УДК 528.94 DOI: 10.30533/0536-101X-2018-62-2-223-232

Разработка программного модуля ГИС «Мониторинг» для геоанализа объектов образования Республики Крым

© 2018 г. М.А. Боярчук*, П.Ю. Орлов

Московский государственный университет геодезии и картографии, Москва, Россия

**stan931@rambler.ru*

Developing a software component for GIS «Monitoring» module designed for geoanalysis of educational facilities in Republic of Crimea

M.A. Boyarchuk*, P.Yu. Orlov

Moscow State University of Geodesy and Cartography, Moscow, Russia

**stan931@rambler.ru*

Received Oktober 20, 2017 *Accepted March 12, 2018*

Keywords: geoanalysis, GIS, Republic of Crimea, web portal.

Summary. Information analysis systems are an independent category of information systems designed to automate analytics aimed at decision management or other possible solutions. The «Monitoring» Information Analysis System developed in MIIGAiK is described in this paper. Furthermore, the development process of a software component designed to plan the examination routes and to carry out consolidated areal indices geoanalysis of educational facilities in Republic of Crimea is represented. The developed software component uses the combination of two programming languages: PHP and JavaScript. The API Yandex.Maps is used to visualize geodata, routes generation and their length and time of travel. A proprietary algorithm, which involves direct searching the minimum routes of three points and then expanding them by locating the nearest free points to search an optimal route was developed. Two consolidated areal indices, which are used in the information system to analyze the status of subordinate educational organizations in the Republic of Crimea, were formed on the basis of the available information analysis. The first one is called «Schools overcrowding» and it is defined as ratio of the number of schools with second and third shifts to the total number of schools in an administrative-territorial entity (ATE). The second index, «Shortage of places in kindergartens for one hundred people in the age group of 4–7 years», is defined as ratio of the sum of ATE shortage places to the population in the age group of 4–7 years. In other words, this index characterizes the number of missing places per one hundred children of pre-school age. The created component cooperates both with the subsystem of the contractor and with the subsystem of the customer and fully completes the tasks set by him.

Citation: *Boyarchuk M.A., Orlov P.Yu.* Developing a software component for GIS «Monitoring» module designed for geoanalysis of educational facilities in Republic of Crimea. *Izvestiya vuzov* «*Geodeziya i aerofotosyemka*». Izvestia vuzov «Geodesy and Aerophotosurveying». 2018, 62 (2): 223–232. [In Russian]. DOI: 10.30533/0536-101X-2018-62-2-223-232.

Поступила 20 октября 2017 г. *Принята к печати 12 марта 2018 г.*

Ключевые слова: геоанализ, ГИС, web-портал.

Рассмотрена созданная в МИИГАиК информационно-аналитическая система «Мониторинг». Описан процесс создания программного модуля для этой системы, который служит для планирования маршрутов инспектирования и базового геоанализа по сводным территориальным показателям объектов образования в Республике Крым. Разработан собственный алгоритм поиска оптимального маршрута. Сформированы два сводных территориальных показателя, используемые в информационной системе для анализа состояния подведомственных образовательных организаций.

Для цитирования: *Боярчук М.А., Орлов П.Ю.* Разработка программного модуля ГИС «Мониторинг» для геоанализа объектов образования Республики Крым // Изв. вузов «Геодезия и аэрофотосъемка». 2018. Т. 62. № 2. С. 223–232. DOI: 10.30533/0536-101X-2018-62-2-223-232.

В статье [1] была рассмотрена Информационно-аналитическая система «Мониторинг», созданная Управлением информатизации МИИГАиК по заказу Министерства образования и науки РФ. Она предназначена для сбора информации, учета объектов недвижимого имущества образовательных организаций Республики Крым[1,2] [2, 3]. Описана ее архитектура, подсистемы, входящие в ее состав, а также круг их пользователей. По просьбе упомянутого заказчика для более эффективного управления объектами образования была разработана подсистема геоанализа, которая в настоящее время подверглась некоторой модернизации [4]. Именно этому и посвящена данная работа.

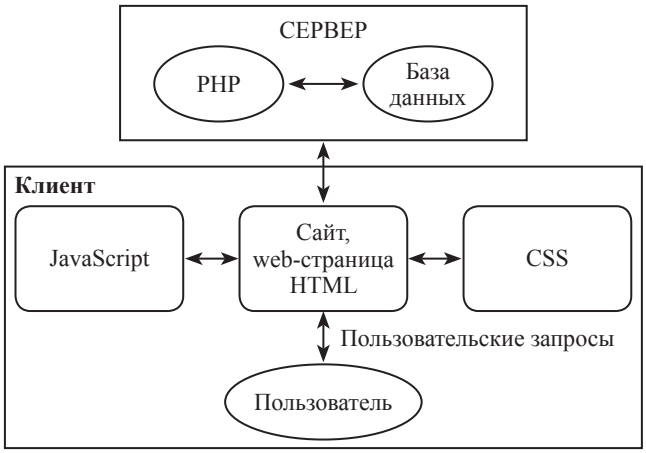

Рис. 1. Место PHP и JavaScript в клиент-серверной архитектуре

Fig. 1. Place of PHP and JavaScript in the client-server architecture

Средства разработки

В разработанном программном модуле применялась связка из двух языков программирования. В целях безопасности для работы с базами данных использовался PHP. В случае JavaScript потребовалось бы указание адреса, логина и пароля к базе данных непосредственно в коде страницы, передаваемой клиентскому браузеру. Поскольку PHP-код завершает свое выполнение еще до передачи страницы клиенту, взаимодействие между языками напрямую невозможно. При этом большая часть алгоритмов и вычислений выполняется также на сервере. Кроме того, при помощи PHP можно вставлять в код страницы не только размеченный текст, но и код на JavaScript, что позволяет варьировать этот код в зависимости от ситуации (рис. 1).

Однако средств PHP не хватает для выполнения всех поставленных задач. Так, визуализацию данных и интерактивность страницы в полной мере можно реализовать только в JavaScript. Кроме того, на JavaScript работает API Яндекс.Карт, который представляет собой набор JavaScript-компонентов, предназначенных для создания интерактивных карт. Компоненты размещены на серверах Яндекса и доступны для использования сразу после их загрузки на страницу [5]. Данный программный интерфейс используется для визуализации геоданных, построения маршрутов и нахождения их длины и времени прохождения.

Применение Яндекс API на фоне конкурентов (например, аналогичного API от Google) связано с несколькими причинами. Во-первых, этот программный интерфейс уже используется в созданной системе. В случае другого API потребовались бы дополнительные работы по интеграции модуля в систему. Во-вторых, Яндекс API разработан российской компанией, что дает основания полагать, что предоставляемые сервисом карты на территорию РФ будут иметь большую точность и обновляться чаще. В-третьих, в последнее время все актуальнее становится использование отечественных продуктов, в том числе и программных, вместо импортных. Наконец, Яндекс.Карты могут предоставить больший объем информации за счет геоинформационного сервиса «Народные карты», позволяющего пользователям дополнять и исправлять карты.

[1]Концепция осуществления Министерством образования и науки Российской Федерации полномочий собственника в отношении имущества подведомственных организаций. М.: Минобрнауки России, 2011. 27 с.

[2]Постановление Правительства Российской Федерации от 12.04.2012 № 289 (ред. от 15.12.2016) «О федеральной государственной информационной системе территориального планирования».

Цели и задачи

Решаемые задачи формулировались следующим образом:

1) разработать модуль для системы «Мониторинг», взаимодействующий с подсистемами исполнителя и заказчика и реализующий функции геоанализа;

2) в подсистеме исполнителя разрабатываемый модуль должен реализовать автоматизированное разбиение посещаемых точек на оптимальные маршруты из *n* точек (число устанавливается пользователем при запуске модуля), а также поиск оптимальных мест ночевки для мобильной группы в промежутках между обследованиями. Точки ночевок выбираются из заранее заданного списка;

3) в подсистеме заказчика реализовать подсчет и визуализацию некоторых статистических показателей, показывающих состояние объектов образования Республики Крым.

Перед описанием реализации модуля рассмотрим задачу более подробно. Имеется база данных по точкам, расположенным в определенном регионе или закрепленным за какой-либо конкретной мобильной группой, и точкам ночевки для обследующей группы. Пользователь вводит подряд в систему число точек, из которых будут состоять маршруты, день начала обследования и число рабочих дней. Разрабатываемый модуль должен:

1) разбить множество точек на минимальные по длине маршруты, содержащие указанное пользователем число точек (под маршрутом будем понимать путь, который позволяет посетить все точки один раз, ни разу не возвращаясь на уже посещенные точки);

2) дополнить построенные маршруты наиболее близкими пунктами ночлега, в которых мобильные группы находятся до и после обследования;

3) составить график объезда организаций, исходя из того, что в один день проводится один маршрут.

В качестве критерия близости точек будем использовать длину пути между ними. Путь прокладывается при помощи API Яндекс.Карт, этот же программный интерфейс предоставляет данные по протяженности и затратам времени на каждый путь, а также применяется для визуализации построенных маршрутов. Перед разработкой программного модуля были проведены поиск и анализ задач и их решений с целью найти уже реализованные алгоритмы, пригодные к применению. Поставленная задача перекликается сразу с несколькими известными задачами теории графов. Однако ни одно решение для классических задач такого рода не может предоставить решение, удовлетворяющее всем необходимым требованиям.

Поскольку не было найдено готового алгоритма, применимого к поставленной задаче, разработана собственная схема действий, которая заключается в нахождении перебором минимальных маршрутов из трех точек с последующим их наращиванием путем нахождения ближайших свободных точек. Достоинства данного варианта – простота и понятность реализации.

Разработка алгоритма поиска оптимального маршрута

Перед реализацией выбранного алгоритма следует *подготовить исходные данные*. Для получения необходимых данных от пользователя используется HTML-форма. Она содержит (подряд): поля для ввода числа точек на маршрут и даты начала обследования, а также выбор значения для указания числа рабочих дней. При помощи кнопки «Построить» данные отправляются в скрипт PHP для дальнейшей обработки. Полный код формы можно найти в работе [4]. Однако это — не все необходимые данные. Дело в том, что у нас отсутствует информация о путях между пунктами и мы сталкиваемся с небольшой сложностью, которая заключается в том, что эти данные находятся при помощи API Яндекс.Карт, который работает в клиентской части, а вычисления выполнены на стороне сервера. В связи с этим нахождение путей вынесено в отдельную связку скриптов.

Ввести данные о путях можно несколькими способами: передать в нужный скрипт на-

прямую с помощью POST-запроса или занести данные в базу. В настоящей работе реализован второй способ, так как он позволяет выполнять сколько угодно расчетов на одном и том же множестве точек, рассчитав пути только один раз. Первым делом скрипту необходимо получить данные о точках, между которыми будут вычисляться пути. Для этого подключаемся к базе данных:

```
$link = mysql_connect ("127.0.0.1", "root", "");
mysql_select_db ("gis", $link).
```

Функция `mysql _ connect` устанавливает подключение к базе данных по указанному адресу, логину и паролю. Данные из базы можно получить при помощи функции `mysql _ query`, которая выполняет переданный ей SQL-запрос. Первым запросом узнаем число записей в базе, а вторым получаем все ее записи. Полученные данные заносим в массивы со связанными индексами:

```
$result = mysql_query ("SELECT COUNT(*) FROM points");
$row = mysql_fetch_row($result);          $count = $row[0];
$query = "SELECT * FROM points ORDER BY ID";
$result = mysql_query($query);
for ($i = 0; $i < $count; $i++){    $row = mysql_fetch_row($result);
$ids[$i] = $row[0];      $X[$i] = $row[1];           $Y[$i] = $row[2];}
В разделе <Head> необходимо подключить API Яндекс.Карт и плагин jQuery:
<script    src="http://api-maps.yandex.ru/2.1/?lang=ru_RU"    type="text/
javascript"> </script>
   <script   type="text/javascript"   src="ajax.googleapis.com/ajax/libs/
jquery/1.6.1/jquery.min.js"></script>
```

В том же разделе в тегах `<script>` пишем свой скрипт на JavaScript. В нем вычисляем необходимые параметры путей. Создаем объект Promise («обещание») как «обертку» для вычислений, и при помощи PHP вводим в него вычисления на Яндекс API:

```
echo "var p1 = new Promise(function(resolve, reject){\n";
<…>
echo "ymaps.route([[".$X[$i]."], ".$Y[$i]."], [".$X[$j]."], ".$Y[$j]."]]).
then(function (route) {l[".$k."] = route.getLength(); t[".$k."] = route.
getTime(); k++; });\n ";
<…>
echo "timer1 = setInterval(function(){if (k >= " . $count*($count-1)/2 .
") {resolve(t, l)}}, 7000);\n });\n ";
```

Функция `ymaps.route` служит для построения путей между указанными точками; `route.getLength()` и `route.getTime()` находят длину и время прохождения пути соответственно. При помощи вложенных циклов по `$i` и `$j` мы вызываем эти функции столько раз, сколько у нас есть пар точек. Во избежание построения путей по два раза (т.е. из А в Б и из Б в А) пути строятся только от точки с меньшим ID до точки с большим. Использование «обертки» в данном случае обусловлено тем, что функция построения пути асинхронна, т.е. возвращает значение не сразу, а в неопределенный момент в будущем. Таким образом, функции могут выполниться не в том порядке, в котором они запрашиваются. Нам же необходимо дождаться момента, когда все пути будут построены. Для этого используются функция `setInterval` и счетчик `$k`. Как только «обещание» завершено, вызывается функция, указанная в методе `.then()`:

```
echo "p1.then(function(){\n";
echo "ldata = JSON.stringify(l);\n"; echo "tdata = JSON.stringify(t);\n";
<…>
echo  "$.post('toDB.PHP',  {len:  ldata,  time:  tdata},  function(data)
{alert(data)});\n});\n ".
```

В этом блоке мы переводим данные в формат JSON и отправляем их в скрипт toDB.PHP при помощи AJAX-метода $.post. В отличие от классического HTML запроса методом POST это позволяет передать данные без перезагрузки текущей страницы. Текущий скрипт полностью приведен в работе [4]. Запись данных в базу проводится в скрипте toDB.PHP. Сначала он получает данные от предыдущего скрипта:

```
$a = json _ decode($ _ POST['len']); $b = json _ decode($ _ POST['time']);
```

Затем известным образом подключаемся к базе данных и пересоздаем таблицу несколькими SQL-запросами:

```
$query = "Drop table routes"; mysql_query ($query);
$query = "CREATE TABLE routes (Counter int not null, Start_id int not
null, end_id int not null, distance real null, time integer null)";
mysql_query ($query).
```

Взяв данные из таблицы, содержащей пункты маршрутов, приступаем к заполнению таблицы путей:

```
for ($i = 0; $i < $count; $i++){    $p1 = $ids[$i];
for ($j = $i+1; $j < $count; $j++){$p2 = $ids[$j];
$query = "INSERT INTO routes (Counter, Start_id, end_id, distance, time)
VALUES (" . $k . "+1, " . $p1 . ", " . $p2 . ", " . $a[$k] . ", " . $b[$k]
. ");";
mysql_query($query);          $k++.
```

Таким образом, получаем таблицы, содержащие протяженности и длительности всех путей. Полный скрипт приведен в работе [4]. Далее рассмотрим скрипт, выполняющий все основные вычисления. Перед началом вычислений получаем данные из формы и базы данных. В этот раз данные из БД заносим в ассоциативные массивы:

```
$row = mysql_fetch_row($result);
$Points[$row[0]]["X"] = $row[1];
$Points[$row[0]]["Y"] = $row[2].
```

Это позволяет нам использовать массивы в некотором смысле как функции — передаем в них ID-точки и координату, которую хотим получить; получаем ее значение. Теперь переходим непосредственно к *алгоритму формирования маршрутов* (рис. 2).

Первый этап алгоритма заключается в следующем: мы перебираем все возможные подмножества из трех точек на исходном множестве и ищем такое, которое соединялась бы маршрутом минимальной протяженности. На подмножестве из трех точек можно построить только три пути, а чтобы соединить три точки нужно всего два пути. Таким образом, кратчайший маршрут может быть найден как сумма двух минимальных путей. Однако этот способ не годится для нахождения минимального маршрута при большем числе точек, так как в данном случае сумма минимальных путей может не дать маршрут или соединять не все точки. Полученные маршруты хранятся в виде списка ID точек в порядке посещения; использованные точки удаляются из списка доступных. Поскольку эту часть скрипта мы описали на естественном языке, а его программный код довольно громоздок, здесь он не приводится.

На втором этапе каждый маршрут поочередно дополняется ближайшими точками ночевки. Для этого создана специальная функция, принимающая на вход список точек маршрута в порядке их посещения и также возвращающая список точек. Поиск ближайшего пункта проводится путем перебора всех доступных пунктов. Необходимо найти такой пункт, среднее расстояние до которого от конца и начала маршрута было бы минимальным:

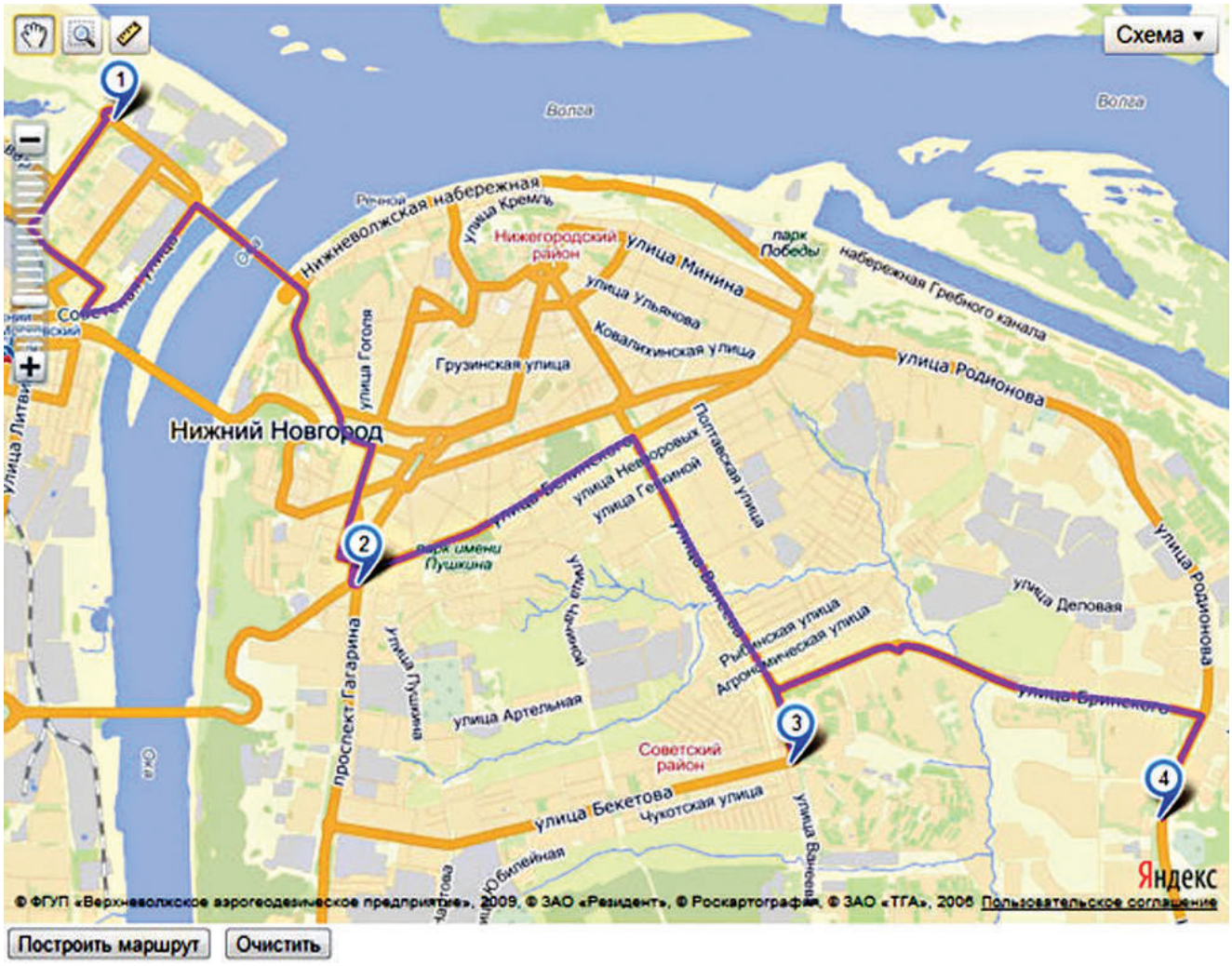

Рис. 2. Пример построения маршрутов при помощи Яндекс API
Fig. 2. An example of route planning performed by Yandex API

```
for ($i = 0; $i < $count; $i++)
$sdlist[$sids[$i]] = ($sDists[$sids[$i]][$plist[0]] + $sDists[$sids[$i]]
[$plist[$lp]])/2.
```

Найденная точка вставляется в начало и конец списка точек маршрута. Однако на этом подбор точек ночевки не заканчивается. Дело в том, что в данном случае не учтены варианты, когда мобильная группа должна переехать в другую точку, чтобы продолжать обследование оттуда. Таким образом, может возникнуть ситуация, когда в *n*-й день группа после обследования приезжает в точку «1», а на следующее утро начинает свой путь уже из точки «2». Чтобы это исправить, отсортируем все маршруты в порядке возрастания ID-точки ночевки. Это обеспечит нам их группировку и исключит ситуацию, когда маршруты с точкой ночевки в «1» идут по графику не подряд, а разбросаны случайным образом. Затем на границах группы меняем последнюю точку последнего маршрута группы на точку начала первого маршрута следующей группы.

Теперь переходим к *визуализации результатов*. В теле JavaScript-скрипта функции `init()` создадим объект «карта» из Яндекс API. Выведем на карту метки точек, построенные маршруты и настроим их отображение [4]. Для отображения построенных маршрутов необходимо добавить их в коллекцию геообъектов карты. Добавлять их напрямую нельзя, поскольку операции по-

строения путей — асинхронные, они выполняются не в порядке их вывода в программном коде и будут занесены в коллекцию в том же порядке. Нам же важно, чтобы они заносились именно в указанном порядке, чтобы индексы путей на карте соответствовали индексам путей в отдельно выводимой таблице. Для этого каждый построенный путь вначале вносим в отдельную коллекцию, а эти коллекции в правильном порядке заносим в коллекцию карты.

Перейдем к *построению графика объезда*. Алгоритм построения крайне прост: каждому дню с начала объезда соответствует определенный маршрут в том порядке, в котором они содержатся в массиве полученных решений. В каждые *N* дней, указанные пользователем, один день считается выходным и пропускается в графике. Полученный график распечатывается с помощью HTML-таблицы. Таблица содержит следующие столбцы: номер маршрута, дата, список посещаемых точек, протяженность маршрута и время на его преодоление. Полученная таблица выводится в виджет Dialog, предоставляемый плагином `jquery-ui`:

```
$( "#table" ).dialog({dialogClass: "no-close", position: { my: "left
top", at: "left bottom"}, autoOpen: true, width: 800, }).
```

Этот виджет представляет собой диалоговое окно, всплывающее поверх страницы с картой. Данное окно не является модальным и, следовательно, не мешает работе со страницей. Также оно может свободно перемещаться по экрану.

Для удобства нахождения нужных маршрутов на карте реализована функция их подсветки при наведении на маршрут в таблице. С этой целью в JavaScript-части создана функция, активирующаяся при наведении на конкретную строку. Данная строка и соответствующий ей маршрут подсвечиваются красным (рис. 3).

```
function h1 (n) {  document.getElementById("r"+n).className="hl";
Rlist.get(n).get(0).getPaths().options.set({strokeColor:      '#FF191D',
opacity: 1});}
```

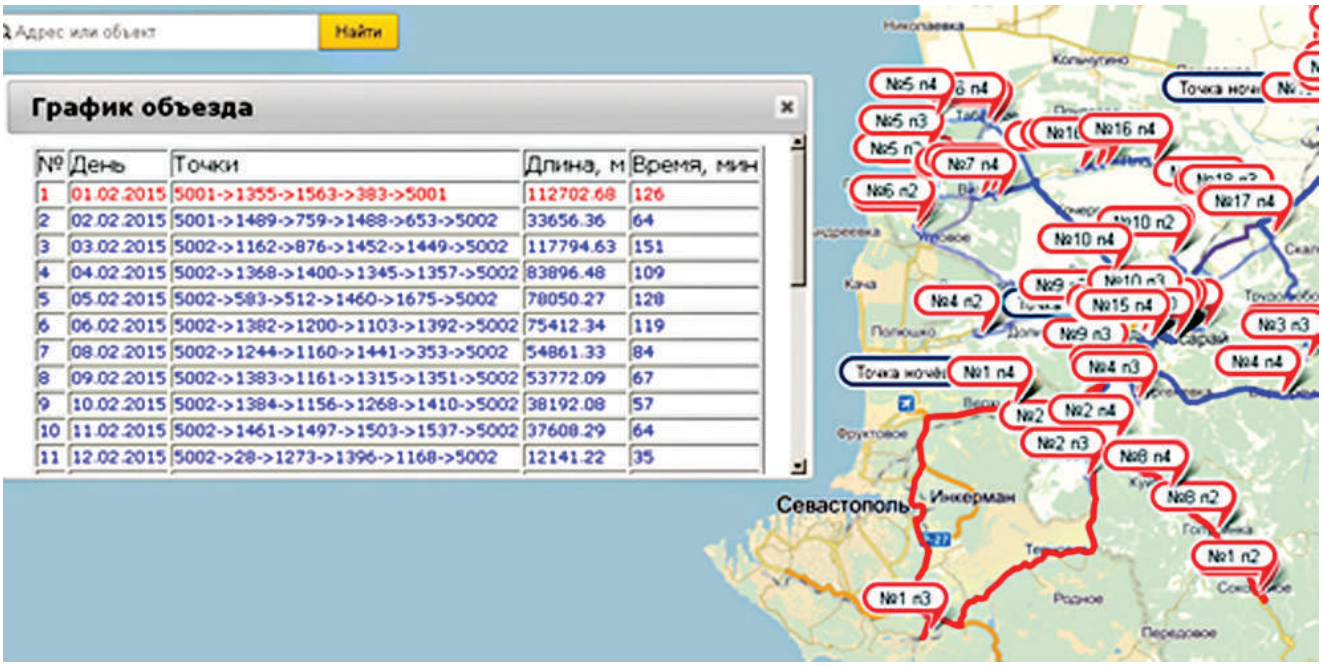

Рис. 3. Внешний вид модуля при построении маршрутов
Fig. 3. Appearance of software component while planning the routes

Разработка функционала базового геоанализа по сводным территориальным показателям

Перейдем к следующей части работы по модернизации подсистемы заказчика, которую можно охарактеризовать *как задачу расчета и визуализации сводных территориальных показателей*, используемых в информационной системе для анализа состояния подведомственных образовательных организаций в Республике Крым [6]. Для начала необходимо определить показатели, с которыми мы будем работать. На основании анализа доступной информации сформированы два показателя. *Первый показатель* условно называется «*Перегруженность школ*» и определяется как отношение школ с наличием второй и третьей смены (для школ с третьей сменой вводится весовой коэффициент, равный двум) к общему числу школ административно-территориальной единицы (АТЕ). Данный показатель может принимать значения от 0 до 2, однако значения больше 1 будут получаться только в случае достаточного числа школ с существующей третьей сменой, поэтому появление значения больше единицы маловероятно. При условии отсутствия школ с третьей сменой этот показатель можно трактовать как доля школ с наличием второй смены от общего числа школ региона. *Второй показатель* — «*Дефицит мест в детских садах на сто человек возрастной группы 4–7 лет*» — определяется как отношение суммы дефицитных мест по АТЕ к численности населения в возрастной группе от четырех до семи лет. Этот показатель характеризует число дефицитных мест на сто детей дошкольного возраста. Диапазон возможных значений находится в интервале от 0 до 100 (так как на одного ребенка может быть максимум одно дефицитное место).

При работе с подсистемой заказчика мы не будем создавать отдельную страницу для нашего модуля, а подключим его к основной. Для этого создадим отдельный JavaScript-модуль и подключим его в коде основной страницы, добавив строку

```
<script src="Analyst.js"></script>.
```

Все наши действия заключены в тело функции GA(), вызываемой с основной страницы. Перед вычислением показателей нужно получить необходимые данные. Для этого используются AJAX-запросы `$.getJSON`, которые позволяют обратиться к удаленному скрипту и получить данные в формате JSON без перезагрузки страницы. Получив данные о необходимых типах образовательных учреждений, мы записываем их в массивы и делаем то же самое с данными по АТЕ. После это просматриваем каждое учреждение и переписываем необходимые для расчетов данные в переменные, связанные с соответствующими АТЕ. Получив необходимые данные, перебираем все АТЕ, вычисляя необходимые показатели:

```
for (i = 0; i < j; i++){
aid = scate[i];
if (scses[i] == 1){atep2[ateid.indexOf(aid)]++}
else  if  (scses[i]  ==  2){atep2[ateid.indexOf(aid)]++;  atep1[ateid.
indexOf(aid)]++}
else  if  (scses[i]  ==  3){atep2[ateid.indexOf(aid)]++;  atep1[ateid.
indexOf(aid)]++; atep1[ateid.indexOf(aid)]++}  }
for (i = 0; i < r; i++){
atepg[i] = atep1[i]/atep2[i]; }
```

Для визуализации мы используем полученную от сервера информацию о границах АТЕ. Кроме координат, она содержит также цвет, которым конкретные регионы заливаются на предусмотренном слое. Мы изменяем этот параметр в зависимости от значения рассчитанного показателя и добавляем его измененные данные в меню выбора слоев как отдельный слой. Для визу-

ализации перегруженности школ нами используются следующие правила: регионы со значением показателя ниже 0,1 окрашиваются зеленым цветом, в промежутке 0,1–0,3 — желтым, выше 0,3 — красным:

```
$.each(data.features, function(key, val){
if (atepg[ateid.indexOf(val.id)] < 0.1){val.properties.color = "#ACFA2F"}
else if (atepg[ateid.indexOf(val.id)] < 0.3){val.properties.color =
"#FAEC2F"}
else val.properties.color = "#FF1612";    });
app.Pg_layer = L.geoJson(null, {onEachFeature:popup_ate, style:style_
ate});
app.Pg_layer.addData(data);
app.toc.addOverlay(app.Pg_layer, "Перегруженность школ");
```

Аналогично вычисляется показатель дефицита мест в детских садах. Отличия заключаются лишь в получаемых данных: вместо данных по школам необходимо запросить данные по детским садам, а также получить расширенную информацию по каждой АТЕ (рис. 4). Полный программный код модуля приведен в работе [4]. Здесь дадим лишь правила визуализации: если значения показателя по району меньше 15 — район окрашивается зеленым цветом, если меньше 25 — желтым, больше — красным.

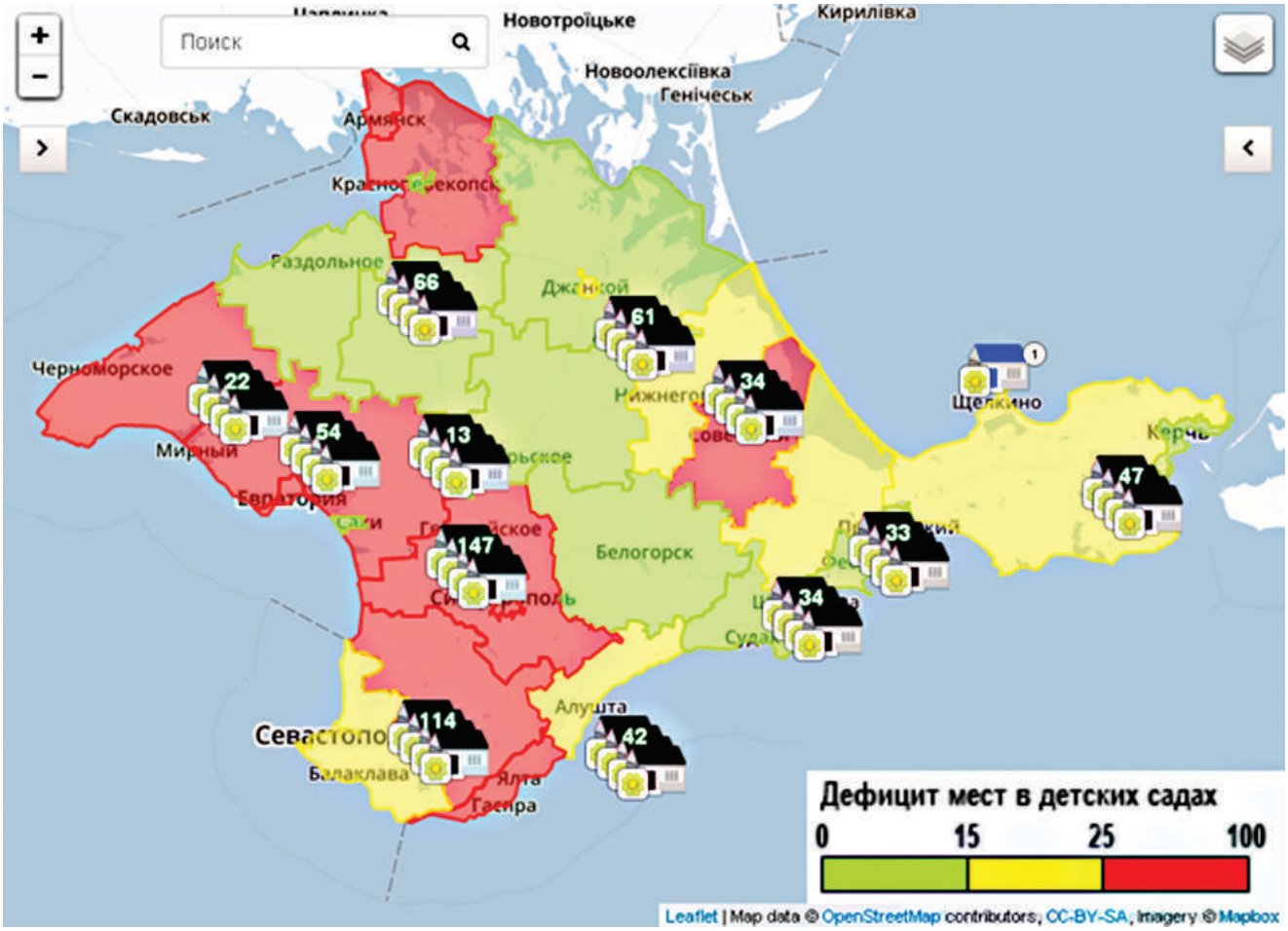

Рис. 4. Визуализация показателя «Дефицит мест в детских садах на сто человек возрастной группы 4–7 лет»
Fig. 4. Visualization of the index «The shortage of places in kindergartens for one hundred people in the age group of 4–7 years»

Заключение

Разработан модуль геоанализа для информационно-аналитической системы «Мониторинг» в рамках проекта Министерства образования и науки РФ по обследованию образовательных учреждений Республики Крым. Созданный модуль взаимодействует как с подсистемой исполнителя, так и с подсистемой заказчика и в полном объеме реализует поставленные заказчиком задачи.

ЛИТЕРАТУРА

1. *Боярчук М.А., Орлов П.Ю.* Обзор Информационно-аналитической системы «Мониторинг» // Славянский форум. 2015. № 4 (10). С. 61–71.

2. *Майоров А.А., Соловьёв И.В., Шкуров Ф.В., Дубов С.С.* Архитектурная концепция единого информационного пространства управления имущественным комплексом Минобрнауки России // Фундаментальные исследования. 2013. № 4–1. С. 53–56.

3. *Майоров А.А., Дубов С.С., Левина Н.И.* Об архитектуре системы автоматизированного мониторинга реализации мероприятий схемы территориального планирования Российской Федерации в области высшего образования. Сб. статей по итогам науч.-техн. конф. М.: МИИГАиК, 2014. № 7–1. С. 174–181.

4. *Боярчук М.А.* Разработка программного модуля ГИС «Мониторинг» для геоанализа объектов образования республики Крым: Дипломная работа. Защищена 03.06.2015, приказ № 55-06.05 от 20.07.2015 г. М.: МИИГАиК, 2015. 84 с.

5. Электронный ресурс: Документация API Яндекс.Карт. https://tech.yandex.ru/maps/doc/jsapi/2.1/quick-start/tasks/quick-start-docpage/

6. *Алексеева Т.В., Амириди Ю.В., Дик В.В., Лужецкий М.Г., Павлековская И.В., Хутинаев Д.А.* Информационные аналитические системы. М.: МФПУ Синергия, 2013. 384 с.

REFERENCES

1. *Boyarchuk M.A., Orlov P.Yu.* The review of an information analysis system «Monitoring». *Slavyanskiy forum*. Slavic forum. 2015, 4 (10): 61–71. [In Russian].

2. *Mayorov A.A., Solov'ev I.V., Shkurov F.V., Dubov S.S.* Architectural concept single information space property management Ministry of Education Russia. *Fundamental'nye issledovaniya*. Fundamental research. 2013, 4–1: 53–56. [In Russian].

3. *Mayorov A.A., Dubov S.S., Levina N.I.* On architecture of automatic monitoring system of plan execution regarding area planning scheme of the Russian Federation in a field of higher education. *Sbornik statey po itogam nauchno-tekhnicheskoy konferentsii*. Collection of articles on the results of the scientific and technical conference. Moscow: MIIGAiK, 2014, 7–1: 174–181. [In Russian].

4. *Boyarchuk M.A. Razrabotka programmnogo modulya GIS «Monitoring» dlya geoanaliza obektov obrazovaniya respubliki Krym. Diplomnaya rabota. Zashchishchena* 03.06.2015, *prikaz* № 55-06.05 ot 20.07.2015. Final project «Development of the GIS «Monitoring» software component for geoanalysis of educational facilities in Republic of Crimea»: Defended 03.06.2015, order No. 55-06.05 of 20.07.2015. Moscow: MIIGAiK, 2015: 84 p. [In Russian].

5. URL: Yandex.Maps API documentation. https://tech.yandex.ru/maps/doc/jsapi/2.1/quick-start/tasks/quick-start-docpage/

6. *Alekseeva T.V., Amiridi Yu.V., Dik V.V., Luzhetskiy M.G., Pavlekovskaya I.V., Khutinaev D.A. Informatsionnye analiticheskie sistemy.* Information analysis systems. Moscow: MFPU Sinergiya, 2013: 384 p. [In Russian].