УДК 004

Построение оптимальных маршрутов передвижения с помощью JavaScript API Яндекс карт

Андриенко Иван Сергеевич
Приамурский государственный университет имени Шолом-Алейхема
студент

Вихляев Дмитрий Романович
Приамурский государственный университет имени Шолом-Алейхема
студент

Глаголев Владимир Александрович
Приамурский государственный университет имени Шолом-Алейхема
К.г.н., доцент, доцент кафедры информационных систем, математики и правовой информатики

Аннотация
В статье проведено анализ использования возможностей API Яндекс.карт для проектирования системы пространственных услуг, позволяющей определять различных виды маршрутов.
Ключевые слова: предприятия, маршрут, клиенты.

Building optimal routes using the Yandex Maps JavaScript API

Andrienko Ivan Sergeevich
Sholom-Aleichem Priamursky State University
Student

Vikhlyaev Dmitry Romanovich
Sholom-Aleichem Priamursky State University
Student

Glagolev Vladimir Aleksandrovich
Sholom-Aleichem Priamursky State University
PhD, Associate Professor, Associate Professor of the Department of Information Systems, Mathematics and Legal Informatics

Abstract
The article analyzes the use of the capabilities of the Yandex.maps API for designing a system of spatial services that allows you to define different types of routes.
Keywords: business, store, client.

1 Введение

1.1 Актуальность

В настоящее время онлайн карты широко используются пользователями. Карты универсально передают информацию и вносят ясность в формы нашего мира. С их помощью даже турист может ориентироваться в мегаполисе. В данной статье можно будет увидеть способ, при котором пользователь может запомнить все точки около тех объектов к которым он хочет попасть, после чего программа составит для него оптимальный маршрут ко всем этим точкам.

1.2 Обзор исследований

М.М. Милихин исследовал разработку WEB-ориентированных ГИС с применением ESRI ArcGIS JavaScript API и "Dojo JavaScript toolkit" [1]. В работе В.А. Зотов изучил реализацию языка JavaScript AJAX и Node.js в результате, при обновлении данных веб-страница не перезагружается полностью, и веб-приложения становятся быстрее и удобнее [2]. В статье Д.А. Веселов, П.В. Михалев освещается принцип работы приложения для визуализации карты [3]. В.А. Мохов, В.Н. Кубил, А.В. Кузнецова, И.В. Георгица отразили в работе тенденцию ориентации современных ИТ-вендоров в направлении развития сервисов геоинформационных систем [4].

1.3 Цель исследования

Цель исследования - создание оптимального маршрута с помощью JavaScript API Яндекс карт.

2 Материалы и методы

Для проведения данного исследования использовался находящийся в открытом источнике сервис проекта Яндекс карты.

3 Результаты и обсуждение

Этот сервис является частью проекта Яндекс карты и черпает информацию именно оттуда. Конструктор карт позволяет наносить маршрут, измерять расстояния, рисовать многоугольники, наносить точки на нужный вам участок карты. Маркеры здесь могут задаваться разного вида и каждый можно сопроводить номером и комментарием, который будет появляться при щелчке по точке.

Возможности сервиса наилучшим образом подходят для создания интерактивных карт, демонстрирующих, например, расположение вашей фирмы или магазина, маршрут движения или точку сбора на общественное мероприятие. Также можно получить информацию об объектах. Например, рестораны, можно узнать время работы или даже меню. Для памятников можно узнать описание памятника и т.д. Всё это осуществимо благодаря электронным картам, в нашем случае Яндекс картам.

В этой статье можно будет увидеть способ, при котором пользователь может запомнить все точки около тех объектов к которым он хочет попасть, после чего программа составит для него оптимальный маршрут ко всем этим точкам.

Для получения API ключа необходимо зарегистрироваться в личном кабинете Яндекса, зайти в кабинет разработчика и нажать на кнопку получить ключ.

Чтобы подключить API к странице сайта добавьте в заголовок head HTML-страницы строку следующего вида:

```
<head>
    <script src="https://api-maps.yandex.ru/2.1/?apikey=ваш API-ключ&lang=ru_RU" type="tex
t/javascript">
    </script>
</head>
```

Листинг 1

Затем создайте видимый контейнер ненулевого размера, в котором будет размещена карта. В качестве контейнера может использоваться любой HTML-элемент блочного типа (например, элемент div). Карта заполнит этот элемент полностью.

```
<body>
    <div id="map" style="width: 600px; height: 400px"></div>
</body>
```

Листинг 2

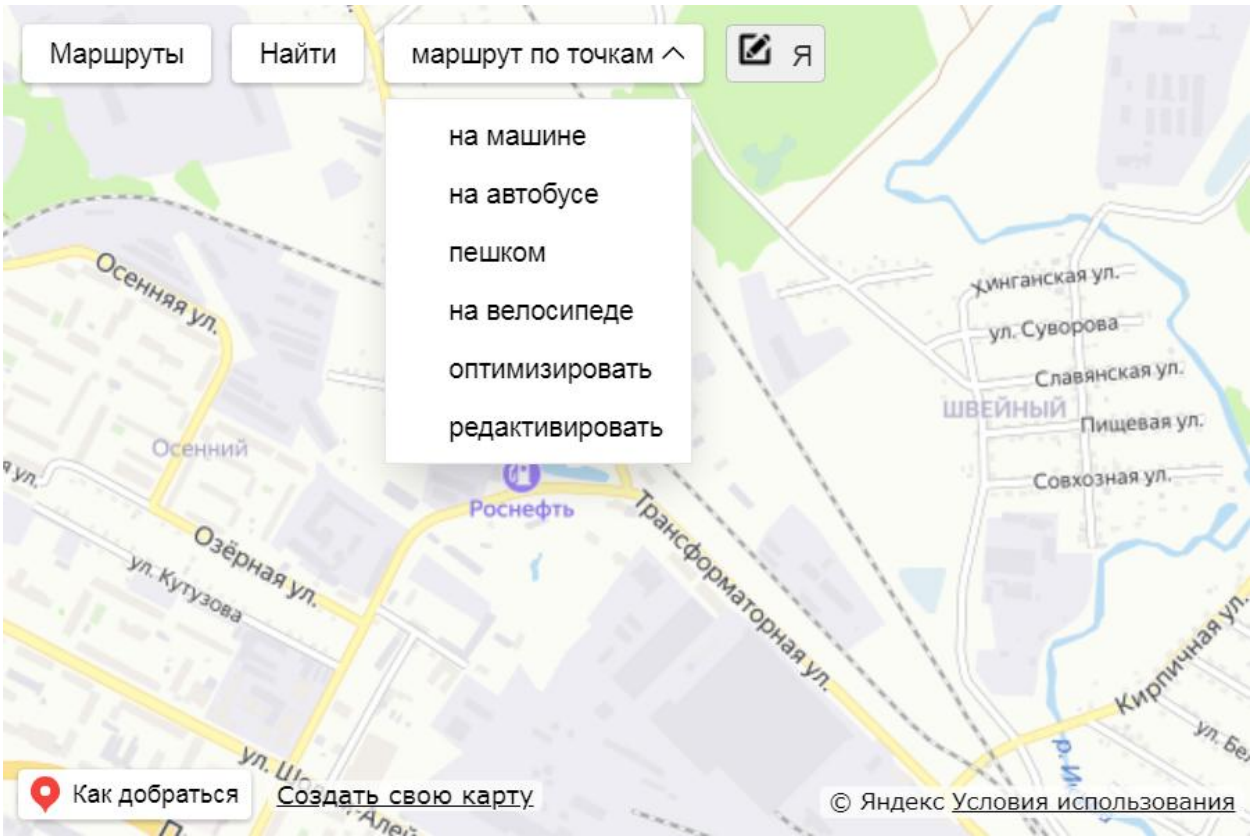
Для создание карты в коде JavaScript конструктору нужно передать:

- идентификатор HTML-контейнера;
- центр карты;
- коэффициент масштабирования.

```
<script type="text/javascript">
    // Функция ymaps.ready() будет вызвана, когда
    // загрузятся все компоненты API, а также когда будет готово DOM-дерево.
    ymaps.ready(init);
    function init(){
        // Создание карты.
        var myMap = new ymaps.Map("map", {
            // Координаты центра карты.
            // Порядок по умолчанию: «широта, долгота».
            // Чтобы не определять координаты центра карты вручную,
            // воспользуйтесь инструментом Определение координат .
            center: [55.76, 37.64],
            // Уровень масштабирования. Допустимые значения:
            // от 0 (весь мир) до 19.
            zoom: 7
        });
    }
</script>
```

Листинг 3

Создадим список кнопок для манипулирования системой. В нём будут находиться, выбор транспорта для построения маршрута и возможность его изменить.



Листинг 4

```
    // Создадим 5 пунктов выпадающего списка.
    var listBoxItems = ['на машине', 'на автобусе', 'пешком', 'на велосипеде',"оптимизировать",'редактивировать']
            .map(function (title) {
                return new ymaps.control.ListBoxItem({
                    data: {
                        content: title
                    },
                    state: {
                        selected: false
                    }
                })
            }),
        reducer = function (filters, filter) {
            filters[filter.data.get('content')] = filter.isSelected();
            return filters;
        },
        // Теперь создадим список, содержащий 5 пунктов.
        listBoxControl = new ymaps.control.ListBox({
            data: {
                content: "маршрут по точкам",
                title: 'Фильтр'
            },
            items: listBoxItems,
            state: {
                // Признак, развернут ли список.
                expanded: false,
                filters: listBoxItems.reduce(reducer, {})
            }
        });
    myMap.controls.add(listBoxControl);
```

Листинг 5

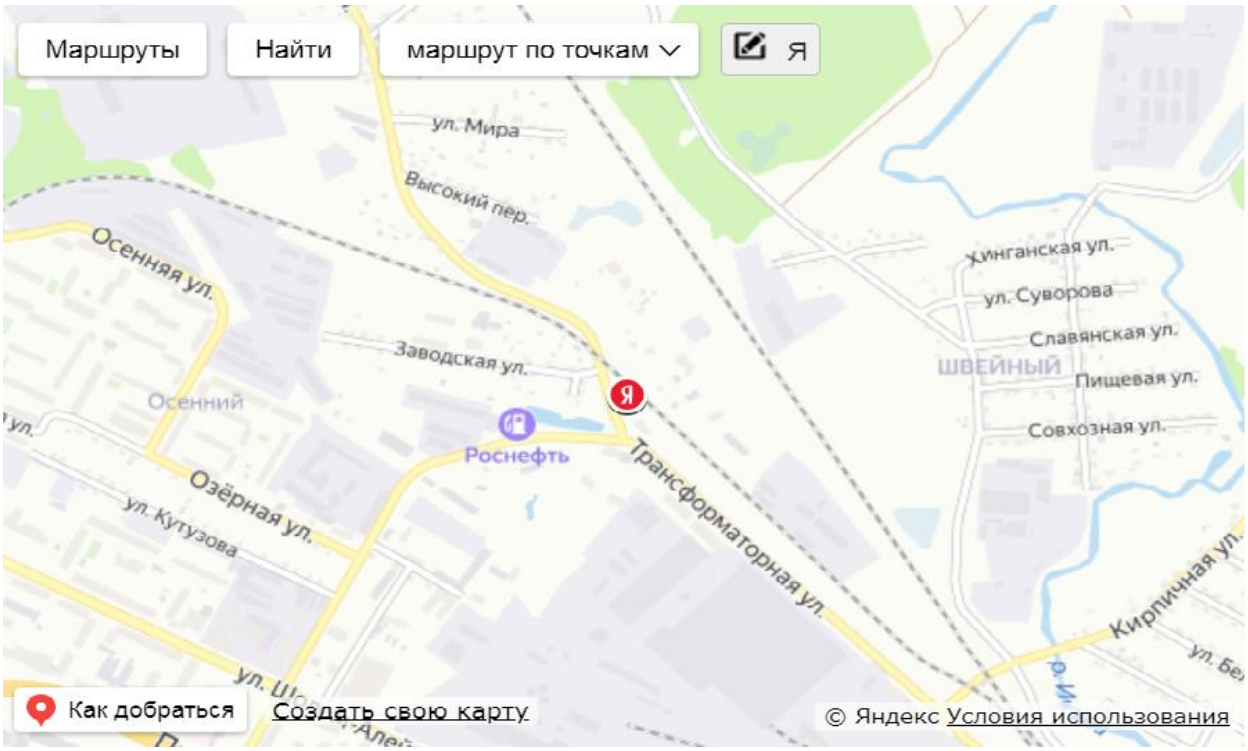
Далее вводим новые переменные и создаём условия для каждой кнопки

```
    // Добавим отслеживание изменения признака, выбран ли пункт списка.
    var x;
    var travel=false;
    var rad=false;
    var optim=false;
    listBoxItems[0].events.add(['select', 'deselect'], function (e) {
        var eType = e.get('type');
        if(eType == 'select'){
        x="auto";
        travel=true;}
        else{
        travel=false;}
    });
     listBoxItems[1].events.add(['select', 'deselect'], function (e) {
        var eType = e.get('type');
        if(eType == 'select'){
        x="masstransit";
        travel=true;}
        else{
        travel=false;}
    });
     listBoxItems[2].events.add(['select', 'deselect'], function (e) {
        var eType = e.get('type');
        if(eType == 'select'){
        x="pedestrian";
        travel=true;}
        else{
        travel=false;}
    });
     listBoxItems[3].events.add(['select', 'deselect'], function (e) {
        var eType = e.get('type');
        if(eType == 'select'){
        x="bicycle";
        travel=true;}
        else{
        travel=false;}
    });
     listBoxItems[4].events.add(['select', 'deselect'], function (e) {
        var eType = e.get('type');
        if(eType == 'select'){
        optim=true;
        travel=false;}
    });

    button.events.add(['select'], function (e) {
```

Листинг 6

Создаём функцию Location.get(), которая будет находить геоданные пользователя, в ней разрешаем переместить карту так чтобы в центре находился пользователь. Получаем его координаты и адрес. Создаём болун и вносим в него информацию. Так при клике на метку своего местонахождения, пользователь мог увидеть адрес улицы.



Листинг 7

```
    location.get({
        mapStateAutoApply: true
    })
.then(
    function(result) {
        // Получение местоположения пользователя.

        var userAddress = result.geoObjects.get(0).properties.get('text');
        var userCoodinates = result.geoObjects.get(0).geometry.getCoordinates();
        // Пропишем полученный адрес в балуне.

        result.geoObjects.get(0).properties.set({
            balloonContentBody: 'Адрес: ' + userAddress +
                                '<br/>Координаты:' + userCoodinates
    });
```

Листинг 8

Не выходя из функции, создаём два массива, один для координат точек до которых нужно добраться, другой для тех же точек, но уже отсортированных в том порядке, при котором маршрут будет оптимален. Помимо них нужно создать переменную которая будет измерять количество заданных точек.

После чего объявим событие .events.add, в параметре поставим 'click', что значит что событие будет происходить при клике на карту.

Там объявим переменную coords, она будет считывать координаты той точки куда кликнули, после создание метки myPlacemark, и присвоим ей координаты coords. А также эти координаты присвоим первому массиву, начиная с индекса 1, так как в индексу 0, присвоятся координаты местонахождения пользователя, тесть arr[0] – это точка старта будущего маршрута.

```
         var arr=new Array(); //массив для координат x
        var but=false; //проверка на клик
        var j=1;
        var arr2= new Array();
          myMap.events.add(['click'], function (e) {
              arr[0]=userCoodinates;
              var eType = e.get('type');
               but=false;
         var coords = e.get('coords');
        if(/*eType == 'click'*/ travel) { //если клик то but=true, i+=1
            but=true;
            arr[j]= coords;
            j++;
        }
          if(but==true && /*j<=5*/ !optim){  //если but=true и i<5 ставим метку
        var myPlacemark = new ymaps.Placemark([coords[0],coords[1]]);
            myMap.geoObjects.add(myPlacemark);
        }
```

Листинг 9

Метки, продолжат ставиться, пока пользователь не нажмёт на кнопку оптимизировать. После этого когда пользователь кликнет на карту метки не появятся, так как будет происходить другая операция. В этой части программы будут искаться ближайшие точки к местоположению пользователя, используя цикл и новые переменные min2 и min1 которые отвечают за начало и ближайшую к началу точку маршрута соответственно.

```
            if(optim){
            optim=false;
         var arPlacemarks = new Array();
        for (i=0; i<arr.length-1; i++)
  arPlacemarks[i] = new ymaps.Placemark(arr[i+1]);

            var min2=arr[0];
            arr2[0]=arr[0];
            for(var k=0; k<j-1; k++){

        var arPlacemarksRez = ymaps.geoQuery(arPlacemarks).addToMap(myMap).setOptions('visible', true);

        //ближайшая точка к клику
        var closestObject = arPlacemarksRez.getClosestTo(min2);
            var min1=closestObject.geometry.getCoordinates();


```

Листинг 10

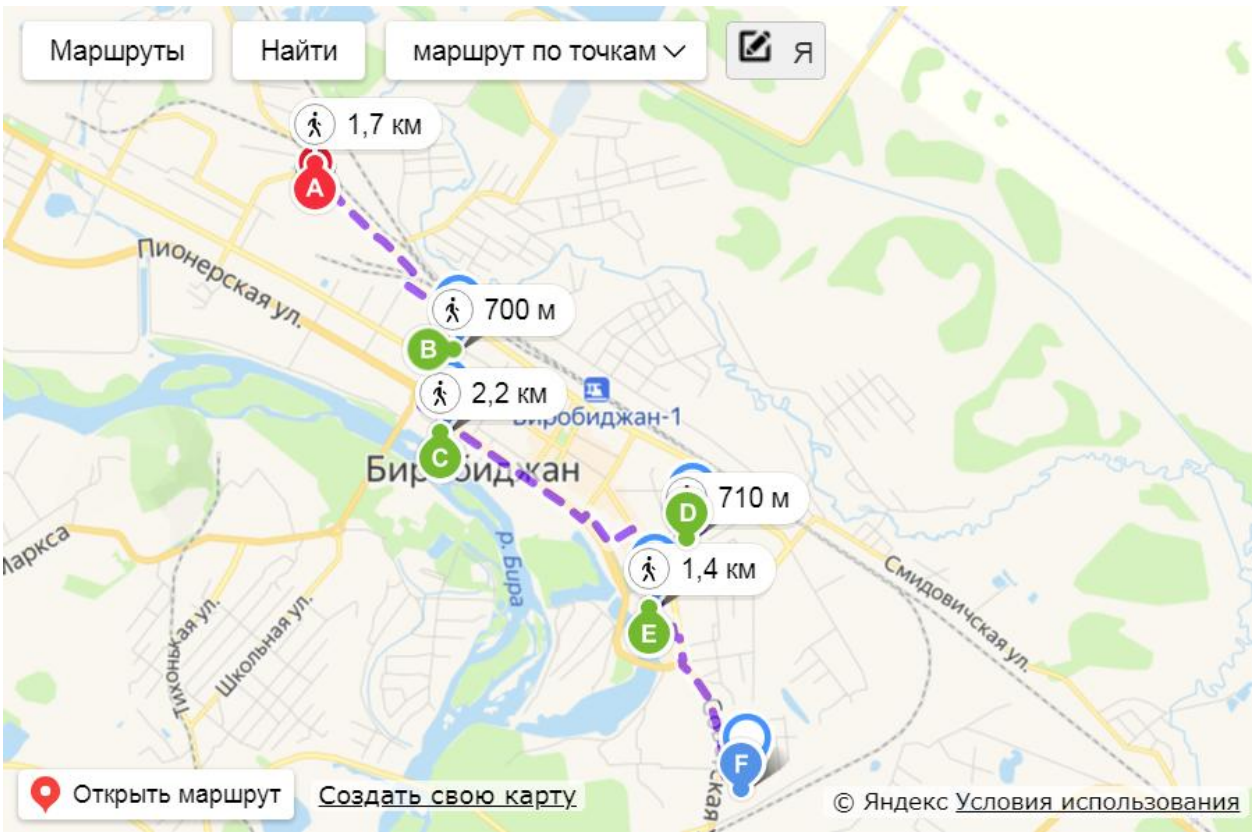
С помощью функции route, создаётся маршрут между двумя точками min2 и min1, выключим возможность отобразить его, так как он нужен лишь для вычислений порядка, в котором будут располагаться точки во втором массиве –arr2. После нахождения следующей порядковой точки, с помощью цикла будем удалять из массива arPlacemarks те точки, которые уже отсортированы, чтобы их не учитывала система в следующей итерации. Точке отправления min2 будут передаваться координаты предыдущей точки прибытия min1.

```
                ymaps.route([min2, min1]).then(
    function (route) { //функция выводит маршрут между двумя точками
        //myMap.geoObjects.add(route);
    });
        min2=min1;
        arr2[k+1]=min1;
                for(var l=0; l<arr.length;l++){
                    if(min1==arr[l]){
                        arPlacemarks.splice(l-1,1,"o");
                    }
                    }
                }
```

Листинг 11

После окончания цикла мы имеем массив arr2, в котором все точки расположены в том порядке в котором маршрут будет оптимален. С помощью функции multiRouter.MultiRoute, которая строит маршрут через множество точек. Выводим оптимальный маршрут, положив в функцию целый массив arr2, функция сама заглянет в каждую ячейку массива, в том порядке, в котором они пронумерованы и выведет его на карте. Добавим параметр params, в нём отметим количество маршрутов results и тип маршрутизации routingMode, которой присваивается значение переменной – x. Переменной x, присваивается значение из списка кнопок, которое раннее было создано. Возможные варианты маршрутизации (Пешеходный, на общественном транспорте, на автомобиле и на велосипеде),

Так же была создана кнопка «редактирования» при активности она принимает значение – true, иначе – false. Если значение true, то маршрут может редактировать сам пользователь, например добавить или удалить путевую точку или перенести её в другое место.



Листинг 12

```
            var multiRoute = new ymaps.multiRouter.MultiRoute({
        referencePoints: arr2,

        // Параметры маршрутизации.
        params: {
            // Ограничение на максимальное количество маршрутов, возвращаемое маршрутизатором.
            results: 1, routingMode: x
        }


       },{// Тип промежуточных точек, которые могут быть добавлены при редактировании.
        editorMidPointsType: "via",
        // В режиме добавления новых путевых точек запрещаем ставить точки поверх объектов карты.
        editorDrawOver: true,
            wayPointDraggable: true,
            boundsAutoApply: true});

            listBoxItems[5].events.add(['select', 'deselect'], function (e) {
        var eType = e.get('type');
        if(eType == 'select'){
         multiRoute.editor.start({
            addWayPoints: true,
            removeWayPoints: true
        });}
        else{
         multiRoute.editor.stop();}
    });


            myMap.geoObjects.add(multiRoute);
        }
         });
        myMap.geoObjects.add(result.geoObjects)
    },
    function(err) {
        console.log('Ошибка: ' + err)
    }

);
    });
}
```

Листинг 13

Таким образом, в данной статье было описано создание оптимального маршрута с помощью JavaScript API Яндекс карт. Построение оптимальных

маршрутов экономит время, появляется доступ к объектам, возможно осуществление планирования деятельности различных сфер.

Библиографический список

1. Милихин М.М. разработка WEB-ориентированных ГИС с применением ESRI ArcGIS JavaScript API и "Dojo JavaScript toolkit" // Научная сессия ТУСУР-2011. 2011. С. 1-2.
2. Зотов В.А. Реализация языка JavaScript AJAX и Node.js // Вестник МГУП Имени Ивана Федорова. 2013.
3. Веселов Д. А., Михалев П. В. Визуализация карты Урала средствами javascript //Молодежная наука в развитии регионов. 2019. Т. 1. С. 32-35.
4. Мохов В.А., Кубил В.Н., Кузнецова А.В., Георгица И.В. Рекурсивный алгоритм синхронизации API-запросов к ГИС-сервису Яндекс карты // Фундаментальные исследования. 2015. Т. 1. №. 9.
5. Muller J. M. Elementary Functions and Approximate Computing //Proceedings of the IEEE. 2020. Т. 108. №. 12. С. 2136-2149.
6. Cormen T. H. Leiserson C. E. Rivest R. L. Stein C. Introduction to Algorithms. MIT press, 2016. 105 с.