Lab - Angular Google Map Street View

Hace aproximadamente un año, Angular introdujo un componente de mapas de Google (@angular/google-maps) como una envoltura de la API javascript de mapas de Google. En este post, vamos a repasar algunos conceptos básicos y consejos sobre cómo utilizar el componente en su aplicación.

Antes de comenzar

Para poder utilizar el componente de mapas de Google, necesitamos obtener primero una clave de API de mapas de Google. Si aún no tienes la clave, sigue los pasos aquí para crear una: https://developers.google.com/maps/documentation/javascript/get-api-key

Set up

Después de obtener la clave de la API, tenemos que añadir el siguiente script a index.html de la aplicación para cargar un mapa de Google. Aquí hay una plantilla de la etiqueta del script. Por favor, sustituye "YOUR_API_KEY" por tu propia clave y añádela a la etiqueta head de la aplicación.

```
<script
src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY"></scrip
t>
```

Instalar @angular/google-maps corriendo el siguiente script.

```
npm i @angular/google-maps
```

En el módulo de su aplicación o en el módulo del componente, importar GoogleMapsModule.

```
import { GoogleMapsModule } from '@angular/google-maps'

      @NgModule({
        declaration: [ ... ]
        imports: [ ..., GoogleMapsModule ],
       })

      export class AppModule { }
```

Ahora se añade la etiqueta google-map a su HTML.

```
<div class="my-google-map">
 <google-map></google-map>
</div>
```

Manejando errores comunes

Antes de pasar a colocar el código, vamos a considerar que errores pueden darse al cargar el mapa.

Error Comun 1

Error: Namespace google not found, cannot construct embedded google map. Please install the Google Maps JavaScript API:

https://developers.google.com/maps/documentation/javascript/tutorial#Loading_the_Maps_API

Solucion: Si tienes el error, tienes que comprobar si tu clave de la API de Google Maps es válida. Además, comprueba que la etiqueta script se ha añadido dentro del elemento head en el DOM.

Error Comun 2

Google Maps JavaScript API error: RefererNotAllowedMapError
https://developers.google.com/maps/documentation/javascript/error-messages#referer-not-allowed-map-error
Your site URL to be authorized: http://localhost:4200/

Solucion: Si recibe este error, significa que sus claves API están restringidas a ciertos sitios web. Por favor, siga este enlace y permita sus sitios web para su clave API. https://console.cloud.google.com/apis/credentials. Hay muchas formas de restringir tus claves API como la URL, la dirección IP o el tipo de dispositivos (ios o android). Si no quieres ninguna restricción, puedes simplemente cambiar la restricción de la aplicación a Ninguna. Nota: cambiar las restricciones puede llevar hasta 5 minutos

Agregando Map y Marker

Al cargar la aplicación, verás un simple mapa de Google. Hay varias entradas disponibles para el componente y las listas son las siguientes.

1. width & height
2. mapTypeId
3. center
4. zoom
5. options

Si planea utilizar el mapa sólo con el centro y el zoom, puede simplemente pasar esas dos entradas al componente. Sin embargo, si desea una configuración más avanzada, puede utilizar el parámetro de opciones para pasar más ajustes al componente. Dado que el parámetro de opciones también incluye las propiedades 'mapTypeId', 'center' y 'zoom', puede incluir todo en las opciones sin necesidad de utilizar entradas separadas. Puede visitar el siguiente enlace para obtener una lista completa de las opciones del propiedades de mapa

https://developers.google.com/maps/documentation/javascript/reference/map#MapOptions

El componente del marcador está disponible como "creador de mapas", que toma la posición y las opciones como entradas. Hay muchos eventos diferentes disponibles para

el componente marcador, pero en este artículo sólo nos centraremos en la posición del marcador. Si quiere ver más sobre otras entradas y eventos para el componente marcador, se puede visitar a:

https://github.com/angular/components/tree/master/src/google-maps/map-marker

app.component.css

```
.my-google-map{
   width: 800px;
   height: 600px;
 }
```

app.component.html

```
<div class="my-google-map">
   <google-map [options]="mapOptions">
      <map-marker [position]="marker.position"></map-marker>
   </google-map>
</div>
```

Agregar a la clase app.component.ts:

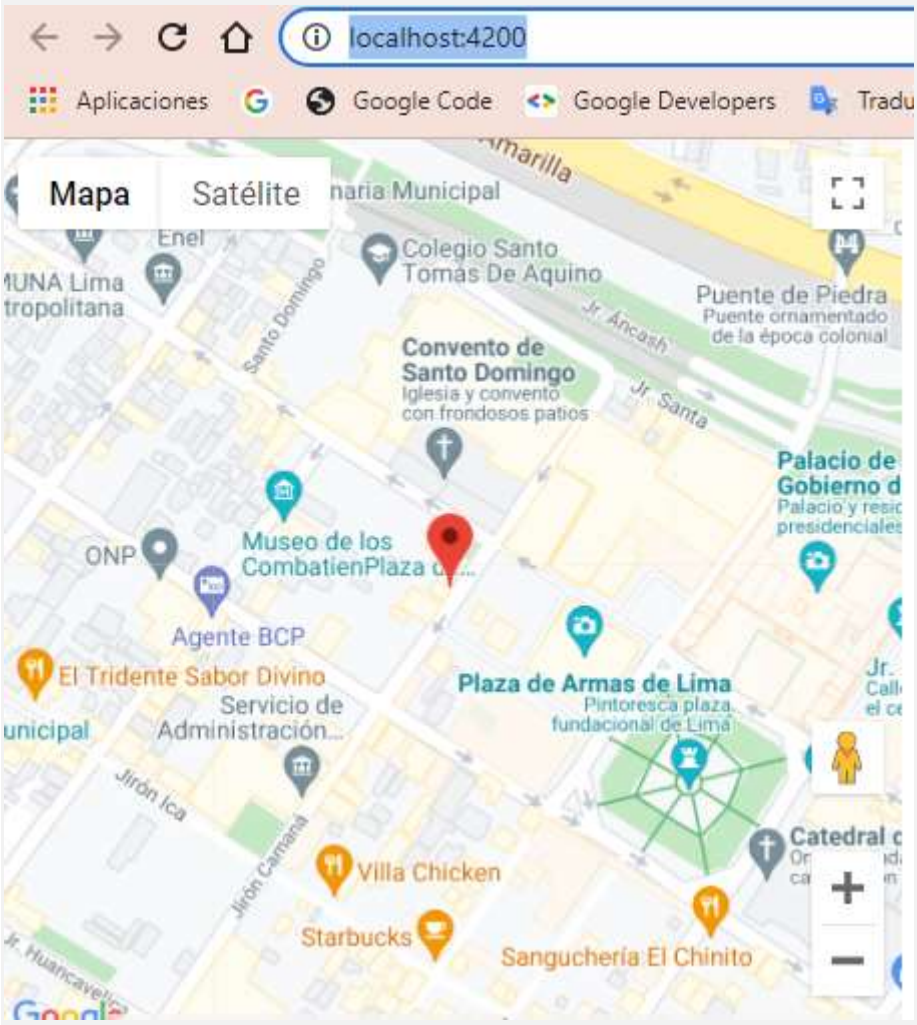
app.component.ts

```
mapOptions: google.maps.MapOptions = {
   center: { lat: -12.0448403, lng: -77.0319752},
   zoom : 14
}
marker = {
   position: { lat: -12.0448403, lng: -77.0319752},
}
```

Para ello la pagina index.html quedara la siguiente manera:

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Uso de Streets-View</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link rel="stylesheet"
href="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css">

</head>
```

```
<body>
  <app-root></app-root>
  <script
  async defer
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyAj6juj6zAvMz8O2ccKY2OZYH1lEaQTqW4">
  </script>
</body>
</html>
```



Implementaciones

1. Customizando el Ancho y Altura del componente

La anchura y la altura del mapa de Google se pueden cambiar pasando números (en píxeles) al componente. También, si simplemente quieres tomar todo el espacio del contenedor padre, puedes pasar 100% a las entradas de anchura y altura. En muchos casos, sin embargo, es posible que desee utilizar un archivo CSS para pasar los estilos al componente. Puede utilizar el CSS pasando null a las propiedades width y height. Para el

ejemplo de abajo, sólo tiene una anchura y altura por defecto, pero puedes tener tus propios tamaños personalizados para diferentes consultas de medios como las vistas de escritorio, tableta y móvil.

app.component.html

```
<div class="my-google-map">
   <google-map              [width]="null"              [height]="null"
[options]="mapOptions"></google-map>
</div>
```

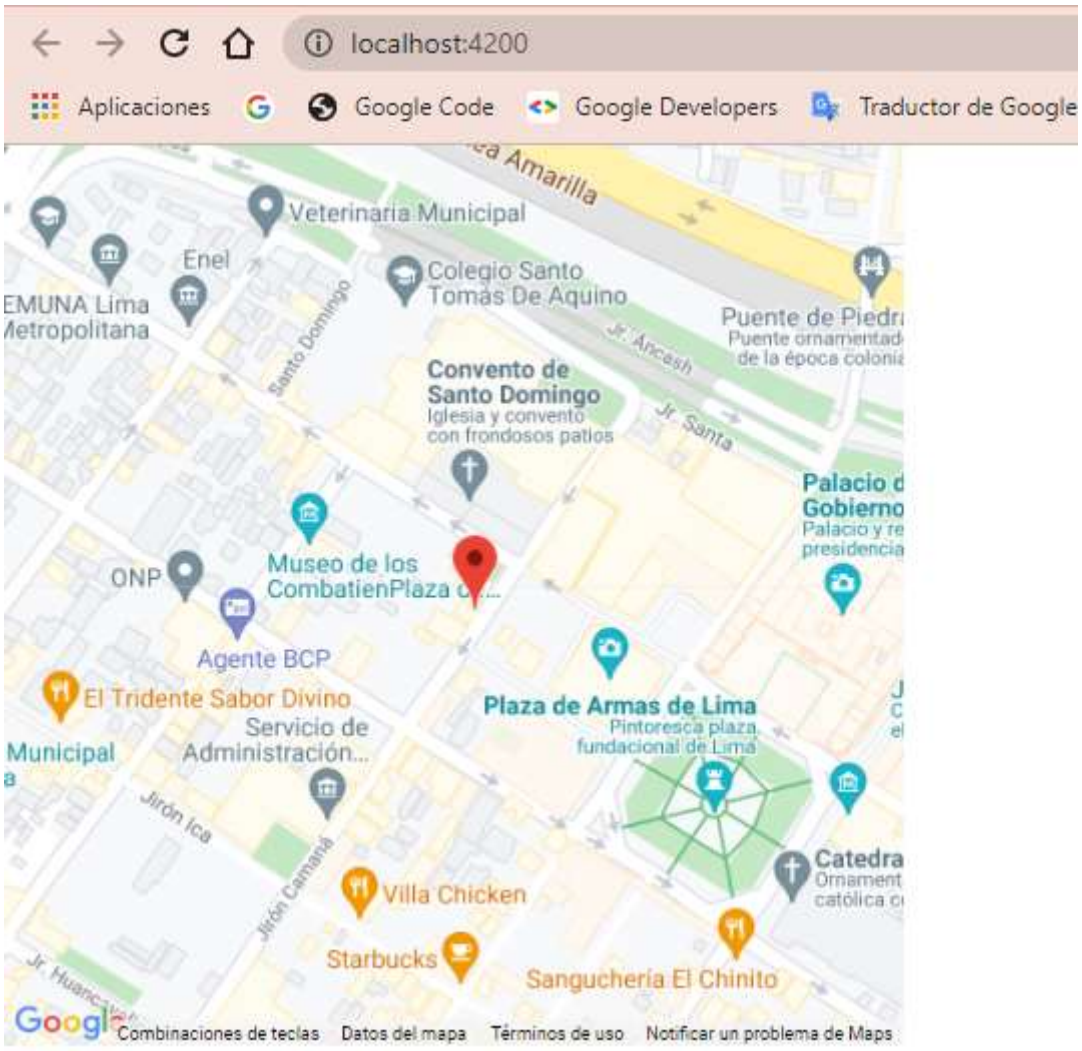
app.component.css

```
.my-google-map{
     width: 800px;
    height: 600px;
}
```

2. Removiendo botones por defecto

Hay un par de maneras de eliminar los botones de la UI que no quieres. Una forma es establecer el disableDefaultUI a true. Esto eliminará todos los controles de zoom, de tipo de mapa, de vista de calle y de pantalla completa del mapa.

```
options: google.maps.MapOptions = {
   ...
   disableDefaultUI: true,
}
```



La otra forma es establecer esos controles a false individualmente. De esta manera, usted puede mantener lo que necesita y eliminar lo que no quiere.

```
options: google.maps.MapOptions = {
   ...
   zoomControl: false,
   mapTypeControl: false,
   streetViewControl: false,
   fullscreenControl: false
}
```

3. Fit the Map to Markers

Si tiene más de 2 marcadores en el mapa, sería muy difícil calcular la posición central y el nivel de zoom para que quepan todos los marcadores en el mapa. En este caso, puede utilizar el método 'fitBounds' del componente de mapa de Google. Toma el tipo LatLngBoundsLiteral que tiene como entrada los valores norte, sur, este y oeste. Aquí hay un ejemplo de ajuste de los marcadores sin proporcionar una posición central específica o nivel de zoom.

app.component.html

```
<div class="my-google-map">
  <google-map [options]="mapOptions">
    <map-marker *ngFor="let marker of markers" [position]="marker.position"></map-marker>
 </google-map>
</div>
```

Agregar a app.component.ts

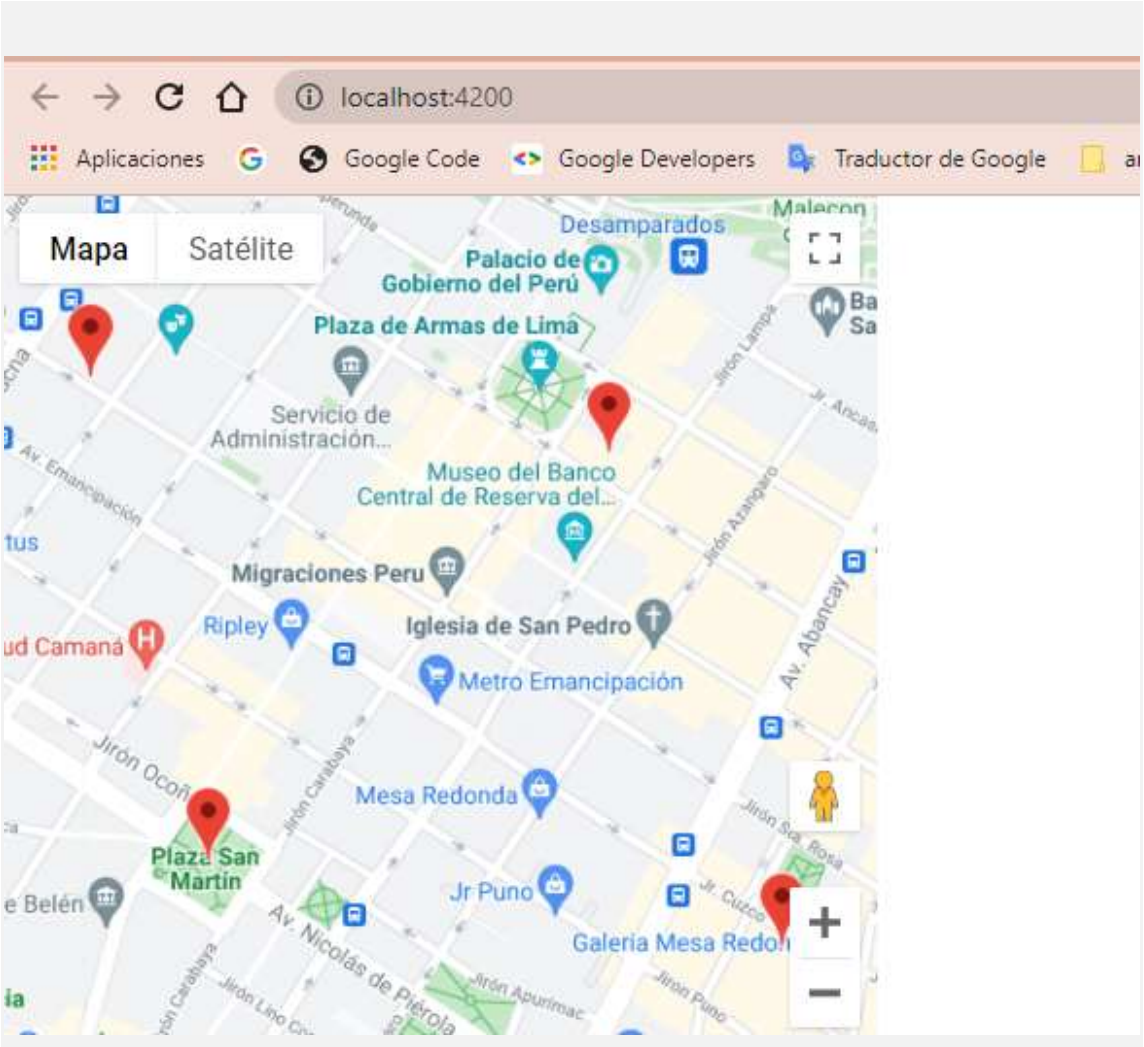
Inicialmente, debe agregarse los imports de esta manera:

```
import { Component, OnInit, ViewChild} from '@angular/core';
import { GoogleMap } from '@angular/google-maps';
```

y luego el siguiente código:

```
  marker1 = { position: { lat: -12.0526789, lng: -77.0275729 } };
  marker2 = { position: { lat: -12.0458476, lng: -77.0360521 } };
  marker3 = { position: { lat: -12.0516596, lng: -77.0346048 } };
  marker4 = { position: { lat: -12.0467646, lng: -77.0297023 } };


  markers: any[] = [this.marker1, this.marker2, this.marker3, this.marker4];

@ViewChild(GoogleMap) map!: GoogleMap;

ngAfterViewInit(){
  const bounds = this.getBounds(this.markers);
  this.map.fitBounds(bounds);
}

getBounds(markers){
  let north;
  let south;
  let east;
  let west;

  for (const marker of markers){
    // set the coordinates to marker's lat and lng on the first run.
    // if the coordinates exist, get max or min depends on the coordinates.
```

```
    north = north !== undefined ? Math.max(north, marker.position.lat) :
marker.position.lat;
    south = south !== undefined ? Math.min(south, marker.position.lat) :
marker.position.lat;
    east = east !== undefined ? Math.max(east, marker.position.lng) :
marker.position.lng;
    west = west !== undefined ? Math.min(west, marker.position.lng) :
marker.position.lng;
  };

  const bounds = { north, south, east, west };
  return bounds;
}
```



4. Starting with Street View

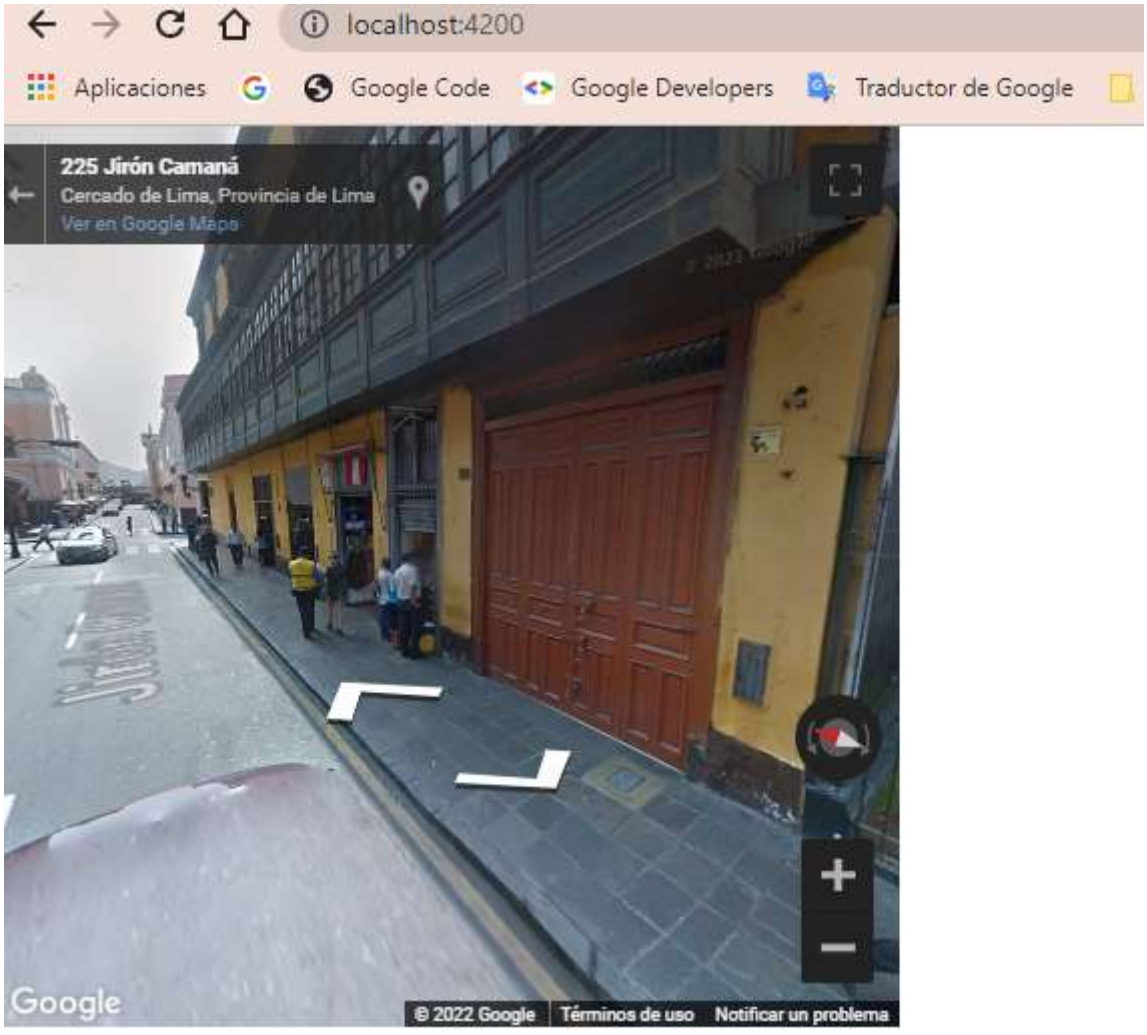
Después de inicializar la vista, podemos obtener una vista de calle del mapa y establecer las opciones para modificar el mapa. Después de establecer las opciones, setVisible(true) convertirá tu mapa en la vista de calle.

Se modificara, el archivo app.component.ts

@ViewChild(GoogleMap) map!: GoogleMap;

```
ngAfterViewInit(){
  const streetView = this.map.getStreetView();

streetView.setOptions({
   position: { lat: -12.0448403, lng: -77.0319752},
   pov: { heading: 70, pitch: -10 },
 });

 streetView.setVisible(true);
}
  streetView.setVisible(true);
}
```

Nota: si obtienes una pantalla negra vacía, es posible que la posición que has pasado no esté disponible para la vista de calle.



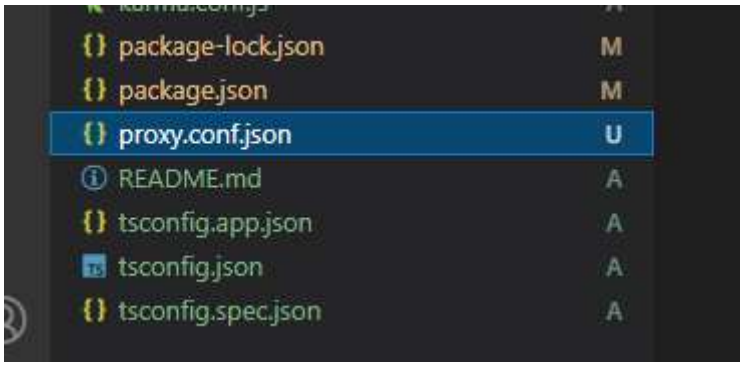
Nota importante:

Es muy probable que la imagen no se produzca, debido al siguiente error:

Access to image at from origin 'http://localhost:4200' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.

Esto se debe a que algunos servicios requieren de confirmación via token.

Para solucionar esto, debe crearse un archivo en el proyecto:



Y luego se colocará el siguiente código:

Luego en package.json se amodificara el script “start” de la siguiente manera:


app.component.html M
app.component.ts M
package.json 1, M
inde
package.json > dependencies
{
"name": "uso-streets",
"version": "0.0.0",
Depuración de
"scripts": {
"ng": "ng",
"start": "ng serve --proxy-config proxy.conf.json",
"build": "ng build",
"watch": "ng build --watch --configuration development",
"test": "ng test"
},
"private": true,
"dependencies": {
"@angular/animations": "~13.1.0",
"@angular/common": "~13.1.0",