AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,
INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

KATEDRA INFORMATYKI STOSOWANEJ

Praca dyplomowa inżynierska

Porównanie budowania i rozwoju aplikacji WWW w języku Elm i technologiach React+Redux

Comparision of building and development of web application in Elm language and React+Redux technologies

Autor: *Kamil Osuch*
Kierunek studiów: *Informatyka*
Opiekun pracy: *dr inż. Piotr Matyasik*

Kraków, 2017

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 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.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystycznego wykonania albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.", a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.): „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej «sądem koleżeńskim».", oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i że nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

Serdecznie dziękuję . . . tu ciąg dalszych podziękowań np. dla promotora, żony, sąsiada itp.

Spis treści

1. Wprowadzenie

W ciągu ostatnich kilku lat sposób tworzenia stron internetowych przechodził intensywne i szybkie zmiany. W związku z rosnącą popularnością internetu na całym świecie okazało się, że zwykłe, proste strony internetowe nie są wystarczające. W związku z tym szybko one awansowały ze statycznych dokumentów hipertekstowych jedynie wyświetlających zawartość użytkownikowi, do poziomu aplikacji internetowych, z którymi może on wejść w interakcję, a nawet używać ich dokładnie tak samo, jak programów zainstalowanych w swoim systemie operacyjnym.

Rynek tworzenia oprogramowania webowego został przejęty głównie przez język JavaScript nazywany dziś przez niektórych asemblerem stron internetowych [1]. Coraz większe wymagania co do działania stron internetowych spowodowały, że tworzenie stron bezpośrednio w JavaScript'cie stało się zbyt skomplikowane i niewystarczające. Z pomocą przychodzą biblioteki, framework'i oraz języki kompilowane do JavaScript'a. Upraszczają one tworzony kod, często zmieniając też jego strukturę. Dzięki temu umożliwiają korzystanie z gotowych funkcjonalności w prosty i wygodny sposób. Przykładami tutaj mogą być biblioteki takie jak jQuery, Angular, React, Redux, czy Ember.js.

W przypadku języków kompilowanych do JavaScript'a zmiany są jeszcze większe. Programista nie zastanawia się nad tym, że ostateczna wersja stworzonego przez niego programu jest zapisana w zupełnie innym języku. Przykładami takich języków są chociażby CoffeeScript, TypeScript czy Elm.

Ilość takich rozwiązań tworzy nieograniczoną liczbę podejść do tworzenia aplikacji webowych, a ich liczba każdego dnia rośnie. W związku z tym powstaje wiele artykułów porównujących różne podejścia, zarówno pod względami architektury kodu, jak i szybkości działania samych aplikacji.

Celem niniejszej pracy jest porównanie budowania i rozwoju aplikacji webowych przy pomocy języka Elm, oraz kombinacji bibliotek React.js i Redux. Technologie te zostaną szczegółowo porównane pod względem dostępnych funkcjonalności, szybkości działania aplikacji, dostępności bibliotek oraz trudności zarówno w tworzeniu pierwszej wersji aplikacji, jak i rozwoju istniejącego już kodu.

Wybór tematu pracy był podyktowany przede wszystkim poszukiwaniem alternatywnych rozwiązań służących do tworzenia aplikacji webowych. Biblioteki takie jak React czy Redux zdominowały rynek, przez co ilość wykorzystywanych sposobów budowania stron internetowych w stosunku do ilości dostępnych możliwości jest bardzo mała.

Ważną częścią pracy jest implementacja dwóch wersji tej samej aplikacji webowej, stworzonych przy pomocy opisywanych w pracy rozwiązań. Pozwoli to na dokładniejszą analizę i porównanie obu podejść, w oparciu o praktyczny przykład.

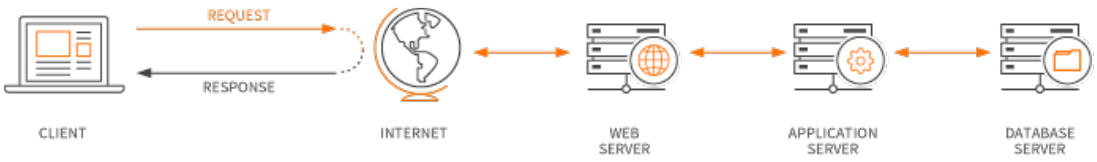
W rozdziale 2 zostały krótko opisane charakterystyka języka JavaScript oraz podstawowe pojęcia związane z tworzeniem aplikacji webowych. Następnie krótko zostały opisane biblioteki React i Redux oraz język Elm. Rozdział **??** zawiera krótkie podsumowanie przeprowadzonej analizy porównawczej. Opisuje on wnioski oraz możliwości rozwoju pracy.

2. Podstawy teoretyczne

W tym rozdziale zostaną omówione podstawowe pojęcia związane z tematem pracy. Pierwsze dwa podrozdziały skupiają się na opisie czym w ogóle jest aplikacja internetowa, oraz opisują jedno z najpopularniejszych podejść do budowania aplikacji internetowej, SPA. W kolejnych podrozdziałach zostały opisane technologie, które są porównywane w dalszej części pracy.

2.1. Aplikacja internetowa

Aplikacja internetowa jest czymś więcej niż zwykłą stroną internetową. Z definicji jest to aplikacja typu klient-serwer, w której klient jest uruchamiany przy pomocy przeglądarki internetowej. Oprogramowanie klienta jest pobierane na komputer klienta podczas wizyty na odpowiedniej stronie internetowej, przy użyciu standardowych procedur, takich jak HTTP. Aktualizacje oprogramowania klienta mogą odbywać się za każdym razem, gdy odwiedzana jest strona internetowa. W czasie trwania sesji przeglądarka internetowa interpretuje i wyświetla strony, oraz działa jako uniwersalny klient dla dowolnej aplikacji internetowej.

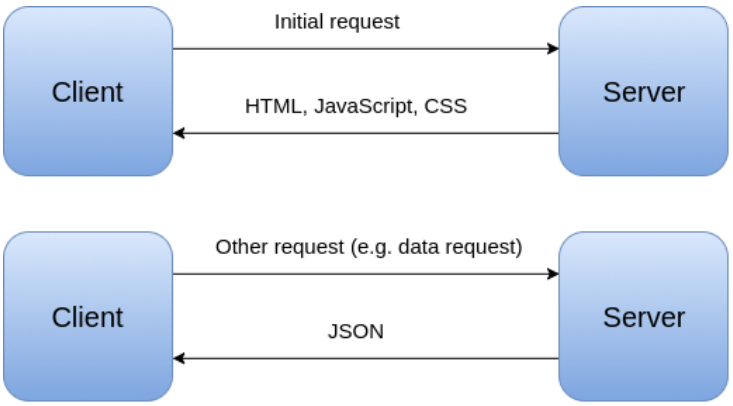


Rysunek 2.1. Podstawowy schemat działania aplikacji internetowej [2]

2.2. SPA

Skrót SPA pochodzi od *Single-page Application*. Jest to aplikacja internetowa działająca wewnątrz przeglądarki, która podczas użytkowania strony nie wymaga odświeżania strony. W tym podejściu, cały niezbędny kod źródłowy – HTML, JavaScript i CSS – jest pobierany przy pojedyńczym załadowaniu strony, lub odpowiednie zasoby są ładowane dynamicznie i dodawane do strony jedynie wtedy, kiedy jest to potrzebne. Idea jaka stoi za takim rozwiązaniem, to przede wszystkim lepszy, bardziej naturalny user

expierience. Użytkownik po wejściu na stronę, nie musi przy każdej interakcji oczekiwać na ponowne załadowanie się strony, czy też jej odświeżanie.



Rysunek 2.2. Podstawowy schemat działania aplikacji internetowej [2]

Choć koncept SPA zaczął być częściej używany po spopularyzowaniu AJAX'a[1], to tak naprawdę dopiero od kilku lat jest on powszechnie wykorzystywany podczas budowania aplikacji internetowych. Serwisy takie jak Gmail, Google Maps, Facebook czy GitHub są właśnie aplikacjami typu single-page application. Także większość najpopularniejszych bibliotek JavaScript'owych umożliwiają implementację aplikacji internetowej zgodnie z zasadami SPA.

2.2.1. Zalety

– Szybkość działania - większość zasobów jest ładowana tylko raz podczas cyklu życia aplikacji, jedynymi informacjami, które są wymieniane z serwerem cały czas, są dane

– Brak ciągłego przeładowywania strony

– Znacznie prostszy proces wdrożenia aplikacji - jedyne co jest potrzebne to statyczny serwer serwujący minimalnie 3 pliki - pojedynczą stronę HTML, oraz 2 pakiety: jeden zawierający wszystkie style, drugi skupiający w sobie cały kod JavaScriptow'y

– Odciążenie strony serwerowej - serwer zamiast generować za każdym razem pełny kod strony, transmituje jedynie potrzebne w danej chwili dane.

2.2.2. Wady

– Powolne początkowe uruchomienie strony - wymaga ono załadowania framework'u, oraz przynajmniej części aplikacji, która później już nie jest ponownie ściągana.

– Ze względu na zależność SPA od JavaScript'a, bardzo łatwo o pojawienie się wycieków pamięci pomiędzy długimi okresami czasu między przeładowaniami strony

[1] **Asynchronous JavaScript And XML**.

–

2.3. JavaScript

2.4. React.js

2.5. Redux

2.6. Elm

Bibliografia

[1] Scott Hanselman. *JavaScript is Assembly Language for the Web: Sematic Markup is Dead! Clean vs. Machine-coded HTML*. Dostęp: 07-12-2017. URL: *https://www.hanselman.com/blog/JavaScriptIsAssemblyLanguageForTheWebSematicMarkupIsDeadCleanVsMachinecodedHTML.aspx*.

[2] Amos Ndegwa. *What is a Web Application?* Dostęp: 13-12-2017. URL: *https://www.maxcdn.com/one/visual-glossary/web-application/*.