

MAPAS DE GOOGLE PASO A PASO

Con mi cuenta de Google Cloud Platform:

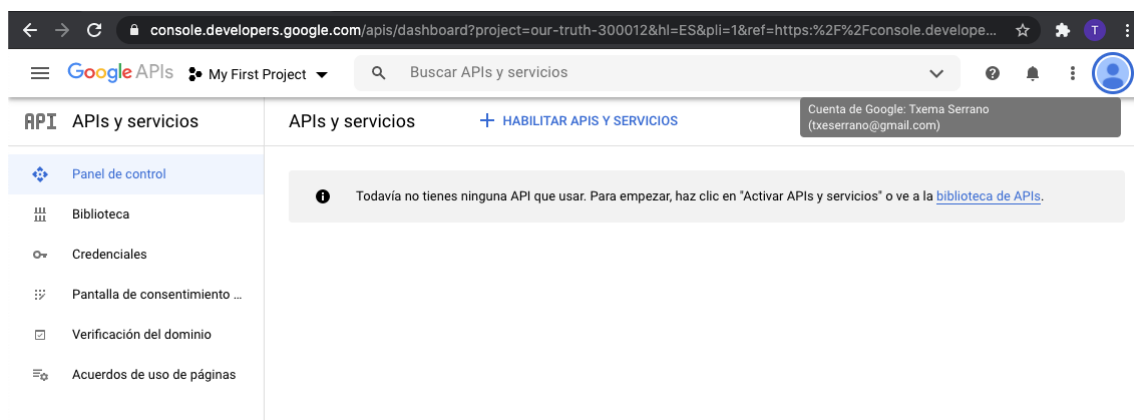


Hola, Txema

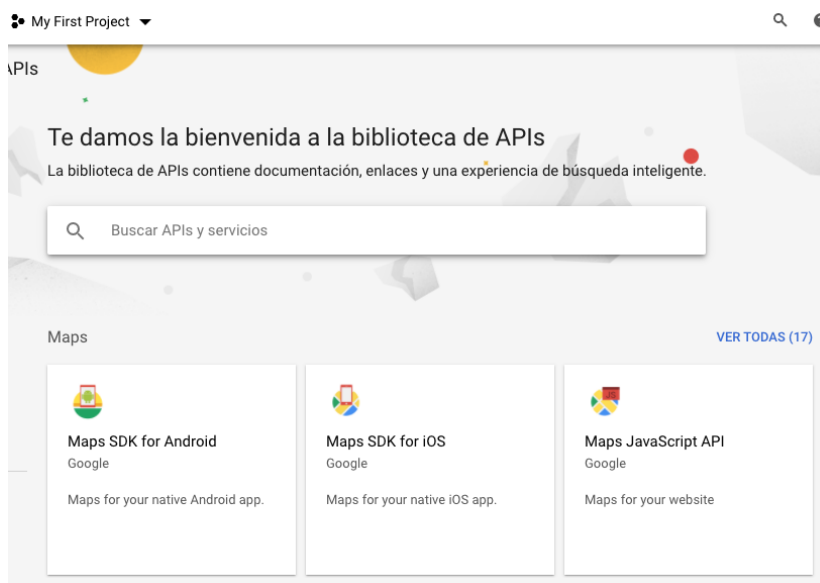
Gracias por registrarte. La prueba gratuita incluye 300 USD en crédito para usarlos durante los próximos 90 días. Si te quedas sin crédito, no te preocupes, no te cobraremos nada a menos que [actives la facturación automática](#).

ENTENDIDO

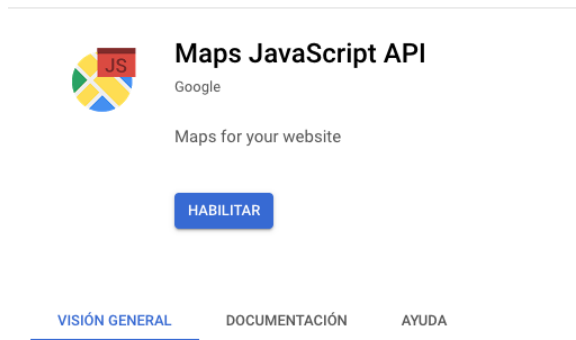
Inicio sesión con esta cuenta en developers.google.com:



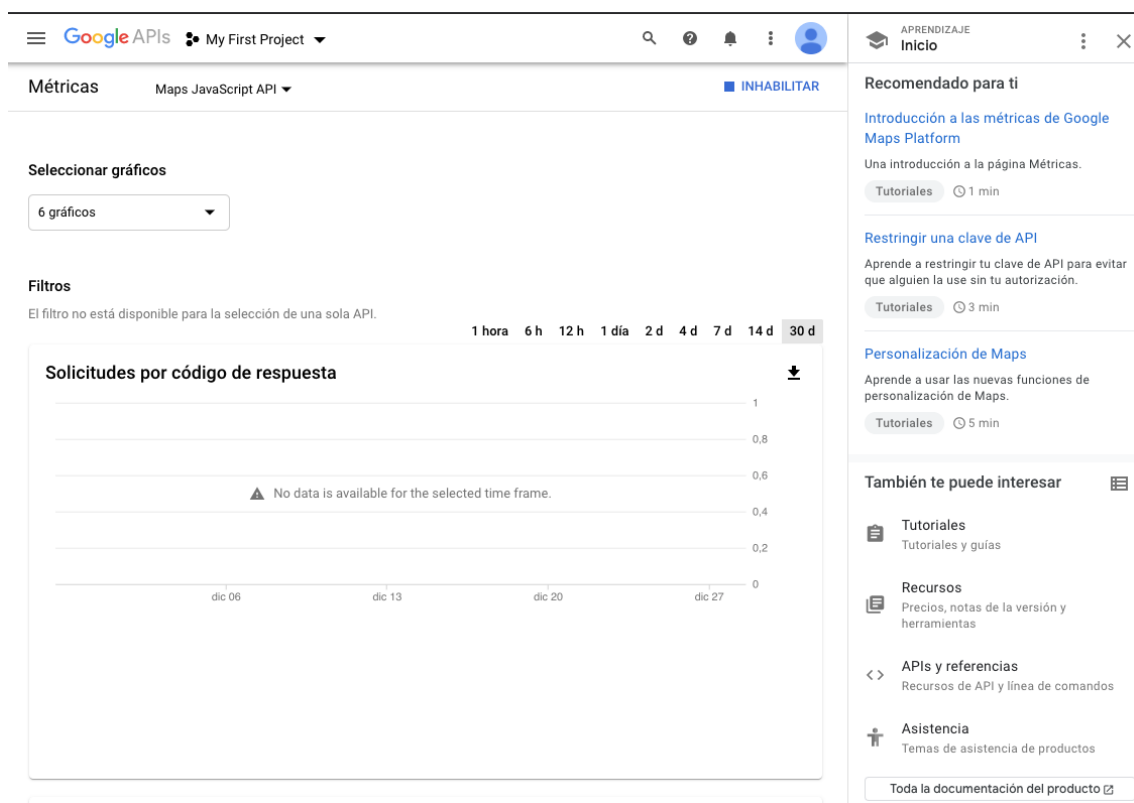
Le damos a Habilitar APIS y Servicios:



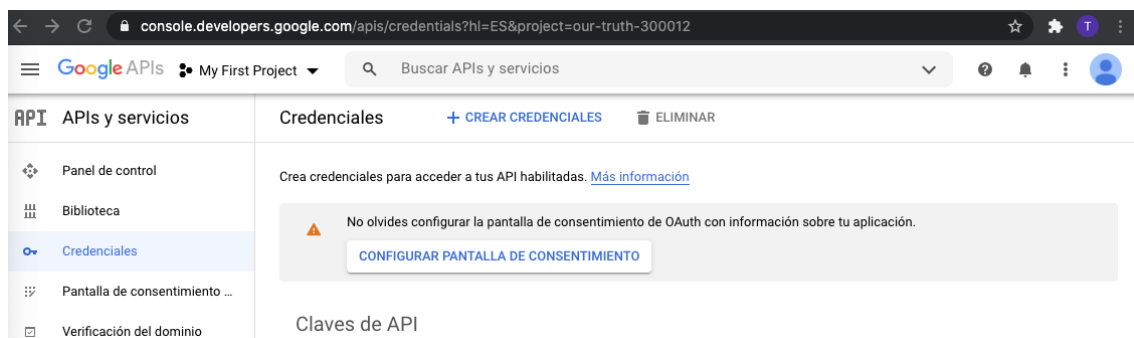
Entramos en Maps Javascripts API:



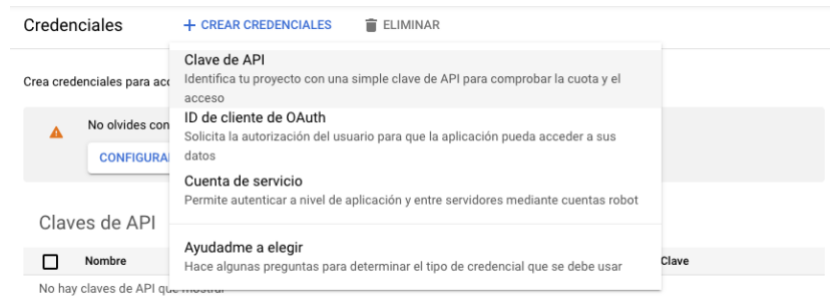
Le damos en Habilitar:



Para obtener la API key, desplegamos APIs y servicios. Credenciales:



Pulsamos crear credenciales



Clave de API

Clave de API creada

Para usar esta clave en tu aplicación, transfírela con el parámetro `key=API_KEY`.

Tu clave de API
AIzaSyI QJ1PA54MchZmcmD53ns

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

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

Restringimos la clave:

Nombre *
API key

API Key
AIzaSy LyLkKFnuDW30E

Para usar esta clave en tu aplicación, transfírela con el parámetro `key=CLAVE_API`.

Fecha de creación	28 de diciembre de 2020, 17:33:11 GMT+1	
Creada por	jsersan@gmail.com (tú)	
Clave anterior	AIzaSyC	OSZ6q0YQ
	Activa hasta el 29 dic. 2020 17:33:11	

Restricciones de clave

⚠ Esta clave no tiene restricciones. Las restricciones ayudan a evitar el uso sin autorización y el robo de cuotas. [Más información](#)

Restricciones de aplicación

Las restricciones de aplicaciones controlan qué sitios web, direcciones IP o aplicaciones pueden usar tu clave de API. Puedes configurar una restricción de aplicaciones por clave.

☒ Ninguna
☐ URL referentes HTTP (sitios web)
☐ Direcciones IP (servidores web, tareas cron, etc.)
☐ Aplicaciones de Android
☐ Aplicaciones de iOS

Restricciones de API

Las restricciones de API especifican las API habilitadas a las que puede llamar esta clave

☐ No restringir la clave
 Esta clave puede llamar a cualquier API
☒ Restringir clave

1 API

API seleccionadas:

Maps JavaScript API

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

[GUARDAR](#) [CANCELAR](#)

Guardamos y nuestra APIkey:

Claves de API

<input type="checkbox"/>	Nombre	Fecha de creación ↓	Restricciones	Clave			
<input type="checkbox"/>	✓ MapaEuskadi	28 dic 2020	Maps JavaScript API	AIzaSyDHau...hZmcmD53ns			

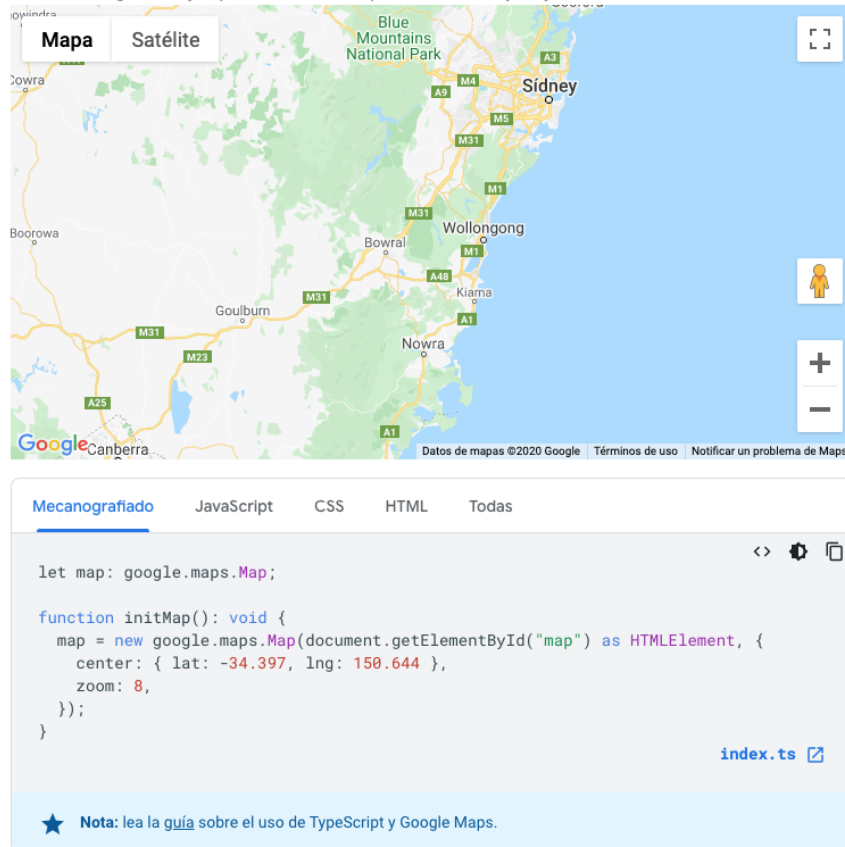
Para utilizar la API key, nos metemos en el portal de formación para desarrolladores:

En la documentación oficial de la API en la sección de API de Javascripts de Maps, encontramos un tutorial paso a paso que vamos a seguir:

Con la API key proporcionada hacemos nuestro primer mapa siguiendo:

Hola Mundo

La forma más sencilla de empezar a aprender acerca de la API de JavaScript de Maps es ver un ejemplo sencillo. El siguiente ejemplo muestra un mapa centrado en Sydney, Nueva Gales del Sur, Australia.



Seguimos el paso a paso:

Las siguientes instrucciones muestran cómo usar Google Cloud Console para crear, personalizar, publicar y administrar sus mapas en cualquier momento usando ID de mapa y Estilos de mapa.

Para crear o administrar cualquier ID de mapa o estilo de mapa en su proyecto de Google Cloud, debe tener el [rol de propietario del proyecto o editor del proyecto de IAM](#).

Crear ID de mapa

Un *ID de mapa* es un identificador único que representa una única instancia de un mapa de Google. Puedes crear ID de mapa y actualizar un estilo asociado con un ID de mapa en cualquier momento en Google Cloud Console sin cambiar el estilo JSON incorporado en el código de tu aplicación.

Para crear un ID de mapa:

1. En Cloud Console, vaya a la página **Administración de mapas**.

[Ir a la página de administración de mapas](#)

2. Haga clic en **Crear nuevo ID de mapa** para mostrar el formulario **Crear nuevo ID de mapa**.

The screenshot shows the Google Cloud Platform interface for Map Management. The 'CREATE NEW MAP ID' button is highlighted with a red box. Below it is a table of existing Map IDs.

Map ID Name	Map ID Number	Type	Requests	Current Map Style
Android Search Map	5588888	Android	1,654,345	Uncustomized
iOS Search Map	5588887	iOS	812,345	My Basic Map Style
Web Search Map	5588886	JS	345,433	Uncustomized
Android Listing Details Map	5588885	Android	389,434	Uncustomized
iOS Listing Details Map	5588884	iOS	154,389	Cherry Blossom Style
Web Listing Details Map	5588883	JS	45,223	Uncustomized
Confirmation Email Map	5588882	Static	10,233	Cherry Blossom Style
Search Results Page 1	5588881	JS	32,345	Cherry Blossom Style
Search Results Page 2	5588880	JS	5,233	My Basic Map Style
Search Results Page 3	5588879	JS	4,389	My Basic Map Style
Search Results Page 4	5588878	JS	6,344	My Basic Map Style
Search Results Page 5	5588876	JS	4,300	Uncustomized
Search Results Page 6	5588875	JS	5,545	Uncustomized
Search Results Page 7	5588874	JS	4389	Uncustomized
Search Results Page 8	5588873	JS	45	Uncustomized

Rows per page: 15 • 1-15 of 24 < >

En el formulario, haga lo siguiente:

- Especifique un nombre de mapa.
- Especifique un tipo de mapa o plataforma.
- Ingrese una descripción del mapa.
- Haga clic en **Siguiente** para mostrar el nuevo ID de mapa.

Uso de ID de mapa en el código de su aplicación

Para crear un mapa con un ID de mapa en el código de su aplicación:

1. Actualice la etiqueta del script:

```
<script  
src="https://maps.googleapis.com/maps/api/js?key=API_KEY&map_ids=MAP_ID&callback=initMap">  
</script>
```

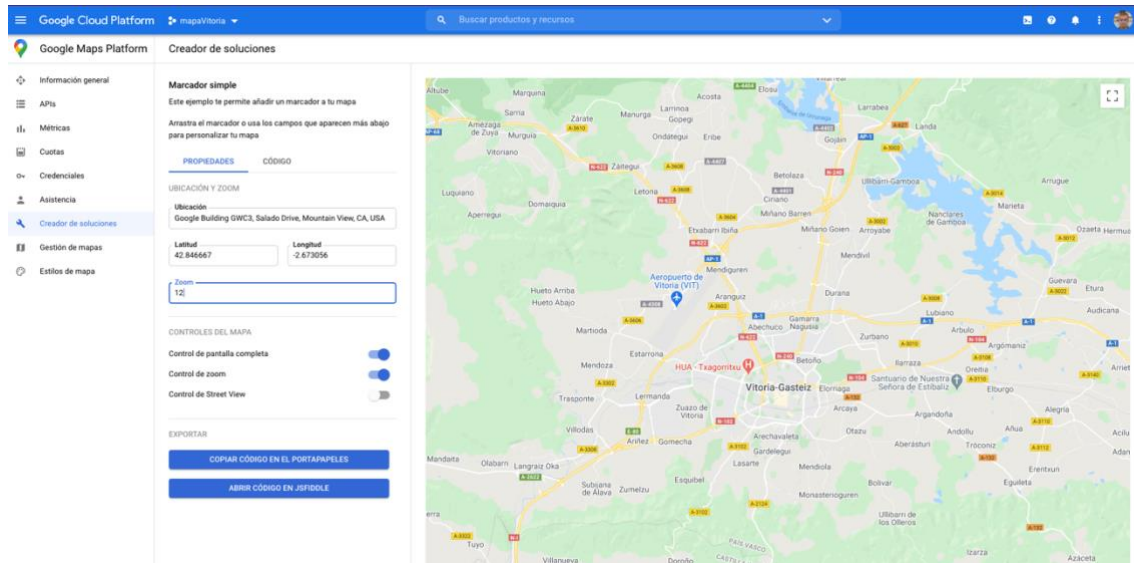
Reemplazar:

- **API_KEY** con la clave API para su proyecto.
 - **MAP_ID** con uno o varios ID de mapa. Por ejemplo: `map_ids=1234` o `map_ids=1234,2345`
2. Si actualmente está personalizando su mapa con código JSON incrustado, elimine la `styles` propiedad de su `MapOptions` objeto; de lo contrario, omita este paso.
 3. Agregue un ID de mapa al mapa usando la `mapId` propiedad. Por ejemplo:

```
map = new google.maps.Map(document.getElementById('map'), {  
  center: {lat: -34.397, lng: 150.644},  
  zoom: 8,  
  mapId: 'MAP_ID'  
});
```

La función anterior muestra un solo mapa y solo admite la inclusión de un solo ID de mapa en la `mapId` opción.

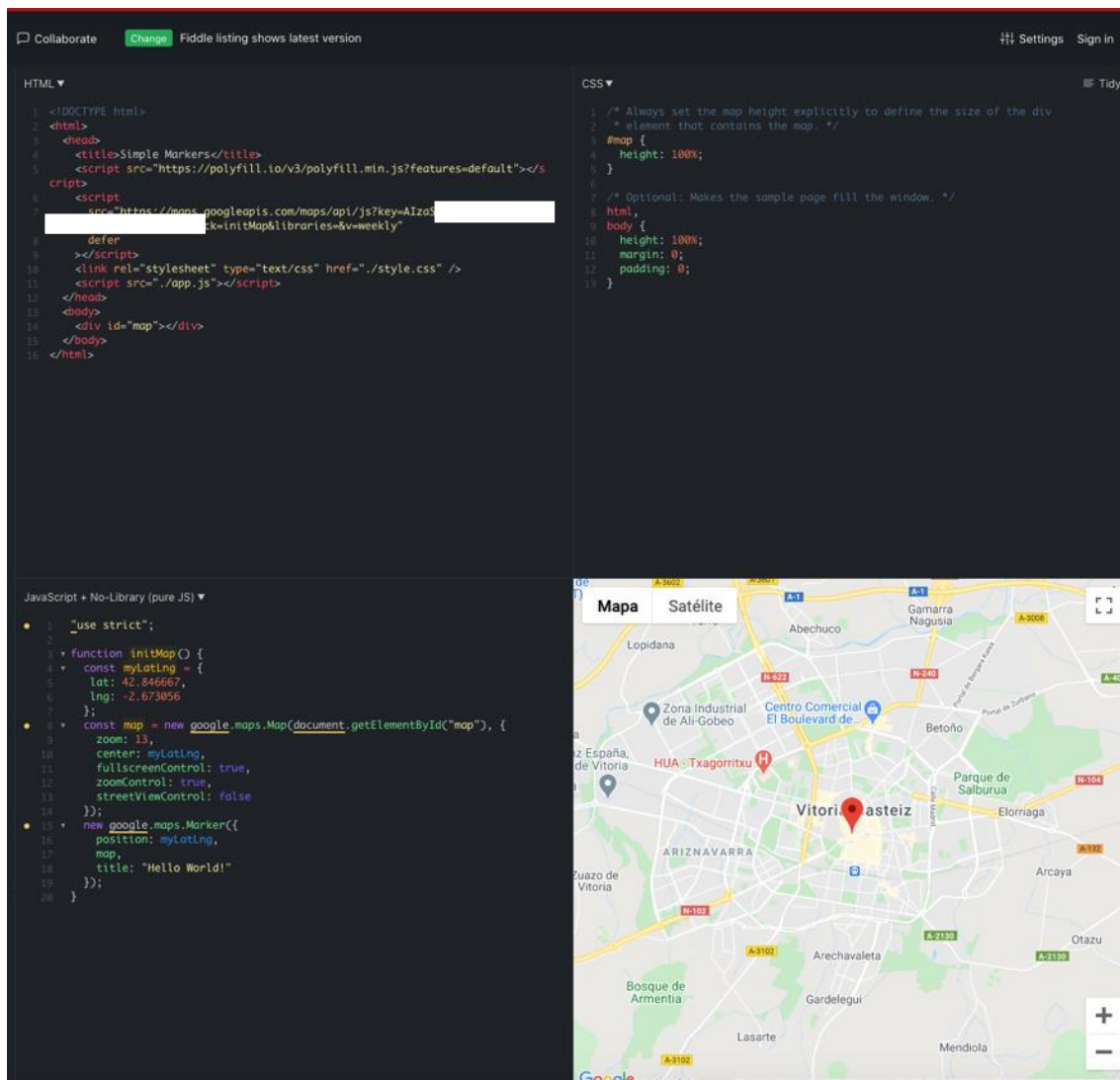
En el creador de soluciones de GoogleMap:



Si le damos a copiar el código en el portapapeles:

```
adi filtrado/... index.html 6.Comunidad de Madrid apiKey.txt index.html ./
index.html > html > head > script
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Simple Markers</title>
5 <script src="https://polyfill.io/v3/polyfill.min.js?features=default">
6 </script>
7 <script
8 src="https://maps.googleapis.com/maps/api/js?key=AIzaSyAeRir0iEZxCi6C
9 defer
10 ></script>
11 <style type="text/css">
12 /* Always set the map height explicitly to define the size of the div
13 * element that contains the map. */
14 #map {
15 height: 100%;
16 }
17
18 /* Optional: Makes the sample page fill the window. */
19 html,
20 body {
21 height: 100%;
22 margin: 0;
23 padding: 0;
24 }
25 </style>
26 <script>
27 "use strict";
28
29 function initMap() {
30 const myLatLng = {
31 lat: NaN,
32 lng: NaN
33 };
```

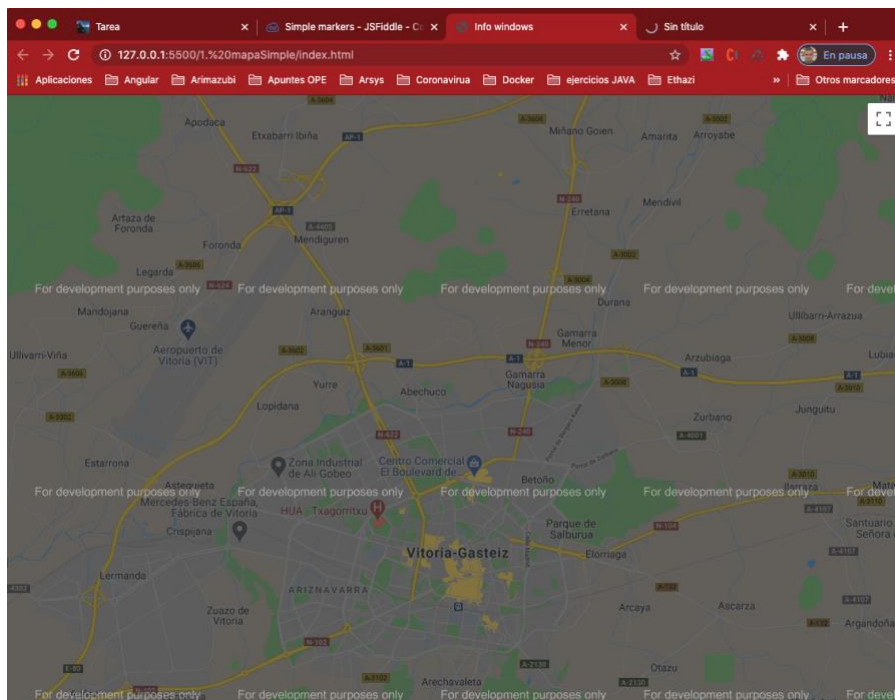

Podemos abrir el proyecto en JSFiddle:



El proyecto consta de tres ficheros:

- index.html
- styles.css
- app.js

La ejecución del fichero en local se vería así:



Sale la marca de agua “For Development purposes only” porque la api key no se ha obtenido con el refrendo de una tarjeta de crédito. Si pongo una API KEY con soporte:



Mapa con pin de marca.

En la función `initMap()` creamos los siguientes elementos:

- Coordenadas donde está centrado el mapa (virgenBlanca).
- Un texto asociado a una ventana de información que será el que se muestre cuando se haga click en el pin que también está centrado en la plaza de la virgen Blanca.
- Se asocia el pin mediante el evento click la aparición de la ventana de información en el pin.

```

7  function initMap() {
8
9      var virgenBlanca = {
10         lat: 42.846667,
11         lng: -2.673056
12     };
13     var map = new google.maps.Map(document.getElementById('map'), {
14         zoom: 13,
15         center: virgenBlanca
16     });
17
18     var contentString = '<div id="content">' +
19         '<div id="siteNotice">' +
20         '</div>' +
21         '<h1 id="firstHeading" class="firstHeading">Gasteiz</h1>' +
22         '<div id="bodyContent">' +
23         '<p><b>Plaza de la Virgen Blanca</b>,' +
24         '<p>Vitoria-Gasteiz es la capital de la comunidad autónoma del País Vasco, situada en el norte
25         '<p>Más información: Gasteiz, <a href="https://es.wikipedia.org/wiki/Vitoria">' +
26         'https://es.wikipedia.org/wiki/Vitoria</a>' +
27         '</div>' +
28         '</div>';
29
30     var infowindow = new google.maps.InfoWindow({
31     });
32
33
34     var marker = new google.maps.Marker({
35         position: virgenBlanca,
36         map: map,
37         title: 'Virgen Blanca'
38     });
39     marker.addListener('click', function () {
40         infowindow.open(map, marker);
41     });
42

```

La salida es:



Mapa con leyenda.

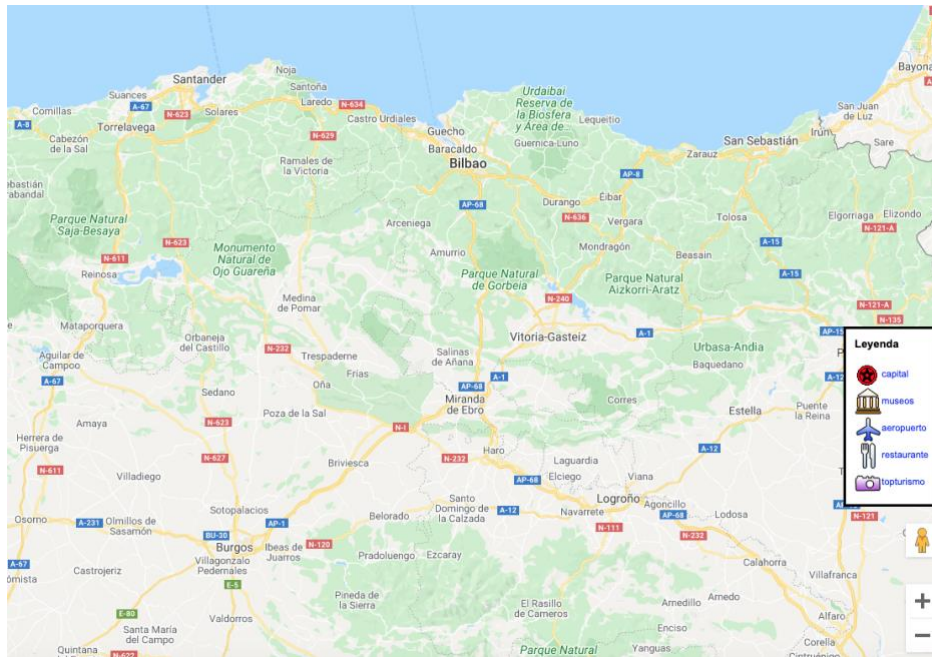
1. Creamos una función **colocarLeyenda()** en app.js.

```

60  function colocarLeyenda() {
61
62      var iconBase = "https://maps.google.com/mapfiles/kml/shapes/";
63      var icons = {
64          capital: {
65              name: "capital",
66              icon: iconBase + "capital_big_highlight.png"
67          },
68          museos: {
69              name: "museos",
70              icon: iconBase + "museum_maps.png"
71          },
72          aeropuerto: {
73              name: "aeropuerto",
74              icon: iconBase + "airports_maps.png"
75          },
76          restaurante: {
77              name: "restaurante",
78              icon: iconBase + "dining_maps.png"
79          },
80          topturismo: {
81              name: "topturismo",
82              icon: iconBase + "camera_maps.png"
83          }
84      };
85
86      if (!inicio) {
87          for (var key in icons) {
88              var type = icons[key];
89              var name = type.name;
90              var icon = type.icon;
91              var div = document.createElement("div");
92              div.setAttribute("id", "leyenda");
93              var enlace =
94                  '<a href="' +
95                  "mapaEuskadi.html?tipo=" +
96                  name +
97                  "><img src='" +
98                  icon +
99                  ">' +
100                  name +
101                  "</a>";
102              div.innerHTML = enlace;
103
104              legend.appendChild(div);
105          }
106
107          inicio = true;
108      }
109
110      this.mapa.controls[google.maps.ControlPosition.RIGHT_BOTTOM].push(legend);
111
112      return;
113  }
114

```

La salida es:



Para acceder a los elementos de la leyenda pasaremos la información en forma de parámetro por la url:

```

10 window.onload = function() {
11     initMap();
12     colocarLeyenda();
13
14     var valores = getGET();
15     if (valores) {
16         //recogemos los valores que nos envia la URL en variables para trabajar
17         //con ellas
18         var tipo = valores['tipo'];
19     }
20     cargarDatos(tipo);
21 }
22
23
24 function getGET() {
25     // capturamos la url
26     var loc = document.location.href;
27
28     // si existe el interrogante
29     if (loc.indexOf("?") > 0) {
30         // cogemos la parte de la url que hay despues del interrogante
31         var getString = loc.split("?")[1];
32
33         // obtenemos un array con cada clave=valor
34         var GET = getString.split("&");
35         var get = {};
36
37         // recorremos todo el array de valores
38         for (var i = 0, l = GET.length; i < l; i++) {
39             var tmp = GET[i].split("=");
40             get[tmp[0]] = unescape(decodeURI(tmp[1]));
41         }
42         return get;
43     }
44 }

```


Ahora que ya sabemos cuál es el tipo de dato seleccionado, lo mostramos por consola:

```

115 function cargarDatos(tipo) {
116     switch (tipo) {
117         case "restaurante":
118             console.log("Restaurante");
119             break;
120
121         case "aeropuerto":
122             console.log("Aeropuerto");
123             break;
124
125         case "museos":
126             console.log("museos");
127             break;
128
129         case "topturismo":
130             console.log("turismo");
131             break;
132
133         case "capital":
134             console.log("capital");
135             break;
136     }
137 }

```

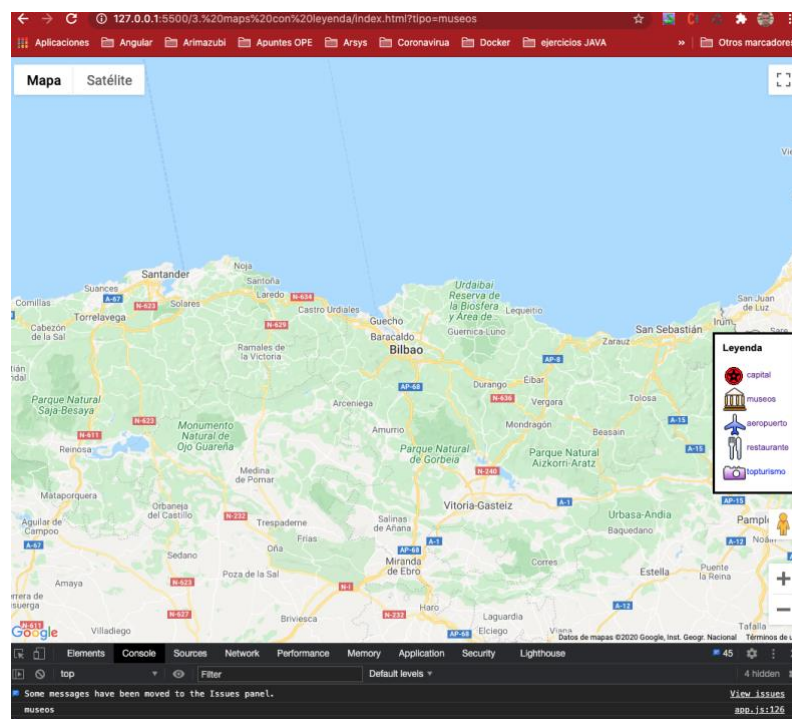
No debemos olvidar declarar las variables utilizadas:

```

1 let mapa;
2 let tipo = "";
3 let markers = [];
4
5 let emoticon; // modificado 20/04
6 let legend = null;
7 legend = document.getElementById("legend");
8 let inicio = false;
9

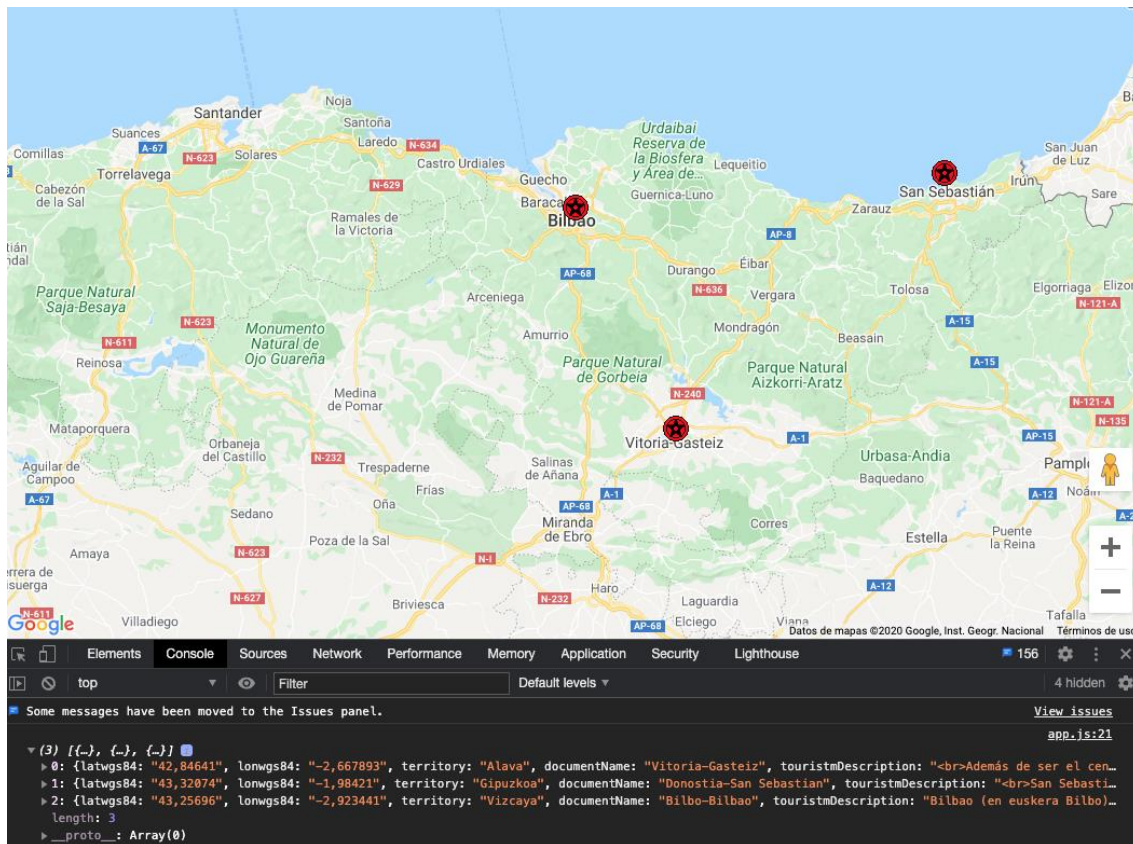
```

La salida es la siguiente si escogemos museos:



Mapa con marcas proveniente de un fichero JSON

La salida es la siguiente:



Si pulsamos sobre cualquier capital:



El método de resolución es como el anterior:

1. Carga vía AJAX del fichero de capitales.
2. Recorrido de ese fichero para colocar los pines en el mapa.

1. Carga AJAX de las localizaciones de las capitales.

El fichero JSON de las capitales es:

```

1  [
2    {
3      "latwgs84": "42,84641",
4      "lonwgs84": "-2,667893",
5      "territory": "Alava",
6      "documentName": "Vitoria-Gasteiz",
7      "tourismDescription": "<br>Además de ser el centro político y administrativo de Euskadi, Vito",
8      "paginaWeb": "https://www.vitoria-gasteiz.org/"
9    },
10   {
11     "latwgs84": "43,32074",
12     "lonwgs84": "-1,98421",
13     "territory": "Gipuzkoa",
14     "documentName": "Donostia-San Sebastian",
15     "tourismDescription": "<br>San Sebastián (en euskera Donostia y oficialmente Donostia/San Set",
16     "paginaWeb": "https://www.donostia.eus/"
17   },
18   {
19     "latwgs84": "43,25696",
20     "lonwgs84": "-2,923441",
21     "territory": "Vizcaya",
22     "documentName": "Bilbo-Bilbao",
23     "tourismDescription": "Bilbao (en euskera Bilbo) es un municipio situado en el norte de Espa",
24     "paginaWeb": "http://bilbao.eus/"
25   }
26 ]

```

La función AJAX de carga de los datos:

```

4- maps con json local > script > JS app.js > ...
1  let mapa;
2  let markers = [];
3  let fichero = 'datoscapitales.json';
4  let data;
5
6  window.onload = function() {
7    initMap();
8    cargarFichero(fichero);
9  }
10
11  function cargarFichero(fichero) {
12    let xhr = new XMLHttpRequest();
13    let datos;
14    xhr.open("GET", fichero, true);
15
16    xhr.onreadystatechange = function() {
17      if (this.readyState === 4 && this.status === 200) {
18        datos = JSON.parse(this.responseText);
19        initMap();
20        colocarPines(datos);
21        console.log(datos);
22      }
23    };
24    xhr.send();
25  }

```

Para pintar los pines usamos colocarPines, que tiene como parámetro la data obtenida:

```

41  function colocarPines(data) {
42    let lat;
43    let lng;
44    let nombre;
45    let infoWindowActivo;
46
47    /***** */
48
49    var iconBase = "https://maps.google.com/mapfiles/kml/shapes/";
50    var icons = {
51      capital: {
52        name: "capital",
53        icon: iconBase + "capital_big_highlight.png"
54      }
55    };

```


2. Recorrido para colocar los pines: recorreremos la data con un bucle foreach dentro de la función colocarPines().

```

41  function colocarPines(data) {
42      let lat;
43      let lng;
44      let nombre;
45      let infoWindowActivo;
46
47      /***** */
48
49      var iconBase = "https://maps.google.com/mapfiles/kml/shapes/";
50      var icons = {
51          capital: {
52              name: "capital",
53              icon: iconBase + "capital_big_highlight.png"
54          }
55      };
56
57      data.forEach(element => {
58          lat = element.latwgs84;
59          lng = element.lonwgs84;
60          nombre = element.documentName;
61          provincia = element.territory;
62          descripcion = element.touristmDescription;
63
64          if (lat != null || lng != null) {
65              lat = lat.replace(",", ".");
66              lng = lng.replace(",", ".");
67          }
68
69          const coordenadas = {
70              lat: Number(lat),
71              lng: Number(lng),
72              tipo: "capital"
73          };
74
75          let icono = icons[coordenadas.tipo];
76          if (icono != undefined) {
77              icono = icono.icon;
78          }
79          let marker = new google.maps.Marker({
80              position: coordenadas,
81              map: this.mapa,
82              icon: icono,
83          });
84          markers.push(marker);
85
86          let infoWindow = crearInfoWindow(
87              nombre,
88              descripcion
89          );
90
91          marker.addListener("click", () => {
92              if (infoWindowActivo) {
93                  infoWindowActivo.close();
94              }
95
96              infoWindow.open(this.mapa, marker);
97              infoWindowActivo = infoWindow;
98          });
99      });
100
101      return;
102  }

```

La función `crearInfoWindow` crea una ventana de información a partir de los datos suministrados en la llamada a la función

```
103
104 ✓ function crearInfoWindow(nombre, descripcion) {
105
106   let markerInfo = `<h1>${nombre}</h1>
107     <br><b>Descripcion</b>: ${descripcion}</br>
108     `;
109
110   infoWindow = new google.maps.InfoWindow({
111     content: markerInfo
112   });
113
114   return infoWindow;
115 }
```