacji aplikacji wizualizującej w React (z TypeScript)
     - 2.2.1. Struktura projektu i kluczowe komponenty
     - 2.2.2. Wykorzystane biblioteki i narzędzia (np. do rysowania wizualizacji)
     - 2.2.3. Podejście do zarządzania stanem
     - 2.2.4. Specyficzne techniki optymalizacyjne
   - 2.3. Opis implementacji aplikacji wizualizującej w Angular (z TypeScript)
     - 2.3.1. Struktura projektu i moduły
     - 2.3.2. Wykorzystane biblioteki i narzędzia
     - 2.3.3. Podejście do zarządzania stanem
     - 2.3.4. Specyficzne techniki optymalizacyjne
   - 2.4. Metody i procedury pomiarowe dla analizy wydajności (benchmarking)
   - 2.5. Metody oceny produktywności programistycznej
4. **Rozdział 3: Prezentacja i analiza wyników badań**
   - 3.1. Wyniki pomiarów wydajności renderowania dla obu implementacji
   - 3.2. Analiza zużycia zasobów i płynności animacji
   - 3.3. Porównanie rozmiarów i złożoności pakietów (bundle analysis)
   - 3.4. Wyniki oceny łatwości rozwoju i produktywności programistycznej
   - 3.5. Porównawcza analiza zastosowania TypeScript w obu frameworkach
5. **Rozdział 4: Dyskusja wyników, wnioski i rekomendacje**
   - 4.1. Odpowiedzi na główne i szczegółowe pytania badawcze
   - 4.2. Porównanie mocnych i słabych stron React i Angular w kontekście dynamicznych wizualizacji
   - 4.3. Implikacje praktyczne dla deweloperów i architektów

- 4.4. Implikacje teoretyczne dla inżynierii oprogramowania
- 4.5. Ograniczenia przeprowadzonego badania

6. **Zakończenie:** Podsumowanie najważniejszych ustaleń, rekapitulacja odpowiedzi na problemy badawcze, wskazanie kierunków dalszych badań i rozwoju tematu.
7. Bibliografia
8. Spis rysunków i tabel
9. Załączniki (np. fragmenty kodu źródłowego, wyniki surowych pomiarów)