GOOGLE MAPS JS API

PROJETO I
LTSIW-ESMAD-2018
Ricardo Queirós

GOOGLE MAPS JS API

AGENDA

1. Introdução
2. O meu primeiro mapa!
3. Mapas
4. Overlays
5. Layers
6. Serviços
7. Integração com outras APIs

GOOGLE MAPS JS API

1. INTRODUÇÃO

- Criado pela Google em Junho de 2005
- Objetivo: permitir aos programadores integraro Google Maps nos seus sites
- Atualmente na versão 3.32 (fevereiro, 2018)
- Disponível de forma gratuita e paga
- Serviço gratuito limitado (25 000 carregamentos de mapas por dia)
- Disponível em Android, iOS, web browsers e via serviços Web (HTTP)

- Mais de 2 milhões de sítios Web usam a Google Maps API

GOOGLE MAPS JS API

AGENDA

1. Introdução
2. **O meu primeiro mapa!**
3. Mapas
4. Overlays
5. Layers
6. Serviços
7. Integração com outras APIs

GOOGLE MAPS JS API

2. O MEU PRIMEIRO MAPA!

Passos para criar um mapa básico:

1. Criar uma página básica HTML
2. Adicionar uma ** ** que vai ser o local da exibição do mapa
3. Definir a função JavaScript para criar o mapa na ** **
4. Carregar a API Maps usando o element **<script>**

GOOGLE MAPS JS API

2. O MEU PRIMEIRO MAPA!

Passos para criar um mapa básico:

1. Criar uma página básica HTML

```
<!DOCTYPE html>
<html>
  <head></head>
  <body>
   …
  </body>
</html>
```

GOOGLE MAPS JS API

2. O MEU PRIMEIRO MAPA!

Passos para criar um mapa básico:

2. Adicionar uma ** ** que vai ser o local da exibição do mapa

```
<body>
    <div id="myMap" style="width:500px; height:500px"></div>
</body>
```

GOOGLE MAPS JS API

2. O MEU PRIMEIRO MAPA!

Passos para criar um mapa básico:

3. Definir a função JavaScript para criar o mapa na

```
<body>
  …
  <script>
    let map
    let myMap = document.getElementById('myMap')
    function initMap() {
      let mapProp = {
          center: new google.maps.LatLng(37.508742,-0.420850),
          zoom: 5
      }
      map = new google.maps.Map(myMap, mapProp)
    }
  </script>
</body>
```

Que nível de zoom devo usar?

1: Mundo
5: Continente
10: Cidade
15: Ruas
20: Edifícios

GOOGLE MAPS JS API

2. O MEU PRIMEIRO MAPA!

Passos para criar um mapa básico:

4. Carregar a API Maps usando o element <script>

```
<body>
  …
  <script>…</script>
  <script src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&callback=initMap" async>
  </script>
</body>
```

obter chave da API

https://console.developers.google.com

GOOGLE MAPS JS API

2. O MEU PRIMEIRO MAPA!

Passos para criar um mapa básico:


```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>DEMO #01 GMJSAPI</title>
</head>
<body>
    <div id="myMap" style="width:500px; height:500px"></div>
    <script>
        let map
        let myMap = document.getElementById("myMap")
        function initMap() {
            let mapProp= {
                center:new google.maps.LatLng(37.508742,-0.420850),
                zoom:5
            }
            map = new google.maps.Map(myMap, mapProp)
        }
    </script>
    <script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyDcYng9pBfj_LmDthyiydpfyy3bftchOmg&callback=initMap" async></script>
</body>
</html>
```

GOOGLE MAPS JS API

AGENDA

1. Introdução
2. O meu primeiro mapa!
3. **Mapas**
4. Overlays
5. Layers
6. Serviços
7. Integração com outras APIs

GOOGLE MAPS JS API

3. MAPAS

- Tipos de mapas
- Controlos
- Eventos
- Estilos
- Geolocalização

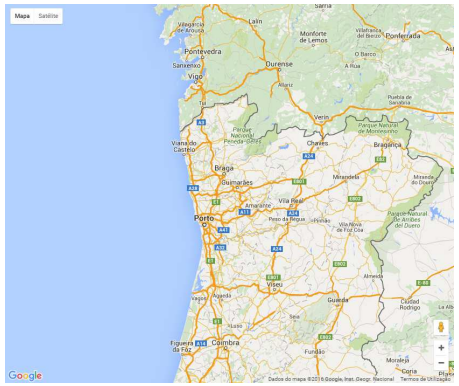
GOOGLE MAPS JS API

3. MAPAS

Tipos de mapas

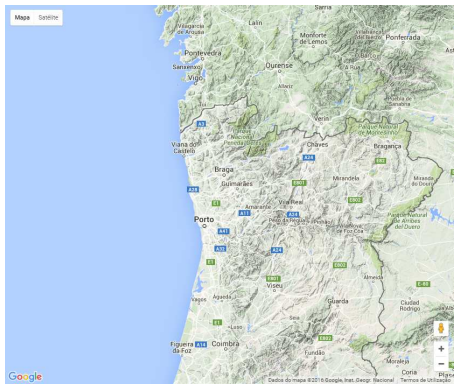
- Tipos de mapas suportados:
  - **ROADMAP** (normal, mapa das estradas por omissão)
  - **SATELLITE** (fotográfico – imagens satélite Google Earth)
  - **HYBRID** (fusão dos dois anteriores)
  - **TERRAIN** (mapa com montanhas, rios, etc.)

ROADMAP

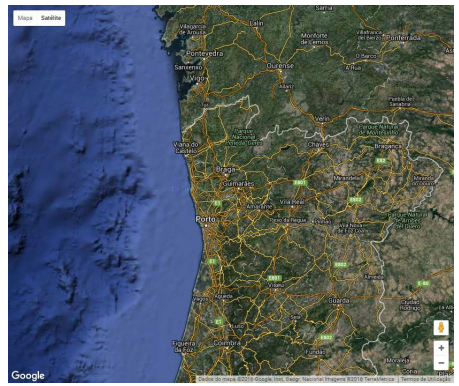


SATELLITE

TERRAIN

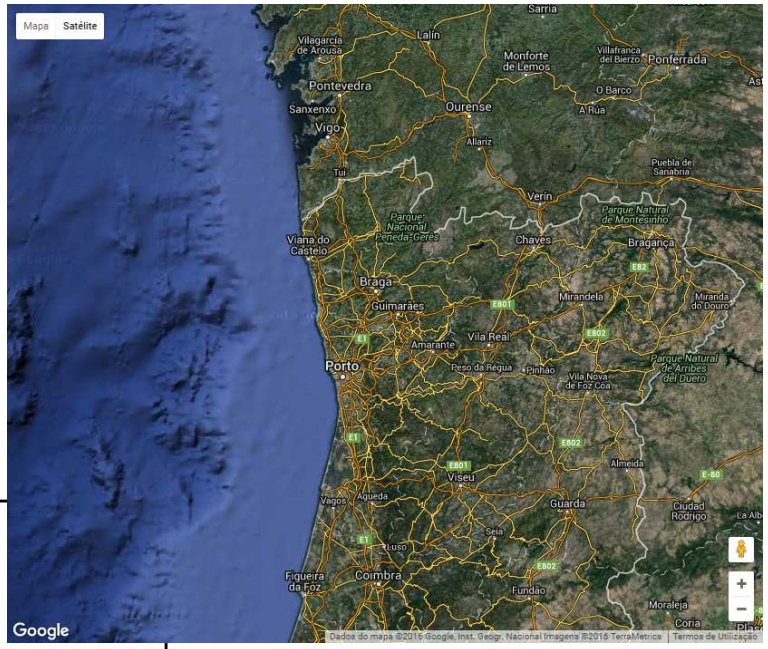


HYBRID



GOOGLE MAPS JS API

3. MAPAS



Tipos de mapas

- Definição do tipo de mapa:
  - Como propriedade das opções do mapa:

```
let myLatlng = new google.maps.LatLng(-34.397, 150.644)
let mapOptions = {
 zoom: 8,
 center: myLatlng,
 mapTypeId: google.maps.MapTypeId.HYBRID
}
let map = new google.maps.Map(document.getElementById("map"),  mapOptions)
```

  - Ou através do método **setMapTypeId()**:

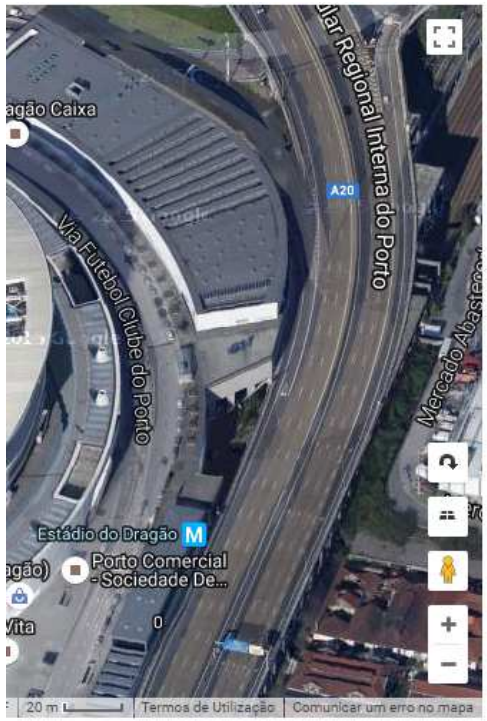
```
map.setMapTypeId(google.maps.MapTypeId.HYBRID)
```

GOOGLE MAPS JS API

3. MAPAS

Controlos

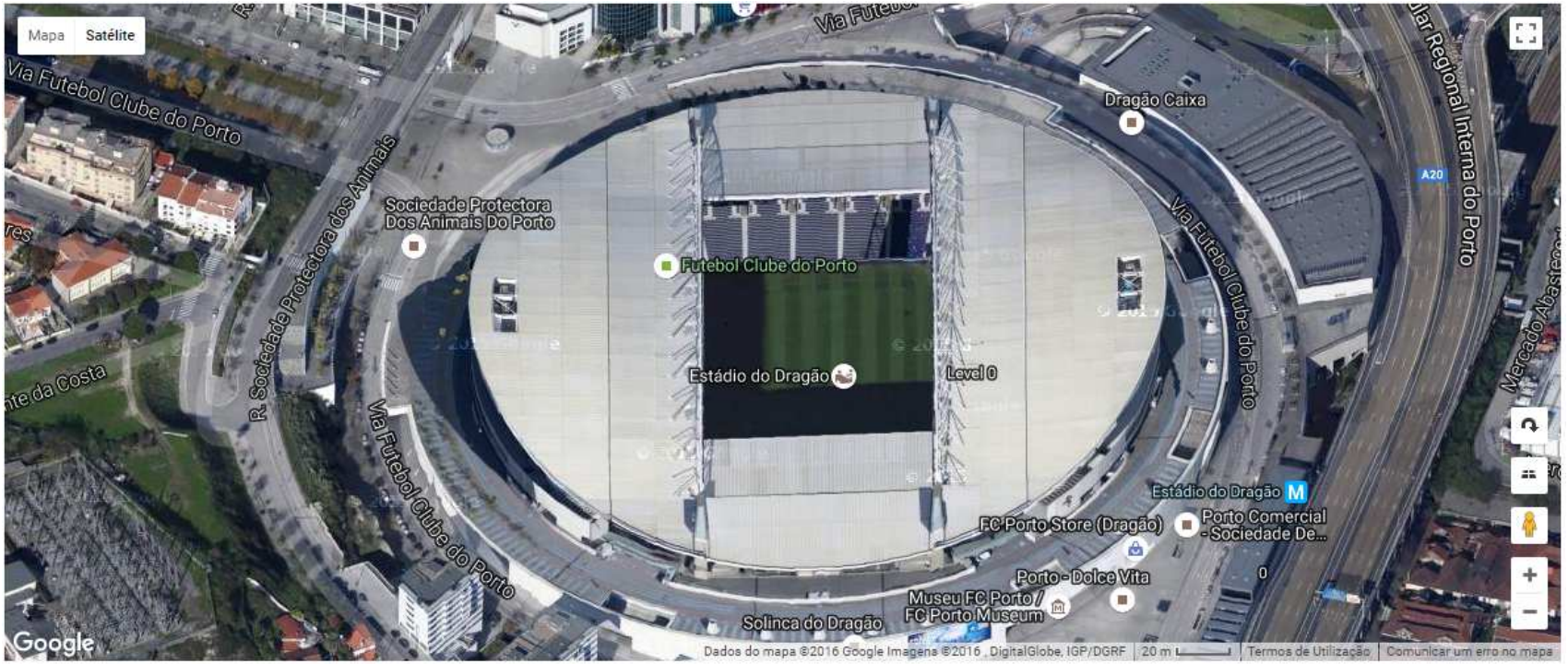
- Controlos comuns:
  - **Zoom** – botões "+/-" para controlar o nível do zoom do mapa
  - **Tipo de mapa** – botões de alternância entre mapas (roadmap e satellite)
  - **Street View** – ícone Pegman arrastável para o mapa para ativar o Street View
- Adicionalmente aos controlos padrão, o GM também tem:
  - **Escala** – exibe uma escala para o mapa
  - **Rotação** – combina opções de rotação e inclinação (tilt) em mapas com perspetiva
  - **FullScreen** – exibe o mapa em ecrã completo



GOOGLE MAPS JS API

3. MAPAS

Controlos



GOOGLE MAPS JS API

3. MAPAS

Controlos

- Desativando os controlos:

```
function initMap() {
 let map = new google.maps.Map(document.getElementById('map'), {
   zoom: 4,
   center: {lat: -33, lng: 151},
   disableDefaultUI: true
 })
}
```

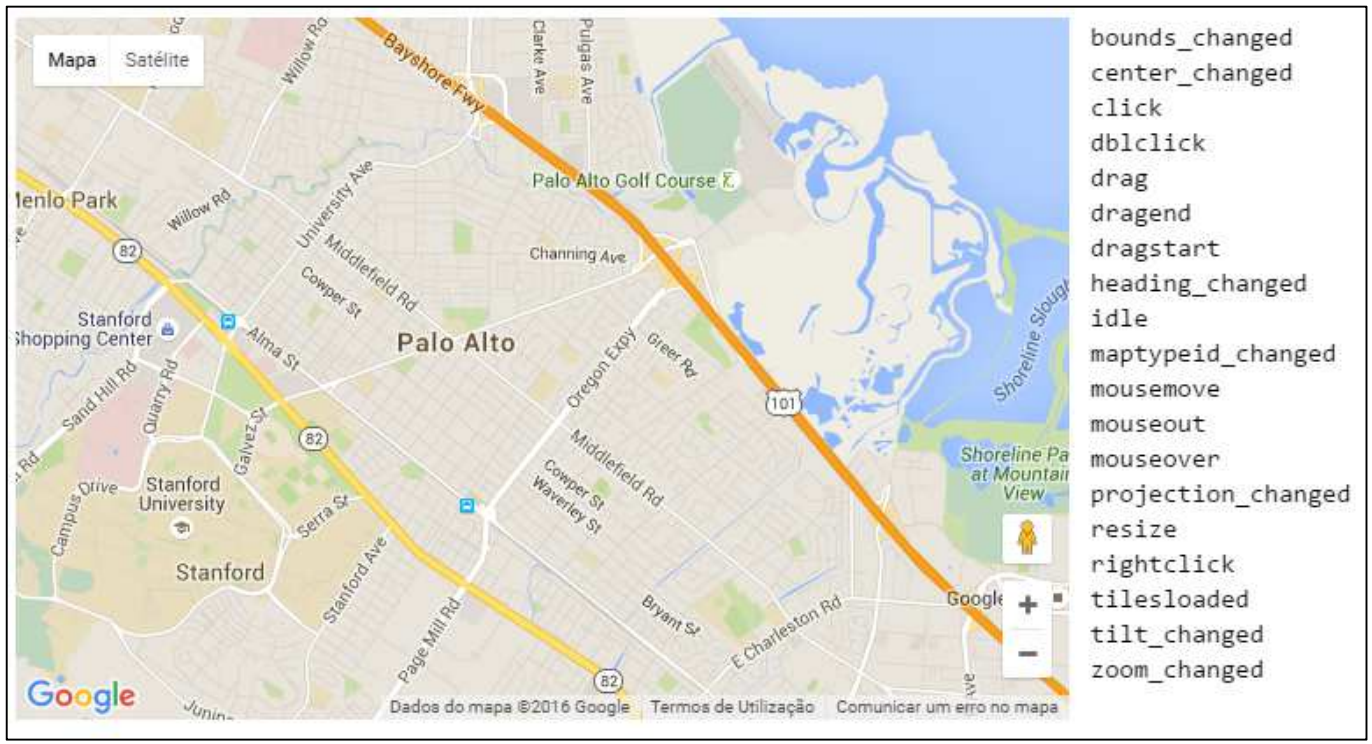
- Ativando todos os controlos:

```
{ …
  zoomControl: true,
  mapTypeControl: true,
  scaleControl: true,
  streetViewControl: true,
  rotateControl: true,
  fullscreenControl: true
}
```

GOOGLE MAPS JS API

3. MAPAS

Eventos




```
bounds_changed
center_changed
click
dblclick
drag
dragend
dragstart
heading_changed
idle
maptypeid_changed
mousemove
mouseout
mouseover
projection_changed
resize
rightclick
tilesloaded
tilt_changed
zoom_changed
```

GOOGLE MAPS JS API

3. MAPAS

Eventos

- Para registar notificações de eventos use o método **addListener()**

```
function initMap() {
  let myLatlng = {lat: -25.363, lng: 131.044}

  let map = new
google.maps.Map(document.getElementById('map'), {
    zoom: 4,
    center: myLatlng
  })

  map.addListener('click', function() {
    map.setZoom(8)
  })
}
```

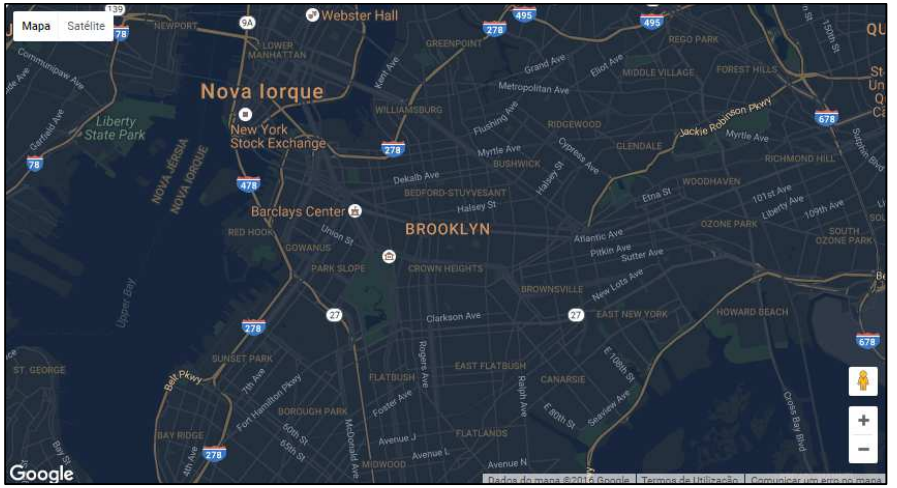
GOOGLE MAPS JS API

3. MAPAS

Estilos

- Pode definer mapas estilizados usando a propriedade **styles**
- Criador de estilos: https://mapstyle.withgoogle.com

```
let map = new google.maps.Map(document.getElementById('map'), {
  zoom: 4,
  center: myLatlng,
  styles: [
        {elementType: 'geometry', stylers: [{color: '#242f3e'}]},
        {elementType: 'labels.text.stroke', stylers: [{color: '#242f3e'}]},
        { featureType: 'road',
          elementType: 'geometry',
          stylers: [{color: '#38414e'}]  }
  ])
}
```



GOOGLE MAPS JS API

3. MAPAS

Geolocalização

- Deteção automática da localização do dispositivo
- Usa GPS ou IP da máquina

```
// Geolocalização HTML5
if (navigator.geolocation) {
      // Se suporta geolocalização tenta obter a sua posição
      navigator.geolocation.getCurrentPosition(showPosition)
} else {
     // Não suporta geolocalização
     console.log("Geolocation is not supported by this browser")
}
```

Se a função tem sucesso: devolve um objeto **Coordinates** para a função especificada no parâmetro (showPosition)

GOOGLE MAPS JS API

3. MAPAS

Geolocalização

- Deteção automática da localização do dispositivo
- Usa GPS ou IP da máquina

Se a função tem sucesso: devolve um objeto **Coordinates** para a função especificada no parâmetro (showPosition)

```
// Geolocalização HTML5
if (navigator.geolocation) {
      // Se suporta geolocalização tenta obter a sua posição
      navigator.geolocation.getCurrentPosition(showPosition)
} else {
     // Não suporta geolocalização
     console.log("Geolocation is not supported by this browser")
}
```

```
function showPosition(position) {
  console.log("LATITUDE:" + position.coords.latitude +
                  "LONGITUDE: " + position.coords.longitude)
}
```

GOOGLE MAPS JS API

3. MAPAS

Geolocalização

- Retorno do método getCurrentPosition()

Property	Returns
coords.latitude	The latitude as a decimal number (always returned)
coords.longitude	The longitude as a decimal number (always returned)
coords.accuracy	The accuracy of position (always returned)
coords.altitude	The altitude in meters above the mean sea level (returned if available)
coords.altitudeAccuracy	The altitude accuracy of position (returned if available)
coords.heading	The heading as degrees clockwise from North (returned if available)
coords.speed	The speed in meters per second (returned if available)
timestamp	The date/time of the response (returned if available)

GOOGLE MAPS JS API

3. MAPAS

Geolocalização

- O objeto **Geolocation** possui também outros métodos:
  - **watchPosition()** – devolve a posição atual do utilizador e continua a devolver a posição atualizada conforme o utilizador se move (como o GPS num carro)
  - **clearWatch()** - Para o método **watchPosition()**

```
if (navigator.geolocation) {
      // Se suporta geolocalização tenta obter a sua posição
      navigator.geolocation.watchPosition(showPosition)
} else {
     // Não suporta geolocalização
     console.log("Geolocation is not supported by this browser")
}
```

```
function showPosition(position) {
  console.log("LATITUDE:" + position.coords.latitude +
                    "LONGITUDE: " + position.coords.longitude)
}
```

GOOGLE MAPS JS API

AGENDA

1. Introdução
2. O meu primeiro mapa!
3. Mapas
4. **Overlays**
5. Layers
6. Serviços
7. Integração com outras APIs

GOOGLE MAPS JS API

4. OVERLAYS

Overlays

- Objetos no mapa ligados a coordenadas (latitude/longitude)
- Tipos de overlays:
  - **Marker** – localizações únicas no mapa
  - **Info Windows** – popup informativa exibida no topo do mapa/marcador
  - **Shapes** – objetos como polilinhas, polígonos, retângulos, circulos, símbolos, etc.

GOOGLE MAPS JS API

4. OVERLAYS

Overlays > Markers

- Identica uma localização no mapa
- Duas formas de colocar um marcador no mapa:
  - Através do construtor **google.maps.Marker**
  - Através do método **setMap** do objeto **Marker**

GOOGLE MAPS JS API

4. OVERLAYS

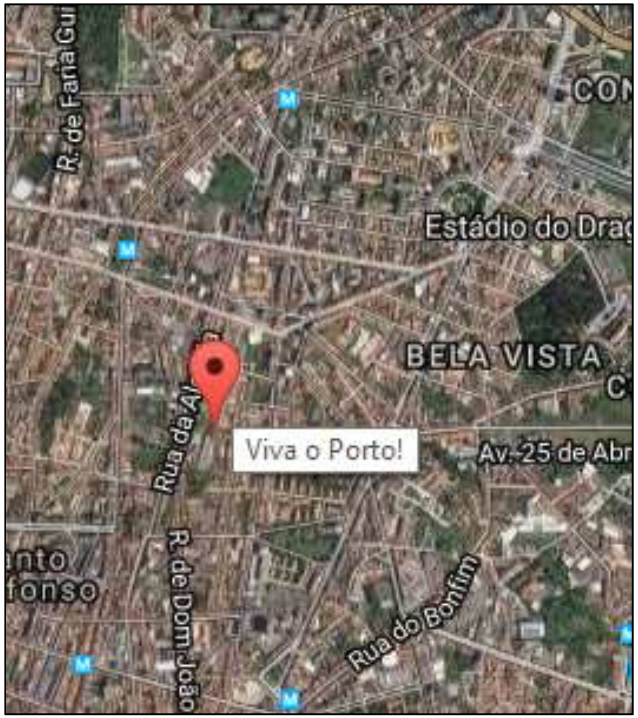
Overlays > Markers

- Através do constructor **google.maps.Marker:**

```
function initMap() {
 let myLatLng = {lat: 41.156111, lng: -8.601111}

 let map = new google.maps.Map(document.getElementById('map'), {
   zoom: 14,
   center: myLatLng
 })

 let marker = new google.maps.Marker({
   position: myLatLng,
   map: map,
   title: 'Viva o Porto!'
 })
}
```



GOOGLE MAPS JS API

4. OVERLAYS

Overlays > Markers

- Através do método **setType** do objeto **Marker:**

```
let myLatlng = new google.maps.LatLng(-25.363882,131.044922)
let mapOptions = {
 zoom: 4,
 center: myLatlng
}
let map = new google.maps.Map(document.getElementById("map"), mapOptions)

let marker = new google.maps.Marker({
   position: myLatlng,
   title:"Hello World!"
})

marker.setMap(map)
```

GOOGLE MAPS JS API

4. OVERLAYS

Overlays > Markers

- Remover um marcador:

```
…
marker.setMap(null)
…
```

GOOGLE MAPS JS API

4. OVERLAYS

Overlays > Markers

- Animar um marcador:
  - **DROP** – o marcador cai do topo para a localização final, na primeira vez que é carregado
  - **BOUNCE** – o marcador salta no lugar e continua até a sua propriedade de animação for defina a nulo

```
function initMap() {
 let map = new google.maps.Map(document.getElementById('map'), {
   zoom: 13,
   center: {lat: 59.325, lng: 18.070}
 })

 marker = new google.maps.Marker({
   map: map,
   animation: google.maps.Animation.BOUNCE,
   position: {lat: 59.327, lng: 18.067}
 })
}
```

GOOGLE MAPS JS API

4. OVERLAYS

Overlays > Markers

- Personalizar um marcador com uma imagem:

```
function initMap() {
  var map = new google.maps.Map(document.getElementById('map'), {
    zoom: 4,
    center: {lat: -33, lng: 151}
  })

  let image = 'https://developers.google.com/…/beachflag.png'
  let beachMarker = new google.maps.Marker({
    position: {lat: -33.890, lng: 151.274},
    map: map,
    icon: image
  })
}
```

GOOGLE MAPS JS API

4. OVERLAYS

Overlays > Markers

- Personalizar um marcador com um label:

```
let marker = new google.maps.Marker({
  position: location,
  label: "P",
  map: map
})
```

- Tornar o marcador arrastável:

```
let marker = new google.maps.Marker({
  position: location,
  draggable: true,
  map: map
})
```

GOOGLE MAPS JS API

4. OVERLAYS

Overlays > Info Windows

- Exibe conteúdo (texto ou imagens) numa janela sob o mapa, numa determinada localização ou associada a um marcador
- Construtor **InfoWindow** usa o objeto **InfoWindowOptions**, que especifica:
  - **content** – texto (simples ou HTML)
  - **position** – objeto **LatLng** onde a janela será ancorada
  - **maxWidth** – largura máxima da janela em píxeis

```
let infowindow = new google.maps.InfoWindow({
  content: "OLA MUNDO!"
})
```

Posso fechar a janela programaticamente?

A **InfoWindow** mantém-se aberta até que o utilizador clique na cruz. Via código pode usar o método **close()**.

GOOGLE MAPS JS API

4. OVERLAYS

Overlays > Info Windows

- Para abrir uma janela use o método **open()**:

```
function initMap() {

 let map = …
 let infowindow = new google.maps.InfoWindow({
   content: 'Estou em Londres!'
 })

 let marker = new google.maps.Marker({
   position: londres,
   map: map,
 })
 marker.addListener('click', function() {
   infowindow.open(map, marker);
 })
}
```

GOOGLE MAPS JS API

4. OVERLAYS

Overlays > Info Windows

```
function initMap() {
 let esmad = {lat: 41.366949, lng: -8.738722}
 let map = new google.maps.Map(document.getElementById('map'), {
   zoom: 17,
   center: esmad
 })
 let contentString = '<div id="content"><div id="siteNotice"></div>'+
     '<h1 id="firstHeading" class="firstHeading">ESMAD</h1>'+
     '<div id="bodyContent">'<p>A <b>ESMAD</b> é uma das escolas do P.PORTO! </p>' +
     '<p><img src="https://www.esmad.ipp.pt/logo-ipp.png"/></p></div></div>'

 let infowindow = new google.maps.InfoWindow({ content: contentString })
 let marker = new google.maps.Marker({
   position: esmad,
   map: map
 })
 marker.addListener('click', function() {
   infowindow.open(map, marker);
 })
}
```

GOOGLE MAPS JS API

4. OVERLAYS

Overlays > Shapes

- Objetos do mapa associados a coordenadas (lat/long)
- Shapes disponíveis:
  - Polilinhas
  - Polígonos
  - Círculos & Retângulos
- Pode-se também configurar as formas podendo os utilizadores editá-las e arrastá-las
- E incluir imagens através de objetos GroundOverlay

GOOGLE MAPS JS API

4. OVERLAYS

Overlays > Shapes > Polyline

- A classe **Polyline** define um overlay linear de segmentos de linha conectados no mapa
- Um objeto **Polyline** consiste num array de objetos **LatLng**

```
let flightPlanCoordinates = [
  {lat: 37.772, lng: -122.214},
  {lat: 21.291, lng: -157.821},
  {lat: -18.142, lng: 178.431},
  {lat: -27.467, lng: 153.027}
]
let flightPath = new google.maps.Polyline({
  path: flightPlanCoordinates,
  strokeColor: '#FF0000',
  strokeOpacity: 1.0,
  strokeWeight: 2
 })
flightPath.setMap(map)
```

GOOGLE MAPS JS API

4. OVERLAYS

Overlays > Shapes > Polygons

- A classe **Polygon** representa uma área fechada definida por uma série de coordenadas

```
// Construção do polígono
let bermudaTriangle = new google.maps.Polygon({
  paths: triangleCoords,
  strokeColor: '#FF0000',
  strokeOpacity: 0.8,
  strokeWeight: 2,
  fillColor: '#FF0000',
  fillOpacity: 0.35
})
bermudaTriangle.setMap(map)
```

GOOGLE MAPS JS API

4. OVERLAYS

Overlays > Shapes > Polygons

- Para criar uma área vazia dentro de um polígono, é necessário criar duas séries de coordenadas:

```
// Define as coordenadas LatLng dos dois polígonos (outer e inner)
let outerCoords = [ {lat: 25.774, lng: -80.190}, {lat: 18.466, lng: -66.118}, {lat: 32.321, lng: -64.757} ]
let innerCoords = [ {lat: 28.745, lng: -70.579}, {lat: 29.570, lng: -67.514}, {lat: 27.339, lng: -66.668} ]

// Construção do polígono com os 2 arrays de coordenadas
let bermudaTriangle = new google.maps.Polygon({
  paths: [outerCoords, innerCoords],
  strokeColor: '#FFC107',
  strokeOpacity: 0.8,
  strokeWeight: 2,
  fillColor: '#FFC107',
  fillOpacity: 0.35
})
bermudaTriangle.setMap(map)
```

GOOGLE MAPS JS API

4. OVERLAYS

Overlays > Shapes > Circles

- A classe **Circle** é similar à classe **Polygon**
- Ao contrário do polígono, não define uma série de coordenadas. Em vez disso, um círculo tem duas propriedades adicionais:
  - **center** – coordenadas **LatLng** para o centro do círculo
  - **radius** – raio do círculo, em metros

```
let esmadCircle = new google.maps.Circle({
    strokeColor: '#FF0000',
    strokeOpacity: 0.8,
    strokeWeight: 2,
    fillColor: '#FF0000',
    fillOpacity: 0.35,
    map: map,
    center: esmad,
    radius: 500
  })
```

GOOGLE MAPS JS API

4. OVERLAYS

Overlays > Shapes > User-editable and draggable shapes

- Qualquer shape (polylines, polygons, circles, e rectangles) pode ser **editável** e **arrastável**

```
let bounds = {
 north: 44.599,
 south: 44.490,
 east: -78.443,
 west: -78.649
}

// Define um rectangle e define a sua propriedade editable a true
let rectangle = new google.maps.Rectangle({
 bounds: bounds,
 draggable: true,
 editable: true
})
```

GOOGLE MAPS JS API

4. OVERLAYS

Overlays > Shapes > Symbols

- A API Maps disponibiliza alguns símbolos predefinidos que pode ser adicionados aos marcadores (icones vectoriais) ou polilinhas através da classe **SymbolPath**

```
let marker = new google.maps.Marker({
  icon: {
    path: google.maps.SymbolPath.CIRCLE,
    scale: 10
  },
  map: map
})
```

Name	Description	Example
google.maps.SymbolPath.CIRCLE	A circle.	○
google.maps.SymbolPath.BACKWARD_CLOSED_ARROW	A backward-pointing arrow that is closed on all sides.	▼
google.maps.SymbolPath.FORWARD_CLOSED_ARROW	A forward-pointing arrow that is closed on all sides.	▲
google.maps.SymbolPath.BACKWARD_OPEN_ARROW	A backward-pointing arrow that is open on one side.	V
google.maps.SymbolPath.FORWARD_OPEN_ARROW	A forward-pointing arrow that is open on one side.	^

GOOGLE MAPS JS API

4. OVERLAYS

Overlays > Shapes > GroundOverlay

- Para colocar uma imagem num mapa pode usar um objeto **GroundOverlay**

```
let imageBounds = {
  north: 40.773941,
  south: 40.712216,
  east: -74.12544,
  west: -74.22655
}
myOverlay = new google.maps.GroundOverlay('https://….jpg', imageBounds)
myOverlay.setMap(map)
```

GOOGLE MAPS JS API

AGENDA

1. Introdução
2. O meu primeiro mapa!
3. Mapas
4. Overlays
5. **Layers**
6. Serviços
7. Integração com outras APIs

GOOGLE MAPS JS API

5. LAYERS

Layers

- Objetos no mapa que consistem em um ou mais items separados, mas manipulados como uma unidade única
- A API Maps suporta os seguintes tipos de layers:
  - **Traffic layer** – exibe no mapa as condições do tráfego
  - **Transit layer** – exibe a rede pública de transportes numa cidade do mapa
  - **Bicycling layer** – renderiza um layer de caminhos de bicicleta
  - **Heatmap layer** – renderiza dados geográficos usando uma visualização Heatmap
  - **Fusion Tables layer** – renderiza dados contidos em Google Fusion Tables

GOOGLE MAPS JS API

5. LAYERS

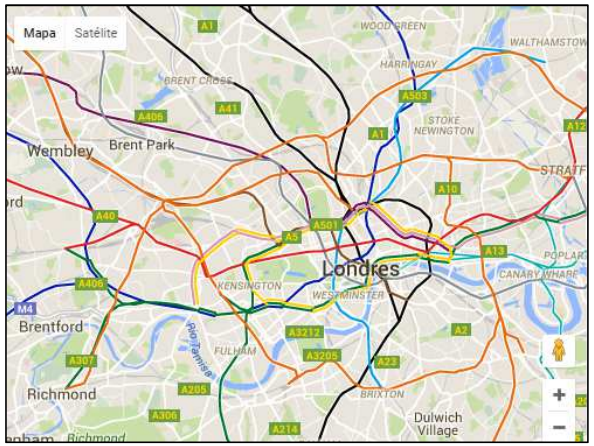
Layers > Transit

- Exibe a rede pública de transportes numa cidade do mapa

```
let mapOptions = {
 zoom: 13,
 center: new google.maps.LatLng(51.5,-0.11)
}

let map = new google.maps.Map(document.getElementById("map"),
mapOptions)

let transitLayer = new google.maps.TransitLayer()
transitLayer.setMap(map)
```



GOOGLE MAPS JS API

5. LAYERS

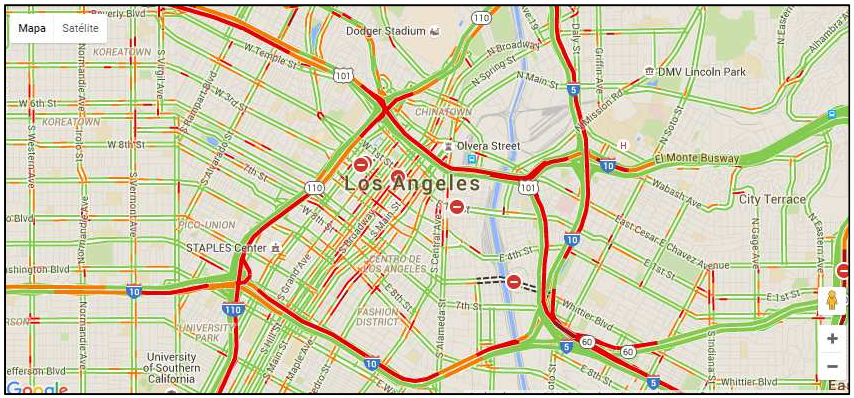
Layers > Outros tipos de layers

- Outros tipos de layers

Transit layer

Bicycling Layer

Traffic layer



GOOGLE MAPS JS API

5. LAYERS

Layers > Fusion Tables Layer

- Renderiza dados contidos no **Google Fusion Tables**
- Fusion Tables são bases de dados relacionais
- Existem tabelas com informação relevante:
  - Espécies de borboletas nos EUA
  - Países e os seus números de habitantes
  - …

GOOGLE MAPS JS API

5. LAYERS

Layers > Fusion Tables Layer

- Link: https://research.google.com/tables?corpus=fusion

GOOGLE MAPS JS API

5. LAYERS

Layers > Fusion Tables Layer

- Link: https://research.google.com/tables?corpus=fusion

```
let layer = new google.maps.FusionTablesLayer({
  query: {
    select: ‘geometry’,
    from: '12e2VhiXyMzHWDl6aponObHH_gvlMDoac9RTrcJ0‘,
    where: “name = ‘Portugal’"
  }
})
layer.setMap(map)
```

GOOGLE MAPS JS API

AGENDA

1. Introdução
2. O meu primeiro mapa!
3. Mapas
4. Overlays
5. Layers
6. **Serviços**
7. Integração com outras APIs

GOOGLE MAPS JS API

6. SERVIÇOS

Serviços

- Conjunto de serviços que podem ser integrados em mapas:
  - Geocoding
  - Direcões
  - Street View
  - Distâncias
  - Elevações
  - …
  - Bibliotecas (Places, AdSense, Panoramio,…)

GOOGLE MAPS JS API

6. SERVIÇOS

Serviços > GeoCoding

- Processo que converte um endereço em texto (ex: Rua da Picua nº 90) em coordenadas **LatLng,** que depois podem ser usadas para colocar marcadores ou posicionar em mapas
- A conversão do par de coordenadas em endereços de texto (human-readable address) chama-se de **reverse geocoding**

GOOGLE MAPS JS API

6. SERVIÇOS

Serviços > GeoCoding

```
// 1. Cria os objetos Map e Geocoder
let map = new
google.maps.Map(document.getElementById('map'), {
        zoom: 8,
        center: {lat: -34.397, lng: 150.644}
})

let geocoder = new google.maps.Geocoder()

geocodeAddress(geocoder, map)
```

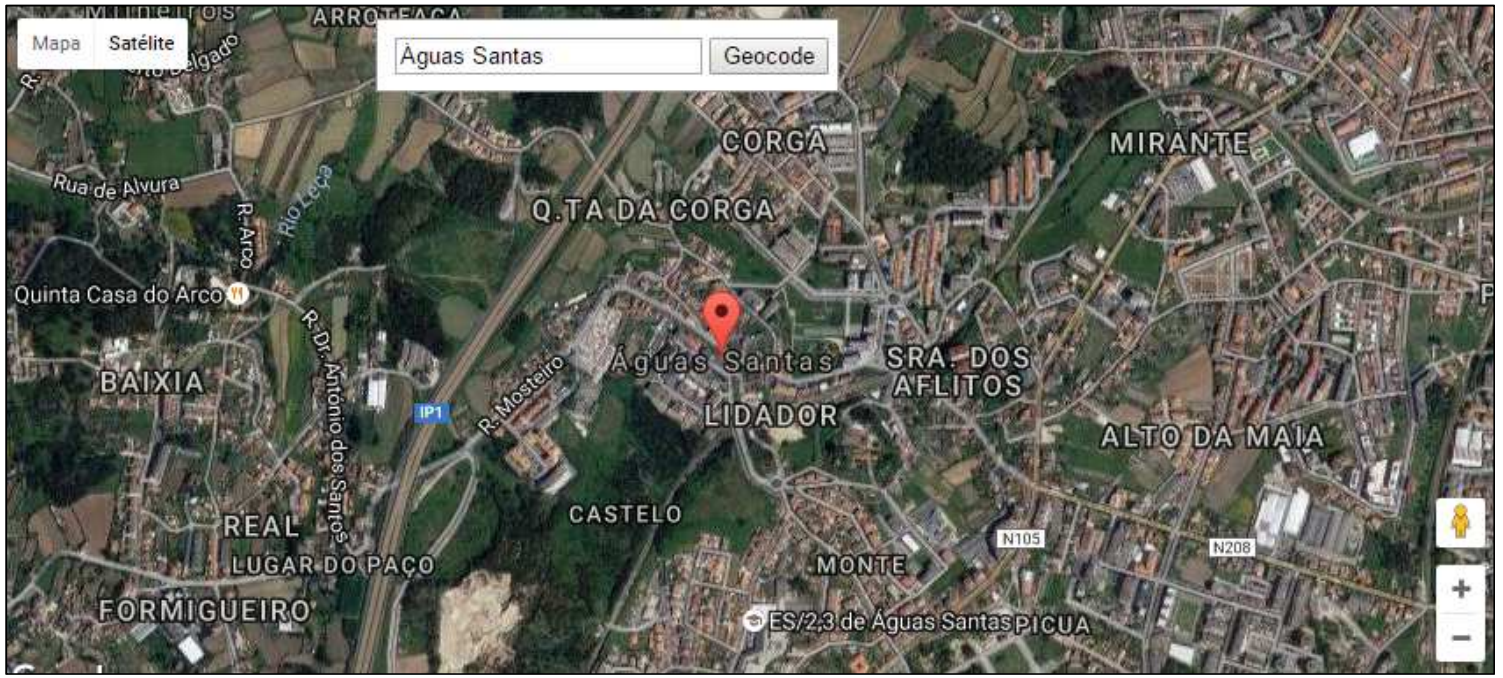
```
// 2. Define texto a usar no geocoding
<p id="address">Águas Santas</p>
```

```
// 3. Faz o pedido ao serviço Geocoding e trata da resposta
function geocodeAddress(geocoder, resultsMap) {
      let address = document.getElementById('address').innerHTML
      geocoder.geocode({'address': address}, function(results,
status) {
        if (status === 'OK') {
          resultsMap.setCenter(results[0].geometry.location)
          let marker = new google.maps.Marker({
            map: resultsMap,
            position: results[0].geometry.location
          })
        } else { alert('Geocode falhou devido a: ' + status) }
      })
    }
  </script>
```

GOOGLE MAPS JS API

6. SERVIÇOS

Serviços > GeoCoding



GOOGLE MAPS JS API

6. SERVIÇOS

Serviços > Direções


```
function initMap() {
 // Define objetos LatLng com origem e destino
 let chicago = {lat: 41.85, lng: -87.65}
 let indianapolis = {lat: 39.79, lng: -86.14}

 // Cria objeto Map
 let map = new google.maps.Map(document.getElementById('map'), {
   center: chicago,
   zoom: 7
 })

 // Associa as direções ao mapa
 let directionsDisplay = new google.maps.DirectionsRenderer({
   map: map
 })

 // Define um objeto Request com origem, destino e modo de viagem
 let request = {
   destination: indianapolis,
   origin: chicago,
   travelMode: google.maps.TravelMode.DRIVING
 }

 // Passa o objeto Request ao serviço Directions
 let directionsService = new google.maps.DirectionsService()
 directionsService.route(request, function (response, status) {
   if (status == google.maps.DirectionsStatus.OK) {
     // Exibe a rota no mapa
     directionsDisplay.setDirections(response);
   }
 })
}
```

GOOGLE MAPS JS API

6. SERVIÇOS

Serviços > StreetView

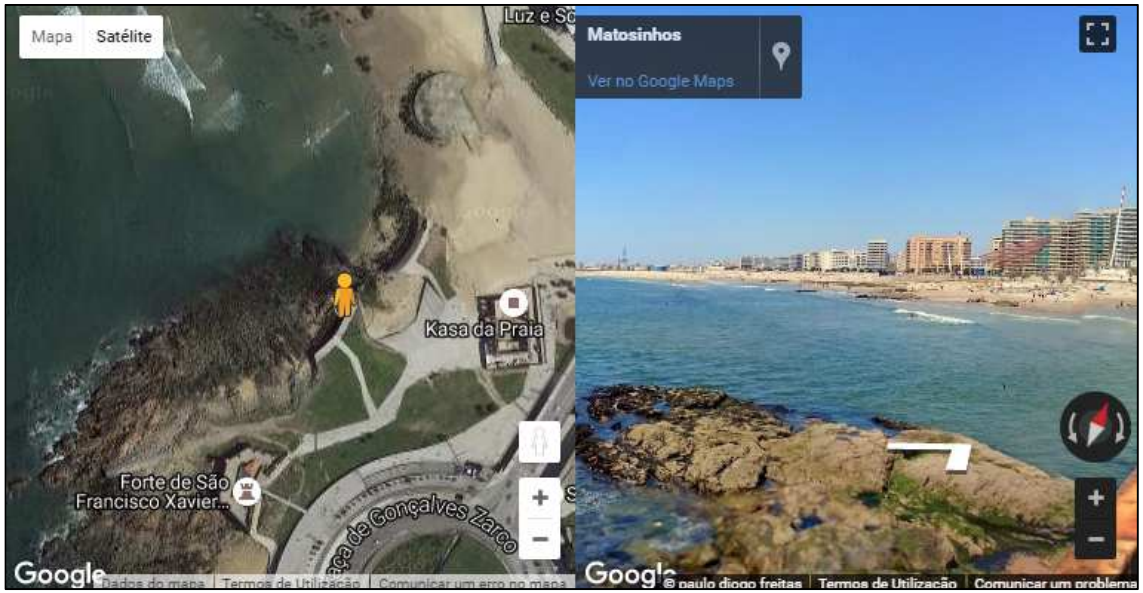
```
let panorama
function initialize() {
  panorama = new google.maps.StreetViewPanorama(
      document.getElementById('map'),
        {
          position: {lat: 37.869260, lng: -122.254811},
          pov: {heading: 165, pitch: 0},
          zoom: 1
        })
   }
```

GOOGLE MAPS JS API

6. SERVIÇOS

Serviços > StreetView (side-by-side)

```
<div id="map"></div>
<div id="pano"></div>
<script>
  function initialize() {
    let matosinhos = {lat: 42.345573, lng: -71.098326}
    let map = new google.maps.Map(document.getElementById('map'), {
      center: matosinhos,
      zoom: 14
    })
    let panorama = new google.maps.StreetViewPanorama(
        document.getElementById('pano'), {
          position: matosinhos,
          pov: {
            heading: 34,
            pitch: 10
          }
        })
    map.setStreetView(panorama)
  }
```



GOOGLE MAPS JS API

AGENDA

1. Introdução
2. O meu primeiro mapa!
3. Mapas
4. Overlays
5. Layers
6. Serviços
7. **Integração com outras APIs**

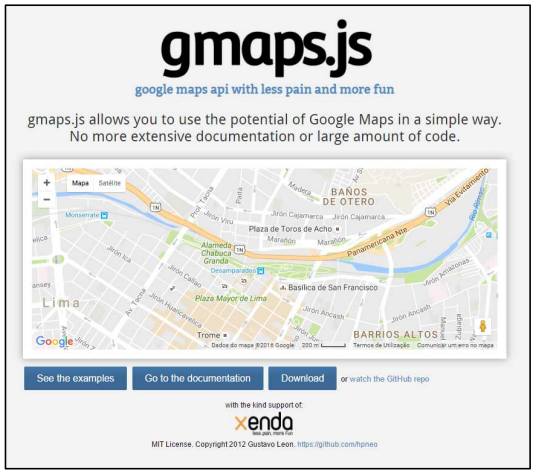
GOOGLE MAPS JS API

7. INTEGRAÇÃO COM OUTRAS API

API Places

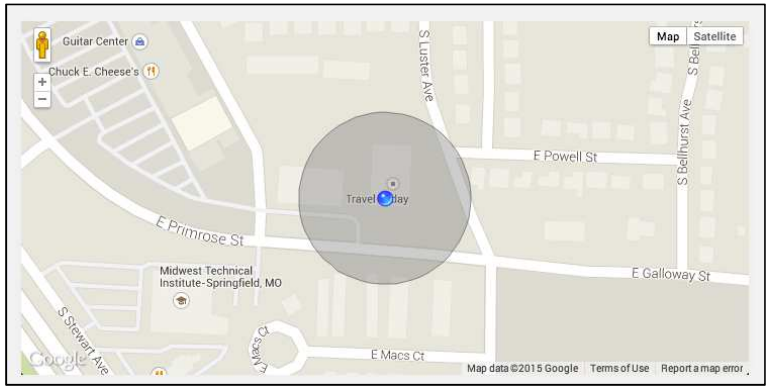
GOOGLE MAPS JS API

BIBLIOTECAS



Bibliotecas

- gmaps.js
  https://hpneo.github.io/gmaps/
- Geolocation marker
  https://chadkillingsworth.github.io/geolocation-marker/
- js-store-locator
  https://github.com/googlemaps/js-store-locator



GOOGLE MAPS JS API
REFERÊNCIAS

Referências

- Google Maps JS API documentation:
  https://developers.google.com/maps/documentation/javascript/

GOOGLE MAPS JS API

Q/A

ricardoqueiros@esmad.ipp.pt