УДК 004.9

Разработка сайта GPS-мониторинга объектов на местности

Искандарова Рената Маратовна
Сахалинский государственный университет
Студент

Кучер Людмила Владимировна
Сахалинский государственный университет
старший преподаватель кафедры информатики

Аннотация
В статье рассматривается процесс проектирования и разработки сайта GPS-мониторинга объектов на местности. Представлены фрагменты программного кода страниц сайта и скриншоты проекта.
Ключевые слова: веб-сайт, GPS-мониторинг, Яндекс-карты, база данных, программный код, скрипт.

Development of a website for GPS monitoring of objects on the territory

Iskandarova Renata Maratovna
Sakhalin State University
Student

Kucher Lyudmila Vladimirovna
Sakhalin State University
IT Department Lecturer

Abstract
The article discusses the process of designing and developing the site GPS-monitoring of objects on the ground. Presents pieces of code pages and screenshots of the project.
Keywords: web site, GPS monitoring, Yandex-maps, database, program code, script.

Развитие информационных технологий позволяет расширять сферу человеческих возможностей. Например, определить местоположение технического устройства собственника на местности, используя способность получать и отправлять свои GPS-координаты. В современных смартфонах предусмотрен GPS-приемник, позволяющий получать свои географические координаты, а принимать и обрабатывать GPS-координаты можно с помощью сайта.

Существует множество сайтов для определения местоположения объектов на местности, например, http://livegpstracks.com,

http://www.gpshome.ru, https://monitoring-gps.ru и другие. Проанализировав возможности данных сайтов определим недостатки: указывается местоположение только тех объектов, в которых предусмотрено наличие GPS-трекера, работа сайтов зависит от определенного ПО установленного на устройствах, имеющих GPS-приемник; услуги сайтов являются платными при использовании в коммерческих целях.

Разобьём процесс разработки сайта на этапы:

1) разработка пользовательского интерфейса с поддержкой интерактивных Яндекс.Карт;
2) создание базы данных хранящей GPS-координаты и другие параметры объекта;
3) разработка алгоритма приема и обработки входящих данных, содержащих GPS-координаты и другие параметры объекта.

Для разработки пользовательского интерфейса с поддержкой интерактивных Яндекс.Карт рассмотрим понятие «Модульная сетка». Модульная сетка в общем понимании -это решетка, состоящая из ячеек, где одна из них - модуль взята за основную единицу измерения, а остальные используются для указания примерного расположения объектов (рис.1).

Рисунок 1. Модульная сетка сайта

«Шапка» предназначена для размещения узнаваемых логотипов и других элементов для привлечения внимания пользователя и расположена в верхней области сайта.

«Подвал» используется для размещения контактной информации.

Основная область сайта используется для расположения следующих объектов:

1) поле для ввода номера телефона;
2) поле для ввода кода доступа;
3) кнопка для отправки данных;
4) интерактивная карта.

Для создания базы данных используется phpMyAdmin - веб-приложение с открытым кодом, написанное на языке PHP и представляющее собой веб-интерфейс для администрирования СУБД MySQL. PhpMyAdmin позволяет через браузер осуществлять администрирование сервера MySQL, запускать команды SQL и просматривать содержимое таблиц и баз данных. Приложение позволяет управлять СУБД MySQL без непосредственного ввода SQL команд, предоставляя дружественный интерфейс.

Рассмотрим элементы процесса создания базы данных:

На рисунке 2. Представлен результат создания базы данных с именем u0156816_Basa и кодировкой Юникод.

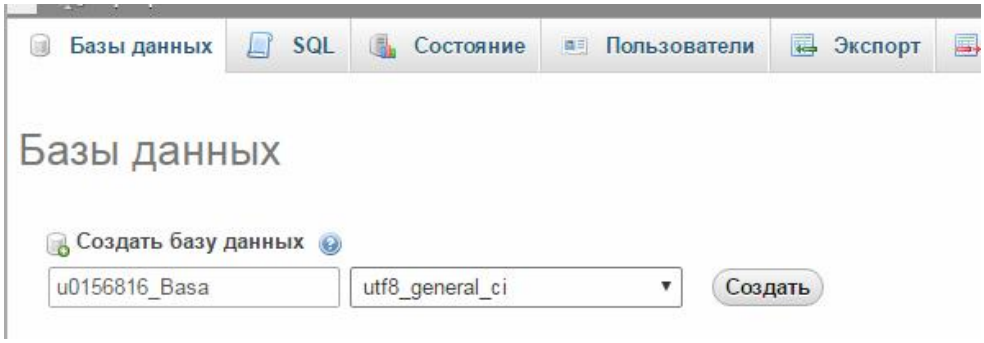

Рисунок 2. Создание базы данных с именем u0156816_Basa

На рисунке 3. представлен результат создания таблицы с именем spisok содержащей 5 столбцов.

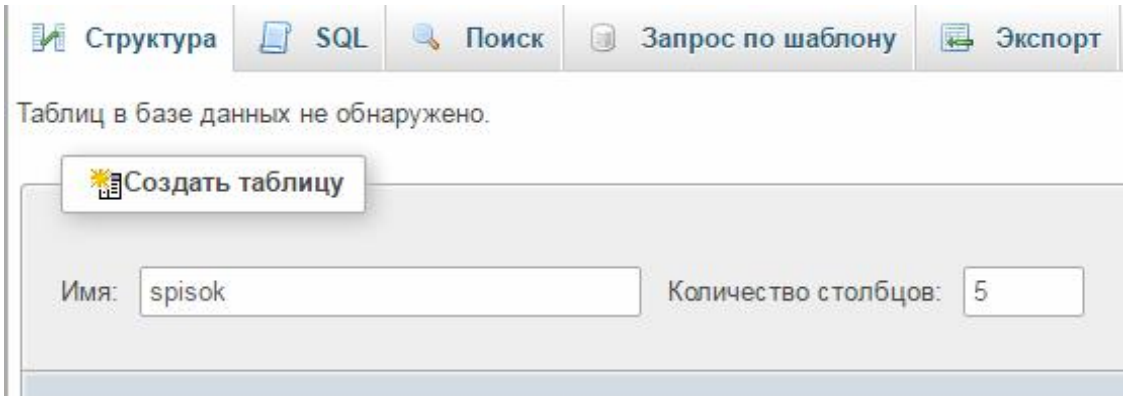

Рисунок 3. Создание таблицы с именем spisok

На рисунке 4 представлен результат описания столбцов таблицы spisok, где:

1) id-уникальный идентификатор;
2) number-номер телефона;
3) code-код доступа;
4) shir-географическая ширина;
5) dolg-географическая долгота.

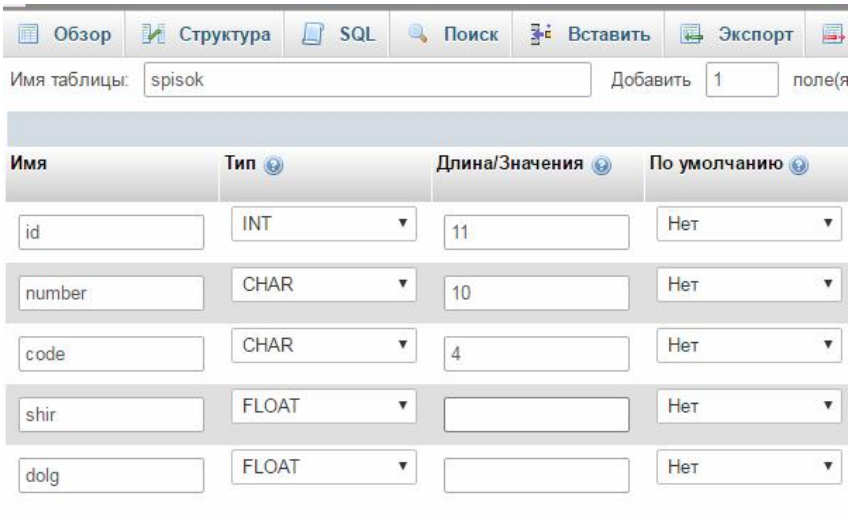

Рисунок 4. Описание столбцов таблицы spisok

Для проверки работоспособности базы необходимо добавить одну строку в таблицу spisok как показано на рисунке 5.

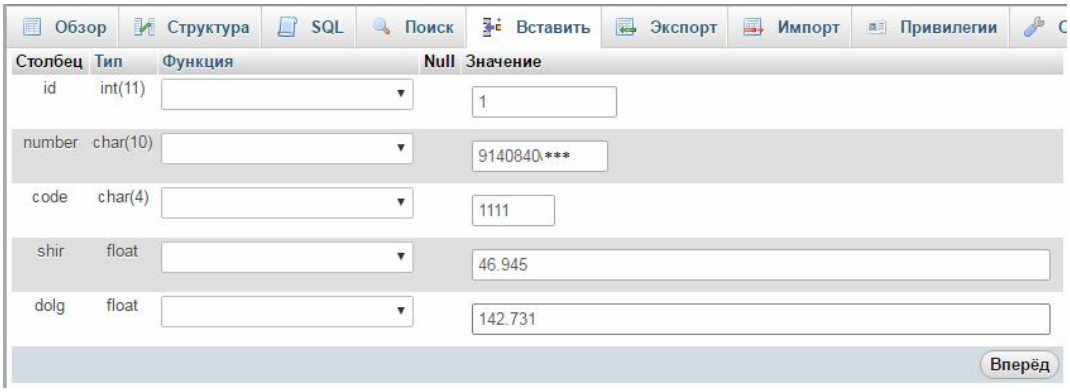

Рисунок 5. Добавление строки в таблицу spisok

Результат показан на рисунке 6.

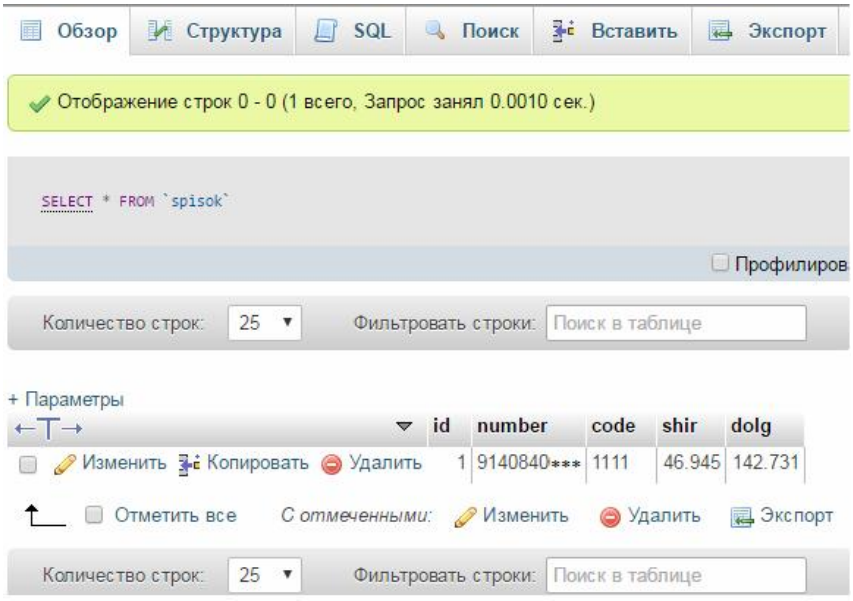

Рисунок 6. Результат запроса к базе данных

После отправки пользователем данных занесем данные в переменные.

Пример 1.

```
<?php
$number=$_POST['number'];
$code=$_POST['code'];
?>
```

В примере 1. переменной *$number* присваивается значение, которое ввел пользователь в текстовое поле для ввода номера, а переменной *$code* значение, которое пользователь ввел в текстовое поле для ввода кода доступа. Для получения информации из базы данных необходимо установить с ней соединение.

Пример 2.

```
$db=mysql_connect("localhost:3306","u0156_Basa","Sweetalinka
");
mysql_select_db("u0156816_Basa",$db);
```

В примере 2. показан код, необходимый для подключения к базе данных.

Алгоритм подключения:

1) соединиться с MySQL сервером и получить идентификатор.

Для того, чтобы подключиться к базе данных необходимо сначала соединиться с MySQL сервером. Для этого существует функция «*mysql_connect*», в которой указывается местонахождение этого сервера *(localhost:3306)*, пользователь, который имеет право работать с сервером(*u0156_Basa*) и пароль этого пользователя(*Sweetalinka*). Результат соединения можно занести в переменную, которая будет идентификатор подключения к MySQL серверу *($db)*;

2) выбрать базу данных.

На сервере может быть сразу несколько баз данных. За выбор БД отвечает функция «*mysql_select_db*». В качестве параметров этой функции указываются: имя базы данных(*u0156816_Basa*) и идентификатор подключения к серверу *($db)*.

После подключения к базе данных можно получить из нее данные удовлетворяющие запросу пользователя.

Пример 3.

```
$res=mysql_query("SELECT * FROM `spisok` WHERE
number='$number' AND code='$code'");
while($row = mysql_fetch_array($res)){
$shir=$row['shir'];
$dolg=$row['dolg'];}
```

В примере 3. функция *mysql_query()* посылает запрос активной базе данных сервера и возвращает указатель на результат. Этот запрос на выборку данных из таблицы *spisok*, где данные столбца *number* и *code* идентичны номеру и коду, которые ввел пользователь.

Функция *mysql_fetch_array()* возвращает значения в массиве с индексами по названию колонок. В примере 3. *mysql_fetch_array($res)*

возвращает в переменную *$row* либо непустой массив , либо FALSE. Далее *while* проверяет *$row*. Если *$row* непустой массив (если номер и код введен пользователем верно), то переменной *$shir* присваивается значение из столбца *shir*, а переменной *$dolg* присваивается значение из столбца *dolg*. Переменные *$shir* и *$dolg* это широта и долгота объекта, параметры которого ввел пользователь. Если FALSE, то осуществляется выход из цикла, а переменные *$shir* и *$dolg* остаются пустыми.

Используя полученную широту и долготу, можно указать местоположение на карте.

Пример 4.

```
if ($shir!=""){
echo "<script type='text/javascript'>
ymaps.ready(metka);
var myPlacemark;
function metka(){
myMap.setCenter([".$shir.",".$dolg."],17);
myPlacemark = new ymaps.Placemark([".$shir.",".$dolg."],{});
myMap.geoObjects.add(myPlacemark);}</script>";}
else{
if($_POST["number"]!=""){
echo "<font color='#ff0000'> неправильный номер или
код</font>";}}
```

В примере 4. сначала проверяется, получены ли данные, соответствующие параметрам, которые ввел пользователь. Если нет, то выводится сообщение, что пользователь ввел неверный номер или код *(echo " неправильный номер или код";)*. Если же данные получены, то используя полученные широту и долготу можно поставить метку на карте.

В API Яндекс.Карт метки реализуются с помощью класса Placemark. Перед тем как добавить метку на карту, необходимо создать экземпляр этого класса *(myPlacemark = new ymaps.Placemark ([".$shir.",".$dolg."],{});)*.

После того как метка была создана, ее можно добавить на карту. Добавление объектов на карту осуществляется через их добавление в глобальную коллекцию объектов карты *myMap.geoObjects(myMap.geoObjects.add(myPlacemark);)*.

В примере 4. для создания и добавления метки используется функция *metka()*.Создавать метку следует после того, как веб-страница и карта загрузится целиком. Чтобы инициализировать метку после загрузки страницы и карты, используется функция *ready()*.Строка *ymaps.ready(metka)* позволяет вызывать функцию *metka()* только после загрузки страницы и карты.

Для того чтобы метка позиционировалась на карте наиболее точно, необходимо установить новый центр карты и новый коэффициент масштабирования. В примере 4. для этого используется функция *myMap.setCenter([".$shir.",".$dolg."],17)*, она устанавливает центр

карты в точке с полученной ранее широтой и долготой и устанавливает коэффициент масштабирования равный 17.

В результате выполнения кода, указанного в примере 4. получаем метку с широтой и долготой, полученной из базы данных.

При правильном введении параметров карта принимает следующий вид (рис. 7)

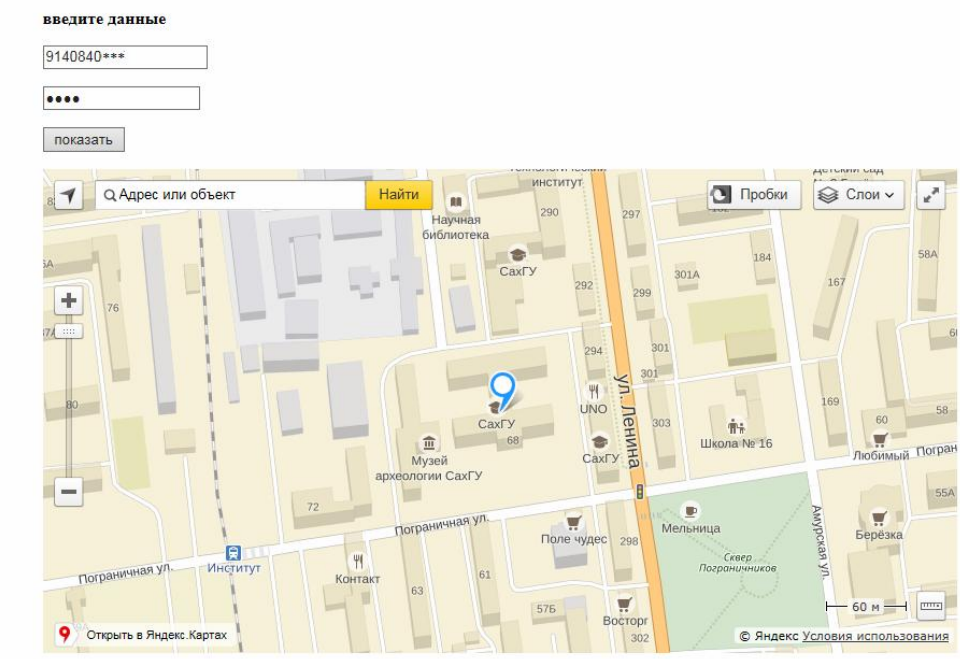

Рисунок 7. Вид карты при правильном введении параметров

При ошибочном введении параметров будет выведено сообщение об ошибке (рис. 8).

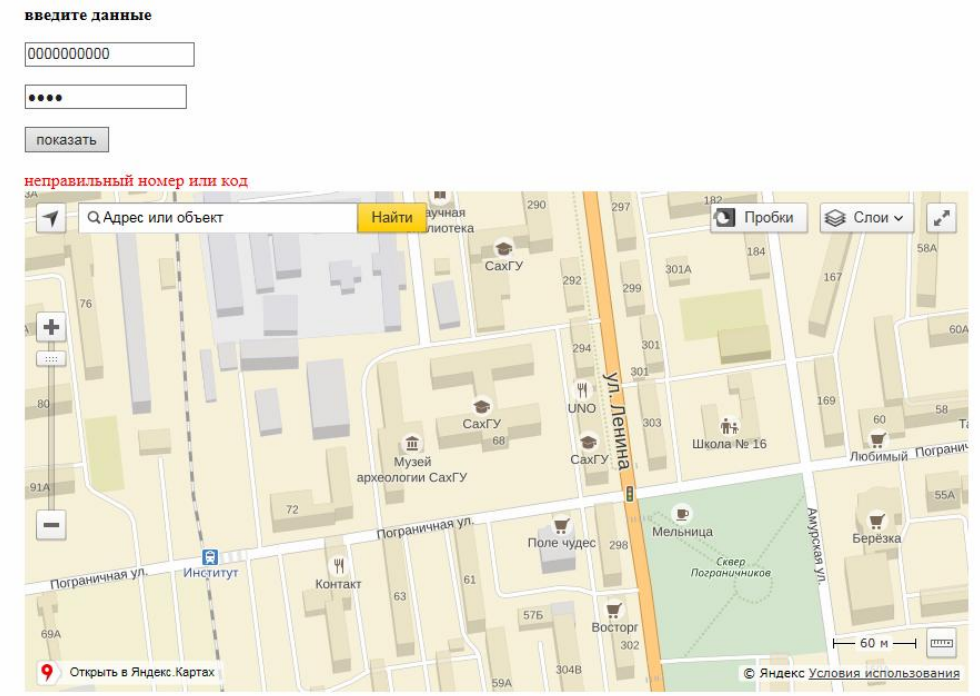

Рисунок 8. Вид карты при ошибочном введении параметров

Для приема и обработки данных используется код php. Прием и обработка данных скрыты от пользователя, поэтому нет необходимости прописывать структуру документа html. Файл для приема и обработки данных назовем request.php. Расширение .php применяется для того, чтобы браузер распознавал встроенный код php.

Данные отправляются в формате JSON. Для отправки данных можно использовать Postman, как показано на рисунке 9.

Рисунок 9. Отправка данных с помощью Postman

В адресной строке прописывается путь к файлу находящемуся на сервере. Также прописывается тип HTTP-запроса(POST) и выбирается тип отправляемых данных(JSON). Сами данные в формате JSON имеют вид как показано в примере 5.

Пример 5.

```
{
"number":"9140840***",
"code":"1234",
"shir":"46.9490",
"dolg":"142.7380"
}
```

В примере 5. ключи имеют следующие значения:

1) *"number"*-номер телефона без +7, в данном случае принимает значение *"9140840***"*;
2) *"code"*–код доступа, в данном случае принимает значение *"1234";*
3) *"shir"*-географическая ширина объекта, в данном случае принимает значение *"46.9490";*
4) *"dolg"*- географическая долгота объекта, в данном случае принимает значение *"142.7380".*

Данные отправляются в указанный файл, активируя прописанный в нем код php.

Пример 6.

```
<?php
$data = file_get_contents('php://input');
?>
```

В примере 6. показан необходимый код для принятия потока данных извне, в результате выполнения которого, значением переменной *$data* становится объект формата JSON.

Для использования данных из объекта формата JSON необходимо преобразовать его в переменную PHP.

Пример 7.

```
<?php
$obj = json_decode ($data,true);
?>
```

В примере 7. показан необходимый код для преобразования объекта формата JSON в переменную PHP. Функция *json_decode()* принимает закодированную в JSON строку и преобразует ее в переменную PHP. Параметры функции:

1) *$data* - json строка для декодирования;
2) *True* - указывает на то что возвращаемые объекты будут преобразованы в ассоциативные массивы.

В результате выполнения кода из примера 7, значением переменной *$obj* становится ассоциативный массив, где индексами будут являться ключи объекта формата JSON.

Пример 8.

```
<?php
$code=$obj["code"];
$number=$obj["number"];
$shir=$obj["shir"];
$dolg=$obj["dolg"];
?>
```

В примере 8. присваиваются значения переменным PHP, отправленным через Postman:

1) *$code* - кода доступа *("1234")*;
2) *$number* - номер телефона без +7 *("9140840***")*;
3) *$shir* - географическая широта объекта *("46.9490")*;
4) *$dolg* - географическая долгота объекта *("142.7380")*.

После получения новых данных их необходимо занести в базу данных, если данные с таким значением `number` отправлены впервые, или обновить данные, если данные с таким значением `number` уже существуют в базе данных.

Для того чтобы добавить или обновить данные в базе данных необходимо установить с ней соединение.

Пример 9.

```
<?php
$db=mysql_connect("localhost:3306","u0156_Basa","Sweetalinka
");
mysql_select_db("u0156816_Basa",$db);
?>
```

В примере 9. показан код необходимый для подключения к базе данных. Алгоритм подключения:

1) соединиться с MySQL сервером и получить идентификатор.

Для того, чтобы подключиться к базе данных необходимо сначала соединиться с MySQL сервером. Для этого существует функция *«mysql_connect»*, в которой указывается местонахождение этого сервера(*localhost:3306*), пользователь, который имеет право работать с сервером(*u0156_Basa*) и пароль этого пользователя(*Sweetalinka*). Результат соединения можно занести в переменную, которая будет идентификатор подключения к MySQL серверу *($db);*

2) выбрать базу данных.

На сервере может быть сразу несколько баз данных. За выбор БД отвечает функция *«mysql_select_db»*. В качестве параметров этой функции указываются: имя базы данных(*u0156816_Basa*) и идентификатор подключения к серверу *($db)*.

После подключения к базе данных, можно узнать существует ли такая запись в базе данных, где значение столбца `number` и значение столбца `code` совпадает со значением переменных `$number` и `$code`.

Пример 10.

```
<?php
$result =mysql_query("SELECT * FROM `spisok` WHERE
number='$number' AND code='$code'");
?>
```

В примере 10. функция *mysql_query()* посылает запрос активной базе данных сервера и возвращает указатель на результат. Этот запрос на выборку данных из таблицы *spisok*, где данные столбца *number* и *code* идентичны переменным *$number и $code*. Результатом выборки может являться либо одна строка, либо пустое значение.

Пример 11.

```
<?php
if (mysql_num_rows($result)) {
mysql_query("UPDATE `spisok` SET shir='$shir', dolg='$dolg'
WHERE number='$number' AND code='$code'");}
?>
```

В примере 11. функция *mysql_num_rows()* возвращает количество рядов результата запроса. Значением *mysql_num_rows($result)* может быть либо 1, либо 0, так как результатом выборки из примера 11 может являться либо одна строка, либо пустое значение.

Обновление столбцов строки осуществляется в том случае, если значение *mysql_num_rows($result)* равно 1(строка существует*)*. В примере 11. функция *mysql_query()* посылает запрос активной базе данных сервера. Этот запрос с использованием оператора UPDATE. Оператор UPDATE обновляет столбцы в соответствии с их новыми значениями в строках существующей таблицы. В примере 11. данные в столбце *shir* и *dolg* из таблицы *spisok*, где данные столбца *number* и *code* идентичны переменным *$number и $code,* заменяются на значения переменных *$shir и $dolg*.

В случае, когда значение `mysql_num_rows($result)` равно 0(строка не существует), необходимо добавить в базу данных новую строку.

Пример 12.

```
<?php
else {
$res=mysql_query('select * from `spisok`');
$id = mysql_num_rows($res)+1;
mysql_query("INSERT INTO `spisok`(`id`, `number`, `code`,
`shir`, `dolg`) VALUES
('$id','$number','$code','$shir','$dolg')");}
?>
```

В примере 12. показан код выполняемый, когда значение *`mysql_num_rows($result)`* равно 0(строка не существует). Для корректного добавления новой строки в базу данных, необходимо определить уникальный идентификатор(id) новой записи. Уникальный идентификатор новой записи будет равен количеству строк в таблице, увеличенному на 1`($id = mysql_num_rows($res)+1;)`.

В примере 12 функция *`mysql_query()`* посылает запрос активной базе данных сервера. Этот запрос с использованием оператора INSERT. Оператор INSERT вставляет новые строки в существующую таблицу. В примере 12. с помощью запроса, в таблицу *`spisok`* добавляется новая строка с такими значениями столбцов:

1) *`id`* – принимает значение переменной *`$id`*;
2) *`number`* – принимает значение переменной *`$number`*;
3) *`code`* – принимает значение переменной *`$code`*;
4) *`shir`* – принимает значение переменной *`$shir`*;
5) *`dolg`* – принимает значение переменной *`$dolg`*.

При отправке данных через Postman как показано на рисунке 10, в таблицу *`spisok`* добавится одна строка, так как в таблице нет такой строки, где значение столбца `number` и значение столбца `code` совпадает со значением переменных `$number` и `$code`. На рисунке 10. показана таблица `spisok` после добавления новой строки.

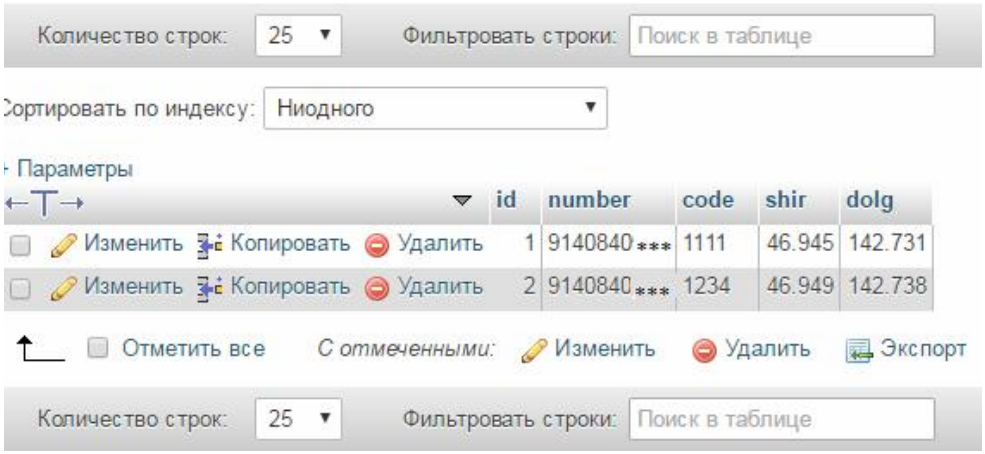

Рисунок 10. Добавление строки в таблицу *`spisok`* после отправки новых данных

При правильном введении параметров на сайте карта принимает вид (рис.11). В качестве параметров используются данные, которые были добавлены в базу данных ранее с использованием Postman.

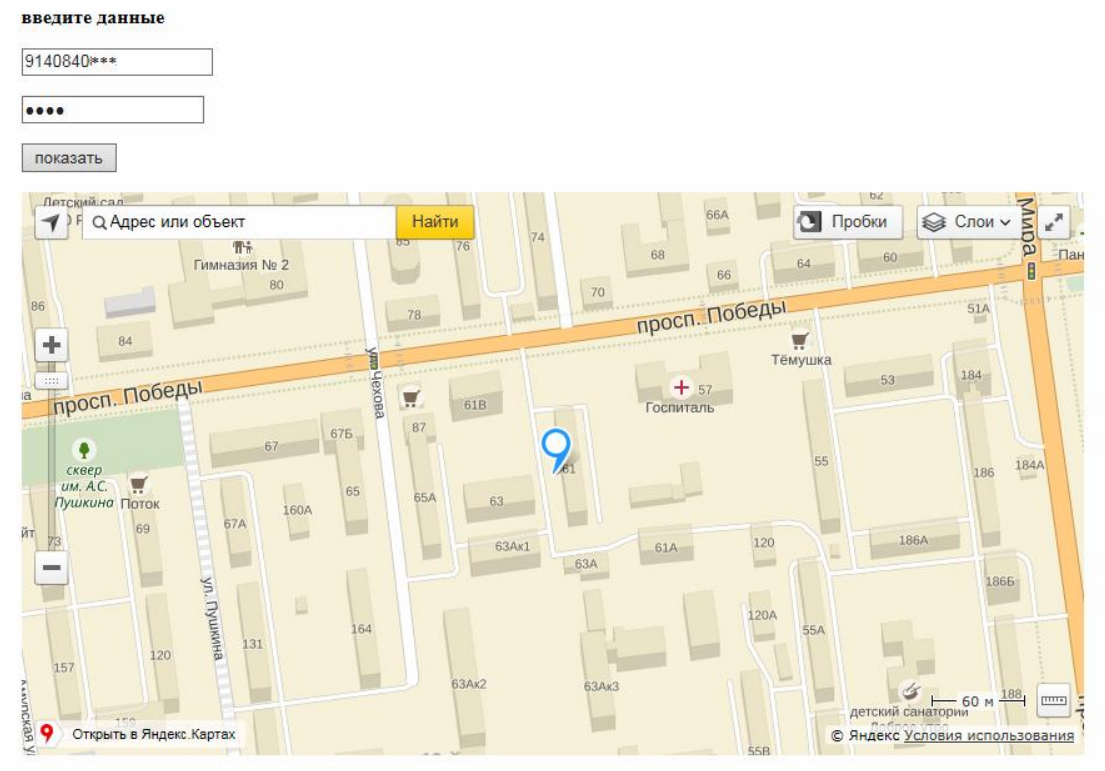

Рисунок 11. Вид карты при отображении принятых данных

При повторной отправке данных через Postman с тем же номером и кодом, но с другими географическими координатами (рис. 12), данные в базе обновляются (рис. 13), и положение метки на карте изменяется (рис. 14).

Рисунок 12. Отправка данных через Postman с тем же номером и кодом, но с другими географическими координатами

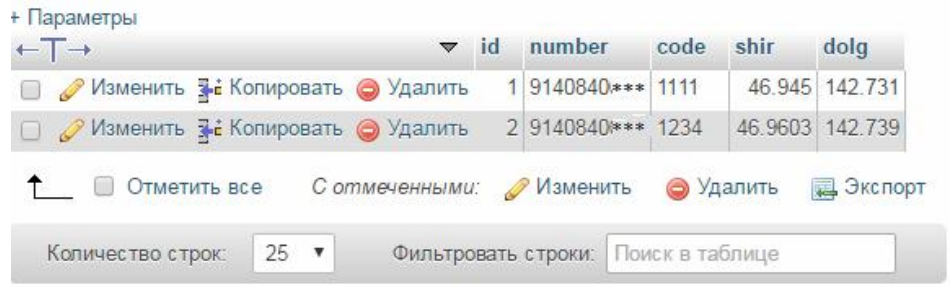

Рисунок 13. База данных с обновленными данными в строке

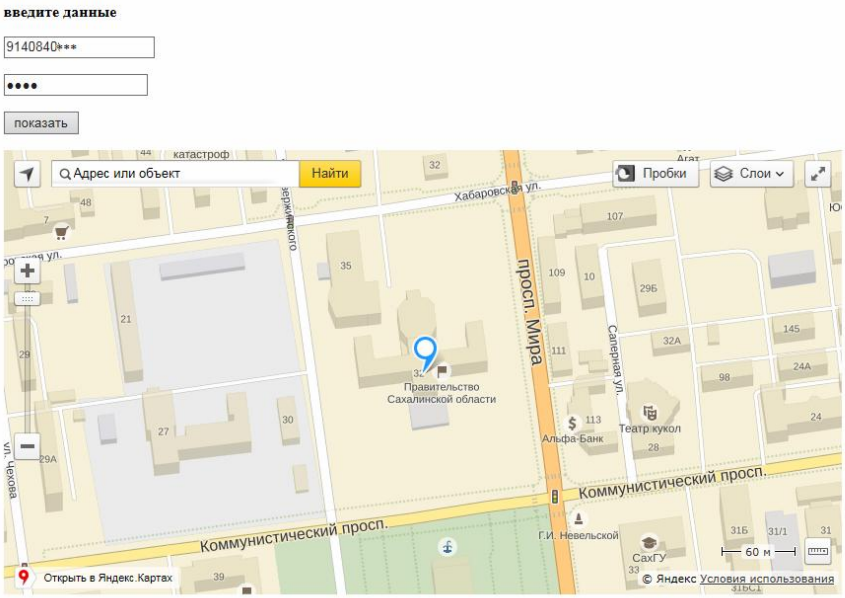

Рисунок 14. Вид карты после обновления данных

Библиографический список

1. Бабаев А., Евдокимов Н., Боде М. Создание сайтов. СПб.: Питер, 2014. 300 с.
2. Сырых Ю.А. Современный веб-дизайн. Вильямс, 2013. 276 с.