Uniwersytet Zielonogórski

Wydział Informatyki, Elektrotechniki i Automatyki

Praca dyplomowa

Kierunek: Informatyka

Aplikacja internetowa umożliwiająca konwersję wyników meczów żużlowych

Kacper Adamiak

Promotor:
Dr hab. inż. Remigiusz Wiśniewski, prof. UZ

Pracę akceptuję:

.....
(data i podpis promotora)

Zielona Góra, Styczeń 2024

Streszczenie

W pracy dyplomowej zaprezentowano proces tworzenia konwertera żużlowego w formie aplikacji internetowej. Opisane zostały technologie oraz narzędzia wykorzystane do stworzenia aplikacji. Omówiony został interfejs aplikacji oraz struktura plików. Została przedstawiona analiza działania skryptu konwertującego, oraz wszystkie wykonane działania na dostarczonych danych.

Słowa kluczowe: praca inżynierska, aplikacja internetowa, konwerter żużlowy, parsowanie danych.

Spis treści

Rozdział 1

Wstęp

1.1. Wprowadzenie

Żużel jest jednym z najbardziej popularnych sportów w Polsce. Historia żużla w Polsce sięga czasów przedwojennych. Serwisy takie jak strona speedwayw.pl[1] przechowują dane nawet najstarszych żużlowych starć. Strona bogata jest w zawartość bezcenną dla ludzi zafascynowanych żużlem, zawiera dane w formacie, które mogą być nieczytelne dla użytkownika na pierwszy rzut oka. Właśnie z tego powodu powstała aplikacja internetowa, konwertująca wyniki meczów. Dzięki konwertowaniu danych z formatu tabelarycznego do formatu biegowego dane stają się bardziej przejrzyste i łatwiejsze do interpretacji.

W ninejszej pracy przedstawiona została aplikacja internetowa, rozwiązująca problem nieczytelnych danych meczów żużlowych. Dokonano prezentacji technologii oraz narzędzi projektowych wykorzystanych do stworzenia konwertera. Przeprowadzony został przegląd interfejsu użytkownika oraz analiza działania części odpowiadającej za konwersję danych. Efektem jest prosta do użytkowania apikacja, spełniająca swoje główne założenie-przekształcenie formatu danych.

1.2. Cel i zakres pracy

Celem pracy inżynierskiej jest stworzenie aplikacji internetowej umożliwiającej konwersję wyników meczów żużlowych. Konwertowanie oparte jest o dane ze strony speedwayw.pl. Obiektem konwersacji są wyniki meczów żużlowych zapisanych w formacie tabelarycznym, mianowicie uzyskane przez poszczególnych zawodników. Wynikiem natomiast jest przekształcenie ich do formatu biegowego, to znaczy wyników uzyskanych w poszczególnych biegach meczu żużlowego.

Zakres pracy obejmuje kluczowe aspekty realizacji pracy dyplomowej. W skład zakresu pracy wchodzą:

- Określenie celu aplikacji,
- Ustalenie funkcjonalności,
- Wybór odpowiedniej technologii do realizacji pracy,
- Wybór narzędzi projektowych,
- Utworzenie interfejsu użytkownika,
- Opracowanie struktury aplikacji,
- Stworzenie algorytmu umożliwiającego konwertowanie danych.
- Wyświetlenie danych w poprawnej formie.
- Przetestowanie rezultatów.

1.3. Struktura pracy

Praca składa się z pięciu rozdziałów. Rozdział drugi opisuje wykorzystane narzędzia projektowe oraz technologie użyte do stworzenia aplikacji.

Rozdział trzeci przedstawia analizę interfejsu użytkownika, czyli struktury aplikacji oraz aspektów wizualnych. Dodatkowo zawiera opis struktury plików stanowiących całość aplikacji.

Czwarty rozdział został poświęcony obsłudze danych oraz logice aplikacji. Zawarte są opisy poszczególnych etapów obsługi danych, takie jak pobieranie i przetwarzanie. Ponadto opisany został algorytm działania konwertera.

Rozdział piąty zawiera podsumowanie pracy. Przedstawione zostały wnioski oraz potencjalne perspektywy rozwoju aplikacji.

Rozdział 2

Środowisko i Narzędzia projektowe

Stworzona aplikacja jest efektem wykorzystania odpowiednich narzędzi oraz technologii. Do stworzenia i testowania rozwiązań wykorzystany został serwer Apache, dostępny w pakiecie programu Xampp. Jak każda aplikacja internetowa podstawą podstawę stanowi HTML, stanowiący strukturę oraz CSS, odpowiadający za wygląd aplikacji. Skrypty napisane w JavaScript zwiększają interaktywność strony oraz dodają funkcjonalności zwiększające komfort korzystania z aplikacji. Część odpowiedzialna za konwertowanie wyników została napisana w języku skryptowym PHP, przy użyciu biblioteki simple_html_dom[2].

2.1. Serwer lokalny

Serwer jest podstawą każdej strony i aplikacji internetowej. Serwer przechowuje zasoby niezbędne do funkcjonowania oraz obsługuje żądania, dzięki którym aplikacja działa prawidłowo. Jednym z najczęściej wykorzystywanych rodzajów serwera obok Nginx i Litespeed jest Apache.

Apache[3] to serwer HTTP umożliwiający obsługę wyświetlanie oraz obsługę stron WWW. Charakteryzuje się swoją multiplatformowością oraz otwartością. Stworzony w 1995 roku działa na wielu systemach operacyjnych np. Windows, Linux czy macOS. Serwer jest darmowy, posiada wiele dodatkowych modułów zwiększających funkcjonalność, dzięki czemu jest najczęściej wybieranym rodzajem serwera.

W czasie procesu tworzenia strony lub aplikacji programiści korzystają z serwera

lokalnego. Umożliwia to pracę nad projektem w znacznie szybszy sposób. Programista ma możliwość eksperymentowania i testowania aplikacji w izolowanym środowisku. Jedną z wielu możliwych opcji do stworzenia serwera lokalnego jest program Xampp.

Xampp[4] to wieloplatformowy pakiet, w skład którego wchodzi serwer Apache, baza danych MariaDB oraz programy interpretujące skrypty języków PHP i Perl. Do stworzenia aplikacji umożliwiającej konwertowanie meczów żużlowych wykorzystany został serwer Apache oraz interpreter skryptów języka PHP.

2.2. Technologie interfejsu użytkownika

Każda aplikacja internetowa składa się z części interfejsu użytkownika, nazywana frontendem oraz części logiki serwera, czyli backendem. Część interfejsu użytkownika odpowiada za wszystko, co użytkownik widzi po wejściu do aplikacji internetowej. Głównymi zadaniami części frontendowej są:

- Struktura i układ aplikacji — stworzenie struktury aplikacji, odpowiednie ułożenie elementów takich jak przyciski, nagłówek czy pole tekstowe. Za tę część odpowiada głównie język znaczników HTML.
- Wygląd aplikacji — nadanie atrybutom wyglądu tak, aby aplikacja była estetyczna i intuicyjna do korzystania dla użytkownika. Za tą część odpowiedzialny jest język arkuszy stylów CSS.
- Interaktywność — skrypty umożliwiające interakcję użytkownika z aplikacją w celu np. zmiany wyglądu strony bez konieczności przeładowania strony. Do realizacji takich funkcjonalności wykorzystywany jest język JavaScript.
- Komunikacja z częścią serwerową — wymiana danych pomiędzy częścią interfejsu użytkownika a częścią serwerową.

Hipertekstowy język znaczników (HTML)[5] to podstawa stron i aplikacji. Stosowany jest do tworzenia struktury stron internetowych. Aspekty takie jak elementy oraz znaczniki umożliwiają poprawne określenie struktury, oraz zawartości doku-

mentu. Elementy formularzu pozwalają na wprowadzenie użytkownikowi danych oraz przesłanie ich do części serwerowej.

Za wygląd elementów w aplikacji odpowiada język arkuszy stylów CSS[6]. Poprzez nadanie odpowiednich atrybutów elementom aplikacja staje się przejrzysta oraz prosta do obsługi przez użytkownika.

JavaScript[7] to język programowania wykorzystywany do tworzenia interaktywnych elementów na stronie internetowej. Umożliwia bez konieczności przeładowania strony na ukrywanie elementów, wyświetlanie alertów czy zmianę stylu aplikacji.

2.3. Technologie części serwerowej

Jedną z różnic między stroną internetową a aplikacją jest część serwerowa. Strony internetowe skupiają się głównie na prezentacji danych, natomiast aplikacje dają możliwość użytkownikowi na większą interakcję oraz posiadają więcej wbudowanych funkcji. Za dodatkowe funkcjonalności aplikacji odpowiada właśnie część serwerowa. Do realizacji celu pracy, którym jest stworzenie konwertera żużlowego, konieczne jest użycie odpowiedniego języka programowania.

Język PHP[8] to skryptowy język programowania stosowany głównie do tworzenia stron i aplikacji internetowych. Działa po stronie serwera zatem nie jest widoczny dla użytkowników. Charakteryzuje się swoją elastycznością, dlatego jest wykorzystywany do różnych zastosowań. PHP jest jednym z najczęściej wykorzystywanych języków programowania.

2.4. Edytor tekstu i rozszerzenia

Do zwiększenia wydajności oraz komfortu pisania kodu stosowane odpowiednie narzędzia takie jak edytor tekstu lub zintegrowane środowisko deweloperskie. Edytor tekstu to narzędzie używane przez programistę do pisania kodu aplikacji. Głównymi funkcjami edytorów tekstów są:

- kolorowanie składni, poprawiające czytelność kodu,
- numerowanie wierszy, ułatwiające odnalezienie konkretnego fragmentu kodu,

- podpowiedzi, które zwiększają płynność pisania kodu,
- instalowanie wtyczek, które rozbudowują możliwości edytora.

Zaletą edytorów tekstów jest ich uniwersalność, stosowane są do różnych języków programowania, w przeciwieństwie do zintegrowanych środowisk deweloperskich, które są skierowane do konkretnego języka. Jednym z najpopularniejszych edytorów tekstów jest Visual Studio Code.

Visual Studio Code[9] to darmowy edytor tekstu stworzony przez firmę Microsoft. Ogromną zaletą tego edytora jest liczba rozszerzeń, zwiększających możliwości tego edytora.

Do stworzenia aplikacji konwertującej wyniki meczów żużlowych zostały wykorzystane rozszerzenia:

- PHP Tools — pełna integracja dla języka PHP,
- LiveServer — umożliwia automatyczne przeładowanie pliku natychmiast po zapisaniu, dzięki czemu zmiany są widoczne bez konieczności manualnego odświeżenia strony.

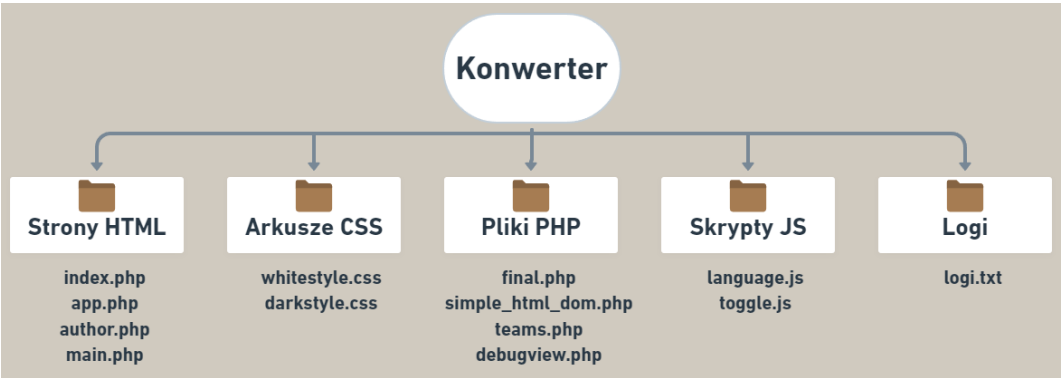
Rozdział 3

Projektowanie Interfejsu

Interfejs jest pierwszą rzeczą, którą zobaczy użytkownik po wejściu do serwisu. Dobrze zorganizowany interfejs graficzny zwiększa szansę na pozytywne doświadczenie podczas korzystania ze strony. Intuicyjność i łatwość w korzystaniu to najważniejsze czynniki, które wpływają na odbiór strony. Poza wymienionymi cechami ważna jest responsywność aplikacji, co pozwoli na korzystanie z serwisu na wielu urządzeniach.

3.1. Struktura aplikacji

W zarządzaniu plikami aplikacji kluczową rolę odgrywa struktura plików. Umożliwia czytelne i logiczne uporządkowanie danych, co ułatwia poruszanie się po całym projekcie. Na Rysunku 3.1 przedstawiono strukturę plików aplikacji.



Rysunek 3.1. Schemat podziału plików aplikacji.

Aplikacja składa się z łącznie 13 plików, razem tworząc całą aplikację internetową. Pliki można podzielić ze względu na ich rolę:

- Strony HTML, w głównej mierze odpowiedzialne są za prezentację danych początkowych takie jak informacje o aplikacji czy autorze,
- Arkusze CSS, definiujące wygląd oraz układ elementów aplikacji,
- Pliki, PHP zawierające całą logikę aplikacji, dzięki której możliwe jest konwertowanie wyników,
- Skrypty JavaScript, odpowiedzialne za interakcję użytkownika z aplikacją, zmianę języka oraz trybu wyświetlania,
- Logi, do których zapisywane są ewentualne ostrzeżenia lub błędy.

3.2. Analiza interfejsu użytkownika

Rolą interfejsu w aplikacji jest zapewnienie jak największego komfortu dla użytkownika. Sam interfejs powinien być responsywny i intuicyjny. Początkowa strona aplikacji podzielona jest na trzy podstrony. Każda z podstron zawiera dwa przyciski, pierwszy służący do zmiany wyglądu strony na tryb ciemny lub jasny, drugi przycisk pozwala na zamianę języka na stronie. Ponadto podstrony zawierają menu pozwalające na przełączanie między podstronami. Poprzez wykorzystanie metody flexbox w CSS, elementy menu są ułożone w jednej osi. Listing 3.1 przedstawia fragment kodu HTML odpowiadający za menu.

```
1 <div class="menu"
2   <a href="index.php" class="option">START</a>
3   <a href="app.php" class="option" id="app">APLIKACJA</a>
4   <a href="author.php" class="option" id="author">AUTOR</a>
5 </div>
```

Listing 3.1. Fragment kodu HTML odpowiadający za menu.

Oprócz tego na stronach znajduję się pasek nawigacyjny, w którym zależnie od podstrony widnieje odpowiednia zawartość, oraz stopka z informacją, że aplikacja została stworzona w ramach pracy inżynierskiej.

Jedyną stroną zawierającą dodatkowy element jest index.php, czyli strona początkowa aplikacji. Oprócz wymienionych wcześniej części strona początkowa posiada formularz, składający się z przycisku oraz pole do wprowadzania adresu strony. Pozwala to użytkownikowi konwertować wybrany przez siebie mecz. W elemencie **form** zostaje nadany atrybut *'method="POST'*, który określa sposób przesyłania danych formularza, a *'action="main.php"'* wskazuje plik, do którego wysłane zostaną dane. Dodatkowo w atrybucie onsubmit, zostaje wywołana funkcja, która zabezpiecza przed wysłaniem pustego formularza. Listing 3.2 prezentuje fragment kodu HTML odpowiedzialny za formularz.

```
<div class="form-container">
  <form method="POST" action="main.php" class="custom-form"
   onsubmit="return validateForm()">
    <input type="text" class="custom-input" id="user-input" name="
   user-input"
      placeholder="">
    <input type="submit" class="custom-button" id="send-button"
   value="">
  </form>
</div>
```

Listing 3.2. Fragment kodu HTML odpowiadający za formularz.

W języku arkuszy stylów nadana została właściwość float do przycisku oraz pola tekstowego, co powoduje, że oba elementy są rozstawione obok siebie. Długość przycisku jest stała, natomiast pole tekstowe dostosowywuje się do szerokości strony uwzględniając rozmiar przycisku. Fragment arkusza stylu CSS został przedstawiony w listingu 3.3

```
.custom-input {
  width: calc(100% - 80px);
  height: 35px;
  box-sizing: border-box;
  padding: 5px;
  border: 1px solid rgb(182, 182, 182);
  float: left;
}
```

```
9
10 .custom-button {
11   color: white;
12   height: 35px;
13   width: 70px;
14   margin-left: 10px;
15   float: right;
16   border: 1px solid rgb(182, 182, 182);
17   background-color: #aaaaaa;
18   box-sizing: border-box;
19 }
```

Listing 3.3. Fragment arkusza stylu CSS odpowiedzialny za pole tekstowe i przycisk.

3.3. Zaimplementowane funkcje JavaScript

Za skrypt zmiany języka oraz trybu wyświetlania odpowiada JavaScript. Zarówno język, jak i tryb zmieniany jest za pomocą przycisków na górze strony. Aplikacja posiada dwa różne style języka arkuszy. Styl whitestyle.css odpowiada za tryb jasny, a darkstyle.css za ciemny. Po kliknięciu przycisku funkcja sprawdza jaki styl był aktualnie przypisany do strony, po czym zamienia go na przeciwny. Wszystko odbywa się bez konieczności przeładowania. Dodatkowo interfejsowi localStorage dane o wybranym trybie wyświetlania zapisywane są w przeglądarce internetowej. Umożliwia to zachowanie ustawień użytkownika nawet po zmianie podstrony.

Aplikacja dostępna jest w dwóch językach: polskim i angielskim. Funkcja zmiany języka wykonywana jest w pliku language.js. Plik zawiera w sobie tablice z danymi dla obydwu języków. Po naciśnięciu przycisku zmiany, wartości tekstowe elementów HTML zmieniają się w zależności od wybranego języka. Tak jak w przypadku przycisku dla trybu wyświetlania, dane o wybranym języku zostają zapisywane do przeglądarki.

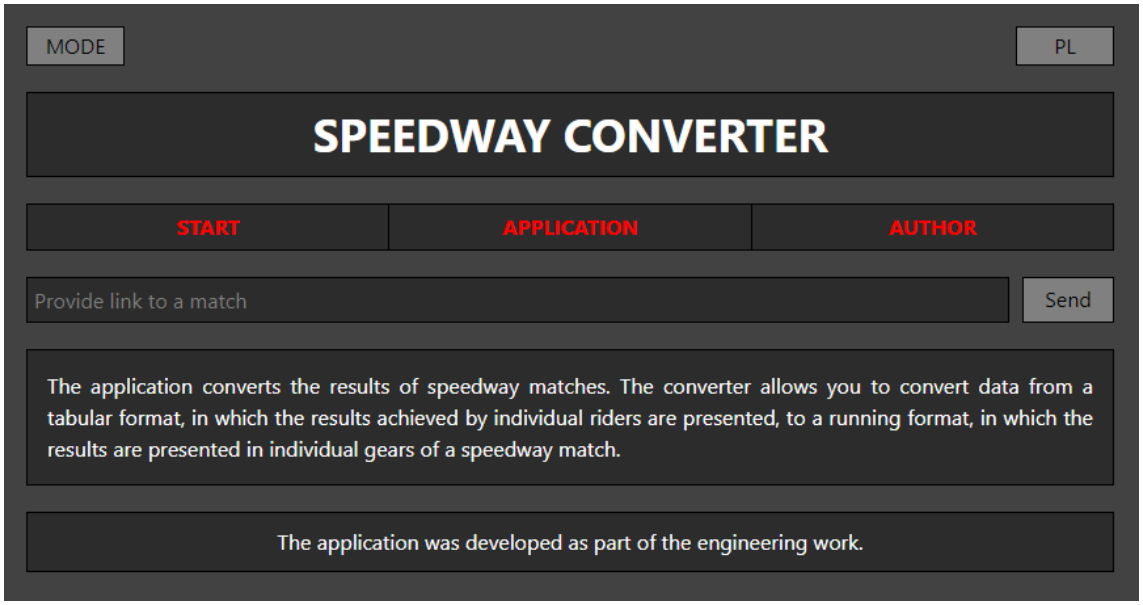
Ze względu na to, że niektóre dane są przechowywane w zasobach przeglądarki, w każdym pliku musi znajdować się odpowiedni skrypt pobierający je przy załadowa-

niu strony. Fragment kodu JavaScript, pobierający dane zachowane w przeglądarce, przedstawiony został w listingu 3.4.

```
window.onload = function () {
  const savedTheme = getSavedTheme();
  if (savedTheme) {
    setTheme(savedTheme);
  } else {
    setTheme("whitestyle.css");
  }
  setLanguage(localStorage.getItem('language'));
}
```

Listing 3.4. Kod JS pobierający dane zachowane w przeglądarce.

Wszystkie wymienione atrybuty oraz funkcję stanowią spójną strukturę aplikacji. Rozmieszczenie elementów HTML sprawia, że widok jest czytelny i przejrzysty dla użytkownika. Atrybuty nadające wygląd aplikacji powodują, że całość jest intuicyjna, responsywna i przede wszystkim prosta do użytkowania. Funkcjonalności, takie jak tryby wyświetlania, czy wybór języka pozwalają na większą interakcję z aplikacją, a zarazem dają możliwość dostosowania jej do własnych potrzeb. Na rysunkach 3.3 oraz 3.2 przedstawiono obie wersje strony głównej aplikacji.



Rysunek 3.2. Wersja ciemna aplikacji w języku angielskim.

Rysunek 3.3. Wersja jasna aplikacji w języku polskim.

Rozdział 4

Obsługa Danych i Logika Aplikacji

Głównym zadaniem aplikacji jest przekształcenie wyników meczu żużlowego z formatu tabelarycznego do formatu biegowego. Całość opiera się na witrynie speedwayw.pl, która zawiera wyniki historycznych meczów żużlowych. Wyniki przestawione na stronie zawierają punktację uzbieraną przez zawodników. Aplikacja po uzyskaniu danych, pobiera je, a następnie na podstawie tablicy biegów konwertuje do formatu biegowego.

4.1. Pobieranie danych

Pierwszym zadaniem części serwerowej jest pobranie danych ze strony, którą użytkownik podał za pomocą formularza. To w niej zapisane są informację o konkretnym meczu, a dokładnie o drużynach, nazwiskach, punktacji i wynikach danego zawodnika. Do wydobycia danych wykorzystana została biblioteka **simple_html_dom.php** [2]. Wpisany w formularzu link zapisywany jest do zmiennej *$inputValue*. Za pomocą funkcji **file_get_html** cała zawartość strony jest przypisywana do zmiennej *$html*. Następnie dzięki funkcji **find('tt')** do tablicy *$table* przypisywane zostają wszystkie znalezione rezultaty elementu. Pobrane dane są zapisane w formie encji HTML. Do dalszego obsługiwania danymi konieczne jest konwersacja ich na zwykły tekst. W tym celu w pętli **for** wykorzystywane są funkcje **html_entity_decode** do usuwania znaczników HTML i zamieniania znaków specjalnych na ich odpowiedniki. Natomiast **preg_replace_callback** i **mb_convert_encoding** służą do zastępowania encji HTML. Listing 4.1 prezentuje fragment kodu

PHP odpowiedzialny za zapisanie oraz konwersję danych.

```php
$html = file_get_html($inputValue);
$table = $html->find('tt');
$table_text = array();

for ($i = 0; $i < count($table); $i++) {
  $table_text[$i] = $table[$i]->plaintext;
  $table_text[$i] = html_entity_decode(str_replace(' ', ' ',
   htmlentities($table_text[$i])));
  $table_text[$i] = preg_replace_callback("/(&#[0-9]+;)/", function
    ($m) {
    return mb_convert_encoding($m[1], "UTF-8", "HTML-ENTITIES"); },
    $table_text[$i]);
}
```

Listing 4.1. Fragment kodu PHP odpowiadający za zapisanie oraz konwersję danych.

Cały algorytm aplikacji opiera się na tablicy biegów. Tablica biegów to spis, według którego zawodnicy brali udział w zawodach. Konwerter oparty jest o tablice z lat 1981-1989. Liczby w kolumnach oznaczają numer konkretnego zawodnika. Tablela z lat 1981-1989 przedstawiona została na rysunku 4.1.

BIEG	TOR A	TOR B	TOR C	TOR D
1.	1	9	2	10
2.	11	3	12	4
3.	5	13	6	14
4.	11	1	12	2
5.	3	13	4	14
6.	9	5	10	6
7.	1	13	2	14
8.	9	3	10	4
9.	5	11	6	12
10.	13	1	10	4
11.	3	9	6	12
12.	11	5	14	2
13.	3	13	6	10
14.	11	1	14	4
15.	5	9	2	12

Rysunek 4.1. Tablica biegów, na której oparty jest konwerter.

W przeciwieństwie do danych z formularza, informacje zapisane na stronie z tablicą są w formacie tekstowym. Z informatycznego punktu widzenia, pierwsza i ostatnia kolumna tablicy nie są istotne, zatem następuję ich usunięcie za pomocą funkcji **array_slice**. Sama tablica natomiast znajduje w elemencie 'td', zatem dzięki funkcji find zapisywana zostaje ona do zmiennej. W pętlach foreach tablica przekształcana zostaje do formatu tablicy dwuwymiarowej. Fragment kodu PHP odpowiedzialny za formatowanie tablicy przedstawiony został w listingu 4.2.

```
$data = array();
foreach ($rows as $row) {
  $cols = $row->find('td');
  $cols_data = array();
  foreach ($cols as $col) {
    $cols_data[] = $col->plaintext;
  }
  $data[] = $cols_data;
}

$data1 = array_slice($data, 1);
$data2 = array();
foreach ($data1 as $row) {
  $data2[] = array_slice($row, 1);
}
$tabelabiegow = $data2;
```

Listing 4.2. Fragment kodu PHP formatujący tablicę biegów.

W późniejszym etapie tabela biegów zostaje posortowana od najmniejszej wartości w wierszu. Zmiana ta umożliwi dostosowanie numeru zawodnika do potencjalnych zmian dokonanych w meczu.

4.2. Przetwarzanie danych

Następnym krokiem jest wykorzystanie pobranych już danych do zrealizowania głównego celu pracy. Wszystkie przechowywane dane zapisane są do zmiennej. Wśród pobranych danych można wyróżnić:

- Nazwy zespołów rywalizujących w konkretnym meczu,
- Numery zawodników z danego spotkania,
- Nazwiska zawodników,
- Suma punktów zdobytych przez zawodnika,
- Punkty, zdobyte w poszczególnym biegu,
- Imię i nazwisko sędziego,
- Zmiany, które zostały dokonane podczas konkretnego meczu.

Rysunek 4.2 prezentuje przykładowe dane meczu w formacie tabelarycznym ze strony speedwayw.pl.

```
1 Z.BŁAŻEJCZAK       12+2 (3,2,1,2,2*,2*)
2 M.MOLKA            5+1  (1,1*,0,-,3)
3 S.DUDEK            9    (3,2,1,2,1)
4 J.SZYMKOWIAK       4    (1,0,3,0)
5 A.HUSZCZA          14+1 (3,1,2,3,3,2*)
6 R.JASINOWSKI       2+1  (2*,0,-,w,0)
7 J.GUZOWSKI         ns
8 J.POŁUBIŃSKI       0    (0)

9 R.JANKOWSKI        8    (2,1,3,2,u/-)
10 D.KASPRZAK        2+1  (0,-,0,-,2*)
11 P.PAWLICKI        2    (0,0,1,w,1)
12 Z.KRAKOWSKI       12   (2,3,3,3,1)
13 Z.KASPRZAK        14+1 (w,3,2*,3,3,3)
14 S.BUŚKIEWICZ      3+1  (1,-,2*,-,0)
15 Dar.BALIŃSKI      2    (1,1)
16 D.ŁOWICKI         1    (1,0)

Sędzia: HENRYK KOWALSKI
ZMIANY:
1-XII 5-XIV 8-IX 9-V 13-VI 15-X,XII 16-XI,XV
```

Rysunek 4.2. Przykładowe dane meczu ze strony speedwayw.pl.

Jednak nie wszystkie pobrane dane są potrzebne do zrealizowania działania konwertera. Kluczowymi danymi z perspektywy działania aplikacji są nazwiska, punkty zdobyte przez zawodników oraz zmiany dokonane w meczu.

Do wydobycia nazwisk z całości pobranych danych wykorzystywany jest odpowiedni wzorzec wyrażenia regularnego. Wyrażenie regularne oznacza ciąg znaków, stosowany do wyszukania konkretnego dopasowania w całości tekstu. W przypadku nazwiska wyrażenie jest zaprojektowane tak, aby wyszukać w tekście ciąg znaków, z uwzględnieniem polskich liter, między którymi znajduje się kropka. Dodatkowo dodana jest alternatywa, gdzie zamiast zapisanego nazwiska zawodnika jest "brak zawodnika". Wszystkie wystąpienia pasujące do wzorca zapisywane są do tablicy. Fragment kodu z przypisywaniem nazwisk do tablicy ukazane zostało w listingu 4.3.

```
1 $pattern_nazwiska = '/([A-ZĄĆĘŁŃÓŚŹŻa-ząćęłńóśźż]+)\.([A-ZĄĆĘŁŃÓŚŹŻ
    a-ząćęłńóśźż]+)|brak zawodnika/';
2 if (preg_match_all($pattern_nazwiska, $table_text[1], $matches)) {
3   $nazwiska = $matches[0];
4 }
```

Listing 4.3. Fragment kodu php przypisujący nazwiska do tablicy.

Cyfry znajdujące się w nawiasach oznaczają punkty zdobyte przez zawodnika. Do wydobycia ich wykorzystywany jest wzorzec, podzielony na trzy alternatywy. Pierwszy wyszukuje fragment, który wpisany jest między nawiasy okrągłe. Drugim przypadkiem jest sytuacja, gdzie zamiast punktacji w nawiasach jest ciąg znaków 'ns' oznaczający, że zawodnik nie został sklasyfikowany, zatem nie uzyskał punktów. Trzecią opcją jest 'brak zawodnika' co również oznacza brak punktów. Zastosowanie tych alternatyw sprawia, że wystąpienie nieklasyfikacji lub brak zawodnika nie jest pomijane w tablicy. Dla tych przypadków przypisywana jest pusta zawartość. Dzięki takiemu zabiegowi numer indeksu tablicy jest również numerem zawodnika (pomniejszonym o jeden, ponieważ indeksowanie tablic w języku PHP rozpoczyna się od zera).

Aczkolwiek w nawiasach znajdują się nie tylko cyfry, ale również inne znaki. System punktowy w żużlu przyznaje kolejno: 3 punkty za pierwsze miejsce, 2 punkty za drugie miejsce, 1 punkt za trzecie miejsce i 0 punktów za czwarte miejsce. Pojawiają się jednak sytuacje, w których zawodnik nie może ukończyć biegu z powodu wypadku lub urazu oraz gdy zawodnik zostaje zdyskwalifikowany. Wszelkie takie zdarzenia zostały odnotowane i zapisane do wyników zawodników. Z perspektywy aplikacji takie sytuacje oznaczają, że zawodnik uzyskał 0 punktów za ten konkretny bieg. W przeciwieństwie do wyżej wymienionych sytuacji, symbole z myślnikiem oznaczają, że zawodnik został zastąpiony przez innego zawodnika. Taka sytuacja jest odnotowywana w punktacji w nawiasach. Do łatwiejszego operowania danymi w późniejszym procesie działania aplikacji każdy symbol niebędący cyfrą zostaje zastąpiony. Listing 4.4 prezentuje fragment kodu PHP zamieniający symbole.

```php
for ($i = 0; $i < count($wyniki); $i++) {
    $wyniki[$i] = str_replace(array("*"), "", $wyniki[$i]);
    $wyniki[$i] = str_replace(array("w"), "0", $wyniki[$i]);
    $wyniki[$i] = str_replace(array("u/-"), "z", $wyniki[$i]);
    $wyniki[$i] = str_replace(array("u"), "0", $wyniki[$i]);
    $wyniki[$i] = str_replace(array("-"), "z", $wyniki[$i]);
    $wyniki[$i] = str_replace(array("d"), "0", $wyniki[$i]);
    $wyniki[$i] = str_replace(array(","), "", $wyniki[$i]);
    $wyniki[$i] = str_replace(array("ns"), "0", $wyniki[$i]);
    $wyniki[$i] = str_replace(array("t"), "0", $wyniki[$i]);
}
```

Listing 4.4. Fragment kodu PHP zamieniający symbole.

Całość działa w pętli for dla wszystkich elementów tablicy *$wyniki*. Zamiana odbywa się za pomocą funkcji **str_replace**. Dla elementów oznaczających, że zawodnik uzyskał 0 punktów za bieg, przypisywane jest zero, a dla zawodników zmienionych przypisywany jest symbol *'z'*.

Ostatnim, kluczowym zestawem danych z perspektywy konwertowania meczów są zmiany. Oznaczają one zawodników, którzy zostali przydzieleni do konkretnego biegu nie na podstawie tablicy biegów. Dane zapisane są w postaci numerów za-

wodników oraz numeru biegu, w którym brali dodatkowo udział. Wyszukanie zmian następuje za pomocą funkcji **strstr**, która wyszukuje pierwsze wystąpienie frazy **"ZMIANY"**. Tekst rozpoczynający się od tej frazy zostaje zapisany do zmiennej ***$zmiany***. Ciąg znaków w zmiennej *$zmiany* zostaje podzielona na elementy tablicy, co umożliwi dalsze działanie w przydzielaniu zawodników do odpowiedniego biegu. Zapis zmian do zmiennej oraz jej podział ukazane zostały w listingu 4.5 przedstawiającym fragment kodu PHP.

```php
$zmiany = strstr($table_text[1], 'ZMIANY:');
$zmiany = str_replace('ZMIANY:', "", $zmiany);
$zmiany = str_replace('-', ",", $zmiany);
$zmiany = preg_replace('/\([^)]+\)/', '', $zmiany);
$zmiany = str_replace(array(" "), "", $zmiany);

$zmianyarray = preg_split('/(?<=[0-9])(?=[a-zA-Z])|(?<=[a-zA-Z])
    (?=[0-9])/', $zmiany);

for ($i = 0; $i < count($zmianyarray); $i++) {
  $zmianyarray[$i] = str_replace(array("-"), " ", $zmianyarray[$i])
    ;
  $zmianyarray[$i] = str_replace(array(","), " ", $zmianyarray[$i])
    ;
}
```

Listing 4.5. Fragment kodu PHP z zapisaniem i podziałem zmiennej ze zmianami.

Numery zawodników zapisane są za pomocą alfabetu arabskiego, natomiast numery biegów zapisane są za pomocą alfabetu rzymskiego. Stąd dane zapisane w alfabecie rzymskim muszą zostać przekształcone do formatu alfabetu arabskiego. Do realizacji konwersji wykorzystywana jest tablica *$romans* typu klucz wartość, w której kluczem jest liczba rzymska, a przypisaną wartością jest jej odpowiednik arabski. Zamiana odbywa się w pętli **foreach**, która przechodzi przez całość tablicy *$zmianyarray*. W przypadku dopasowania wywołana zostaje funkcja, która korzysta z tablicy *$romans*. Listing 4.6 prezentuje zamianę liczb rzymskich na arabskie w kodzie PHP.

```php
$romans = array("I" => 1, "II" => 2, "III" => 3, "IV" => 4, "V" =>
    5, "VI" => 6, "VII" => 7, "VIII" => 8, "IX" => 9, "X" => 10, "XI
    " => 11, "XII" => 12, "XIII" => 13, "XIV" => 14, "XV" => 15, "
    XVI" => 16);

foreach ($zmianyarray as &$element) {
  $element = preg_replace_callback("/(\b[IVX]+\b)/", function (
    $matches) use ($romans) {
     return $romans[$matches[0]];
  }, $element);
}
```

Listing 4.6. Fragment kodu PHP zamieniający liczby rzymskie na arabskie.

W dalszym etapie tablica *$zmianyarray* zostaje zamieniona na tablicę dwuwymiarową, gdzie pierwsza kolumna zawsze oznacza numer zawodnika. To powoduje, że niektóre wiersze tablicy mają większą liczbę kolumn od reszty. Dzieje się tak, gdy jeden zawodnik uwzględniony w zmianach występuje w większej liczbie biegów. Dlatego tablica ze zmianami zostaje przekształcona w taki sposób, aby występowały maksymalnie dwie kolumny, pierwsza z numerem zawodnika, a druga z numerem biegu. Jeżeli wiersz tablicy ma trzeci lub kolejny element, wtedy wartość pierwszego (numer zawodnika) oraz trzeciego lub kolejnego (numer biegu) elementu zostają zapisane. Następnie tworzony zostaje nowy wiersz wraz z zapisanymi danymi. Uzyskana w ten sposób tablica dwuwymiarowa zostaje posortowana według najmniejszej wartości drugiej kolumny (numeru biegu). Do tego celu została stworzona specjalna funkcja **sortBySecondColumn**. Po dokonaniu sortowania tablica zostaje podzielona na dwie osobne tablice, reprezentujące dwie drużyny. Zawodnicy z numerami od 1 do 8 zostają przydzieleni do tablicy *$zmianyteam1*, a zawodnicy z numerami od 9 do 16 do tablicy *$zmianyteam2*. Cały proces tworzący tablicę dwuwierszową oraz sortujący na podstawie najmniejszego numeru biegu w kodzie ukazany został w listingu 4.7.

```php
$zmiany2wiersze = array();

```

```php
foreach ($zmianyarraymatrix as $row) {
  $firstValue = array_shift($row);
  $newRow = array($firstValue);

  foreach ($row as $colValue) {
    $zmiany2wiersze[] = array($firstValue, $colValue);
  }
}
function sortBySecondColumn($a, $b)
{
  return $a[1] - $b[1];
}
usort($zmiany2wiersze, 'sortBySecondColumn');
$zmianyteam1 = [];
$zmianyteam2 = [];

for ($i = 0; $i < count($zmiany2wiersze); $i++) {
  if ($zmiany2wiersze[$i][0] < 9) {
    $zmianyteam1[] = $zmiany2wiersze[$i];
  } else {
    $zmianyteam2[] = $zmiany2wiersze[$i];
  }
}
```

Listing 4.7. Fragment kodu PHP tworzący tablicę dwuwierszową oraz sortujący.

4.3. Działanie algorytmu

Najważniejszym aspektem całej aplikacji jest fragment kodu odpowiedzialny za ustawienie zawodników oraz punktacji w odpowiedniej kolejności. Pierwotna wersja algorytmu, w przeciwieństwie do wersji finalnej, nie opierała się na tablicy biegów. Algorytm miał za zadanie przypisać biegi z tablicy biegów do danych każdego zawodnika, a następnie biegi ze zmian. Po ich dodaniu następowało sprawdzenie, czy numer biegu w biegach zawodnika znajdował się w biegach uwzględnionych w zmianach. W przypadku, gdy nie było takiego wystąpienia, algorytm przechodził do następnego zawodnika. Jednak, gdy taki przypadek miał miejsce, liczba biegów za-

wodnika porównywana była do ilości punktów w znajdujących się w nawiasach. Jeśli porównanie wskazywało równość, algorytm przechodził do następnego zawodnika. Aczkolwiek jeśli nie było równości, ostatni bieg zawodnika był usuwany. Algorytm działał poprawnie dla przykładu, na którym był tworzony, jednak w przypadku innych meczów wyświetlane wartości były niezgodne z prawdą. Powodem okazała się błędna kolejność przydzielania zmian, co prowadziło do wielu niechcianych konsekwencji. Dużo lepszym podejściem do stworzenia poprawnego algorytmu było oparcie go na tablicy biegów.

Finalny algorytm, zawarty w dwóch zagnieżdżonych pętlach **for**, opiera kolejność działania na podstawie tablicy biegów. Przed rozpoczęciem działania, stworzone zostają tablice *$punktacja* oraz *$tabelanazwisk*. Tablica *$punktacja* zapisuje wartości uzyskane przez zawodnika w danym biegu, a tablica *$tabelanazwisk*, nazwisko zawodnika biorącego udział w danym biegu. Pierwszym warunkiem **if** jest sprawdzenie, czy pierwszy element tablicy *$wyniki* jest równy *'z'* (co oznacza zmianę) lub czy tablica jest pusta. Jeżeli warunek nie jest spełniony, następuje działanie zapisane w klauzuli **else**, zatem do tablicy *$punktacja* zapisywany jest wynik, a do tablicy *$tabelanazwisk* nazwisko zawodnika w danym biegu. Dodatkowo pierwszy element tablicy *$wyniki* jest usuwany. Natomiast gdy warunek został spełniony, numer zawodnika zostaje zamieniony na podstawie posortowanej tablicy ze zmianami. Pierwszy wiersz zmian zostaje usunięty, a całość opiera się teraz na podstawie numeru zawodnika ze zmian. Następuje przypisanie wartości do tablic *$tabelanazwisk* oraz *$punktacja* tak jak to miało miejsce w przypadku klauzuli **else**. Jednak wszystko odbywa się, nie dla numeru zawodnika z tablicy biegów, tylko zawodnika uwzględnionego w zmianach. Fragment kodu odpowiedzialny za algorytm przypisywania dla jednej z drużyn przedstawiony został w listingu 4.8.

```
1 for ($i = 0; $i < 15; $i++) {
2   $zmienna1 = $i + 1;
3   for ($j = 2; $j < 4; $j++) {
4     $nr = $tabelabiegow[$i][$j];
5     if ($wyniki[$nr - 1][0] === 'z' || strlen($wyniki[$nr - 1]) ===
      0) {
6       if ($wyniki[$nr - 1][0] === 'z') {
7         $wyniki[$nr - 1] = substr($wyniki[$nr - 1], 1);
```

```
8       }
9       if ($zmianyteam2[0][1] === $zmienna1) {
10        $nr = $zmianyteam2[0][0];
11        array_shift($zmianyteam2);
12        $tabelanazwisk[$i][$j] = $nazwiska[$nr - 1];
13        $punktacja[$i][$j] = $wyniki[$nr - 1][0];
14        $wyniki[$nr - 1] = substr($wyniki[$nr - 1], 1);
15      } else {
16        $tabelanazwisk[$i][$j] = 'brak danych';
17        $punktacja[$i][$j] = '?';
18      }
19    } else {
20      $tabelanazwisk[$i][$j] = $nazwiska[$nr - 1];
21      $punktacja[$i][$j] = $wyniki[$nr - 1][0];
22      $wyniki[$nr - 1] = substr($wyniki[$nr - 1], 1);
23    }
24  }
25 }
```

Listing 4.8. Fragment kodu PHP z algorytmem dla jednej z drużyn.

Dodatkowo w późniejszym etapie dodane zostało sprawdzenie, czy numer biegu ze zmian odpowiada numerowi biegu tablicy biegów. Dzięki czemu w przypadku błędnego zapisu na stronie aplikacja w miejsce błędnych danych zapisuje wartości *'?'* w przypadku punktacji, a *'brak danych'* w przypadku nazwiska zawodnika. Ponadto takie działanie odbywa się osobno każdej z drużyn, co pozwoliło rozwiązać problem błędnie przypisywanych wartości.

4.4. Wyświetlanie danych

Z powstałych tablic z nazwiskami i punktacją w odpowiedniej kolejności, tworzona jest tablica asocjacyjna *$wyniki_asocjacyjne*. Tablica asocjacyjna to tablica, w której przechowywane są pary klucz-wartość. Tego typu tablice są indeksowane za pomocą klucza. W ostatecznej formie tabeli w formacie biegowym oprócz zawodników i ich punktacji są również wyniki za poszczególny bieg oraz wynik na danym etapie zawodów. Zapisywanie wyników oraz stworzenie tablicy asocjacyjnej

odbywa się w pętlach **for**. W przypadku wyników sumowane są rezultaty obu zawodników danej drużyny, a całość zapisywana w formie łańcucha znaków w zmiennej *$wyniki_biegow* oraz *$wyniki_ogólne*. Każde kolejne przejście pętli zewnętrznej zeruje wartości w zmiennych *$bieg1* i *$bieg2*, by przechowywały one jedynie wyniki poszczególnego biegu. Tworzenie danych do wyświetlenia przedstawiono w listingu 4.9.

```
$wyniki_asocjacyjne = array();
$wyniki_biegow = array();
$wyniki_ogolne = array();
$team1 = 0;
$team2 = 0;

for ($i = 0; $i < 15; $i++) {
  $wyniki_wiersza = array();
  $bieg1 = 0;
  $bieg2 = 0;

  for ($j = 3; $j > -1; $j--) {
    $wyniki_wiersza[] = array(
      "nazwisko" => $tabelanazwisk[$i][$j],
      "wynikzawodnika" => $punktacja[$i][$j]
    );
    if ($j > 1) {
      $team1 = $team1 + $punktacja[$i][$j];
      $bieg1 = $bieg1 + $punktacja[$i][$j];
    } else {
      $team2 = $team2 + $punktacja[$i][$j];
      $bieg2 = $bieg2 + $punktacja[$i][$j];
    }
  }
  $wyniki_asocjacyjne[] = $wyniki_wiersza;
  $wyniki_biegow[$i] = $bieg1 . ' : ' . $bieg2;
  $wyniki_ogolne[$i] = $team1 . ' : ' . $team2;
}
```

Listing 4.9. Fragment kodu PHP z tworzeniem danych do wyświetlenia.

Aby zwiększyć czytelność uzyskanych danych, tablica zostaje posortowana według zawodnika, który zdobył najwięcej punktów w danym biegu. Tablica wyświetlana jest na ekranie za pomocą wyrażenia **echo**. Pierwszą kolumną jest numer danego biegu. Następnie wyświetlane są posortowani zawodnicy wraz z przypadającymi im wynikami. Dwie ostatnie kolumny ukazują wynik danego biegu oraz wynik na danym etapie zawodów. Fragment kodu PHP wyświetlający tabelę ukazuje listing 4.10.

```
1  echo '<tbody class="result">';
2
3  for ($i = 0; $i < $liczbaWierszy; $i++) {
4    echo '<tr>';
5    echo '<td>' . ($i + 1) . '</td>';
6    $liczbaKolumn = count($wyniki_asocjacyjne[$i]);
7
8    for ($j = 0; $j < $liczbaKolumn; $j++) {
9
10     echo '<td>' . $wyniki_asocjacyjne[$i][$j]['nazwisko'] . '</td>'
       ;
11     echo '<td>' . $wyniki_asocjacyjne[$i][$j]['wynikzawodnika'] . '
       </td>';
12   }
13   echo '<td>' . $wyniki_biegow[$i] . '</td>';
14   echo '<td>' . $wyniki_ogolne[$i] . '</td>';
15   echo '</tr>';
16 }
17 echo '</tbody>';
```

Listing 4.10. Fragment kodu PHP do wyświetlania tabeli w formacie biegowym.

Z uwagi na dużą liczbę kolumn tabeli dodany został specjalny atrybut w arkuszu CSS, pozwalający na przewijanie tabeli w przypadku, gdy szerokość tabeli przekracza szerokość strony. Rysunek 4.3 przedstawia finalną tabelę w formacie biegowym.

B	ZAWODNIK	P	ZAWODNIK	P	ZAWODNIK	P	ZAWODNIK	P	BIEG	WYNIK
1	Z.BŁAŻEJCZAK	3	R.JANKOWSKI	2	M.MOLKA	1	D.KASPRZAK	0	2 : 4	2 : 4
2	S.DUDEK	3	Z.KRAKOWSKI	2	J.SZYMKOWIAK	1	P.PAWLICKI	0	2 : 4	4 : 8
3	A.HUSZCZA	3	R.JASINOWSKI	2	S.BUŚKIEWICZ	1	Z.KASPRZAK	0	1 : 5	5 : 13
4	Z.KRAKOWSKI	3	Z.BŁAŻEJCZAK	2	M.MOLKA	1	P.PAWLICKI	0	3 : 3	8 : 16
5	Z.KASPRZAK	3	S.DUDEK	2	R.JANKOWSKI	1	J.SZYMKOWIAK	0	4 : 2	12 : 18
6	R.JANKOWSKI	3	Z.KASPRZAK	2	A.HUSZCZA	1	R.JASINOWSKI	0	5 : 1	17 : 19
7	Z.KASPRZAK	3	S.BUŚKIEWICZ	2	Z.BŁAŻEJCZAK	1	M.MOLKA	0	5 : 1	22 : 20
8	J.SZYMKOWIAK	3	R.JANKOWSKI	2	S.DUDEK	1	D.KASPRZAK	0	2 : 4	24 : 24
9	Z.KRAKOWSKI	3	A.HUSZCZA	2	P.PAWLICKI	1	J.POŁUBIŃSKI	0	4 : 2	28 : 26
10	Z.KASPRZAK	3	Z.BŁAŻEJCZAK	2	Dar.BALIŃSKI	1	J.SZYMKOWIAK	0	4 : 2	32 : 28
11	Z.KRAKOWSKI	3	S.DUDEK	2	D.ŁOWICKI	1	R.JASINOWSKI	0	4 : 2	36 : 30
12	A.HUSZCZA	3	Z.BŁAŻEJCZAK	2	Dar.BALIŃSKI	1	P.PAWLICKI	0	1 : 5	37 : 35
13	Z.KASPRZAK	3	D.KASPRZAK	2	S.DUDEK	1	R.JASINOWSKI	0	5 : 1	42 : 36
14	A.HUSZCZA	3	Z.BŁAŻEJCZAK	2	P.PAWLICKI	1	S.BUŚKIEWICZ	0	1 : 5	43 : 41
15	M.MOLKA	3	A.HUSZCZA	2	Z.KRAKOWSKI	1	D.ŁOWICKI	0	1 : 5	44 : 46

Rysunek 4.3. Tabela w formacie biegowym.

Poza wyświetlaniem tabeli w formacie biegowym, na górze strony przekopiowana zostaje tablica w formacie tabelarycznym. Tablica została podzielona na dwie części, dla obu drużyn dane wyświetlane są osobno. Użytkownik ma możliwość ukrycia tych tabel za pomocą przycisku. Skrypt ukrywania tabeli napisany został w JavaScript. Tabela dla jednej z drużyn zaprezentowana została na rysunku 4.4.

Zielona Góra:			
NR	NAZWISKO	PUNKTY	WYNIKI
9	R.JANKOWSKI	8	2,1,3,2,u/-
10	D.KASPRZAK	2+1	0,-,0,-,2*
11	P.PAWLICKI	2	0,0,1,w,1
12	Z.KRAKOWSKI	12	2,3,3,3,1
13	Z.KASPRZAK	14+1	w,3,2*,3,3,3
14	S.BUŚKIEWICZ	3+1	1,-,2*,-,0
15	Dar.BALIŃSKI	2	1,1
16	D.ŁOWICKI	1	1,0

Rysunek 4.4. Dane w formacie tabelarycznym dla jednej z drużyn.

4.5. Weryfikacjia poprawności

Do rozwiązywania następujących w trakcie tworzenia aplikacji problemów powstała specjalna podstrona. Po wyświetleniu tabeli w formacie biegowym użytkownik ma możliwość przejścia do roboczej wersji aplikacji. Wersja robocza charakteryzuje się tym, że zawiera wyświetlanie każdego etapu działania aplikacji, wyświetlane są między innymi:

- wszystkie pobrane dane w formie łańcucha znaków,
- posortowana tablica biegów,
- tablica z nazwiskami i wynikami po przekształceniu,
- tablica ze zmianami dla obydwu drużyn,
- działanie algorytmu ustawiającego zawodników oraz punktację.

Za pomocą części z przebiegiem działania algorytmu został rozwiązany problem błędnego przypisywania zmian. Do kodu został dodany warunek sprawdzający, czy podany ze zmian bieg zgadza się z aktualnym etapem przypisywania do tablicy. Rysunek 4.5 przedstawia działanie algorytmu w wersji roboczej.

BIEG: 1 | Nr Zawodnika: 9	bez zmian	2132z <- Wyniki zawodnika	Przydzielony zawodnik: R.JANKOWSKI	Punkty za bieg: 2
BIEG: 1 | Nr Zawodnika: 10	bez zmian	0z0z2 <- Wyniki zawodnika	Przydzielony zawodnik: D.KASPRZAK	Punkty za bieg: 0
BIEG: 2 | Nr Zawodnika: 11	bez zmian	00101 <- Wyniki zawodnika	Przydzielony zawodnik: P.PAWLICKI	Punkty za bieg: 0
BIEG: 2 | Nr Zawodnika: 12	bez zmian	23331 <- Wyniki zawodnika	Przydzielony zawodnik: Z.KRAKOWSKI	Punkty za bieg: 2
BIEG: 3 | Nr Zawodnika: 13	bez zmian	032333 <- Wyniki zawodnika	Przydzielony zawodnik: Z.KASPRZAK	Punkty za bieg: 0
BIEG: 3 | Nr Zawodnika: 14	bez zmian	1z2z0 <- Wyniki zawodnika	Przydzielony zawodnik: S.BUŚKIEWICZ	Punkty za bieg: 1
BIEG: 4 | Nr Zawodnika: 11	bez zmian	0101 <- Wyniki zawodnika	Przydzielony zawodnik: P.PAWLICKI	Punkty za bieg: 0
BIEG: 4 | Nr Zawodnika: 12	bez zmian	3331 <- Wyniki zawodnika	Przydzielony zawodnik: Z.KRAKOWSKI	Punkty za bieg: 3
BIEG: 5 | Nr Zawodnika: 13	bez zmian	32333 <- Wyniki zawodnika	Przydzielony zawodnik: Z.KASPRZAK	Punkty za bieg: 3
BIEG: 5 | Nr Zawodnika: 14	WYKRYTO ZMIANĘ (NA PODSTAWIE PUNKTACJI [z])	Nr biegu wg zmian: 5:5 Nr biegu wg zmiennej		
BIEG: 6 | Nr Zawodnika: 9	bez zmian	32z <- Wyniki zawodnika	Przydzielony zawodnik: R.JANKOWSKI	Punkty za bieg: 3
BIEG: 6 | Nr Zawodnika: 10	WYKRYTO ZMIANĘ (NA PODSTAWIE PUNKTACJI [z])	Nr biegu wg zmian: 6:6 Nr biegu wg zmiennej		
BIEG: 7 | Nr Zawodnika: 13	bez zmian	333 <- Wyniki zawodnika	Przydzielony zawodnik: Z.KASPRZAK	Punkty za bieg: 3
BIEG: 7 | Nr Zawodnika: 14	bez zmian	2z0 <- Wyniki zawodnika	Przydzielony zawodnik: S.BUŚKIEWICZ	Punkty za bieg: 2
BIEG: 8 | Nr Zawodnika: 9	bez zmian	2z <- Wyniki zawodnika	Przydzielony zawodnik: R.JANKOWSKI	Punkty za bieg: 2
BIEG: 8 | Nr Zawodnika: 10	bez zmian	0z2 <- Wyniki zawodnika	Przydzielony zawodnik: D.KASPRZAK	Punkty za bieg: 0
BIEG: 9 | Nr Zawodnika: 11	bez zmian	101 <- Wyniki zawodnika	Przydzielony zawodnik: P.PAWLICKI	Punkty za bieg: 1
BIEG: 9 | Nr Zawodnika: 12	bez zmian	331 <- Wyniki zawodnika	Przydzielony zawodnik: Z.KRAKOWSKI	Punkty za bieg: 3
BIEG: 10 | Nr Zawodnika: 10	WYKRYTO ZMIANĘ (NA PODSTAWIE PUNKTACJI [z])	Nr biegu wg zmian: 10:10 Nr biegu wg zmiennej		
BIEG: 10 | Nr Zawodnika: 13	bez zmian	33 <- Wyniki zawodnika	Przydzielony zawodnik: Z.KASPRZAK	Punkty za bieg: 3
BIEG: 11 | Nr Zawodnika: 9	WYKRYTO ZMIANĘ (NA PODSTAWIE PUNKTACJI [z])	WYKRYTO ZMIANĘ (PUSTA WARTOŚĆ PUNKTACJI)	Nr biegu wg zmian: 11:11 Nr biegu wg zmiennej	
BIEG: 11 | Nr Zawodnika: 12	bez zmian	31 <- Wyniki zawodnika	Przydzielony zawodnik: Z.KRAKOWSKI	Punkty za bieg: 3
BIEG: 12 | Nr Zawodnika: 11	bez zmian	01 <- Wyniki zawodnika	Przydzielony zawodnik: P.PAWLICKI	Punkty za bieg: 0
BIEG: 12 | Nr Zawodnika: 14	WYKRYTO ZMIANĘ (NA PODSTAWIE PUNKTACJI [z])	Nr biegu wg zmian: 12:12 Nr biegu wg zmiennej		
BIEG: 13 | Nr Zawodnika: 10	bez zmian	2 <- Wyniki zawodnika	Przydzielony zawodnik: D.KASPRZAK	Punkty za bieg: 2
BIEG: 13 | Nr Zawodnika: 13	bez zmian	3 <- Wyniki zawodnika	Przydzielony zawodnik: Z.KASPRZAK	Punkty za bieg: 3
BIEG: 14 | Nr Zawodnika: 11	bez zmian	1 <- Wyniki zawodnika	Przydzielony zawodnik: P.PAWLICKI	Punkty za bieg: 1
BIEG: 14 | Nr Zawodnika: 14	bez zmian	0 <- Wyniki zawodnika	Przydzielony zawodnik: S.BUŚKIEWICZ	Punkty za bieg: 0
BIEG: 15 | Nr Zawodnika: 9	WYKRYTO ZMIANĘ (PUSTA WARTOŚĆ PUNKTACJI)	Nr biegu wg zmian: 15:15 Nr biegu wg zmiennej		
BIEG: 15 | Nr Zawodnika: 12	bez zmian	1 <- Wyniki zawodnika	Przydzielony zawodnik: Z.KRAKOWSKI	Punkty za bieg: 1

Rysunek 4.5. Działanie algorytmu w wersji roboczej.

Ponadto dzięki wersji roboczej aplikacji do zapisywania zawodników dodane zostały odpowiednie wyjątki. Brak zawodnika lub nieklasyfikacja oznaczały przypisanie 'brak zawodnika' do tablicy z nazwiskami, dzięki czemu indeks tablicy zgadzał się z numerem zawodnika (pomniejszone o jeden, ponieważ indeksowanie tablic w PHP rozpoczyna się od zera).

Rozdział 5

Podsumowanie

Głównym celem pracy było stworzenie konwertera żużlowego, przekształcającego dane w formacie tabelarycznym do formatu biegowego. Do realizacji części praktycznej pracy wykorzystane zostały narzędzia takie jak serwer lokalny Apache dostępny w pakiecie programu Xampp. Podstawę aplikacji stanowią HTML oraz CSS z dodatkowymi skryptami JavaScript. Algorytm aplikacji zrealizowany został w języku programowania PHP.

5.1. Wnioski

Część poświęcona interfejsowi użytkownika nie sprawiała większych problemów. Odpowiednie zastosowanie elementów HTML oraz atrybutów CSS pozwoliło na stworzenie czytelnej i responsywnej wersji aplikacji. Ponadto zastosowane skrypty JavaScript umożliwiają użytkownikowi dostosowanie aplikacji do swoich potrzeb.

Wykorzystanie bibloteki **simle_html_dom** oraz jej dokumentacji sprawiło, że parsowanie danych odbyło się bez większych trudności. Najbardziej wymagającym etapem realizacji aplikacji stworzenie algorytmu, który podczas przypisywania danych brał pod uwagę rotację zawodników uwzględnioną w zmianach. Zmiany musiały zostać wywołane w odpowiednim czasie, tak aby wyświetlane wyniki pokrywały się z faktycznym rezultatem danego spotkania. Główny cel pracy został zrealizowany, konwerter działa poprawnie dla większości spotkań. Problemy pojawiają się w momencie, gdy pobrane dane zapisane są w niepoprawnym formacie. W takich przypadkach, wyświetlane jest 'brak danych' w odpowiednich miejscach.

5.2. Perspektywy rozwoju

Algorytm działania oparty jest o tablicę biegów, która zawiera w sobie kolejność występowania zawodników w danym biegu. Aktualna wersja aplikacji skupiona jest na latach 1981-1989. Aby algorytm działał poprawnie dla wszystkich lat, konieczne jest dostosowanie algorytmu do różnych tablic biegów, stosowanych w przeszłości.

Bibliografia

[1] Roman Lach. *Historia Speedway'a w Polsce*. http://www.speedwayw.pl/.

[2] PHP Simple HTML DOM Parser. *Simple HTML DOM documentation*. https://simplehtmldom.sourceforge.io/docs/1.9/index.html.

[3] Apache. *About the Apache HTTP Server Project*. https://httpd.apache.org/ABOUT_APACHE.html.

[4] Xampp. *About the XAMPP project*. https://www.apachefriends.org/pl/about.html.

[5] Hostinger. *What is HTML?* https://www.hostinger.com/tutorials/what-is-html.

[6] Hostinger. *What is CSS?* https://www.hostinger.com/tutorials/what-is-css.

[7] MDN. *What is JavaScript?* https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript.

[8] PHP.net. *What is PHP*. https://www.php.net/manual/en/intro-whatis.php.

[9] Visual Studio. *Documentation for VSC*. https://code.visualstudio.com/docs.