Politechnika Gdańska

Wydział Elektroniki,
Telekomunikacji i Informatyki

Katedra: Algorytmów i Modelowania Systemów

Imię i nazwisko dyplomanta: Marcin Jurczak

Nr albumu: 171641

Forma i poziom studiów: Stacjonarne jednolite studia magisterskie

Kierunek studiów: Informatyka

Specjalność: Algorytmów i Technologii Internetowych

Praca dyplomowa magisterska

Temat pracy:

Wykorzystanie języka Elm do tworzenia aplikacji frontendowych.

Title of thesis:

Programming the front-end applications with Elm language.

Opiekun pracy:

dr inż. Krzysztof Manuszewski

Data ostatecznego zatwierdzenia raportu podobieństw w JSA: //TODO

Gdańsk, 2022

Streszczenie

Celem niniejszej pracy magisterskiej było stworzenie front-end'owej aplikacji internetowej z wykorzystaniem funkcyjnego języka Elm, porównanie tejże technologii z istniejącymi, bardziej powszechnymi rozwiązaniami tego typu, a także przygotowanie instrukcji laboratoryjnej, która mogłaby zostać wykorzystana w ramach zajęć *Współczesne Aplikacje Programowania Funkcyjnego* przeprowadzanych na Wydziale Elektroniki, Telekomunikacji i Informatyki Politechniki Gdańskiej. Wytworzona aplikacja to strona internetowa typu *startpage*, czyli startowa strona przeglądarki, zawierająca najpotrzebniejsze informacje, takie jak czas, pogoda oraz odnośniki do wyszukiwarki i najczęściej odwiedzanych stron.

Słowa kluczowe: Elm, programowanie funkcyjne, wytwarzane aplikacji internetowych

Dziedzina nauki i techniki: Nauki inżynieryjne i techniczne, inżynieria informatyczna.

Abstract

The goal of this master thesis is to use the Elm language to create a front-end web application, comparing this technology to existing, more popular solutions, as well as preparing a lab instruction, which could be used at *Modern applications of functional programming* class at Gdańsk University of Technology's Faculty of Electronics, Telecommunications and Informatics. The created application is a *startpage*, meaning a starting page of a web browser consisting of the most useful information, such as time, weather and references to search engine and the most visited websites.

Keywords: Elm, functional programming, web development

Field of Science and Technology: Engineering and Technology, Information engineering.

Spis treści

Wykaz najważniejszych skrótów

CD – ang. Continuous Deployment, pol. ciągłe wdrażanie.

CI – ang. Continuous Integration, pol. ciągła integracja.

CSS – ang. Cascading Style Sheets, pol. kaskadowe arkusze stylów

HTML – ang. Hypertext Markup Language, pol. hipertekstowy język znaczników.

HTTP – ang. Hypertext Transfer Protocol, pol. protokół przesyłania hipertekstu.

JSON – ang. JavaScript Object Notation, pol. tekstowy format zapisu danych

Rozdział 1

Wstęp i cel pracy

Głównym celem niniejszej pracy jest zapoznanie się z funkcyjnym językiem programowania Elm oraz stworzenie przykładowej frontendowej aplikacji internetowej. Ponadto chciałbym przeprowadzić porównanie tej technologii z innymi, bardziej powszechnie używanymi rozwiązaniami do tworzenia aplikacji internetowych. Ostatnim celem pracy jest przygotowanie części dydaktycznej w postaci instrukcji laboratoryjnej, która mogłaby zostać potencjalnie wykorzystana w ramach przedmiotu Współczesne Aplikacje Programowania Funkcyjnego, prowadzonego przez mojego promotora, dra inż. Krzysztofa Manuszewskiego.

W drugim rozdziale skupiam się na przedstawieniu technologii powszechnie używanych do tworzenia frontendowych aplikacji internetowych, t.j. React, Angular oraz Vue. Prezentuję ich zalety i wady względem siebie, a także przedstawiam cechy wyróżniające je między sobą.

Trzeci rozdział poświęcam na wysokopoziomowe wprowadzenie do języka Elm. Mówię o idei jaka przyświęcała stworzeniu tego języka, jakie są jego potencjalne zastosowania, gdzie sprawdza się najlepiej oraz przedstawiam narzędzia wspomagające tworzenie oprogramowania z użyciem tejże technologii.

W czwartm rozdziale przedstawiam implementację przygotowanej aplikacji frontendowej napisanej w Elm'ie. Jest to poniekąd rozwinięcie poprzedniego rozdziału, ponieważ głównym celem jest dalej zapoznanie się z Elm'em, jednak tutaj uwagę skupiam na przedstawieniu konkretnych rozwiązań, jakie zostały wykorzystane do osiągniecia wybranego celu.

Piąty rozdział zawiera instrukcję laboratoryjną, w której przeprowadzam czytelnika nieposiadającego żadnego doświadczenia z Elm'em przez proces tworzenia oprogramowania z wykorzystaniem tej technologii, zaczynając od przygotowania środowiska, przez podstawy języka, aż po stworzenie aplikacji frontendowej przedstawionej we wcześniejszym rozdziale.

Szósty rozdział poświęcam na omówienie zagadnień związanych z ciągłą integracją oraz ciągłym wdrażaniem, a także przedstawiam narzędzia wykorzystywane przeze mnie w tym celu podczas tworzenia omawianej aplikacji. Pokazuję, że Elm nie stanowi przeszkody w wykorzystywaniu tych technologii i całkowicie nadaje się do użytku produkcyjnego.

Ostatni rozdział dotyczy przede wszystkim podsumowania niniejszej pracy magisterskiej. Przedstawiam produkt dwóch semestrów moich działań oraz wyciągam wnioski na temat Elm'a jako języka przeznaczonego do tworzenia aplikacji frontendowych.

Na końcu dokumentu znajdują się spisy użytych grafik i listingów, jak i bibliografia.

Rozdział 2

Powszechne rozwiązania

2.1 React

[1] [2]

2.2 Angular

[3] [4]

2.3 Vue.js

[5]

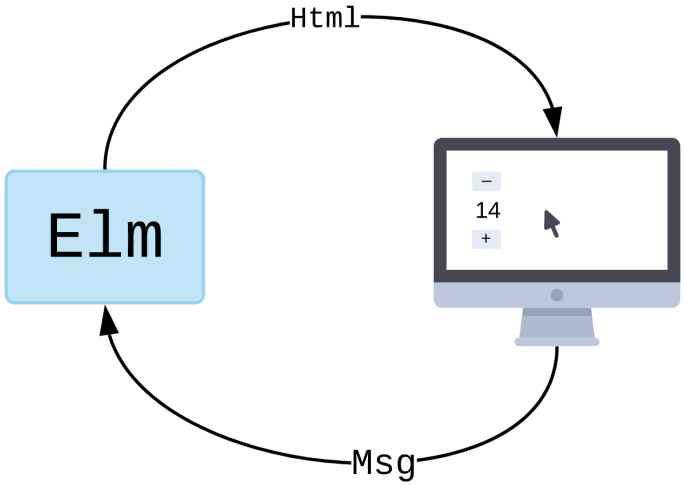
2.4 Podobieństwa i różnice

Rozdział 3

Elm

Elm[6] jest czysto funkcyjnym językiem programowania przeznaczonym do tworzenia graficznych interfejsów użytkownika. Powstał w roku 2012 wraz z opublikowaniem przez Evana Czaplickiego pracy „Elm: Concurrent FRP for Functional GUIs”[7]. Podczas jego tworzenia nacisk został położony na użyteczność, wydajność oraz niską podatność na błędy.

3.1 The Elm Architecture



Rysunek 3.1: Diagram działania programu w Elm’ie

The Elm Architecture jest schematem tworzenia interaktywnych aplikacji internetowych lub gier. Zgodnie z rysunkiem 3.1 typowa aplikacja Elm działa w następujący sposób: Program generuje pewien kod HTML, który zostaje wyświetlony na ekranie, a następnie komputer zwraca wiadomości

informujące o tym co się dzieje, np. użytkownik wcisnął guzik.

A co się dzieje wewnątrz wspomnianego programu Elm'owego? Zawsze składa się z trzech podstawowych elementów:

- Model – opisujący stan aplikacji
- Update – opisujący logikę aplikacji
- View – opisujący wygląd aplikacji

W kolejnych podrozdziałach przedstawiam powyższe elementy architektury Elm'a na podstawie prostego programu, którego zadaniem jest wyświetlenie na ekranie dwóch guzików oraz licznika, który może się zwiększać i zmniejszać, w zależności od tego, który guzik zostanie naciśnięty przez użytkownika.

3.1.1 Model

Celem modelu jest zdefiniowanie danych w naszej aplikacji. W tym przypadku model będzie bardzo prosty – jedna wartość całkowitoliczbowa, która będzie mogła zostać zwiększona lub zmniejszona.

Listing 3.1: *The Elm Architecture* - Model

```
type alias Model = Int


init : Model
init =
  0
```

3.1.2 Update

Listing 3.2: *The Elm Architecture* - Update

```
type Msg
  = Increment
  | Decrement


update : Msg -> Model -> Model
update msg model =
  case msg of
    Increment ->
      model + 1

    Decrement ->
      model - 1
```

Funkcja `update` ma za zadanie opisywać jak nasz model będzie się zmieniał w czasie. Może odebrać dwa typy wiadomości – *Increment* i *Decrement*. W wyniku operacji `update` otrzymujemy nowy, zaktualizowany model.

3.1.3 View

Funkcja `view` jako argument przyjmuje model i zwraca kod HTML. Wykorzystany został tutaj handler `onClick`, który po kliknięciu generuje odpowiednią wiadomość. Znak plusa generuje wiadomość *Increment*, znak minusa *Decrement*. Wybrana wiadomość trafia do funkcji `update`.

Listing 3.3: *The Elm Architecture* - View

```
view : Model -> Html Msg
view model =
  div []
    [ button [ onClick Decrement ] [ text "-" ]
    , div [] [ text (String.fromInt model) ]
    , button [ onClick Increment ] [ text "+" ]
    ]
```

3.2 Narzędzia

Platforma Elm jest dostarczana wraz z zestawem narzędzi pozwalających m.in. na kompilację plików źródłowych czy instalację dodatkowych modułów. Poniżej postaram się opisać większość z tych narzędzi, tj. dostarczanych przez Elm'a, ale wskazać również te dostarczane przez zewnętrznych twórców, a które znacząco ułatwiły mi pracę z tym językiem.

`elm repl` – otwiera interaktywną sesję programistyczną.

`elm init` – inicjalizuje bieżący katalog jako nowy projekt Elm'a poprzez stworzenie pliku `elm.json` opisującego projekt i jego zależności, a także tworzy katalog `src/`, w którym będą znajdowały się pliki `.elm`.

`elm reactor` – uruchamia serwer deweloperski, który poprzez przeglądarkę pozwala wybrać dany plik źródłowy, skompilować go i sprawdzić jak wygląda po zbudowaniu.

`elm make` – pozwala na kompilację kodu źródłowego do HTML'a lub JavaScript'u. Jest to najbardziej ogólna forma kompilacji, jaką udostępnia Elm, ale jest to niezwykle przydatne narzędzie, kiedy projekt stanie się zbyt skomplikowany na korzystanie z `elm reactor`.

`elm install` – pozwala instalować paczki dostępne na stronie `package.elm-lang.org`, które udostępniają nowe funkcjonalności, jak np. obsługa plików JSON czy praca z zapytaniami HTTP.

`elm-live [8]` – test

`elm-format [9]` – test

Rozdział 4

Implementacja

W ramach części praktycznej niniejszej pracy stworzona została aplikacja internetowa typu *startpage*, czyli spersonalizowanej strony startowej przeglądarki zawierającej najpotrzebniejsze i najczęściej używane elementy oraz skróty. Na potrzeby aplikacji postanowiłem stworzyć cztery moduły:

- Cyfrowy zegar wskazujący aktualny czas w strefie czasowej użytkownika
- Pogoda w Gdańsku przedstawiona w formie krótkiego opisu oraz temperatury w stopniach Celsjusza
- Wyszukiwarka Google
- Zakładki zawierające odnośniki do wybranych stron internetowych

W poniższych podrozdziałach skupię się na opisie implementacji każdego modułu.

4.1 Zegar

4.2 Pogoda

4.3 Wyszukiwarka

4.4 Zakładki

4.5 Dokument hipertekstowy

Listing 4.1: Zawartość pliku `index.html`

```
<head>
    <link rel="stylesheet" href="assets/styles.css" />
    <script src="assets/main.js"></script>
    <script src="assets/bookmarks.js"></script>
</head>
<body>
    <div id="myapp"></div>
    <script>
        var app = Elm.Main.init({
            node: document.getElementById('myapp'),
            flags: bookmarks
        });
    </script>
</body>
```

4.6 Style

4.7 Całość

Rozdział 5

Instrukcja laboratoryjna

W poniższym rozdziale przedstawiam przykładową instrukcję laboratoryjną, która krok po kroku przeprowadza czytelnika przez proces tworzenia aplikacji w Elmie, zaczynając od przygotowania środowiska deweloperskiego, przez podstawy języka wraz z ćwiczeniami pozwalającymi na lepsze zrozumienie składni, aż po stworzenie większej aplikacji frontendowej.

5.1 Przygotowanie środowiska

Pierwszą rzeczą, którą należy się zająć przed rozpoczęciem nowego projektu jest przygotowanie odpowiedniego środowiska deweloperskiego. Należy upewnić się, że wszystkie narzędzia potrzebne do wykonania pracy są zainstalowane i prawidłowo skonfigurowane. W przypadku pracy z Elm'em będziemy korzystać przede wszystkim z kompilatora, edytora tekstu wspierającego podświetlanie składni, a także innych narzędzi wspomagających proces tworzenia oprogramowania z użyciem tej technologii.

5.1.1 Linux

Najprostszym sposobem instalacji platformy Elm na systemie operacyjnym Linux jest wykorzystanie narzędzia `npm` – powszechnie używanego menadżera pakietów służącego do zarządzania warstwą frontendową aplikacji internetowych. Aby zainstalować `npm` należy skorzystać z systemowego menadżera pakietów. Na przykładzie dystrybucji Ubuntu będą to komendy:

```
$ sudo apt update
$ sudo apt install npm
```

Kiedy narzędzie zostanie już pomyślnie zainstalowane, można przejść do instalacji platformy Elm. Posłuży do tego polecenie:

```
$ npm install -g elm
```

Zgodnie z dokumentacją `npm`[10], flaga `-g` oznacza, że pakiet zostanie zainstalowany globalnie, dzięki czemu będzie dostępny z każdego miejsca z systemu. Aby sprawdzić, czy rzeczywiście tak się stało, wystarczy uruchomić polecenie `elm -version`. Jeżeli w odpowiedzi otrzymamy prawidłową wersję kompilatora oznacza to, że platforma Elm jest gotowa do działania i można przejść do kolejnego kroku.

5.1.2 Windows

5.1.3 Edytor

5.1.4 Tworzenie projektu

5.2 Podstawy języka Elm

5.3 Aplikacja frontendowa

Rozdział 6

Automatyzacja

W poniższym rozdziale chciałbym opisać procesy *CI/CD* oraz korzyści płynące z używania ich, a także zaprezententować implementację takich rozwiązań na przykładzie stworzonej wcześniej aplikacji w Elm'ie.

6.1 CI/CD

6.2 GitHub Actions

6.3 GitHub Pages

Rozdział 7

Podsumowanie

7.1 Wnioski

Spis rysunków

Spis listingów

Bibliografia

[1] Alex Banks i Eve Porcello. *Learning React: functional web development with React and Redux*. "O'Reilly Media, Inc.", 2017.

[2] Jordan Walke. *React documentation*. 2022. URL: `https://angular.io/docs`.

[3] Ashok and Mehta Jain Nilesh and Bhansali. „AngularJS: A modern MVC framework in JavaScript". W: *Journal of Global Research in Computer Science* (2014).

[4] Google. *Angular documentation*. 2022. URL: `https://angular.io/docs`.

[5] Evan You. *Vue.js documentation*. 2022. URL: `https://vuejs.org/guide`.

[6] Evan Czaplicki. *Elm documentation*. 2022. URL: `https://elm-lang.org/docs`.

[7] Evan Czaplicki. „Elm : Concurrent FRP for Functional GUIs". W: *Elm : Concurrent FRP for Functional GUIs*. 2012.

[8] Will King. *elm-live documentation*. 2022. URL: `https://www.elm-live.com/`.

[9] Aaron VonderHaar. *elm-format code repository*. 2022. URL: `https://github.com/avh4/elm-format`.

[10] Isaac Z. Schlueter. *npm documentation*. URL: `https://docs.npmjs.com/`.