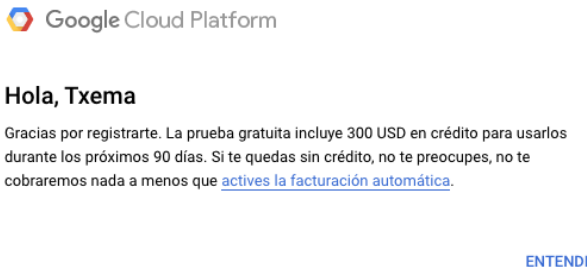
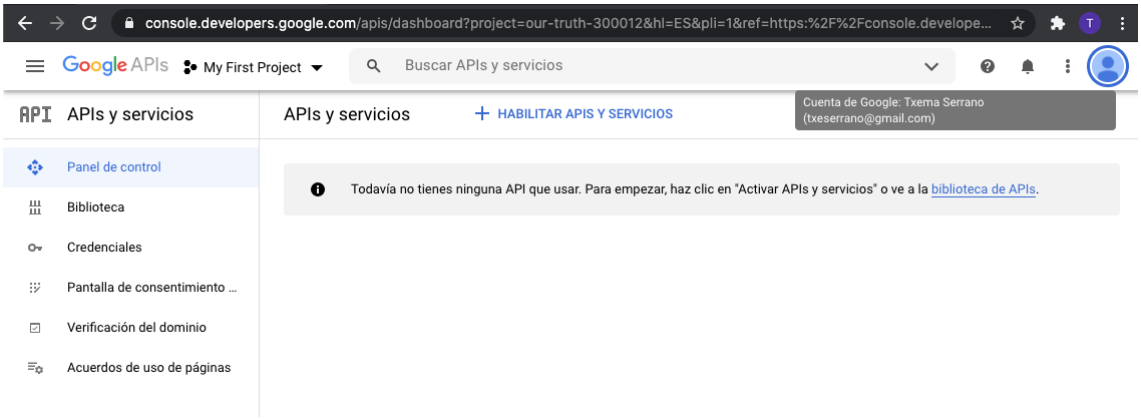
MAPAS DE GOOGLE PASO A PASO

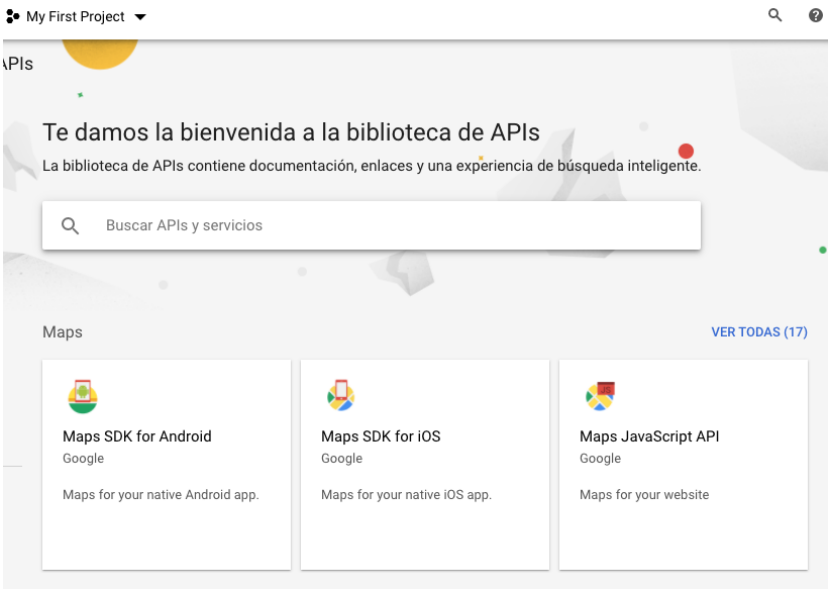
Con mi cuenta de Google Cloud Platform:



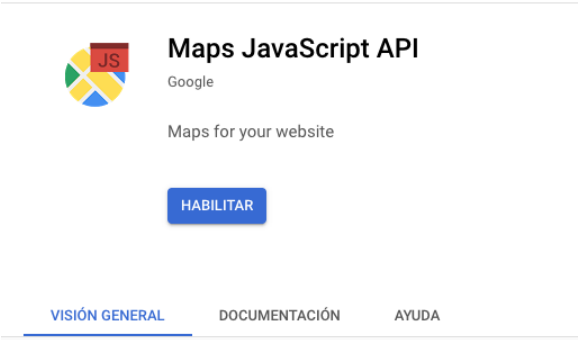
Inicio sesión con esta cuenta en developers.google.com:



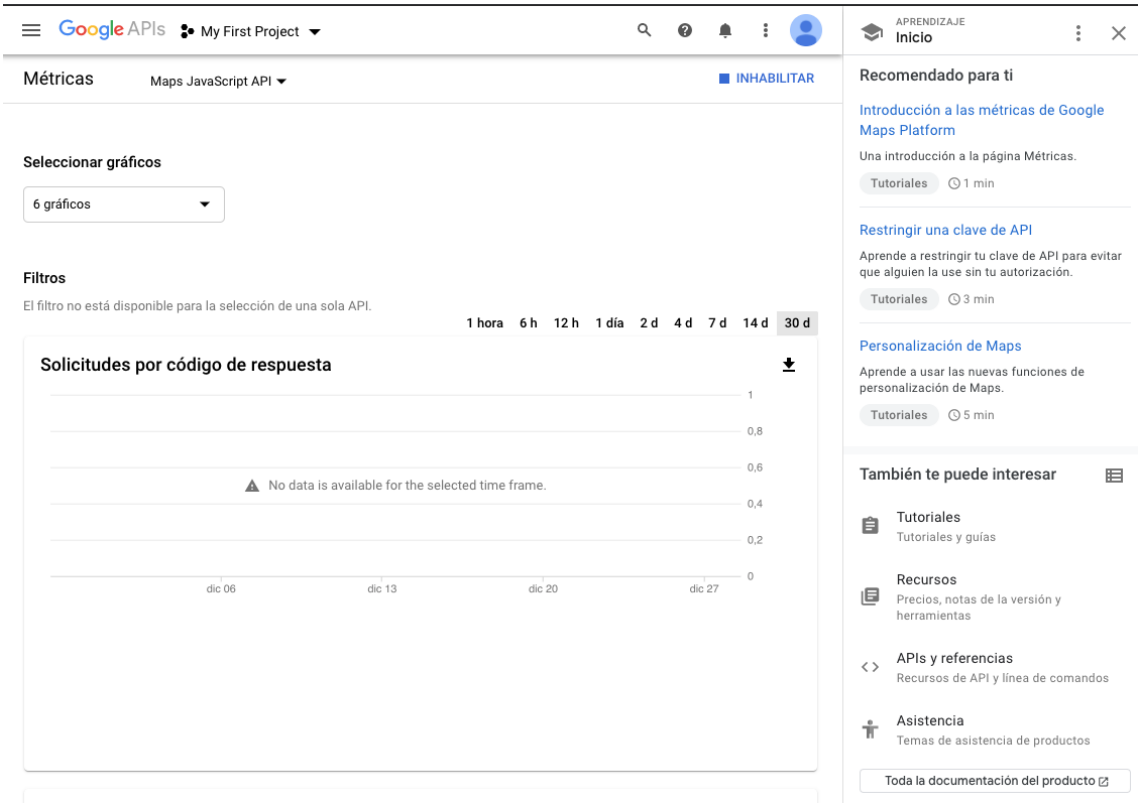
Le damos a Habilitar APIS y Servicios:



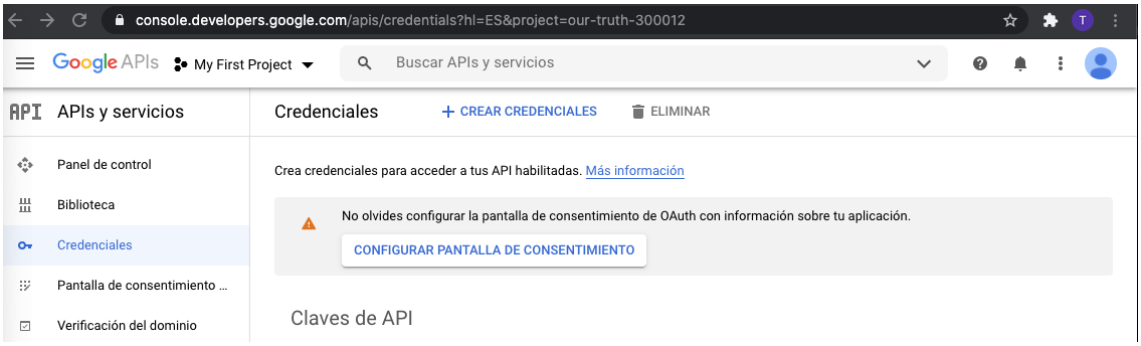
Entramos en Maps Javascripts API:



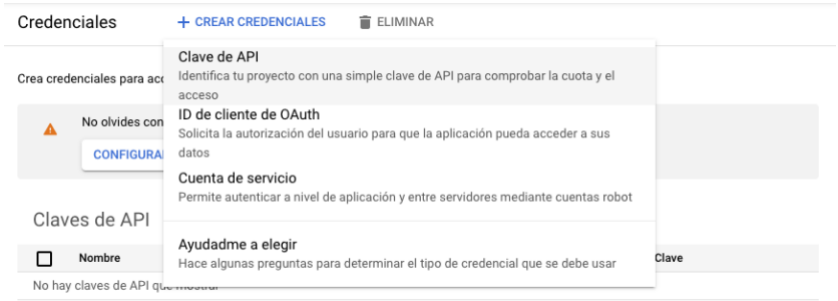
Le damos en Habilitar:



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

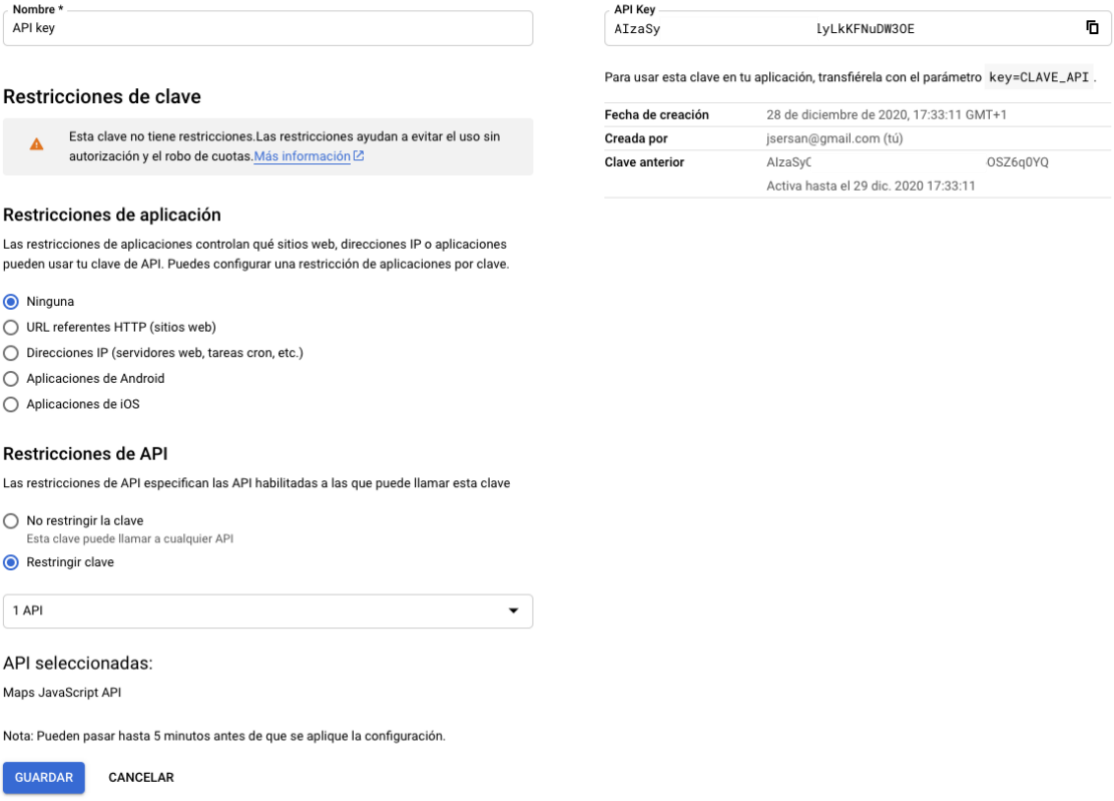


Pulsamos crear credenciales

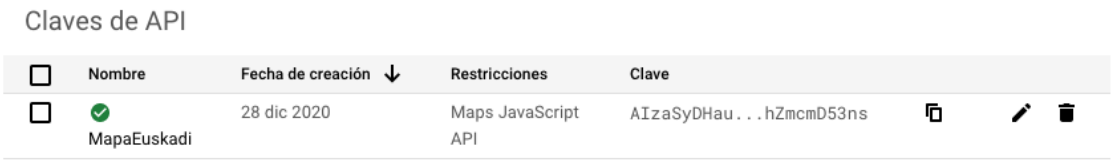


Clave de API

Restringimos la clave:

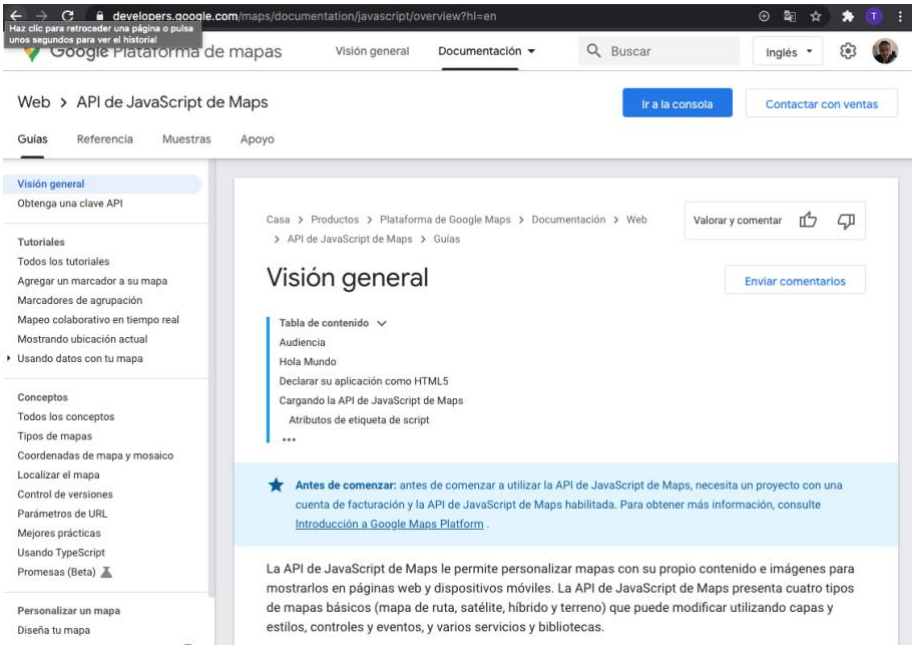


Guardamos y nuestra APIkey:



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.

Mecanografiado JavaScript CSS HTML Todas

```
let map: google.maps.Map;

function initMap(): void {
  map = new google.maps.Map(document.getElementById("map") as HTMLElement, {
    center: { lat: -34.397, lng: 150.644 },
    zoom: 8,
  });
}
```

index.ts

Nota: lea la guía sobre el uso de TypeScript y Google Maps.

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 .

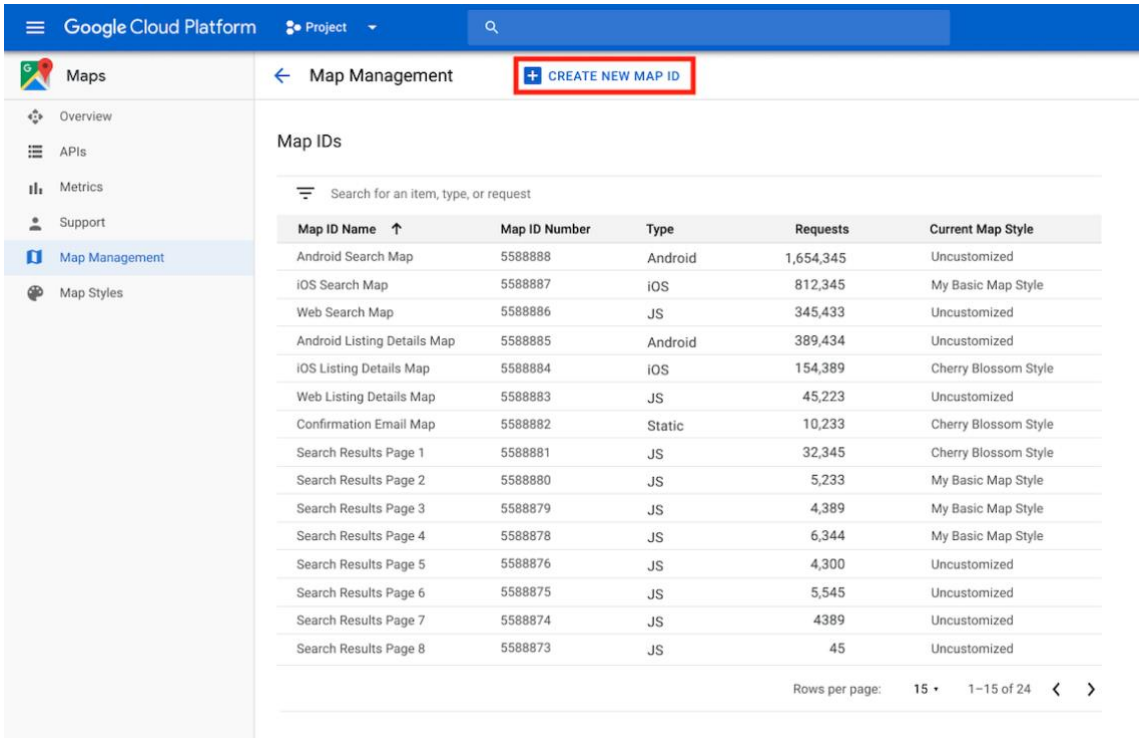
Crear ID de mapa

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**.



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:

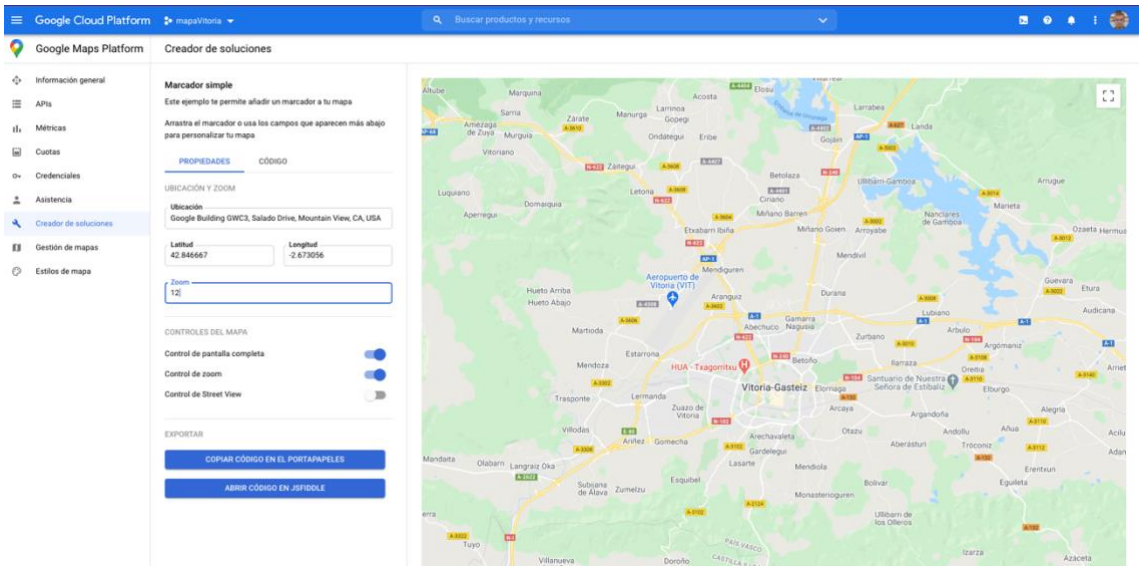
- ***API_KEY*** con la clave API para su proyecto.
- ***MAP_ID*** con uno o varios ID de mapa. Por ejemplo: `map_ids=1234` o `map_ids=1234,2345`

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:



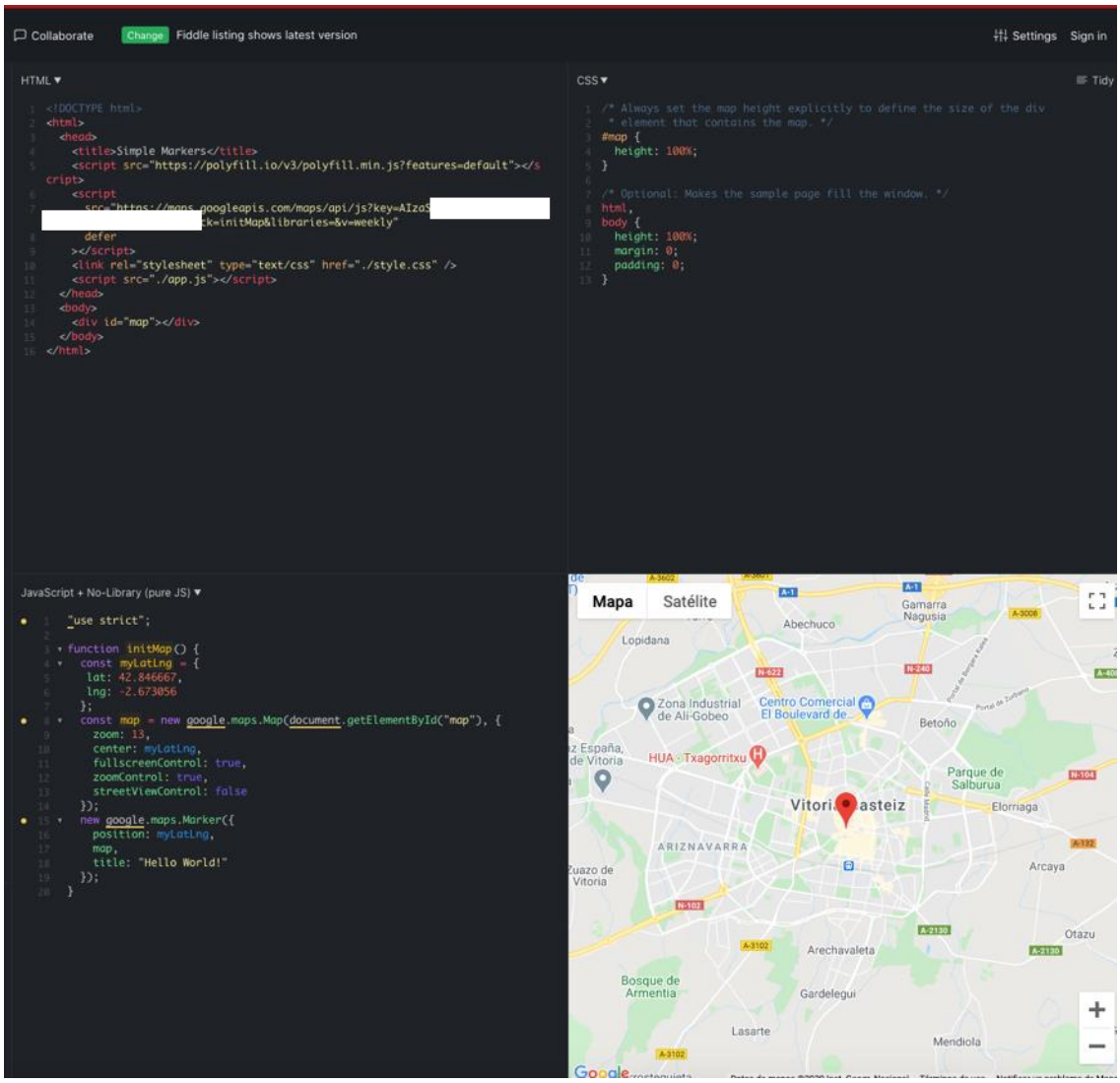
Si le damos a copiar el código en el portapapeles:

:adi filtrado/... index.html 6.Comunidad de Madrid apiKey.txt index.html ./

index.html > html > head > script

```
<!DOCTYPE html>
<html>
  <head>
    <title>Simple Markers</title>
    <script src="https://polyfill.io/v3/polyfill.min.js?features=default">
    </script>
    <script
      src="https://maps.googleapis.com/maps/api/js?key=AIzaSyAeRir0iEZxCi6c
      defer
    ></script>
    <style type="text/css">
      /* Always set the map height explicitly to define the size of the div
       * element that contains the map. */
      #map {
        height: 100%;
      }

      /* Optional: Makes the sample page fill the window. */
      html,
      body {
        height: 100%;
        margin: 0;
        padding: 0;
      }
    </style>
    <script>
      "use strict";

      function initMap() {
        const myLatLng = {
          lat: NaN,
          lng: NaN
        };
```

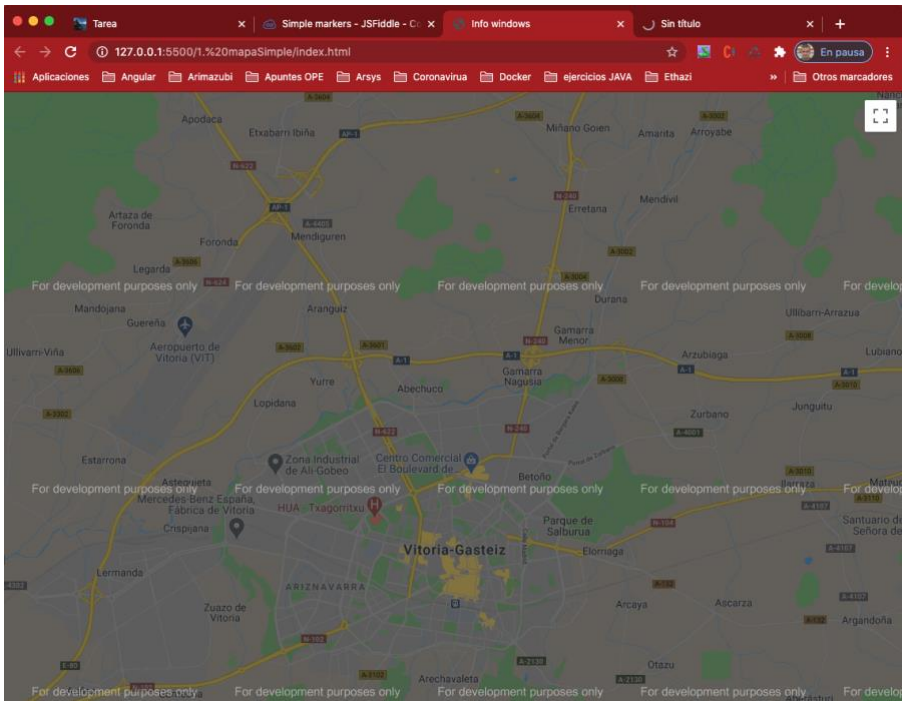
Podemos abrir el proyecto en JSFiddle:



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 vemtama de información en el pin.

```
function initMap() {

    var virgenBlanca = {
        lat: 42.846667,
        lng: -2.673056
    };
    var map = new google.maps.Map(document.getElementById('map'), {
        zoom: 13,
        center: virgenBlanca
    });

    var contentString = '<div id="content">' +
        '<div id="siteNotice">' +
        '</div>' +
        '<h1 id="firstHeading" class="firstHeading">Gasteiz</h1>' +
        '<div id="bodyContent">' +
        '<p><b>Plaza de la Virgen Blanca</b>,' +
        '<p>Vitoria-Gasteiz es la capital de la comunidad autónoma del País Vasco, situada en el norte
        '<p>Más información: Gasteiz, <a href="https://es.wikipedia.org/wiki/Vitoria">' +
        'https://es.wikipedia.org/wiki/Vitoria</a> '
    '</div>' +
    '</div>';

    var infowindow = new google.maps.InfoWindow({…
    });

    var marker = new google.maps.Marker({
        position: virgenBlanca,
        map: map,
        title: 'Virgen Blanca'
    });
    marker.addListener('click', function () {
        infowindow.open(map, marker);
    });
}
```

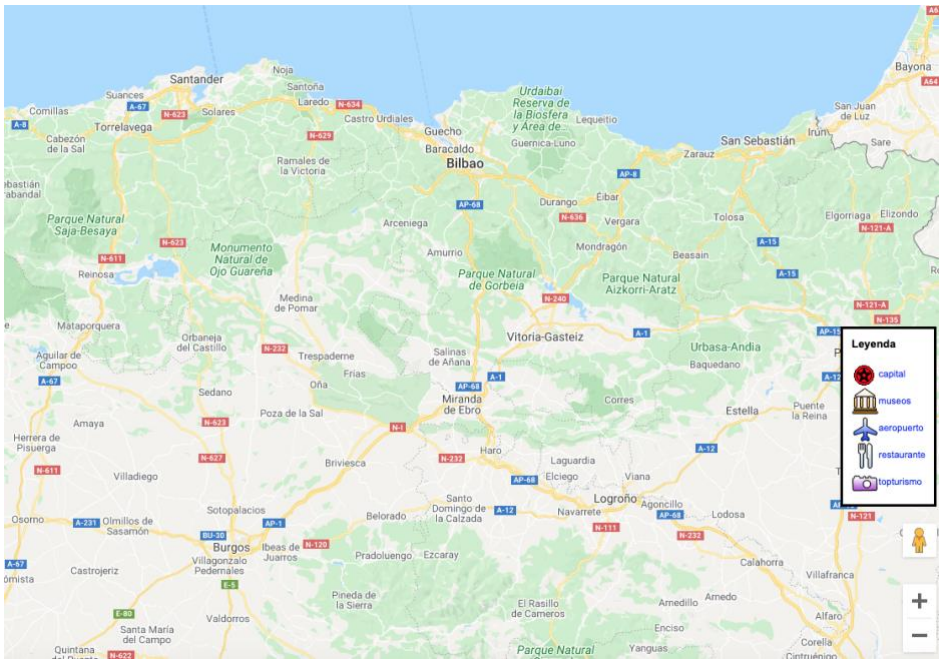
La salida es:

Mapa con leyenda.

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

```
function colocarLeyenda() {

    var iconBase = "https://maps.google.com/mapfiles/kml/shapes/";
    var icons = {
        capital: {
            name: "capital",
            icon: iconBase + "capital_big_highlight.png"
        },
        museos: {
            name: "museos",
            icon: iconBase + "museum_maps.png"
        },
        aeropuerto: {
            name: "aeropuerto",
            icon: iconBase + "airports_maps.png"
        },
        restaurante: {
            name: "restaurante",
            icon: iconBase + "dining_maps.png"
        },
        topturismo: {
            name: "topturismo",
            icon: iconBase + "camera_maps.png"
        }
    };

    if (!inicio) {
        for (var key in icons) {
            var type = icons[key];
            var name = type.name;
            var icon = type.icon;
            var div = document.createElement("div");
            div.setAttribute("id", "leyenda");
            var enlace =
                '<a href="' +
                "mapaEuskadi.html?tipo=" +
                name +
                '"><img src="' +
                icon +
                '">' +
                name +
                "</a>";
            div.innerHTML = enlace;

            legend.appendChild(div);
        }

        inicio = true;
    }

    this.mapa.controls[google.maps.ControlPosition.RIGHT_BOTTOM].push(legend);

    return;
}

```

La salida es:



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

```
window.onload = function() {
    initMap();
    colocarLeyenda();

    var valores = getGET();
    if (valores) {
        //recogemos los valores que nos envia la URL en variables para trabajar
        //con ellas
        var tipo = valores['tipo'];
    }
    cargarDatos(tipo);
}


function getGET() {
    // capturamos la url
    var loc = document.location.href;

    // si existe el interrogante
    if (loc.indexOf("?") > 0) {
        // cogemos la parte de la url que hay despues del interrogante
        var getString = loc.split("?")[1];

        // obtenemos un array con cada clave=valor
        var GET = getString.split("&");
        var get = {};

        // recorremos todo el array de valores
        for (var i = 0, l = GET.length; i < l; i++) {
            var tmp = GET[i].split("=");
            get[tmp[0]] = unescape(decodeURI(tmp[1]));
        }
        return get;
    }
}
```

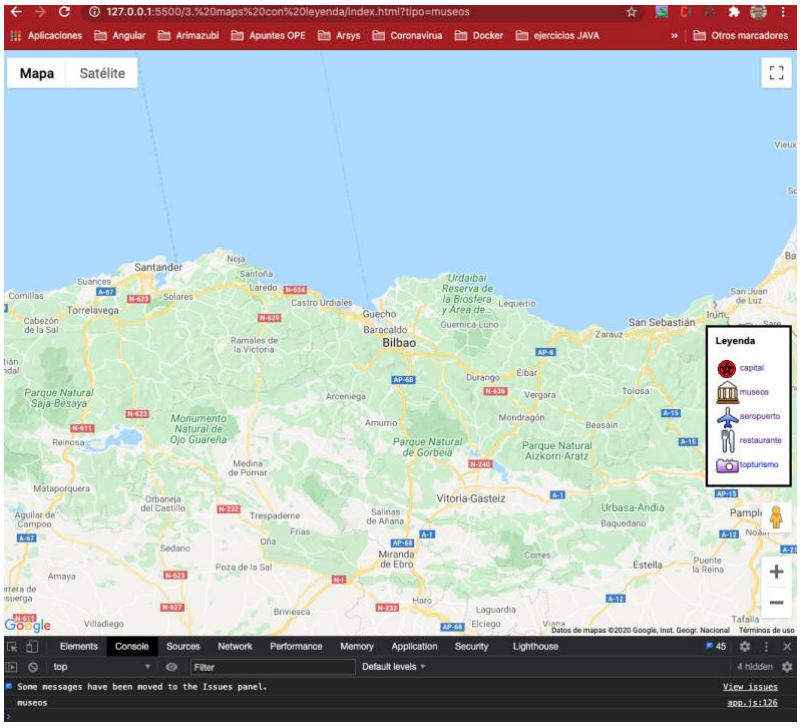
Ahora que ya sabemos cuál es el tipo de dato seleccionado, lo mostramos por consola:

```
function cargarDatos(tipo) {
    switch (tipo) {
        case "restaurante":
            console.log("Restaurante");
            break;

        case "aeropuerto":
            console.log("Aeropuerto");
            break;

        case "museos":
            console.log("museos");
            break;

        case "topturismo":
            console.log("turismo");
            break;

        case "capital":
            console.log("capital");
            break;
    }
}
```

No debemos olvidar declarar las variables utilizadas:

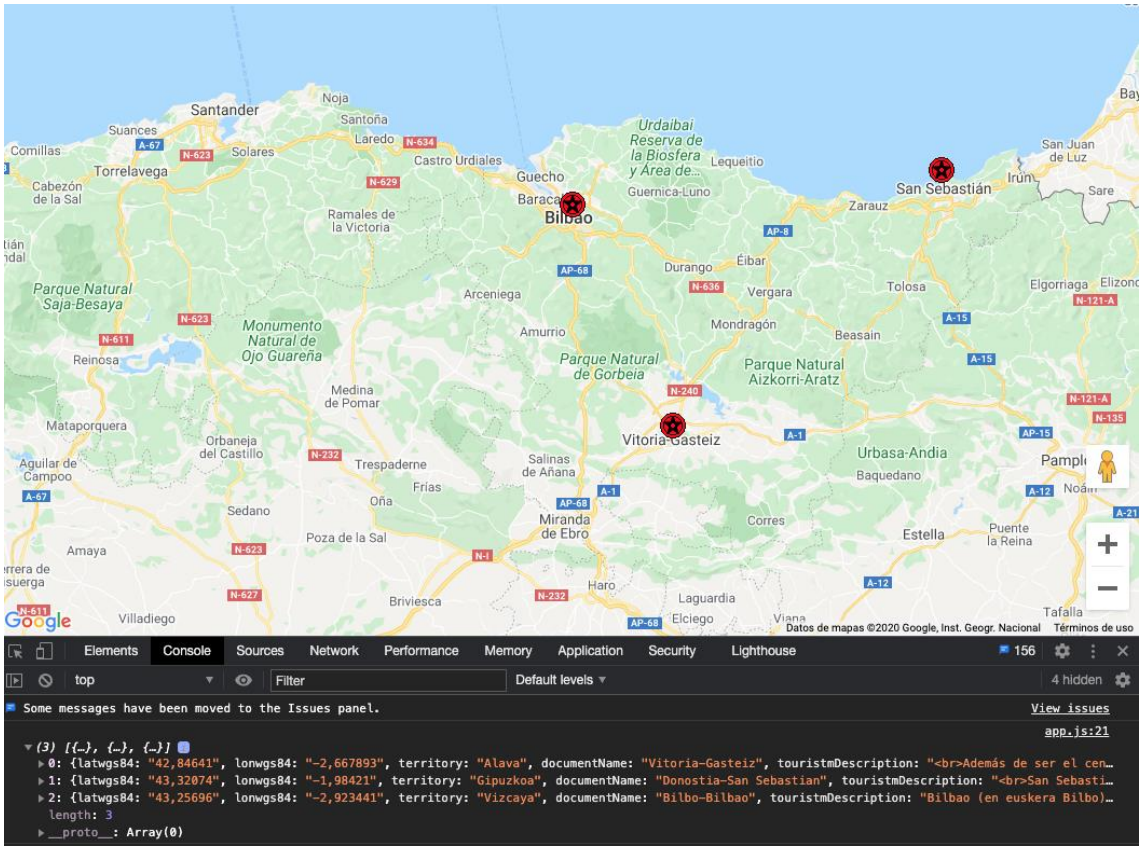
```
let mapa;
let tipo = "";
let markers = [];

let emoticon; // modificado 20/04
let legend = null;
legend = document.getElementById("legend");
let inicio = false;
```

La salida es la siguiente si escogemos museos:

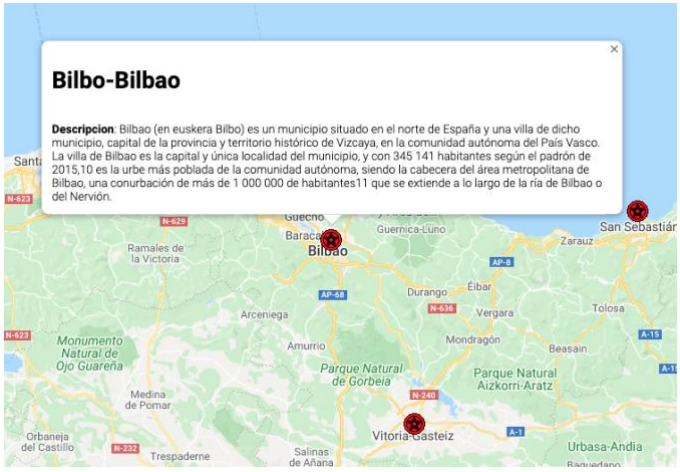


Mapa con marcas proveniente de un fichero JSON

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:

```json
[{
    "latwgs84": "42,84641",
    "lonwgs84": "-2,667893",
    "territory": "Alava",
    "documentName": "Vitoria-Gasteiz",
    "touristmDescription": "<br>Además de ser el centro político y administrativo de Euskadi, Vito
    "paginaWeb": "https://www.vitoria-gasteiz.org/"
},
{
    "latwgs84": "43,32074",
    "lonwgs84": "-1,98421",
    "territory": "Gipuzkoa",
    "documentName": "Donostia-San Sebastian",
    "touristmDescription": "<br>San Sebastián (en euskera Donostia y oficialmente Donostia/San Seb
    "paginaWeb": "https://www.donostia.eus/"
},
{
    "latwgs84": "43,25696",
    "lonwgs84": "-2,923441",
    "territory": "Vizcaya",
    "documentName": "Bilbo-Bilbao",
    "touristmDescription": "Bilbao (en euskera Bilbo) es un municipio situado en el norte de Espa
    "paginaWeb": "http://bilbao.eus/"
}
]
```

La función AJAX de carga de los datos:

4- maps con json local > script > JS app.js > ...

```js
let mapa;
let markers = [];
let fichero = 'datoscapitales.json';
let data;

window.onload = function() {
    initMap();
    cargarFichero(fichero);
}

function cargarFichero(fichero) {
    let xhr = new XMLHttpRequest();
    let datos;
    xhr.open("GET", fichero, true);

    xhr.onreadystatechange = function() {
        if (this.readyState === 4 && this.status === 200) {
            datos = JSON.parse(this.responseText);
            initMap();
            colocarPines(datos);
            console.log(datos);
        }
    };
    xhr.send();
}
```

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

```js
function colocarPines(data) {
    let lat;
    let lng;
    let nombre;
    let infoWindowActivo;

    /********************** */

    var iconBase = "https://maps.google.com/mapfiles/kml/shapes/";
    var icons = {
        capital: {
            name: "capital",
            icon: iconBase + "capital_big_highlight.png"
        }
    };
```

2. Recorrido para colocar los pines: recorremos 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

```
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 }
```