STUDIA INFORMATICA 2012
Volume 33 Number 2B (106)

Katarzyna HARĘŻLAK, Dawid DZIURDZIA, Michał STELMACH
Politechnika Śląska, Instytut Informatyki

SPOŁECZNOŚCIOWY SERWIS DLA ROWERZYSTÓW

Streszczenie. Celem artykułu jest prezentacja serwisu zbudowanego z myślą o stworzeniu platformy, wokół której gromadzić się mogą osoby zainteresowane uprawianiem turystyki rowerowej. W ramach funkcjonalności serwisu uwzględniono możliwość gromadzenia bazy tras, organizacji wycieczek rowerowych oraz prowadzenia konwersacji wśród użytkowników serwisu. Ważnym jego elementem jest moduł statystyk dotyczący aktywności rowerzystów oraz danych opisujących trasę. W pracy omówiono realizację wybranych elementów serwisu.

Słowa kluczowe: turystyka rowerowa, mapy cyfrowe, pliki gpx, serwis społecznościowy

THE SOCIAL WEBSITE FOR BIKERS

Summary. The purpose of the paper was to present the website built in order to create a platform for gathering bikers community. Among the website functionality creating a set of routes, trip management and tools for users' conversation can be found. One of the important elements of the website is a statistical module responsible for presenting user's bike activities and the data describing routes. Realization of the chosen functionality was presented as well.

Keywords: cycling, digital maps, gpx files, social website

1. Wstęp

Turystyka rowerowa jest metodą spędzania wolnego czasu przez podróżowanie rowerem dla samej przyjemności jeżdżenia, a nie w celach ścigania się lub w wyniku istnienia różnych potrzeb. Taki sposób spędzania wolnego czasu staje się w naszym kraju coraz bardziej popularny i mimo że Polska, w stosunku do innych państw Unii Europejskiej, posiada dość słabo rozwiniętą sieć dróg rowerowych, to jednak ich liczba w ciągu ostatnich kilku lat znacząco

wzrosła. Niemniej, pomimo poprawy sytuacji, wciąż relatywnie mało osób jeździ regularnie na rowerze.

Analizując ten problem, można dojść do kliku wniosków. Po pierwsze, większość rowerzystów nie lubi jeździć samemu. Rowerzysta nawet mając ochotę na przejażdżkę, kiedy nie może znaleźć towarzystwa, rezygnuje z pomysłu samotnej jazdy. Kolejną przyczyną jest brak pomysłu na ciekawą trasę. Rowerzysta, który przejechał większość znanych mu tras, demotywuje się perspektywą jazdy kolejny raz tą samą ścieżką i pozostaje w domu.

Sytuacje te świadczą o potrzebie udostępnienia serwisu społecznościowego dla osób spędzających aktywnie czas na rowerze. Dalsza część artykułu skupia się na opisie funkcjonalności oraz realizacji takiego serwisu, którego głównymi celami są stworzenie społeczności rowerzystów, informowanie o wspólnych wyjazdach i wydarzeniach rowerowych oraz gromadzenie bogatej bazy tras.

W chwili obecnej w Polsce istnieją trzy duże serwisy społecznościowe dla rowerzystów. Najprostszym i posiadającym najmniej możliwości jest portal *Bikemap.net*. Serwis ten nastawiony jest głównie na gromadzenie tras rowerowych udostępnianych przez rowerzystów. Użytkownik serwisu może zarówno znaleźć trasę według wybranych kryteriów, jak i dodać własną trasę. Portal nie posiada rozbudowanej funkcjonalności społecznościowej. Same trasy nie posiadają również opisu ani możliwości dodania zdjęć.

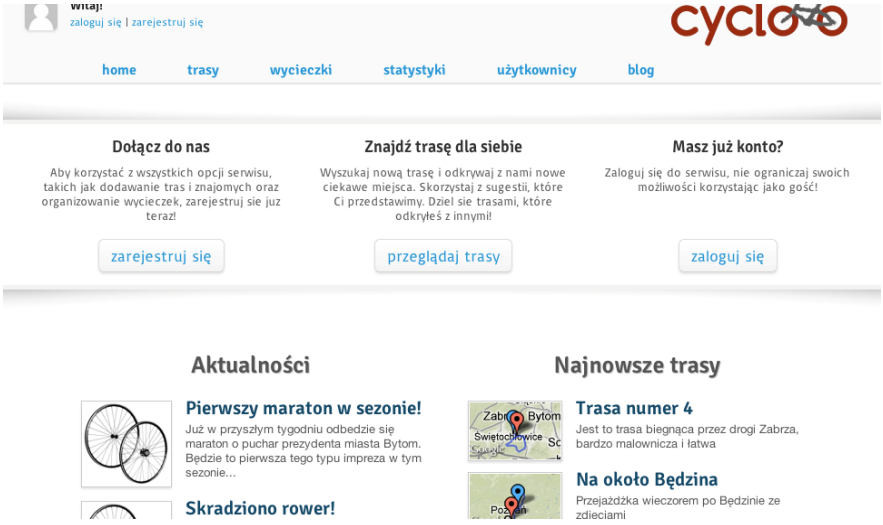
Serwis *Pedaluj.pl* posiada bardziej rozbudowaną, w stosunku do poprzedniego systemu, funkcjonalność. Można wśród niej znaleźć zarządzanie własnym profilem oraz możliwość wymiany opinii na forum. Brakuje w nim jednak możliwości organizacji wycieczek, a przede wszystkim dostępu do wszelkiego rodzaju statystyk.

Portal *Bikestats.pl* to największy tego typu portal na naszym rynku. Strona istnieje od 2005 roku i w czasie, gdy powstawała, nie miała żadnego konkurenta. Jednak podstawowa funkcjonalność serwisu jest inna niż w prezentowanym w niniejszej pracy rozwiązaniu, ponieważ skupia się on przede wszystkim na prowadzeniu dziennika rowerowego – tworzeniu artykułów na temat przejechanych tras – a statystyki rowerowe są tylko mało przejrzystym dodatkiem. Portal nie posiada także możliwości importu danych z urządzenia GPS czy zaznaczenia trasy na mapie.

2. Funkcjonalność serwisu

W prezentowanym w artykule serwisie (rys. 1) zaproponowano rozbudowaną funkcjonalność obejmującą zarządzanie trasami, organizację wycieczek, moduł statystyk, tworzenie planów treningowych, zarządzanie profilem użytkownika, moduły czatu oraz wiadomości.

Niniejsze opracowanie poświęcone jednak będzie tylko tym cechom serwisu, które wyróżniają go pośród innych takich rozwiązań.



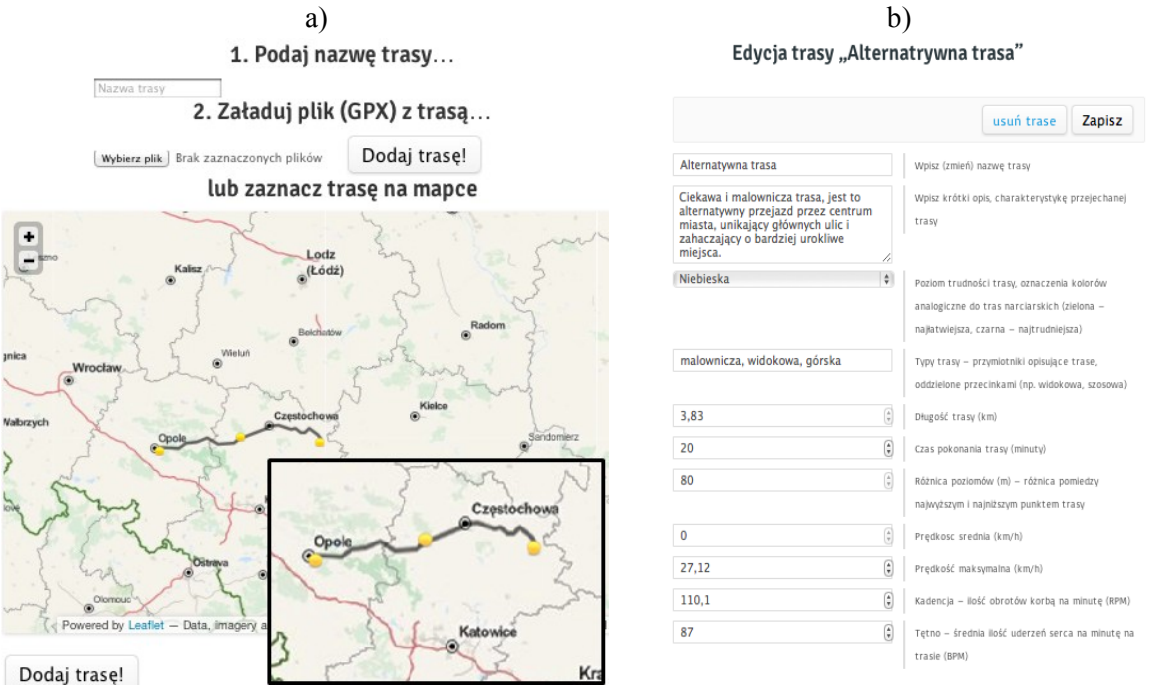
Rys. 1. Strona domowa serwisu
Fig. 1. The service main website

2.1. Zarządzanie trasami

Jednym z ważniejszych elementów serwisu, które mogą zadecydować o jego sukcesie, jest dobrze opracowana baza tras. Do jej utworzenia uruchomione zostały dwie opcje. Pierwsza z nich to zaimportowanie zawartości pliku *.gpx* z urządzenia GPS (lub nawigacji zainstalowanej na telefonie z odbiornikiem GPS) — w tym wypadku system dodatkowo automatycznie przygotowuje wszystkie możliwe do policzenia statystyki i wprowadza do bazy danych ich wartości. Format *.gpx* posiada odpowiadający mu ustandaryzowany schemat XML, stworzony w celu ułatwienia wymiany danych pomiędzy aplikacjami używającymi danych GPS. Jak zaprezentowano na rysunku 2a, użytkownik podaje nazwę trasy, a reszta danych ładowana jest z pliku. W serwisie istnieje możliwość ich uszczegółowienia o dodatkowe informacje, jak typ czy trudność trasy (rys. 2b). Na dole strony edycji zaznaczony jest także jej przebieg.

Drugi sposób dodawania trasy to zaznaczenie odpowiednich punktów na mapie (rys. 2a), co spowoduje narysowanie przez system właściwej drogi oraz podanie jej przybliżonej długości. Użytkownik jest zobligowany do podania prędkości, dokładnej długości i innych danych trasy, korzystając na przykład z komputerka rowerowego (rys. 2b).

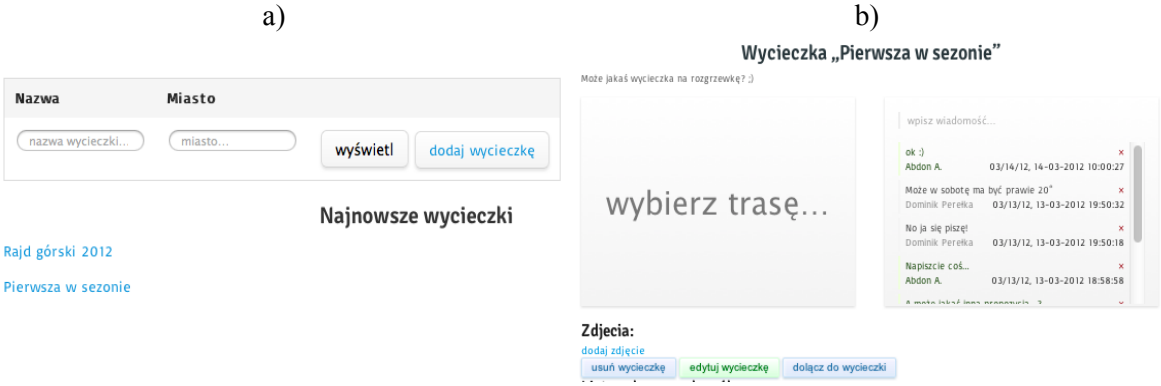
Trasy dodawane do systemu mogą być całkiem nowymi propozycjami lub mogą posiadać referencje do już istniejącej trasy, różniąc się w stosunku do niej niektórymi odcinkami. Tak przygotowana baza tras może posłużyć użytkownikom w planowaniu swoich wycieczek. W tym celu udostępniono mechanizm wyszukiwania tras według ich bliskości względem danego miejsca, długości, oceny, stopnia trudności oraz typu trasy. Wyszukane trasy wyświetlane są na mapie z możliwością podejrzenia ich przebiegu.



Rys. 2. Okna zarządzania trasą: a) okno dodawanie trasy, b) okno edycji trasy
Fig. 2. Route management windows: a) window for adding a route, b) editing a route

2.2. Organizacja wycieczek

Funkcjonalność dotycząca organizacji wycieczek tworzy platformę miejsc spotkań rowerzystów chcących wspólnie z innymi osobami uczestniczyć w różnych wyjazdach rowerowych.



Rys. 3. Okna zarządzania wycieczkami: a) strona wycieczek, b) edycja wycieczki
Fig. 3. Trips management windows: a) a trip Web, b) editing a trip

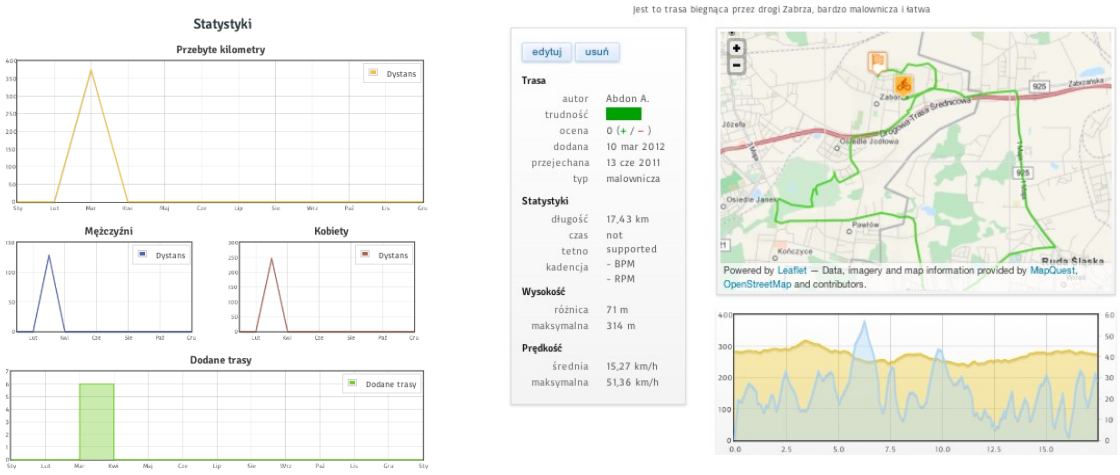
W części a) rysunku 3 widoczna jest główna strona zarządzającą wycieczkami, skąd istnieje możliwość przejścia do formularza dodania wycieczki (rys. 3b), z opcją dodania sugerowanej trasy przejazdu. Również w tym oknie wszyscy uczestnicy, dzięki funkcji chatu, otrzymują możliwość wymiany wrażeń lub dyskusji na temat danej wycieczki. Wycieczka może zostać oznaczona jako *prywatna*, co będzie powodować, że tylko zaproszone przez autora osoby będą mogły do niej dołączyć. Jeśli wycieczka zdefiniowana została z atrybutem

publiczna, będzie ona widoczna dla wszystkich użytkowników portalu, którzy w przypadku zainteresowania się nią, będą mogli się na nią zapisać.

2.3. Moduł statystyk

Ostatnim ważnym, opisywanym, elementem prezentowanego serwisu są statystyki, które dzielą się na statystyki użytkownika, statystyki porównawcze dwóch użytkowników, statystyki trasy oraz statystyki globalne serwisu. Statystyki globalne prezentują kilometry przejechane przez wszystkich użytkowników w poszczególnych miesiącach, wraz z podziałem dokonanym według płci (rys. 4a). Wykresy generowane są na podstawie tras dodawanych przez użytkowników. Na osi pionowej wykresów znajduje się liczba kilometrów, a na osi poziomej miesiąc danego roku kalendarzowego. Statystyki użytkownika znajdują się w jego profilu, gdzie istnieje również dostęp do wykresu porównawczego z innym, wybranym użytkownikiem serwisu.

a) b)



Rys. 4. Okna przeglądania statystyk: a) statystyki całego serwisu, b) statystki trasy
Fig. 4. Statistics windows: a) general statistics, b) a route statistics

Na koniec przedstawione zostaną statystyki trasy, które zawierają wykres prędkości i wysokości w zależności od dystansu na trasie (rys. 4b) oraz:

- maksymalną i średnią prędkość,
- maksymalną wysokość i różnicę między najniższym i najwyższym miejscem trasy,
- czas przejazdu,
- średnie kadencję (czyli liczbę obrotów pedałami na minutę jazdy) i tętno na trasie,
- długość trasy.

3. Realizacja serwisu

Serwis został zaprojektowany zgodnie ze wzorcem projektowym Model-Widok-Kontroler [5, 6], w związku z czym został on podzielony na następujące części:

- modele – stanowią warstwę pośredniczącą w dostępie do bazy danych i umożliwiających uniezależnienie serwisu od konkretnego systemu bazodanowego,
- kontrolery – implementują logikę aplikacji, odpowiadających za przetwarzanie danych dostarczonych przez model i przekazywanie ich do widoku,
- widoki – których zadaniem jest zaprezentowanie wyników dostarczonych przez kontroler w formie zrozumiałej przez klienta.

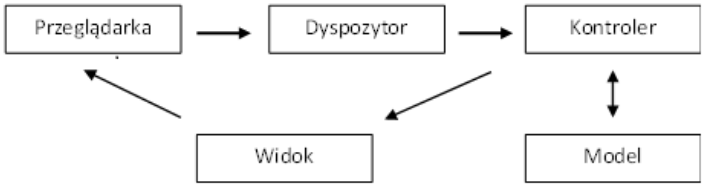
W omawianym serwisie widok korzysta z wzorca projektowego Strategia [5, 6], umożliwiającego definicję formatu danych wysyłanych do klienta.

Za wybór kontrolera i uruchamianie odpowiedniej jego akcji odpowiada kolejny wzorzec projektowy – Dyspozytor [4, 6].

Na rysunku 5 przedstawiono obsługę zapytania wykonanego przez przeglądarkę użytkownika. Mechanizm tworzenia odpowiedzi na żądanie wygląda następująco:

- zapytanie przeglądarki zostaje odebrane przez dyspozytora,
- dyspozytor przekierowuje zapytanie do odpowiedniego kontrolera,
- kontroler komunikuje się z modelem (pobiera lub zapisuje dane),
- kontroler przekazuje rezultaty do widoku,
- widok generuje dane wyjściowe,
- dane zostają przesłane jako odpowiedź do przeglądarki.

Podstawą pracy całości serwisu jest rozbudowana relacyjna baza danych, w której gromadzone są wszystkie niezbędne do uzyskania prezentowanej funkcjonalności dane [2].



Rys. 5. Schemat wzorców projektowych MVC i Dyspozytor
Fig. 5. Scheme of design patterns: MVC and Dispatcher

Całość systemu zrealizowana została z wykorzystaniem języka PHP [1]. Do komunikacji z bazą danych, umieszczoną na serwerze MySQL, został wybrany system ORM Propel. System ten działa na zasadzie generowania klas mapowanych na tabele bazy danych, przy użyciu specjalnie przygotowanego pliku schematu *schema.xml* [12]. Plik ten zawiera definicje bazy danych oraz tabel w niej zawartych. Do tworzenia stron internetowych wybrano język HTML wraz z arkuszem stylów CSS. Rolę serwera WWW pełni program Nginx [10]. Jest to serwer,

którego główną zaletą jest obsługa zapytań HTTP za pomocą zdarzeń, co znaczy, że klient nie blokuje serwera w czasie oczekiwania, lecz serwer przechodzi do kolejnego zapytania, wracając w momencie, gdy czas bezczynności się skończy. Zaprojektowany serwis w dużej mierze korzysta także z możliwości, jakie daje język JavaScript.

3.1. Implementacja wzorców projektowych

Implementacja wzorców projektowych została wykonana jako część, stworzonego na potrzeby serwisu, szkieletu typu framework. Szkielet ten składa się z klas, które ułatwiają szybką implementację serwisu WWW.

Klasa implementująca wzorzec *Dyspozytor*, uruchamia odpowiednią akcję na podstawie adresu URL, jaki wybrał użytkownik. Do akcji tej, wybieranej z tablicy mapowań wyrażeń regularnych na akcje, przekazywane są parametry z adresu WWW. Ponadto, następuje przefiltrowanie oraz oczyszczenie danych przekazywanych przez metody POST i GET. Dzięki zastosowaniu wzorca Dyspozytor istnieje tylko jeden punkt wejścia do aplikacji, dlatego to tam następuje filtrowanie danych. Na początku sprawdzany jest typ danych: jeśli jest to liczba, konwertowana jest ona odpowiednio do typu *integer* bądź *double*, natomiast, jeżeli jest to ciąg znaków, wycinane są z niego wszystkie znaczniki HTML, z wyłączeniem tych, które znajdują się na białej liście (znaczniki paragrafów, list, łamania linii). Dzięki temu, jeśli użytkownik wpisze w formularze znaczniki, które mają załadować i wykonać na stronie obcy kod (XSS) [3], zostaną one usunięte. Usuwane są także ciągi ‚../' i ‚./', aby uniemożliwić odwoływanie się do folderów ponad katalogiem głównym witryny. Dodatkowo, cudzysłowy proste (",') zamieniane są na ich typograficzne odpowiedniki (encje HTML: ”, &rsquo) – poza funkcją estetyczną – ma to na celu utrudnienie wpisania fragmentów zapytań (SQL Injection) [3]. Jeśli typem danych wejściowych jest tablica, to funkcje filtrujące zostają wywołane dla każdego elementu tej tablicy. W ten sposób filtrowane są tzw. zmienne super globalne PHP, które zawierają dane przesłane metodami HTTP: GET i POST.

W przypadku implementacji wzorca projektowego Model-Widok-Kontroler:

- jako *model* został użyty system ORM Propel,
- *kontroler* został stworzony jako klasa wzorcowa, z której dziedziczą klasy konkretnych kontrolerów – definiuje ona podstawowe funkcje potrzebne do prawidłowego działania kontrolera: stworzenie instancji klasy wraz z wyborem strategii wyświetlania (wzorzec Strategia), wywołanie akcji kontrolera oraz wygenerowanie wyjścia za pomocą widoku,
- *widok* został zrealizowany z wykorzystaniem wzorca Strategia. W serwisie stworzono dwie klasy widoku uruchamiane w zależności od wybranej strategii: **widok HTML** i *widok JSON*. Różnią się one sposobem generowania zwracanych danych. Pierwszy opiera się na szablonach HTML, w których zostają umieszczone dane z kontrolera, natomiast

drugi pobiera dane z kontrolera i koduje je do formatu JSON. Widok posiada funkcje, które odpowiadają za przypisanie danych z kontrolera do widoku.

3.2. Obsługa plików GPX

Plik GPX jest to plik XML opisujący trasę za pomocą punktów. Każdy punkt zawiera informacje o długości i szerokości geograficznej, czasie, w którym nastąpiła rejestracja danego punktu, oraz wysokości nad poziomem morza. W pliku GPX możemy wyróżnić elementy *gpx* zawierające informacje o wersji standardu GPX i programie, który utworzył plik. Podrzędne w stosunku do nich elementy *trk* gromadzą informacje o nazwie trasy oraz dacie przejazdu, które z kolei zawierają elementy *trkseg*. Elementy te odpowiedzialne są za przechowywanie poszczególnych punktów trasy zawartych w elementach *trkpt*. Element *trkpt* ma za zadanie przechowywanie informacji o współrzędnych geograficznych oraz opcjonalnie o wysokości i czasie rejestracji punktu, w elementach *ele* i *time*.

Prezentowany dalej plik GPX pokazuje, w jaki sposób ułożone są elementy względem siebie.

```
<?xml version="1.0" encoding="UTF-8"?>
<gpx
  version="1.1"
  creator="RunKeeper - http://www.runkeeper.com"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns=http://www.topografix.com/GPX/1/1
  xsi:schemaLocation="http://www.topografix.com/GPX/1/1 http://www.topografix.co
  m/GPX/1/1/gpx.xsd">
  xsi:schemaLocation="http://www.topografix.com/GPX/1/1 http://www.topografix.co
  m/GPX/1/1/gpx.xsd">
  <trk>
   <name><![CDATA[Cycling 24/6/11 8:56 pm]]></name>
   <time>2011-06-24T20:56:40Z</time>
   <trkseg>
    <trkpt lat="31.111943000" lon="28.223410000">
        <ele>255.7</ele><time>2010-05-21T20:16:45Z</time>
    </trkpt>
    <trkpt lat="31.162573000" lon="28.226150000">
       <ele>255.6</ele><time>2010-05-21T20:17:05Z</time>
    </trkpt>
   …
   </trkseg>
  </trk>
```

Do obsługi tak zbudowanych plików użyto narzędzia Simple API for XML (SAX) [14]. Przetwarzanie plików XML polega na zdefiniowaniu funkcji odpowiadających za otwarcie/zamknięcie elementu, obsługę atrybutów i danych tekstowych w elementach. Przetwarzanie rozpoczyna się od stworzenia parsera metodą PHP *xml_parser_create*( ). Następnie ustawia się funkcje, które zostają wywołane w momencie, gdy parser natrafi na znacznik otwierający/zamykający lub na treść pomiędzy znacznikami:

```
$parser = xml_parser_create(); //utworzenie parsera plików xml
//ustawienie funkcji start_tag i end_tag do obsługi elementów
xml_set_element_handler($parser, array(&$this,'start_tag'), ar-
```

```
ray(&$this,'end_tag'));
//ustawienie funkcji tag_text do obsługi treści miedzy znacznikami definiującymi
elementy
xml_set_character_data_handler($parser, array(&$this,'tag_text'));
```

W dalszej części następuje otwarcie pliku, wczytanie danych do parsera i przetworzenie ich oraz zwolnienie zasobów:

```
//otwarcie pliku lub zwrócenie wyjątku jesli się nie uda
if(!($fp = fopen($this->gpxfile->getPath(), 'r')))
  throw new GPXException('Nie można otworzyć podanego pliku', 500);
//dopóki są dane należy je przetwarzyć
while($data = fread($fp, 4096))
 if(!xml_parse($parser, $data, feof($fp))) //jesli się nie uda to zwracamy błąd
  throw new GPXException('Błąd xml: '.xml_get_error_code($parser).
    ' (linia: '.xml_get_current_line_number($parser).')', 500);
//zwalniamy zasób parsera
xml_parser_free($parser);
```

Funkcja odpowiadająca za otwarcie znacznika reprezentującego dany element zawiera instrukcje warunkowe, które sprawdzają typ znacznika i jeśli natrafią na element *trkpt*, to zapisują do obiektu klasy GPXPoint jego atrybuty (lat, lon). Następnie, jeśli w środku elementu *trkpt* parser natrafi na elementy *time* lub *elev*, to zapamiętuje ten fakt w odpowiedniej zmiennej. Funkcja odpowiedzialna za przetwarzanie tekstu wewnątrz elementów, korzystając z tej zmiennej, doda tekst elementu do odpowiedniej zmiennej w obiekcie klasy GPXPoint. Proces powtarza się aż do zakończenia pliku GPX. Klasa GPXPoint odpowiedzialna jest za przechowywanie informacji o pojedynczym punkcie trasy:

```
class GPXPoint {
   private $lat; //długość geograficzna
   private $lon; //szerokość geo.
   private $elev; //wysokość n.m.p
   private $time; //czas rejestracji punktu
   private $distance; //dystans miedzy poprzednim,a tym punktem
   private $speed; //prędkość chwilowa w punkcie

   public function __construct($lat=0,$lon=0,$elev=0,$time=''){…}}
```

3.3. Wyznaczanie statystyk na podstawie współrzędnych GPS

Następnym krokiem po przetworzeniu pliku GPX jest wyznaczenie, na podstawie punktów pobranych z tego pliku, statystyk trasy. W tym celu wyliczane są interwały czasu pomiędzy kolejnymi punktami oraz odległość między nimi. Posiadając te dwie dane, bardzo łatwo jest policzyć prędkość między punktami. Jednak, ze względu na stosunkowo dużą niedokładność pojedynczych odczytów przez urządzenia GPS, prędkość chwilowa jest wyznaczana jako uśrednienie, według wzoru 1:

$$v_i = \begin{cases} v_0, & dla\ i = 0 \\ v_i = \dfrac{v_0 + 2v_1}{3} & dla\ i = 1, \\ v_i = \dfrac{v_{i-1} + v_i + v_{i+1}}{3} & dla\ 1 < i < n, \\ v_i = \dfrac{v_{i-1} + 2v_i}{3} & dla\ i = n, \end{cases} \tag{1}$$

gdzie: n – liczba punktów na trasie pomniejszona o 1, v_i – prędkość chwilowa liczona na odcinku pomiędzy punktami: i oraz i+1.

Odległość jest liczona za pomocą wzoru Haversina [15], który daje dokładność wyliczeń na poziomie pojedynczych metrów – trochę większą niż dokładność konsumenckich odbiorników GPS – tak więc w zupełności wystarczającą w tym zastosowaniu

$$\begin{aligned} a &= \sin^2 \frac{\Delta\theta}{2} + \cos\frac{\theta_1}{2} + \cos\frac{\theta_2}{2} + \sin^2 \frac{\Delta\varphi}{2}, \\ c &= 2\arcsin\left[\min\left(1, \sqrt{a}\right)\right] \\ length &= Rc, \end{aligned} \tag{2}$$

gdzie: R – średnia długość promienia ziemi, θ_1, θ_2 – długość geograficzna (odpowiednio 1 i 2 punktu), $\Delta\theta$ – różnica długości geograficznych między punktami, $\Delta\varphi$ – różnica szerokości geograficznych między punktami, *length* – odległość w linii prostej między punktami.

Przy obliczaniu odległości pomiędzy dwoma punktami pośrednimi można również wykorzystać wzór (3), który uwzględnia różnicę wysokości pomiędzy nimi:

$$\begin{aligned} \Delta h_i &= \left|h_{i-1} - h_i\right|, \\ D_i &= \sqrt{d_i^{\,2} + \Delta h_i^{\,2}}, \\ D_{cał.} &= \sum_{i=1}^{n} Di, \end{aligned} \tag{3}$$

gdzie: h_i – wysokość punktu i, d_i – odległość między punktami i oraz $i-1$.

Pomimo dokładniejszych pomiarów, metoda ta nie jest wykorzystywana w obecnej wersji prezentowanego serwisu w celu zachowania zgodności z odległościami udostępnianymi przez mapy terenowe. Problem wpływu ukształtowania terenu na rzeczywistą odległość uwzględniony zostanie w kolejnych wersjach portalu.

Wygenerowane w opisany powyżej sposób statystyki zostają zapisane w odpowiednich tabelach bazy danych: DBUserStats – statystyki użytkownika, DBRouteStats – statystyki trasy, DBGPS – statystyki punktów. Statystyki trasy zawierają całkowity dystans, maksymalną i średnią prędkość, maksymalną i minimalną wysokość n.p.m na trasie, średnią kadencję i tętno. Statystyki użytkownika zawierają te same informacje co statystyki trasy, tylko dla konkretnego użytkownika, w danym miesiącu kalendarzowym. Natomiast każdy punkt GPS

zawiera statystyki chwilowe: dystans pomiędzy tym punktem a jego poprzednikiem (gdy jest to pierwszy punkt to 0) oraz prędkość chwilową w tym punkcie.

3.4. Dodawanie trasy za pomocą mapy

W pierwszej wersji systemu do zarządzania trasami na mapie cyfrowej wykorzystywano oprogramowanie udostępniane przez firmę Google−API Google Maps. Jednak, ze względu na wprowadzenie przez firmę Google opłat dotyczących korzystania z jej produktów, zdecydowano się zmienić interfejs na alternatywny, jakim jest serwis MapQuest [9] wraz z biblioteką Leaflet [8]. Co prawda limity darmowego wykorzystywania Google Maps Api są dość wysokie (do 25 tys. wyświetleń dziennie za darmo, każde dodatkowe 1000 wyświetleń to koszt 4$), ale dzięki zastosowaniu drugiego z rozwiązań wykorzystywanie funkcjonalności map jest w pełni darmowe.

Dodanie trasy z wykorzystaniem mapy cyfrowej odbywa się przez zaznaczenie jej kilku punktów pośrednich:

```
marker = new L.Marker(e.latlng,{ //współrzędne miejsca kliknięcia
  icon:markers['pos'],           //ustalenie odpowiedniej ikony znacznika
  clickable:true,        // udostępnienie możliwości kliknięcia w znacznik
  draggable:true });     // oraz "przeciągnięcia" go w inne miejsce
marker.on('dragend', f); //funkcja wywołana po przeciągnięciu znacznika
map.addLayer(marker);    //dodanie znacznika na mapę
markery.push(marker);    //dodanie znacznika do tablicy punktów trasy
```

Następnie zostaje wygenerowany adres usługi sieciowej MapQuest Direction Web Service przez podanie punktu początkowego i „sklejonych” pozostałych punktów trasy. W kolejnym kroku wywoływane jest zapytanie *get*, a wyniki zostają zapisane do bazy danych:

```
for(var i=1;i<markery.length;i++) {
   to += '&to='+ markery[i].getLatLng().lat + ',' + markery[i].getLatLng().lng;
   }
//ścieżka do usługi sieciowej wyznaczania trasy MapQuest
var url=
   'http://open.mapquestapi.com/directions/v0/route?outFormat=json&routeType=bi
cycle&timeType=0&shapeFormat=raw&generalize=200&unit=k&from='+from+to;
                                //z doklejonymi punktami trasy do wyznaczenia
```

3.5. Wyświetlanie trasy

Proces wyświetlania trasy polega na pobraniu z bazy danych opisujących ją cech, narysowaniu na mapie profilu trasy oraz stworzeniu wykresów dotyczących statystki trasy. Do rysowania na mapie trasy użyto *polinii*. Parametry funkcji określają mapę, na której wyświetlona ma zostać trasa, punkty, przez które przechodzić ma linia, a także jej grubość i kolor:

```
for (var i =0; i<lat.length; i++){ //dodajemy współrzędne punktów do łamanej
       latlngs.push(new L.LatLng(lat[i],lon[i]))};

var polyline = new L.Polyline(latlngs, {color: '#4b1', opacity:0.7,weight:3});
map.fitBounds(new L.LatLngBounds(latlngs)); //ustawienie przybliżenia oraz
```

```
                                środka mapy, tak aby łamana wypełniła całe okno
map.addLayer(polyline); //narysowanie trasy
```

Prezentowanie wykresów możliwe jest dzięki bibliotece Flot [7], która umożliwia zaawansowane rysowanie różnego typu wykresów w języku JavaScript. W stworzonym portalu wykorzystywany jest wykres liniowy o dwóch osiach Y, którego sposób użycia przedstawiono w zamieszczonym dalej kodzie. W definicji obiektu można wyróżnić następujące parametry:

- element strony (kontener), w którym zostanie narysowany wykres,
- tablicę z kolejnymi seriami danych,
- tablicę asocjacyjną z opcjami wykresu (ustawienia legendy wykresu, osi, podpisów):

```
//zmienna X (przebyta droga)
x = <?php echo($this->route->getDistances())?$this->route-
                                                    >getDistances():'[]';?>;
//Y1 - wysokość n.p.m
elevations = <?php echo($this->route->getElevations())?$this->route-
                                                    >getElevations():'[]';?>;
//Y2 - prędkość
speeds = <?php echo($this->route->getSpeeds())?$this->route-
                                >getSpeeds():'[]';?>;

//wpełnienie zmiennych e i s parami: e = (X,Y1) i s = (X,Y2)
var e = new Array();
var s = new Array();
for(i=0;i<x.length;i++) {
   e[i] = [x[i], elevations[i]];
   s[i] = [x[i], speeds[i]];
}
//ustawienie opcji serii danych
  ...
// ustawienie opcji wykresu
  ...
$.plot($('#chart'), [ e_options, s_options ], options); //rysowanie wykresu
```

3.6. Obsługa wiadomości pomiędzy użytkownikami

Mechanizm obsługi wiadomości wykorzystuje technologię *HTTP Push*. Polega ona na nawiązywaniu przez przeglądarkę dodatkowego połączenia HTTP, które nie jest zamykane po otrzymaniu odpowiedzi na zapytanie, jak w normalnym trybie, tylko oczekuje na kolejne. Użycie tej technologii wymaga dołączenia do serwera WWW *Nginx* modułu NGINX_HTTP_Push_Module [11]. Dodatkowo należy w konfiguracji serwera, dla domeny obsługującej serwis, dodać linie, które odpowiedzialne są za ustawienie odpowiednich adresów URL dla usług zapisu i odczytu wiadomości pochodzących z kolejki tworzonej dla każdej konwersacji:

```
location/pool {
  push_subscriber;      # pod tym adresem można pobierać wiadomości
                        # wiadomość jest nadawana do każdego klienta tego kanału
  push_subscriber_concurrency broadcast;
  set $push_channel_id $arg_id;      # id jest w parametrze: ?id=
  default_type application/json;     # typ zwracanych danych to json
}
```

```
location /message_send {
  #id będzie w parametrze id: ?id=[id_wpisu]
  set $push_channel_id $arg_id;
  push_publisher;           #informujemy że pod tym adresem można
                            #publikować wiadomości
  push_store_messages on;       #włączone będzie kolejkowanie wiadomości
  push_message_timeout 2h;      #wiadomości wygasają po 2 godzinach
  push_max_message_buffer_length 10; #i maksymalnie trzymamy 10 wiadomości
}
```

Taka konfiguracja pozwala na wpisywanie wiadomości pod adres /message_send i odbieranie ich z adresu /pool. Dodatkowo należy podać identyfikator kanału. W przypadku obsługi wiadomości przez opisywany w tej pracy serwis, identyfikator kanału to skrót MD5 pewnej stałej znakowej (hasła) sklejonej z identyfikatorami użytkowników, których dotyczy konwersacja, w kolejności: odbiorca, nadawca. Daje to wynik w postaci dwóch kanałów wiadomości dla każdej konwersacji.

W akcji SEND kontrolera obsługującego wiadomości znajduje się część odpowiedzialna za publikowanie wiadomości w odpowiednim kanale. Pierwszym krokiem musi być nawiązanie połączenia z adresem publikacji. W kolejnym kroku należy ustawić typ żądania HTTP na POST, którego daną stanowi tablica z czasem wysłania i treścią wiadomości, zakodowana w formacie JSON. Wiadomość zostaje zapisana w kolejce wiadomości oczekujących, skąd pobierana jest przez adresata. Jej odbiór wymaga uruchomienia kodu, który zmienia typ zapytania na GET, ustawia identyfikator kanału na odpowiednią wartość oraz definiuje funkcję, która pobiera dane, dodaje je do strony WWW, jako kolejną wiadomość, oraz wywołuje następne zapytanie (w ramach tego samego połączenia HTTP), które będzie oczekiwało na kolejną wiadomość. Przykładowa wymiana wiadomości zaprezentowana została na rys. . 6.

Rys. 6. Przykładowa wymiana wiadomości
Fig. 6. Sample conversation

4. Podsumowanie

Ciągle wzrastająca liczba tras rowerowych, dostosowywanie przepisów ruchu drogowego, tak aby były bardziej przyjazne dla rowerzystów, coraz większa aktywność rynku w zakresie sprzedaży sprzętu rowerowego oraz istnienie oddolnych inicjatyw, takich jak „Zagoń rower do roboty", pokazują, że zmienia się świadomość Polaków dotycząca użytkowania roweru.

Na tym gruncie pojawił się pomysł stworzenia miejsca w Internecie, gdzie, ciągle jeszcze mała, społeczność rowerzystów będzie mogła dzielić się odkrytymi trasami, swoimi rekordami i sukcesami. Zaprezentowany w artykule serwis powstał z myślą o promowaniu jazdy na rowerze jako sposobu na przyjemne, aktywne spędzanie czasu, dlatego nacisk położony został na ułatwienie rowerzystom umawiania się na wspólne wyjazdy oraz dzielenie się informacjami o trasach już przebytych. Do jego zalet zaliczyć można umiejętność automatycznego generowania trasy na podstawie danych uzyskanych z urządzenia GPS oraz definiowanie jej na podstawie punktów określonych na mapie cyfrowej. Niewątpliwie ważnymi funkcjonalnościami serwisu są gromadzenie i prezentacja różnego rodzaju statystyk, które mogą pełnić rolę motywującą lub pomocniczą przy wyborze trasy dla planowanej wycieczki czy treningu. Dzięki udostępnieniu funkcji chatu uzyskano wygodne narzędzie do komunikacji pomiędzy członkami społeczności zgromadzonej wokół proponowanego serwisu.

Serwis, który stworzony został na podstawie nowoczesnych narzędzi, takich jak PHP w wersji 5.4 oraz obecnie wykorzystywane technologie prezentacji treści w Internecie (HTML 5, CSS 3 i JavaScript), zostanie wkrótce uruchomiony. Dzięki zastosowaniu wzorców projektowych oraz podziale na moduły, jest on przystosowany do dalszego rozwoju obecnych jego elementów oraz rozszerzenie o nowe, takie jak:

- stworzenie bazy punktów użyteczności publicznej (POI),
- dopracowanie metod przechowywania i stworzenie algorytmu rozpoznającego takie same trasy,
- ulepszenie algorytmu sugerowania tras,
- zarządzanie planami treningowymi.

BIBLIOGRAFIA

1. Atkinson L.: PHP. Programowanie. Helion, Gliwice 2008.
2. Garcia-Molina H., Ullman J., Widom J.: Database Systems: The Complete Book. Prentice Hall, 2009.
3. Nowocień T.: Bezpieczeństwo aplikacji webowych. Poznań 2009.

4. Mancl D.: Design Patterns. 11.2007, 2011, s. 28÷29.
5. Shalloway A., Trott J. R.: Design Patterns Explained. 2001, s. 238÷239.
6. Walczak T.: Wzorce projektowe. Elementy oprogramowania obiektowego wielokrotnego użytku. Helion, 2010.
7. Flot. Attractive Javascript plotting for jQuery, http://code.google.com/p/flot/, 2011.
8. Leaflet. A Modern, Lightweight Open-Source JavaScript Library for Interactive Maps by CloudMade, http://leaflet.cloudmade.com/reference.html, 2012.
9. MapQuest-OSM Tiles + MapQuest Open Aerial Tiles, http://developer.mapquest.com/web/products/open/map, 2012.
10. Nginx: http://www.nginx.com/, 2011.
11. NGiNX_HTTP_Push_Module, http://pushmodule.slact.net/, 2011.
12. Propelorm.com, Database Schema, http://www.propelorm.org/reference/schema.html, 2011.
13. php.net, What can php do?: http://pl.php.net/manual/pl/intro-whatcando.php, 2011.
14. php.net, XML Parser – Instalacja, http://www.php.net/manual/pl/xml.installation.php, 2011.
15. PHP – Calculating Distance Between Two Locations Given Their GPS Coordinates, 2009, http://sgowtham.net/blog/2009/08/04/php-calculating-distance-between-two-locations-given-their-gps-coordinates/, 2012.

Wpłynęło do Redakcji 22 stycznia 2012 r.

Abstract

Increasing number of bicycle paths, adjustment of traffic regulations to bikers' needs and existence of social initiatives showed that awareness of the Poles with the regard to cycling has changed. This is why the idea of creating the website which will serve as a place for bikers community meeting was developed. The website presented in the paper was prepared to promote cycling as a way for nice time spending. For this reason great effort was focused on preparing functionality facilitating trips arrangement and exchanging the data on routes. An advantage of the website is possibility of defining routes on the basis of gpx files including geographical information regarding points of a bike path. One of the important elements of the website is the statistical module responsible for presenting user's bike activities and the data describing particular routes. Besides, thanks to the chat functionality a convenient tool for user communication was obtained.

The website, implemented with usage of PHP language and such technologies like HTML, CSS and Java Script will, be soon made accessible. Owing to utilizing design patterns and modular architecture of the website it can be easily extended.

Adresy

Katarzyna HARĘŻLAK: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, katarzyna.harezlak@polsl.pl.
Dawid DZIURDZIA: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, ddziurdzia@me.com.
Michał STELMACH: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, michstel256@gmail.com.