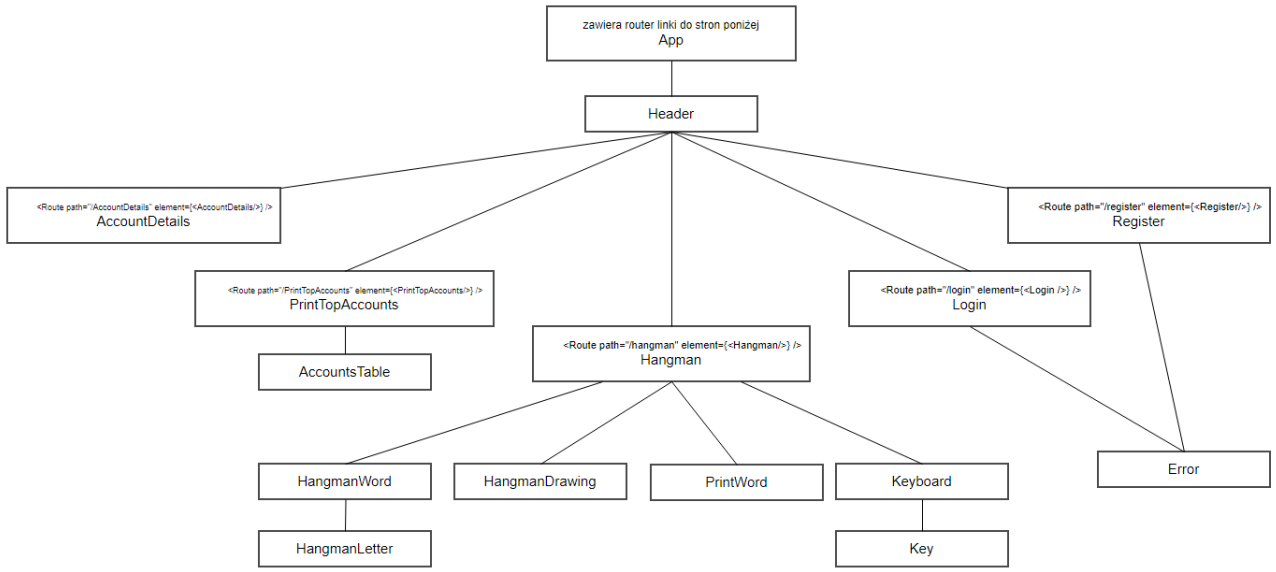
# Dokumentacja

## Temat: **Wisielec**

Skład zespołu:
Mateusz Rojek
Michał Olszewski
Dawid Piecek

### a) architektura komponentów, przekazywane właściwości i metody




### b) ścieżki i komponenty związane z routingiem

```
<Routes>
      <Route path="/" element={<Hangman />} />
      <Route path="/register" element={<Register />} />
      <Route path="/login" element={<Login />} />
      <Route path="/account/:id" element={<AccountDetails />} />
      <Route path="/top" element={<PrintTopAccounts />} />
 </Routes>
```

### d) API serwera

Zewnętrzne API, z którego korzystamy to: https://dictionaryapi.dev/ , zwraca ono informacje na temat danego słowa, używamy go do wyświetlenia definicji odgadniętego hasła.

## e) wybrane przez autorów, szczególnie ciekawe fragmenty kodu

- zrealizowanie routingu z parametrem poprzez wyświetlanie szczegółów kont:

zadeklarowanie routingu w App:

```
<Route path="/account/:id" element={<AccountDetails />} />
```

Przekierowanie na stronę konkretnego użytkownika po wciśnięciu przycisku podczas wyświetlania top 5 użytkowników:

```
<NavLink
  to={`/account/${account.id}`}
  style={{
    color: "yellow",
    margin: "0 20px",
    textDecoration: "none",
    marginLeft: "20px",
  }}
>
```

Pobranie id użytkownika, którego wybraliśmy i wyszukanie informacji interesującego nas gracz

```
const { id } = useParams();
const user = accounts.find((account) => account.id === Number(id));
```

- przykładowe zapytanie do naszego serwera o listę kont:

```
export const getAccounts = async () => {
  try {
    const { data } = await axios.get("http://localhost:8000/accounts");
    return data;
  } catch (err) {
    console.error(err);
  }
};
```

- przykład reużywalnego komponentu do którego możemy przekazać słowo i jego kolor, następnie wywołanie go w komponencie Hangman, gdzie ustawiony jest odpowiedni komunikat:

```
export function PrintWord({ message, messageColor }: Prop_Message) {
  return (
    <div>
      <div style={{ color: messageColor }}>{message}</div>
    </div>
  );
}
export default PrintWord;
```

```
<PrintWord message={message} messageColor={messageColor} />
```

## f) wypunktowane elementy techniczne, które zostały zrealizowane w projekcie wraz z krótkim komentarzem odnośnie realizacji: jak zrealizowano i w którym pliku

Punktacja elementów technicznych (18pkt):

- własna walidacja danych wprowadzanych przez użytkownika ( w każdym przypadku wprowadzania danych, co najmniej 4 różne przypadki danych) - 2pkt,

  zrealizowane w komponentach **Login.tsx**, **Register.tsx**,

- obowiązkowa weryfikacja typu danych (PropTypes) przekazywanych do wszystkich komponentów (nie stosujemy typu 'any') - 2pkt

  zrealizowane we wszystkich komponentach,

- właściwe użycie typów komponentów (czy każdy z komponentów jest właściwie odwzorowany na komponent prezentacyjny lub stanowy) - 1pkt,

  zrealizowane we wszystkich komponentach,

- co najmniej 4 komponenty reużywalne (komponenty, które mogą być użyte bez zmian w innym miejscu projektu) - 2pkt

  zrealizowane w komponentach **Error.tsx**, **Key.tsx**, **PrintWord.tsx, HangmanLetter.tsx**, komponenty te są uniwersalne i można ich użyć w innych miejscach lub w innym projekcie.

- operacje modyfikacji danych za pomocą 4 rodzajów żądań http - 1pkt

  zrealizowane w komponencie **Services.tsx**, wykorzystane zostały zapytania get, post, put, delete.

- żądania do serwera są zapisane w jednym oddzielnym pliku - 1pkt

  zrealizowane w komponencie **Services.tsx**, w tym pliku znajdują się wszystkie żądania.

- routing (ścieżki 'routes', w tym jedna z parametrem) - 1pkt

  zrealizowane w komponencie **App.tsx**, ścieżka route z parametrem użyta jest podczas wyświetlania szczegółów kont.

- wykorzystanie dwóch zmiennych właściwości routingu (np. navigate, params) - 1pkt

  Zrealizowane m.in w komponencie **Login.tsx**, **Register.tsx**, **AccountDetails.tsx**. Navigate użyte jest np. po wciśnięciu przycisków do przekierowywania na kolejne strony, useParams użyte jest do pobrania id wyświetlanych informacji konkretnego użytkownika.

- brak błędów/ostrzeżeń w konsoli przeglądarki - 1pkt

## g) Dodatkowe biblioteki użyte w aplikacji: link oraz zdanie opisu biblioteki i celu użycia.

Json-server - biblioteka dzięki której zaimplementowaliśmy komunikacje z serwerem, służy nam jako prosta baza danych.
https://www.npmjs.com/package/json-server

react-router-dom - Biblioteka wspomagająca routing w aplikacjach react
https://www.npmjs.com/package/react-router-dom

## h) Podział pracy w zespole

Mateusz Rojek: Login, Register, Header, ReusableComponents
Walidacja, stylowanie
Michał Olszewski: Hangman, HangmanDrawing, Keyboard, HangmanWord
Routing, stylowanie
Dawid Piecek: AccountDetails, AccountTable, PrintTopAccounts
Komunikacja z serwerem