



universidad
cenfotec_
tecnologías digitales



Bachillerato en Desarrollo de Software

Proyecto de Ingeniería del Software 2

Investigación sobre GMaps

Profesores

Álvaro Cordero Peña

Dennis Córdoba López

Raquel Robles Flores

Estudiantes

Keissy Arroyo Rodríguez

Gabriel Calvo Vargas

Miranda Castaing Karlsson

David Solano Frini

Mónica Zúñiga Arguedas

Fecha de Entrega

1 de noviembre del 2019

Período Lectivo

III Cuatrimestre

Tabla de Contenidos

I.	¿Qué es GMaps?	1
II.	¿Para qué se utiliza GMaps?	2
III.	Otras Librerías Similares	3
IV.	Implementación	4
V.	Bibliografía	9

I. ¿Qué es GMaps?

GMaps es una librería JavaScript que se basa en el API de Google Maps y permite publicar mapas en la Web de forma fácil y amigable para los desarrolladores.

En términos generales, el API de Google Maps es amigable y posee una valiosa documentación para su implementación. Sin embargo, es un API que presenta una gran cantidad de funcionalidades que, en la mayoría de los casos, no se utilizan y tienden a confundir al programador. Es por esta razón que GMaps fue creada.

GMaps permite crear mapas con marcadores, rutas, geolocalización, perfiles longitudinales y entre otras funciones mediante el uso de jQuery lo cual permite reducir considerablemente la cantidad de código y lo hace más fácil de comprender.

Es importante aclarar que GMaps se basa en el API de Google Maps, por lo que no es un API totalmente independiente. Lo que permite es implementar el API de Google Maps de manera más sencilla.

II. ¿Para qué se utiliza GMaps?

GMaps se puede utilizar para lograr diferentes objetivos. A continuación, se encuentran varias funcionalidades que GMaps ofrece.

- Una de las principales funcionalidades es la de consultar direcciones y calcular rutas. Se puede averiguar acerca de la ubicación de cualquier lugar, así como calcular la ruta desde diferentes puntos.
- También, puede ejercerse como GPS. Puede indicar la dirección a tomar de camino a algún lugar, teniendo en cuenta el modo de desplazamiento de la persona.
- Consulta en detalle de cómo es un territorio geológicamente, gracias a las fotografías satelitales que Google Maps brinda.
- Localización de negocios, como tiendas y restaurantes, con el motivo de visualizar cómo son por dentro, gracias al proyecto Business View de Google. Este proyecto consiste en brindar a los dueños de empresas la posibilidad de fotografiar sus instalaciones por fotógrafos 'de confianza' de Google.

III. Otras Librerías Similares

La aparición de ciertas librerías de fuente libre de JavaScript ha permitido lograr funcionalidades similares a las de GMaps. A continuación, se encuentran algunas de estas librerías.

- **Leaflet.** Es la biblioteca líder de fuente libre de JavaScript de mapas interactivos compatibles con dispositivos móviles. Leaflet fue diseñada teniendo en mente la simplicidad, el rendimiento y la usabilidad para los programadores. También, tiene todas las características de mapeo que la mayoría de los desarrolladores necesitan y presenta un tamaño aproximado de 38 KB de código JavaScript.
- **OpenLayers.** Facilita la implementación de un mapa dinámico en cualquier página web. Puede cargar mosaicos de mapas, datos vectoriales y marcadores desde cualquier fuente. OpenLayers ha sido desarrollada con el motivo de facilitar el acceso a la información geográfica de cualquier tipo. Además, es completamente gratuita debido a que es un JavaScript de fuente libre y fue lanzada bajo la licencia BSD de dos cláusulas (también conocida como FreeBSD).

IV. Implementación

Antes de empezar, se debe agregar el JavaScript gmaps.js al proyecto deseado debido a que el núcleo de la aplicación está formado por un objeto que es creado mediante una instancia de GMaps. Ésta es la clase fundamental de la librería y la encargada de manejar el mapa. GMaps requiere de la definición de las siguientes cuatro propiedades principales:

1. el, la cual indica la referencia del mapa a través del id del div en HTML.
2. lat.
3. lng.
4. zoom (con el que debería cargar el mapa).

lat y lng son las coordenadas geográficas para establecer el centro del mapa.

Además del archivo .js de GMaps, se debe agregar el código que se encuentra a continuación.

```
//Iniciación del mapa
var map = new GMaps({
  el: '#map',           //él toma el tag con el id #map e inicializa un nuevo mapa
  zoom: 14,             //se le puede setear qué tanto zoom se quiere
  lat: 9.909580,         //latitud del centro del mapa
  lng: -84.054062,       //longitud del centro del mapa
  click: addMarker       //se añade el evento click y se referencia a la función addMarker
});

function addMarker(e) {
  if (map.markers.length <= 2) {
    if (map.markers.length == 0) {
      addStartPoint(e);
    }
    else if (map.markers.length == 1) {
      addEndPoint(e);
      drawRoute();
      calculateDistance();
    }
  }
}
```

Paso #1: Crear un API Key

Para lograr implementar un mapa en un sitio web usando el API de Google Maps, se debe crear una clave de API (API key) gratuita siguiendo los pasos que se encuentran a continuación.

1. Crear una cuenta de Google ingresando un correo electrónico único y una contraseña.

2. Ingresar a la Consola de la API de Google y se debe realizar el registro marcando las casillas de la siguiente manera:

Registra tu aplicación para utilizar Google Maps JavaScript API en Consola de la API de Google

Consola de la API de Google te permite administrar tu aplicación y supervisar el uso de la API.

No tienes ningún proyecto. Se creará un proyecto nuevo llamado "My Project".

Quiero recibir por correo electrónico información sobre las funciones del producto, sugerencias de rendimiento, encuestas para aportar mis observaciones y ofertas especiales.

☐ Sí ☒ No

Acepto que el uso que haga de cualquier servicio y las API relacionadas queda sujeto a mi cumplimiento de las Condiciones de Servicio correspondientes.

☒ Sí ☐ No

[Aceptar y continuar](#)

3. Ingresar un nombre significativo que identifique al mapa, restringir el uso de la clave de API a solo el dominio del proyecto deseado y oprimir el botón de crear.

API Administrador de API:

Panel de control

Biblioteca

Credenciales

Credenciales

Clave de API

Esta clave de API se puede usar en este proyecto y con cualquier API compatible. Para usar esta clave en tu aplicación, transfírela con el parámetro `key=API_KEY`.

Nombre

Mapa para la web [www.example.com](#)

Restricción de clave

Si restringes una clave, puedes especificar qué sitios web, direcciones IP o aplicaciones pueden usarla. [Más información](#)

☐ Ninguna

☒ URLs de referencia HTTP (sitios web)

☐ Direcciones IP (servidores web, tareas cron, etc.)

☐ Aplicaciones para Android

☐ Aplicaciones para iOS

Aceptar solicitudes de estas URL de referencia HTTP (sitios web) (Opcional)

Usa asteriscos como caracteres comodín. Si lo dejas en blanco, se aceptarán solicitudes de cualquier URL de referencia. Añade URLs de referencia antes de usar esta clave en producción.

[*.example.com/*](#)

[*.example.com/*](#)

Nota: Pueden pasar hasta 5 minutos antes de que se aplique la configuración

[Crear](#) [Cancelar](#)

4. Aparecerá una ventana pequeña que contendrá la clave de API deseada la cual se podrá utilizar para mostrar el mapa en el sitio web. Éste deberá ser añadido al src del script del HTML que referencia a Google Maps.

Clave de API creada

Para usar esta clave en tu aplicación, transfírela como un parámetro

key=API_KEY

Tu clave de API

[Redacted API Key]

⚠ Restringe la clave para impedir el uso no autorizado en producción.

[CERRAR](#) [RESTRINGIR CLAVE](#)

Paso #2: Agregar Marcadores

Para poder realizar el cálculo de rutas, se necesita tener dos coordenadas o marcadores. GMaps ofrece la funcionalidad de `addMarker`, la cual crea un marcador y, lo añade al mapa y a la colección de marcadores. Esta función permite una variedad de opciones.

A continuación, se encuentra la continuación del código con los marcadores mencionados anteriormente.

```
function addStartPoint(e) {
    //Se añade el punto inicial
    map.addMarker({
        //llamada a función de Gmaps --> addMarker
        lat: e.latLng.lat(),
        lng: e.latLng.lng(),
        title: "Start Point",
        icon: {
            //Esta propiedad es opcional pero ayuda a referenciarla luego
            //Añadir el icono (también es configurable)
            path: google.maps.SymbolPath.CIRCLE,
            scale: 5, //tamaño
            strokeColor: '#f00', //color del borde
            strokeWeight: 10
        }
    });
}

function addEndPoint(e) {
    //Se añade el punto inicial
    map.addMarker({
        //llamada a función de Gmaps --> addMarker
        lat: e.latLng.lat(),
        lng: e.latLng.lng(),
        title: "End Point",
        icon: {
            //Lo mismo con la longitud
            path: google.maps.SymbolPath.CIRCLE,
            scale: 5, //tamaño
            strokeColor: '#009975', //color del borde
            strokeWeight: 10
        }
    });
}
```


Paso #3: Calcular Rutas

Para lograr realizar un cálculo de rutas, GMaps brinda una función que se llama `getRoutes`, la cual retorna un arreglo de rutas. Lo que esta función realiza es simplemente obtener rutas entre dos coordenadas.

A continuación, se encuentran las opciones que siempre se deberían incluir para que la función retorne las rutas. Sin embargo, `getRoutes` tiene la ventaja de aceptar una variedad más de opciones.

Opciones de <code>getRoutes</code>		
Nombre	Tipo	Descripción
origin	Arreglo	Arreglo con dos elementos: latitud y longitud.
destination	Arreglo	Arreglo con dos elementos: latitud y longitud.
callback	Función	Función que se dispara después del retorno de resultados.

`getRoutes` también acepta cualquier propiedad definida en [google.maps.DirectionsRequest](#).

Ahora, se debe hablar acerca de los *polylines*. La función `drawRoute` dibuja una ruta usando *polylines* (líneas que representan rutas). Al igual que `getRoutes`, ésta también acepta una gran variedad de opciones pero solo se mencionarán las principales a continuación.

Opciones de <code>drawRoute</code>		
Nombre	Tipo	Descripción
origin	Arreglo	Arreglo con dos elementos: latitud y longitud.
destination	Arreglo	Arreglo con dos elementos: latitud y longitud.
travelMode	<i>String</i>	Puede ser <i>driving</i> , <i>bicycling</i> o <i>walking</i> .
waypoints	Arreglo	Arreglo de objetos google.maps.DirectionsWaypoint .
strokeColor	<i>String</i>	Color del <i>polyline</i> . Puede ser hexadecimal o el nombre en CSS.
strokeOpacity	<i>Float</i>	Opacidad del <i>polyline</i> desde 0.0 al 1.0.
strokeWeight	Entero	<i>Width</i> del <i>polyline</i> en pixeles.

Finalmente, a continuación se encuentra la última parte del código en orden para que se pueda calcular una ruta a partir de dos marcadores utilizando las funciones de GMaps drawRoute y addMarker.

```
function drawRoute() {
    //Se dibuja la ruta
    var start = map.markers[0].position; //De la lista de markers se toma el primer elemento como punto inicial
    var end = map.markers[1].position; //De la lista de markers se toma el segundo elemento como punto inicial

    map.drawRoute({
        //llamada a función de Gmaps --> drawRoute
        origin: [start.lat(), start.lng()], //Se toma la lat y long del punto inicial
        destination: [end.lat(), end.lng()], //Lo mismo con el punto final
        travelMode: 'driving', //Se detalla el tipo de modo de traslado
        strokeColor: '#131540', //Se setean estilos de la rutas
        strokeOpacity: 0.6,
        strokeWeight: 6
    });
}

/* Se calcula la distancia entre los 2 puntos tomando las coordenadas
(se toma la latitud y longitud del punto inicial y del punto final).
El cálculo se hace por trigonometría utilizando la tierra como una esfera
*/
function calculateDistance() {
    var start = map.markers[0].position;
    var end = map.markers[1].position;
    var lat1 = 0, lat2 = 0, long1 = 0, lon2 = 0, dist = 0;

    lat1 = start.lat();
    lat2 = end.lat();
    lon1 = start.lng();
    lon2 = end.lng();
    if ((lat1 == lat2) && (lon1 == lon2)) {
    }
    else {
        var radlat1 = Math.PI * lat1 / 180;
        var radlat2 = Math.PI * lat2 / 180;
        var theta = lon1 - lon2;
        var radtheta = Math.PI * theta / 180;
        var dist = Math.sin(radlat1) * Math.sin(radlat2) + Math.cos(radlat1) * Math.cos(radlat2) * Math.cos(radtheta);
        if (dist > 1) {
            dist = 1;
        }
        dist = Math.acos(dist);
        dist = dist * 180 / Math.PI;
        dist = dist * 60 * 1.1515;
        dist = dist * 1.609344;
    }
}

//Se pega el resultado en el HTML usando jQuery, tanto la distancia como el tiempo de llegada
$("#distance").html("Distancia: " + Math.round(dist).toFixed(1) + "km");
$("#time").html("Tiempo estimado : " + Math.round(dist * 4).toFixed(0) + "min");

/* Nota: el tiempo estimado depende de las condiciones del tráfico, tipo de transporte, etc.
Para efectos de este ejemplo, se toma un tiempo promedio de 4 minutos por km.
Este número se puede refinar */
```

El movimiento de un marcador se puede realizar utilizando las mismas funcionalidades, pero añadiendo un solo marcador al mapa y cambiándole la latitud y longitud por una de la ruta cada cierto tiempo, es decir, a medida que la persona avance.

V. Bibliografía

¿Cómo obtener una clave API para Google Maps? (s.f). España: Artesans. Recuperado de <https://www.artesans.eu/como-obtener-una-clave-api-para-google-maps/>

García, J. L. (s.f). *GMaps.js: una forma muy fácil de publicar mapas en la web*. España: MappingGIS. Recuperado de <https://mappinggis.com/2018/03/gmaps-js-una-forma-muy-facil-de-publicar-mapas-en-la-web/>

GMaps Documentation. (s.f). GMaps. Recuperado de <https://hpneo.dev/gmaps/documentation.html>