Politechnika Gdańska

**Wydział Elektroniki,
Telekomunikacji i Informatyki**

Katedra: Algorytmów i Technologii Internetowych

Imię i nazwisko dyplomanta: Marcin Jurczak

Nr albumu: 171641

Forma i poziom studiów: Stacjonarne jednolite studia magisterskie

Kierunek studiów: Informatyka

Praca dyplomowa magisterska

Temat pracy:
Wykorzystanie języka ELM do tworzenia aplikacji frontendowych.

Promotor:
dr inż. Krzysztof Manuszewski

Gdańsk, 2022

Spis treści

Rozdział 1

Wstęp i cel pracy

Celem niniejszej pracy jest zapoznanie się z funkcyjnym językiem programowania Elm, porównanie go z powszechnymi rozwiązaniami do tworzenia aplikacji frontendowych, a także przygotowanie instrukcji laboratoryjnej, która mogłaby zostać potencjalnie wykorzystana w ramach przedmiotu Współczesne Aplikacje Programowania Funkcyjnego, prowadzonego przez mojego promotora, dra inż. Krzysztofa Manuszewskiego.

W pierwszym rozdziale

W drugim rozdziale

W trzecim rozdziale

Rozdział 2

Aplikacje frontendowe

2.1 Powszechne rozwiązania

2.1.1 TypeScript

2.1.2 React

Rozdział 3

Programowanie funkcyjne

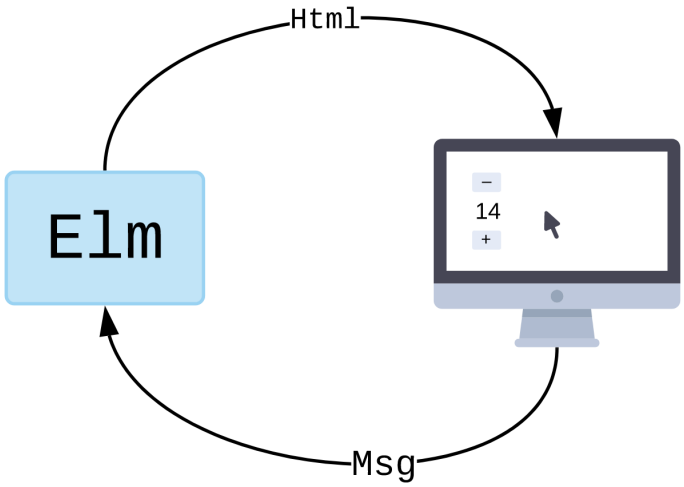
Rozdział 4

Elm

Elm jest czysto funkcyjnym językiem programowania przeznaczonym do tworzenia graficznych interfejsów użytkownika. Podczas jego tworzenia nacisk został położony na użyteczność, wydajność oraz niską podatność na błędy.

4.1 The Elm Architecture

The Elm Architecture jest schematem tworzenia interaktywnych aplikacji internetowych lub gier.



Rysunek 4.1: The Elm Architecture

Architektura Elma składa się z trzech podstawowych elementów:

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

4.1.1 Model

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

Listing 4.1: Model

```
type alias Model = Int


init : Model
init =
  0
```

4.1.2 Update

Listing 4.2: 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.

4.1.3 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 4.3: View

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

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 Instalacja środowiska

5.1.1 Linux

5.1.2 Windows

5.2 Podstawy języka Elm

5.3 Aplikacja frontendowa

Rozdział 6

Podsumowanie

Spis rysunków

Spis tabel

Spis algorytmów

Bibliografia