# Manejo de Mapas en Páginas Web con OSM y Leaflet

Alejandro Schmidt, *ajoscram@gmail.com*

***Abstract*—OpenStreetMap (también conocido por sus siglas OSM) es uno de los más populares servicios de mapas online de internet y un proyecto encaminado a la creación, publicación y visualización de mapas para su uso en diferentes plataformas y dispositivos. Es de colaboración de código abierto y libre, en el que puede participar y aportar cualquier persona que cuente con una conexión a Internet con la que poder subir los mapas editados. Los mapas de OpenStreetMaps pueden mostrarse en un sitio WEB, insertándolos como un elemento o a través de librerías gratuitas de JS como Leaflet; que es de código abierto y despliega mapas interactivos con gran facilidad de implementación.**

***Index Terms*—McRuta, Javascript, HTML, CSS, JSON, OpenStreetMap, Leaflet, Routing Machine**

## I. Introducción

EL presente artículo documenta el trabajo realizado para el laboratorio 2 del curso de Introducción al Desarrollo Web en modalidad verano con la profesora Ericka Solano. El proyecto consiste en marcar algunos puntos en OSM que representen sitios representativos de un contexto, por ejemplo: puntos de hidratación en una carrera para la que se ha definido una ruta, paradas intermedias de una ruta de un servicio de transporte público, puntos donde se encuentran pistas en un rally, etc. Se debe realizar una experimentación editando puntos representativos usando OpenStreetMaps con el objetivo de que éstos sean identificados claramente a la hora de definir la ruta. Posteriormente, los datos de la ruta serán definidos en un archivo JSON, para ser usados en una página donde se muestre un mapa que contiene lugares específicos que fueron marcados desde OSM.

## II. Contexto

El contexto seleccionado para este trabajo es la **Mc Ruta**, una carrera ficticia propuesta por McDonalds donde los participantes recorren a pie varios McDonalds en el Valle Central y comer un Big Mac en cada uno de ellos. Se seleccionó este tema porque como hay muchos McDonalds en el área era fácil trazar una ruta entre ellos lo suficientemente larga para ilustrar lo que se puede hacer con las librerías seleccionadas. El criterio de selección de puntos fue simplemente que crearan entre ellos una línea directa entre San José y Cartago. Los puntos son los siguientes (en orden en que se recorrerán):

*Nota: los pares ordenados en cada punto representan latitud y longitud, en ese orden.*

1) **El Prado:** (9.9280092, -84.0445633)
2) **Momemtum Pinares:** (9.9110719, -84.0180413)
3) **Terramall:** (9.90173, -83.99630)
4) **Paseo Metrópoli:** (9.86695, -83.94350)
5) **Ruinas de Cartago:** (9.86391, -83.92078)
6) **Caballo Blanco:** (9.855664, -83.90433)

## III. Solución desarrollada

Se creó una pequeña página web encargada de mostrar la ruta definida. Esta puede ser accedida mediante este link. Si desea ver el código fuente puede hacerlo mediante este otro link.

### A. Librerías externas utilizadas

Se utilizaron las siguientes librerías de software para crear la página:

- **Leaflet:** Librería utilizada para mostrar el mapa en la pantalla. Requiere un provedor de mapas para esto y es completamente open source.
- **Routing Machine:** Es un plug-in de Leaflet súmamente personalizable diseñado para dibujar rutas entre puntos.
- **OpenStreetMap:** Ofrece gratis información de mapas a nivel mundial. Cada sector del mapa se conoce como una loseta, y existen muchos dependiendo del zoom que haga el usuario sobre el mapa. La información provista por esta plataforma es utilizada por Leaflet y se debe indicar como tal, pero no se interviene con su API de ninguna otra forma.

### B. Archivos incluídos

Los siguientes archivos fueron necesarios para desarrollar la página:

*1) index.html:* Esta es la página principal con el mapa. Incluye la estructura general de la página. Es de vital importancia notar que acá se importarán todas las librerías necesarias. Todas las importaciones de hojas de estilo o código son las siguientes y deben hacerse en este orden exacto para que se puedan reconocer en cascada por el browser.

Esto importa la hoja de estilos utilizada por Leaflet para su mapa:

```
<link rel="stylesheet" type="text/css"
href="https://unpkg.com/leaflet@1.6.0/dist/
    leaflet.css"
integrity="sha512-xwE/Az9zrjBIph
    AcBb3F6JVqxf46+CDLwfLMHloN
    u6KEQCAWi6HcDUbeOfBIptF7tc
    CzusKFjFw2yuvEpDL9wQ=="
crossorigin=""/>
```

Esto se encarga de importar la hoja de estilos del paquete Routing Machine:

```
<link rel="stylesheet" href="https://unpkg.com/
    leaflet-routing-machine@latest/dist/
    leaflet-routing-machine.css"/>
```

Esta es la importación de la librería de Leaflet:

```
<script type="text/javascript"
src="https://unpkg.com/leaflet@1.6.0/dist/
    leaflet.js"
integrity="sha512-gZwIG9x3wUXg2hdXF6+r
    VkLF/0Vi9U8D2Ntg4Ga5I5BZpVkVxlJWbS
    QtXPSiUTtC0TjtGOmxa1AJPuV0CPthew=="
crossorigin=""></script>
```

Esto se encarga de importar la librería de Routing Machine:

```
<script src="https://unpkg.com/
    leaflet-routing-machine@latest/
    dist/leaflet-routing-machine.js"></script>
```

Note como no se hay incluído importaciones para nada relacionado con OpenStreetMap. Esto se debe a que se debe llamar desde el código (en este caso el archivo index.js), que será visto después.

El cuerpo del documento es muy simple, solo incluye un espacio para el mapa y el logo y título de la aplicación, así:

```
<body onLoad="main()">
    <div id="map"></div>
    <div id="title_container">
        <img id="logo" src="assets/logo.png"
    alt="logo">
        <h1 id="title">Mc Ruta</h1>
    </div>
</body>
```

La función `main()` invocada desde `onLoad` está definida en index.js y se encarga de correr todo el código de Leaflet necesario para poner el mapa en la pantalla.

*2) index.js:* Este archivo es el corazón de la aplicación. Primero es necesario definir los íconos a utilizar en el mapa como variables globales, que serán utilizados después para representar cada restaurante. Son tres, uno para el primer restaurante, uno para el último y otro para los demás. Así:

```
const start = L.icon({
    iconUrl: 'assets/start.svg',
    iconSize: [35, 35],
    iconAnchor: [17.5, 17.5],
    popupAnchor: [0, -17.5]
});

const finish = L.icon({
    iconUrl: 'assets/finish.svg',
    iconSize: [35, 35],
    iconAnchor: [17.5, 17.5],
    popupAnchor: [0, -17.5]
});

const burger = L.icon({
    iconUrl: 'assets/burger.svg',
    iconSize: [30, 30],
    iconAnchor: [15, 15],
    popupAnchor: [0, -15]
});
```

La siguiente función carga los contenidos del JSON (points.json) a un objeto y lo devuelve. Es asíncrona, o en otras palabras, no devuelve su resultado directamente sino una promesa de que lo devolverá eventualmente, así:

```
async function getPoints(){
    const response = await fetch("points.json");
    const json = await response.json();
    return json;
}
```

La palabra reservada `await` simplemente espera a que se termine de resolver lo que está después de ella. Esto resuelve el problema clásico de Javascript conocido como callback hell.

Finalmente está la función `main()`, que sirve como punto de entrada y se llama a la hora de cargar el **HTML**. Como es una función relativamente larga se analizará por partes. La primera se encarga de crear un mapa invocando a Leaflet, así:

```
const map = L.map('map');
const attribution = '&copy; <a href=
    "http://osm.org/copyright">OpenStreetMap</a> |
    Map icons by <a
    href="https://www.flaticon.com/authors/freepik"
    title= "Freepik">Freepik</a>';
const tileUrl =
    'https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png';
const tiles = L.tileLayer(tileUrl, { attribution });
tiles.addTo(map);
```

La primera línea crea la variable del mapa. El string `'map'` que se le pasa coresponde al identificador del div en index.html donde se colocará el mapa. Después se declaran los strings de `attribution` y `tileUrl` El primero atribuye a OpenStreetMap la posesión de las losetas usadas para el mapa (y adicionalmente se aprovechó para atribuir a Freepik por los íconos usados en la aplicación, pero eso no es necesario). Finalmente las últimas dos líneas se encargan de crear el objeto con la capa de losetas necesarias y agregarla al mapa.

Por otra parte, se llama asíncronamente a `getPoints()` y se maneja su promesa, así:

```
getPoints().then(json => {
    const points = json.points;
    const route = [];
    for(i = 0; i < points.length; i++){
        let waypoint =
    {latLng:L.latLng(points[i].latitude,
    points[i].longitude),
    name:"<b>"+points[i].location+"</b>"};
        route.push(waypoint);
    }
    L.Routing.control({
        addWaypoints: false,
        draggableWaypoints: false,
        waypoints: route,
        createMarker:
            (i, waypoint, n) => {
                let icon = null;
                if(i == 0)
                    icon = start;
                else if(i == n-1)
                    icon = finish;
                else
                    icon = burger;
                return L.marker(waypoint.latLng,
    {icon}).bindPopup(waypoint.name);
```

```
            }
    }).addTo(map);
});
```

Una vez que se tienen los puntos devueltos por la promesa, se extraen en una variable llamada `route` y sus datos contenidos son formateados para que sean utilizados por Routing Machine. Finalmente se le indica al módulo `L.Routing` que modifique como reaccionará ante varios eventos mediante otro objeto que se le pasa al método `control()`. Lo que hace cada una de las entradas en el objeto puede ser encontrado aquí.

*3) main.css:* Este archivo incluye información de estilos, de la cual solo es necesario lo siguiente:

```
html, body {
    height: 100%;
    width: 100%;
 }

#map {
    height: 100%;
    width: 100%;
}
```

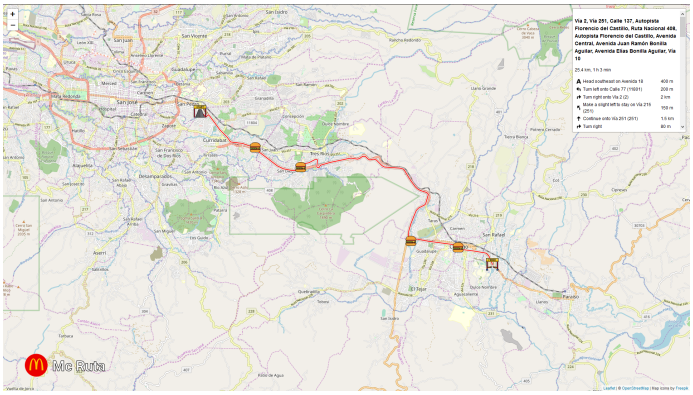
La primera regla indica que el cuerpo de la página ocupará el 100% de la misma. Esto es necesario para que se le pueda indicar al mapa que haga lo mismo, porque el cuerpo del **HTML** es padre del div del mapa en la jerarquía propuesta. La segunda regla simplemente hace lo mismo.

*4) points.json:* Finalmente, este archivo incluye la información de todos los restaurantes McDonalds seleccionados para la Mc Ruta en un arreglo. Solo incluyen nombre, latitud y longitud, de esta forma para cada restaurante agregado:

```
{
    "location":"El Prado",
    "latitude":9.9280092,
    "longitude":-84.0445633
}
```

## IV. Resultado final

Al final, la página se ve así:




## V. Conclusión

OpenStreetMap y Leaflet son tecnologías poderosas y muy fáciles de utilizar. A pesar de no ser mucho código, se consiguió una aplicación que en otros frameworks o metodologías de trabajo podría tomar meses desarrollando. Se espera que este pequeño prototipo sea útil para otros desarrolladores y que lo descrito a través de este documento ayude a otros con un objetivo parecido.

## References

[1] gafonkin V., *Leaflet Quick Start Guide*. Recuperado de https://leafletjs.com/examples/quick-start/.

[2] iedman P., *Leaflet Routing Machine API*. Recuperado de http://www.liedman.net/leaflet-routing-machine/api/.