Programação Móvel

Recursos Nativos

Prof. Dr. Diego R. Antunes
drantunes@utfpr.edu.br

Departamento de Computação
Universidade Tecnológica Federal do Paraná

Anteriormente

Recursos Nativos

Plugins para Compartilhamento;

Plugin para Envio de SMS;

Plugin para Ligação;

Splash

Splash

Possibilita controlar a exibição da ”spash screen” durante o loading do aplicativo.

ionicframework.com/docs/v2/native/splashscreen/

Splash

Possibilita controlar a exibição da ”spash screen” durante o loading do aplicativo.

ionicframework.com/docs/v2/native/splashscreen/

Resources

Para customizar sua própria imagem para o splash, bem como os ícones do aplicativo, alterar a imagem modelo disponível na pasta resources do seu projeto (splash.png).

Resources

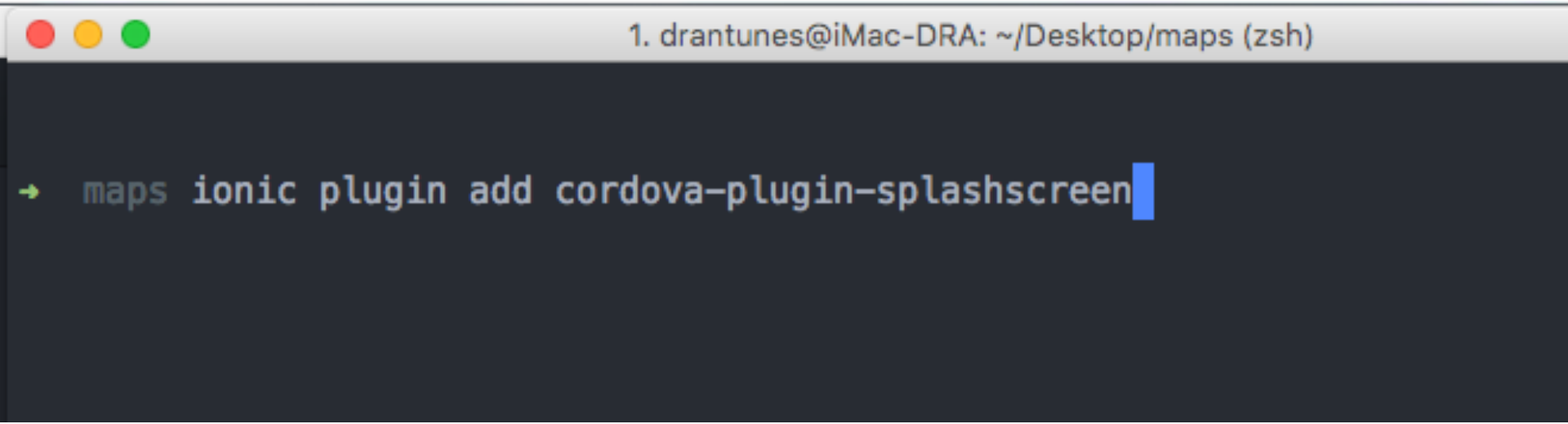
Esta é uma imagem de alta resolução, que serve de modelo para o Ionic gerar todos os ícones e splash para diferentes resoluções.

Resources

Para gerar os resources, basta executar o comando ionic resources. O Ionic então irá gerar todos os arquivos automaticamente para você.

Spash

Primeiro, você deve instalar o plugin no seu projeto:



TS home.ts | TS app.ts | home.html

```
1  import { Component } from '@angular/core';
2  import { ionicBootstrap, Platform } from 'ionic-angular';
3  import { StatusBar, Splashscreen } from 'ionic-native';
4
5  import { HomePage } from './pages/home/home';
6
7
8  @Component({
9      template: '<ion-nav [root]="rootPage"></ion-nav>'
10 })
11 export class MyApp {
12     rootPage: any = HomePage;
13
14     constructor(public platform: Platform) {
15
16
17
18
19
20 }
21 }
```

Para utilizar o plugin você deve importar Splashscreen do Ionic Native tanto em app.ts quanto em home.ts

TS home.ts | TS app.ts | <> home.html

Então podemos mostrar o splash no construtor do app.ts.

```
@Component({
    template: '<ion-nav [root]="rootPage"></ion-nav>'
})
export class MyApp {
    rootPage: any = HomePage;

    constructor(public platform: Platform) {
        // Mostra a Splash Screen
        Splashscreen.show();
        platform.ready().then(() => {
            StatusBar.styleDefault();
        });
    }
}
```

Ao entrar na página inicial, home.ts, podemos esconder a splash no evento ionViewDidEnter.

```
1
2
3   import { Splashscreen } from 'ionic-native';
4
5   @Component({
6       templateUrl: 'build/pages/home/home.html'
7   })
8   export class HomePage {
9       ionViewDidEnter() {
10          Splashscreen.hide();
11      }
12  }
13
14
15
16
```

Geolocation

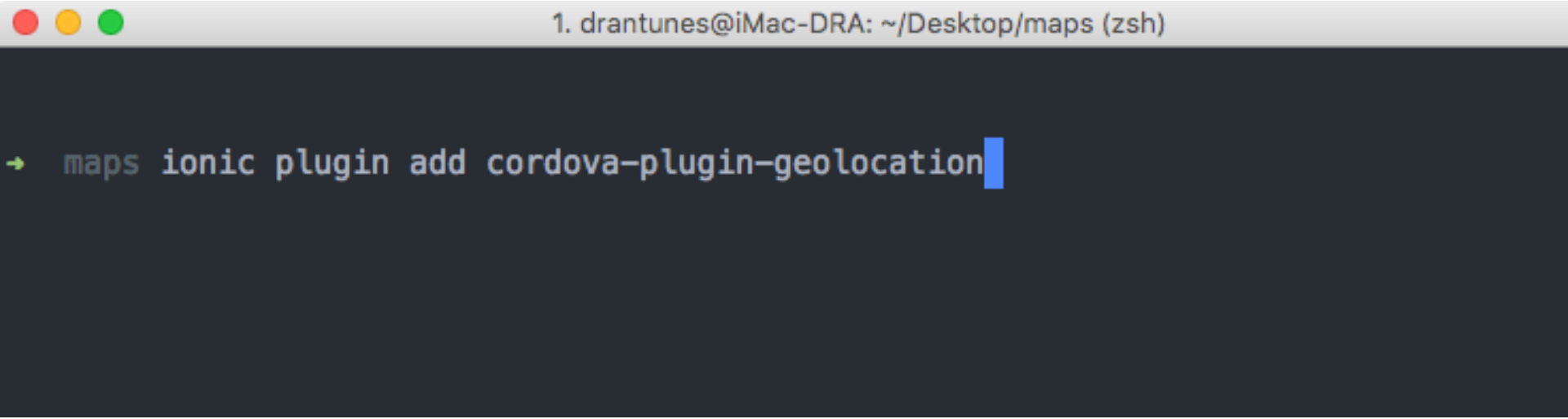
Geolocation

Possibilita recuperar a geolocalização do dispositivo, retornando a longitude e latitude.

ionicframework.com/docs/v2/native/geolocation

Geolocation

Primeiro, você deve instalar o plugin no seu projeto:



TS home.ts <> home.html

```ts
import { Component } from '@angular/core';
import { AlertController } from 'ionic-angular';
import { Geolocation } from 'ionic-native';

@Component({
    templateUrl: 'build/pages/home/home.html'
})
export class HomePage {
    public latitude;
    public longitude;

    constructor() {
        Geolocation.getCurrentPosition().then((position) => {
```

Para utilizar o plugin você deve importar Geolocation do Ionic Native na página que deseja obter a localização.

TS home.ts <> home.html

```
1  import { Component } from '@angular/core';
2  import { AlertController } from 'ionic-angular';
3  import { Geolocation } from 'ionic-native';
4
5  @Component({
6      templateUrl: 'build/pages/home/home.html'
7  })
8  export class HomePage {
9      public latitude;
10     public longitude;
11
12     constructor() {
13         Geolocation.getCurrentPosition().then((position) => {
14
15
16
17
18
19
20 }
```

Então declaramos duas variáveis para armazenar a latitude e a longitude.

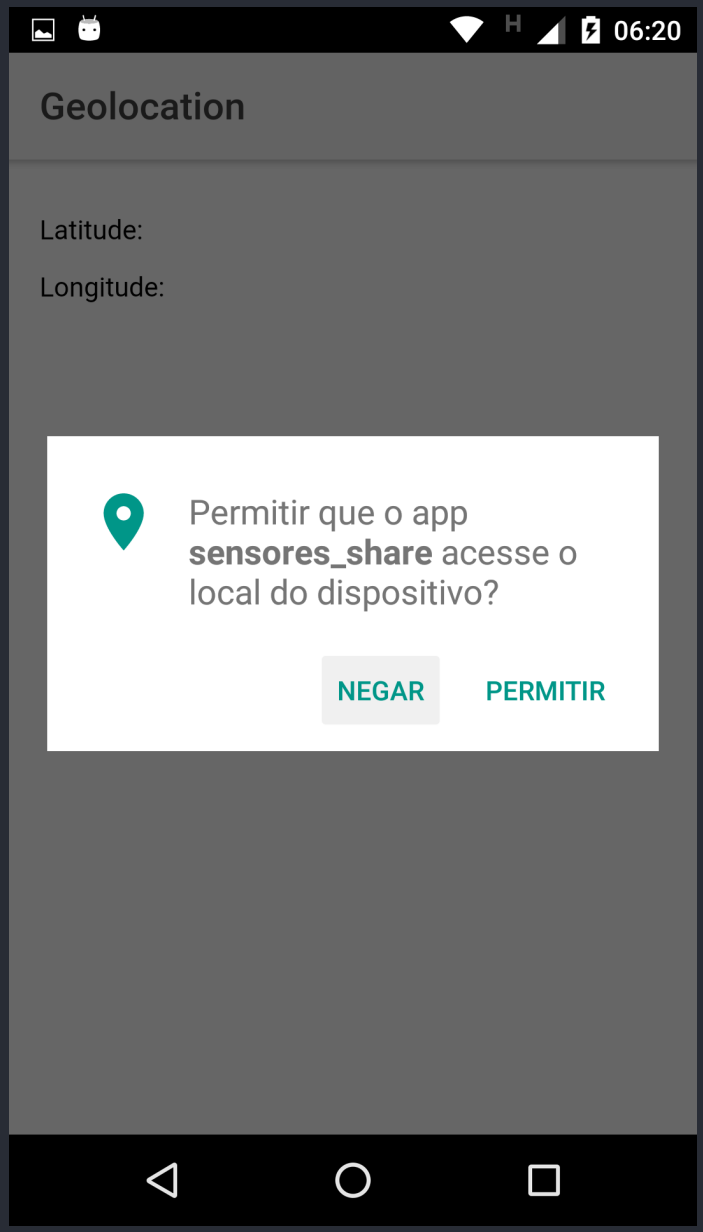
Então podemos chamar o método

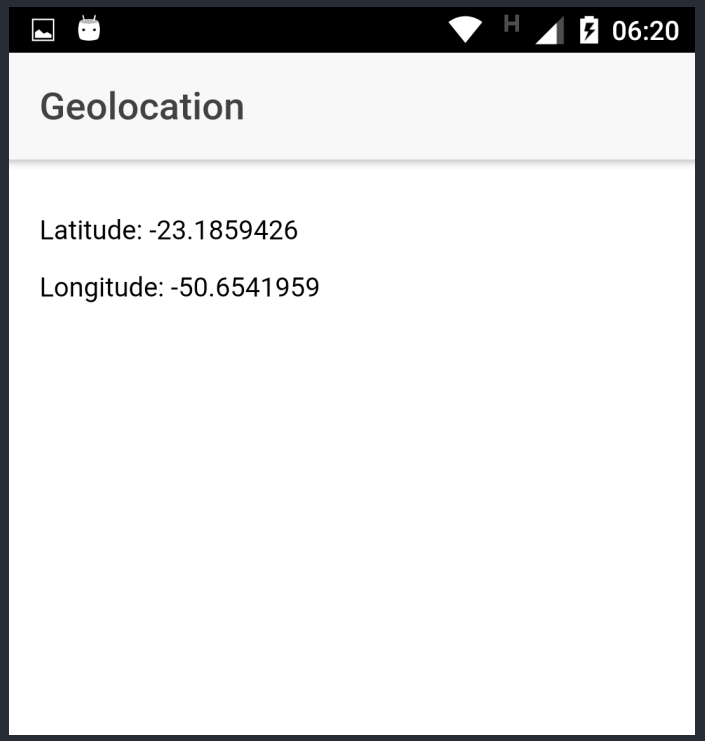
Geolocation.getCurrentPosition( )

```
templateUrl: 'build/pages/home/home.html'
})
export class HomePage {
    public latitude;
    public longitude;

    constructor() {
        Geolocation.getCurrentPosition().then((position) => {
            this.latitude = position.coords.latitude;
            this.longitude = position.coords.longitude;
        }).catch((error) => {
            console.log('Erro: ', error);
        });
    }
}
```

O método retorna a position do dispositivo, que fornece os dados de latitude e longitude.

```
templateUrl: 'build/pages/home/home.html'
})
export class HomePage {
    public latitude;
    public longitude;

    constructor() {
        Geolocation.getCurrentPosition().then((position) => {
            this.latitude = position.coords.latitude;
            this.longitude = position.coords.longitude;
        }).catch((error) => {
            console.log('Erro: ', error);
        });
    }
}
```






```
8
9   <ion-content padding>
10
11    <p>Latitude: {{ latitude }}</p>
12    <p>Longitude: {{ longitude }}</p>
13
14  </ion-content>
15
```

Google Maps

Google Maps

Possibilita exibir e manipular mapas em aplicativos Ionic, utilizando funções como marcadores, distância, rotas, entre outras.

Google Maps

Existem duas maneiras de utilizar o Google Maps:

- SDK Nativo (Android ou iOS);
- SDK Javascript

SDK Nativo

Possibilita interagir com o SDK Nativo, tanto para Android quanto para iOS. Isto possibilita maior desempenho e suavidade no carregamento do mapa.

SDK Nativo (Desvantagens)

Pode apresentar bugs, pois depende do plugin do cordova. Assim, cria uma camada extra que pode inserir alguns erros.

SDK Nativo (Desvantagens)

Não apresenta todas as funcionalidades disponíveis facilmente no Javascript SDK, como plugins de cluster, distância, rota, street view, windows customizadas, entre outros.

SDK Nativo (Desvantagens)

Funciona somente no dispositivo físico ou emulador, enquanto o Javascript SDK funciona no navegador.

SDK Nativo (Plugin)

ionicframework.com/docs/v2/native/google-maps

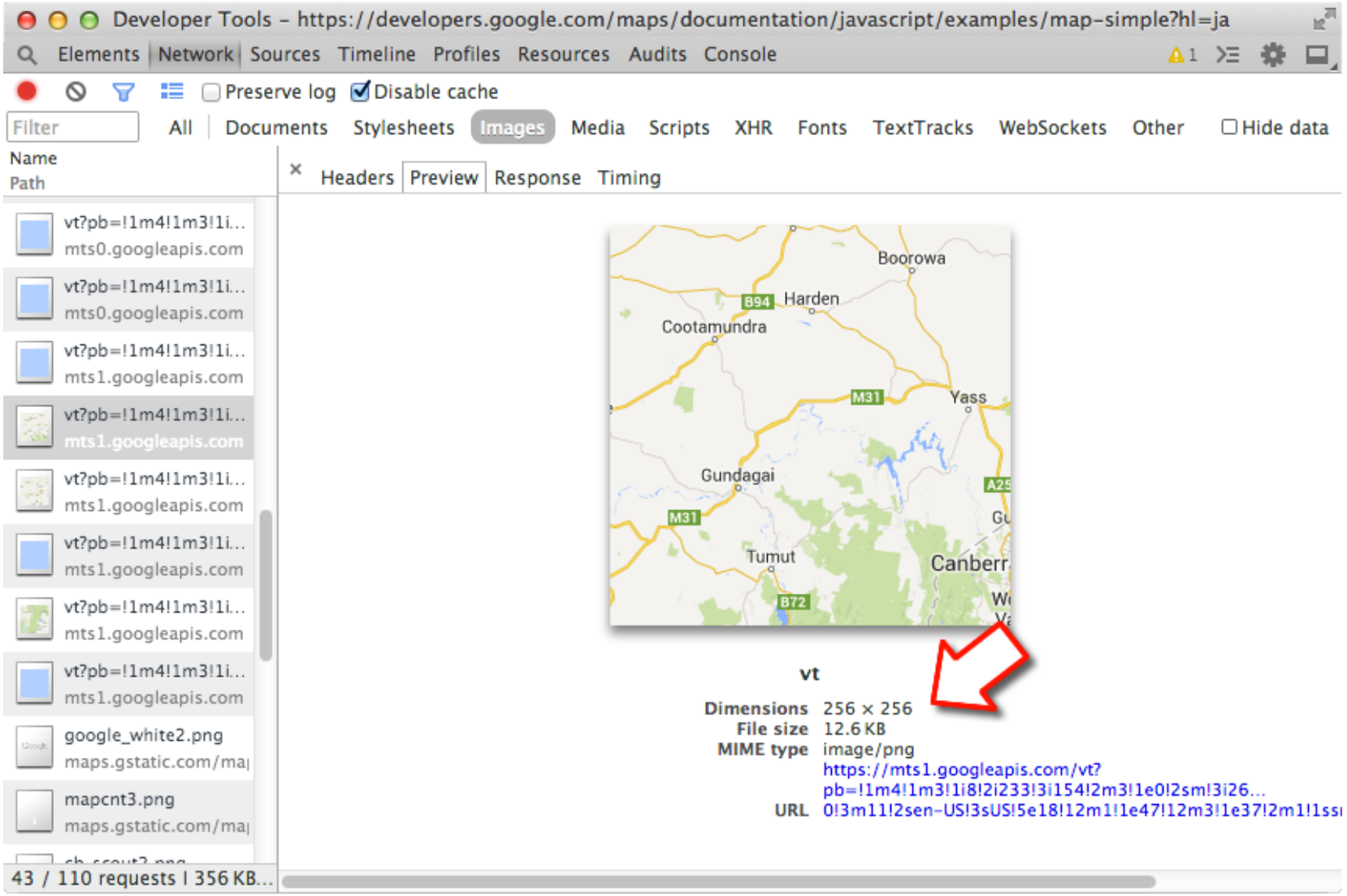
Nesta página você obtém as informações da API do plugin nativo, bem como as instruções de instalação.

A API é muito semelhante ao Javascript SDK.

Diferenças

Nativo: Carrega os mapas através de vetores

Javascript: Carrega porções de imagens em PNG.


Developer Tools - https://developers.google.com/maps/documentation/javascript/examples/map-simple?hl=ja
Elements
Network
Sources
Timeline
Profiles
Resources
Audits
Console
Preserve log
Disable cache
Filter
All
Documents
Stylesheets
Images
Media
Scripts
XHR
Fonts
TextTracks
WebSockets
Other
Hide data
Name
Path
vt?pb=!1m4!1m3!1i...
mts0.googleapis.com
vt?pb=!1m4!1m3!1i...
mts0.googleapis.com
vt?pb=!1m4!1m3!1i...
mts1.googleapis.com
vt?pb=!1m4!1m3!1i...
mts1.googleapis.com
vt?pb=!1m4!1m3!1i...
mts1.googleapis.com
vt?pb=!1m4!1m3!1i...
mts1.googleapis.com
vt?pb=!1m4!1m3!1i...
mts1.googleapis.com
vt?pb=!1m4!1m3!1i...
mts1.googleapis.com
google_white2.png
maps.gstatic.com/ma
mapcnt3.png
maps.gstatic.com/ma
43 / 110 requests | 356 KB...
Headers
Preview
Response
Timing
Boorowa
B94
Harden
Cootamundra
M31
Yass
Gundagai
M31
Tumut
B72
Canberr
vt
Dimensions 256 × 256
File size 12.6 KB
MIME type image/png
URL https://mts1.googleapis.com/vt?pb=!1m4!1m3!1i8!2i233!3i154!2m3!1e0!2sm!3i26...0!3m11!2sen-US!3sUS!5e18!12m1!1e47!12m3!1e37!2m1!1ss

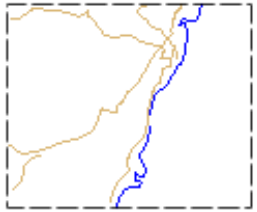
Google Maps JavaScript API

Image data: 12.6KB * n

Google

loading time: 2 - 3 seconds

Google Maps Mobile SDKs



Vector data: < 12.6KB

Google

loading time: 1 second

Nativo:


12:44
Step7: Draw a polygon
Draw a polygon. Click it!
var encodedPolygon = "uyb~FupbzYqAoK|H{@~FaFjCx
map.addPolygon({
'points': plugin.google.maps.geometry.encoding
'strokeColor' : '#AA00FF',
'strokeWidth': 5,
'fillColor' : '#880000'
}, onPolygonAdded);
function onPolygonAdded(polygon) {
map.animateCamera({
target: polygon.getPoints()
});
polygon.one(plugin.google.maps.event.OVERLAY_C
}
function onOverlayClicked(polygon, latLng) {
polygon.setFillColor("blue");
polygon.setStrokeColor("green");
polygon.setStrokeWidth(10);
}
Execute
Google

SDK JavaScript

Basta incluir o script do Google no rodapé da página e utilizar a API para exibição e manipulação de mapas.

developers.google.com/maps/documentation/javascript/tutorial?hl=pt-br

SDK JavaScript

No site do Google estão documentadas todas as informações para customização de marcadores, controles, tipos de mapas, sobreposição, etc.

SDK JavaScript

A vantagem é a existência de diversos tutoriais e vários plugins para o SDK JavaScript, seja para marcadores, mapas de calor, cálculo de distância, rota, street view, mapas estáticos, entre outros.

SDK JavaScript

A desvantagem é que pode apresentar um desempenho ruim em redes móveis (3G, 4G, etc). É importante verificar a conexão antes de tentar carregar o mapa.

SDK JavaScript

Além disso, o SDK JavaScript possui cotas de uso, ou seja, há um limite de requisições dependendo do plano.

Instalação

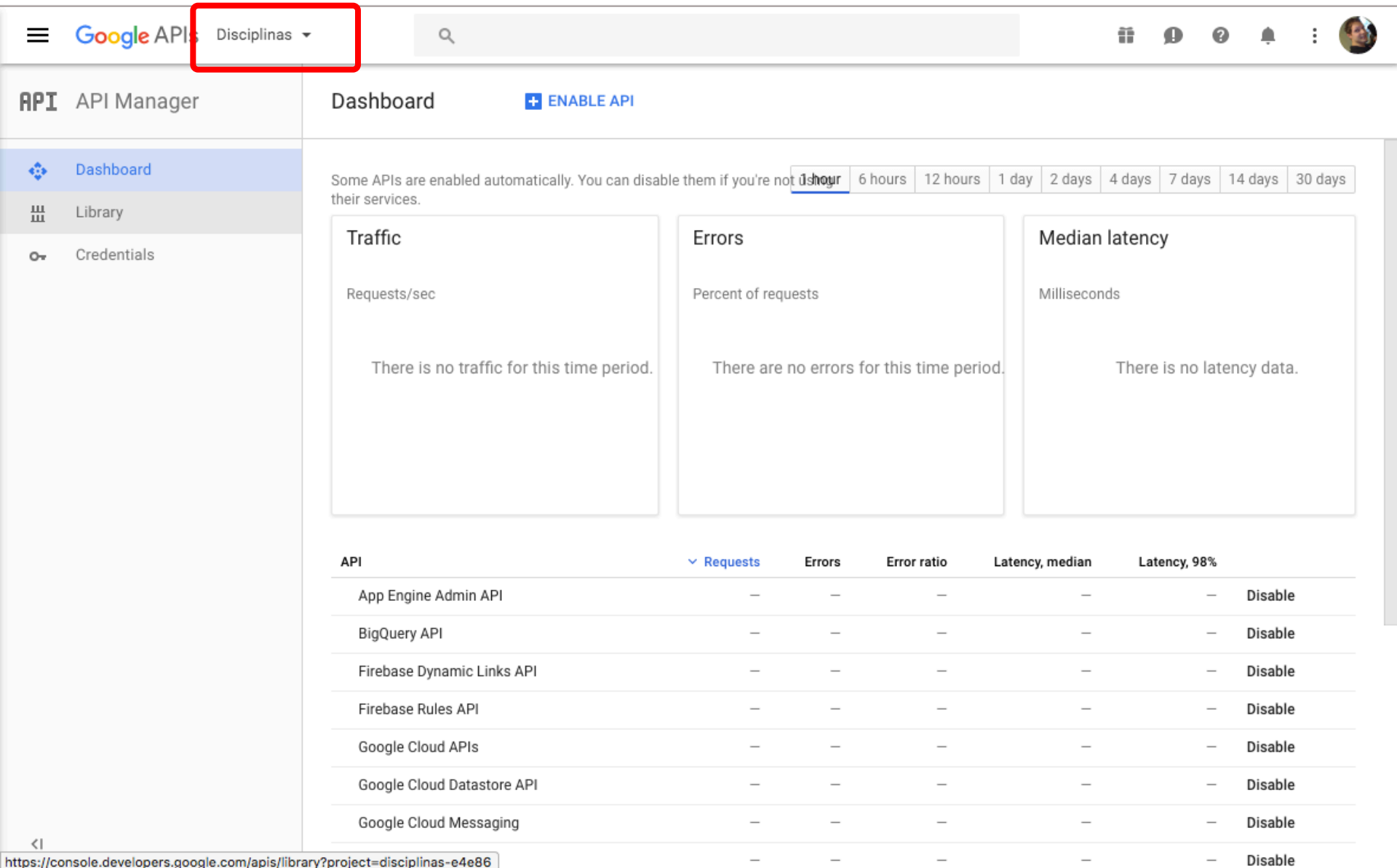
Primeiro, você deve inserir o script do Google Maps no rodapé do seu arquivo index.html, para ficar disponível em toda a aplicação.

```
26  <body>
27    <ion-app></ion-app>
28
29    <script async defer
30      src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&callback=initMap">
31    </script>
32
33    <!-- cordova.js required for cordova apps -->
34    <script src="cordova.js"></script>
35    <!-- Polyfill needed for platforms without Promise and Collection support -->
36    <script src="build/js/es6-shim.min.js"></script>
37    <!-- Zone.js and Reflect-metadata  -->
38    <script src="build/js/Reflect.js"></script>
39    <script src="build/js/zone.js"></script>
40    <!-- the bundle which is built from the app's source code -->
41    <script src="build/js/app.bundle.js"></script>
42  </body>
```

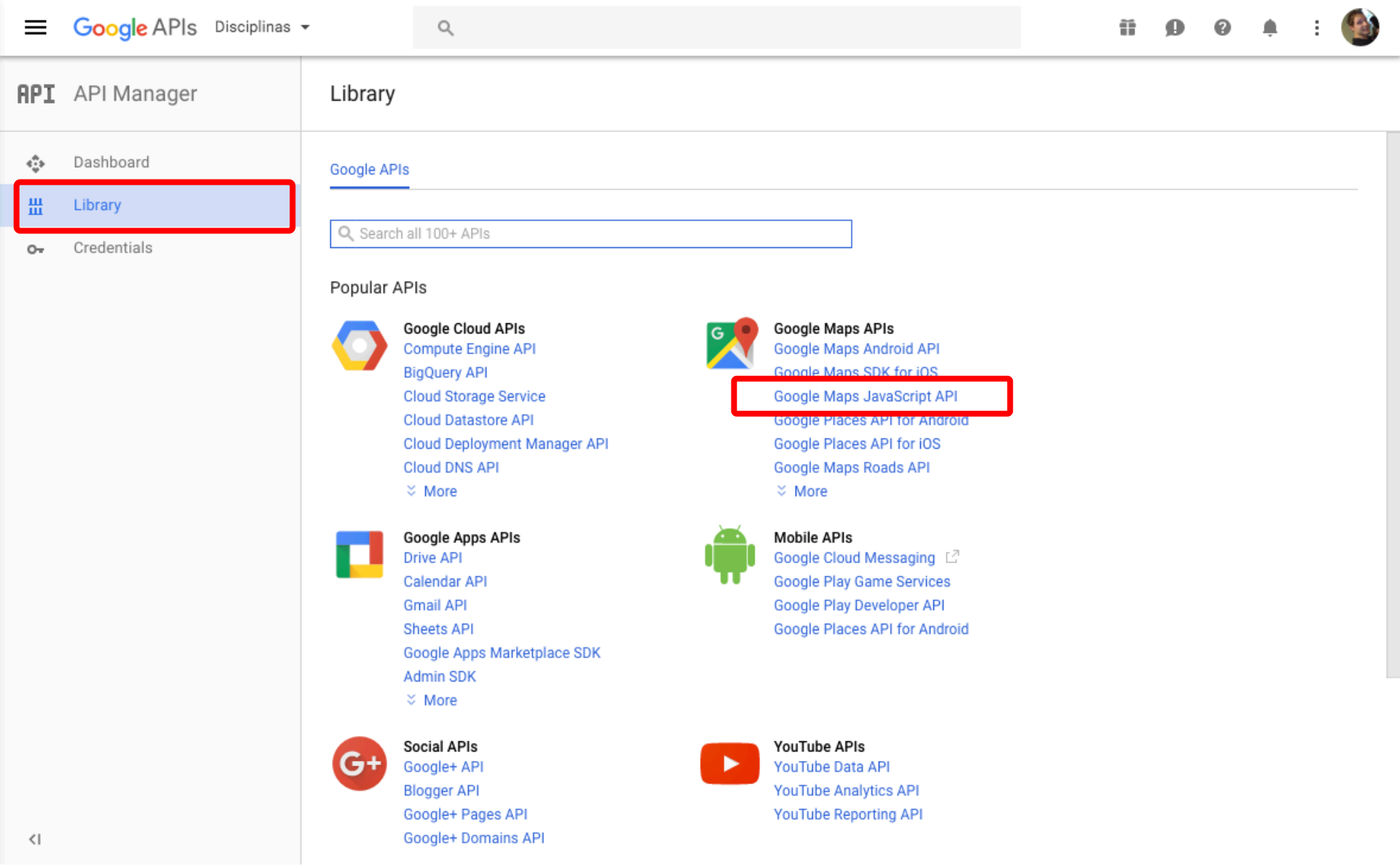
API Key

Você deve habilitar e obter uma chave de API no Google, em console.developers.google.com

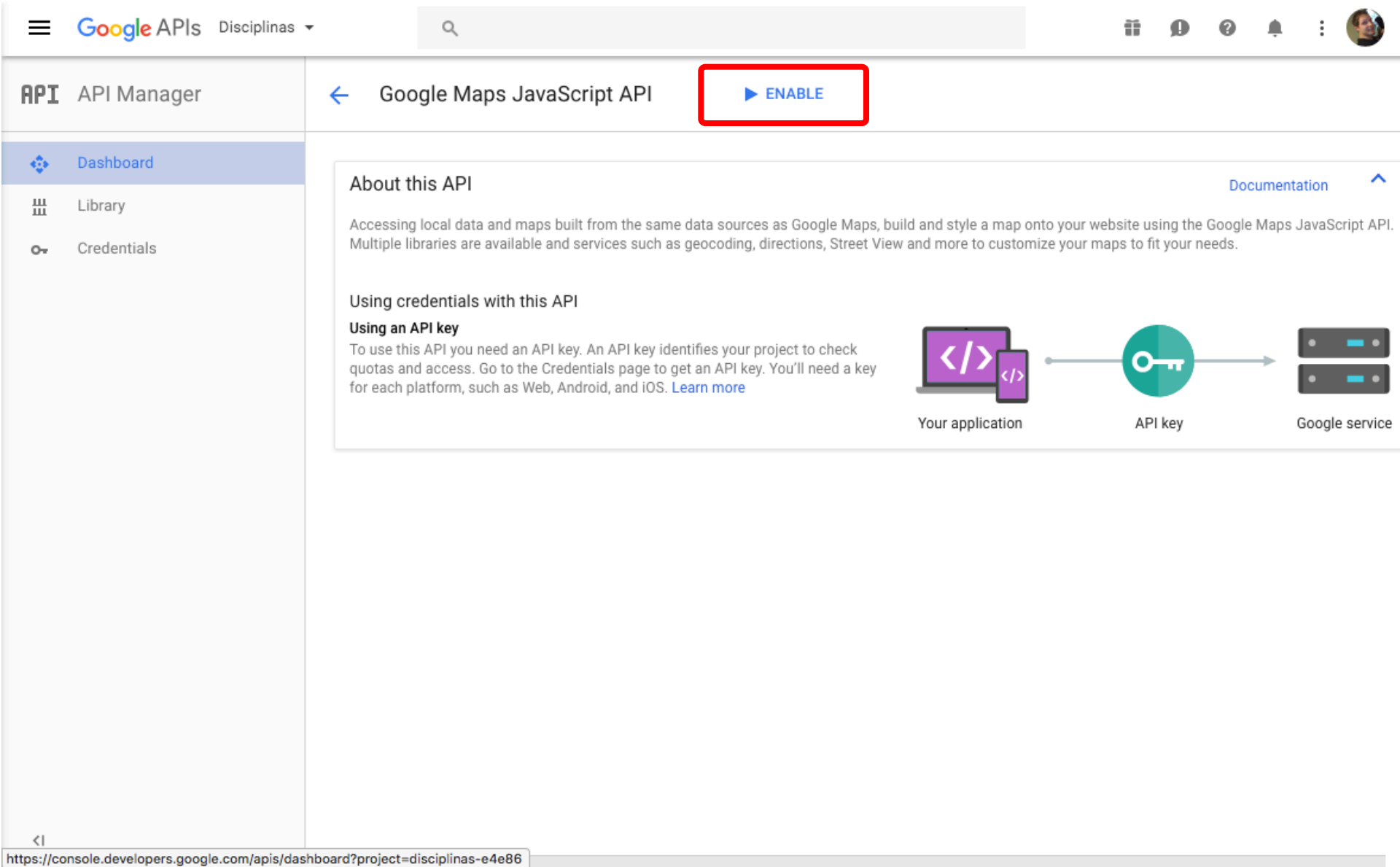
Você pode criar um projeto específico



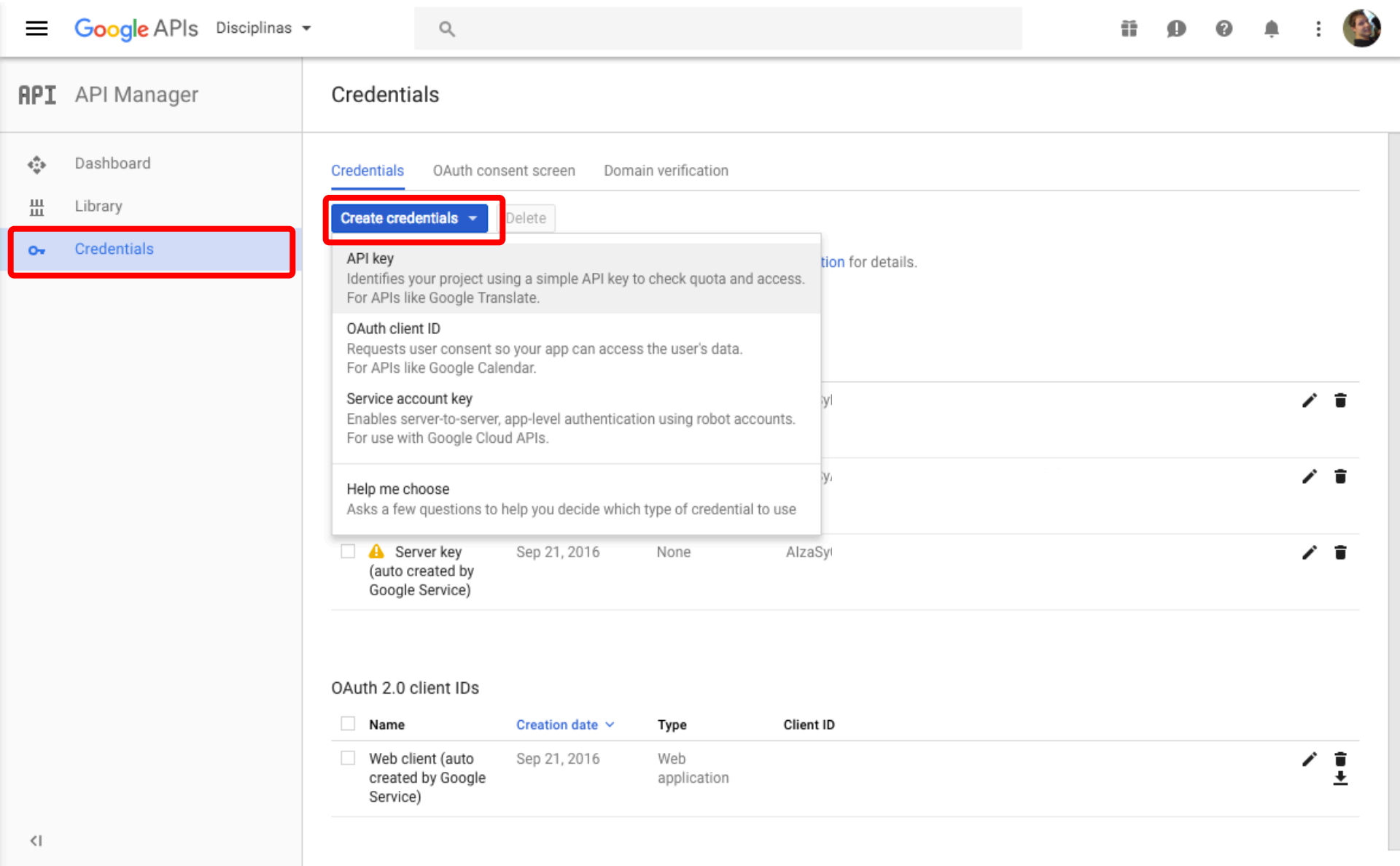
Então entre em Biblioteca -> Google Maps JavaScript



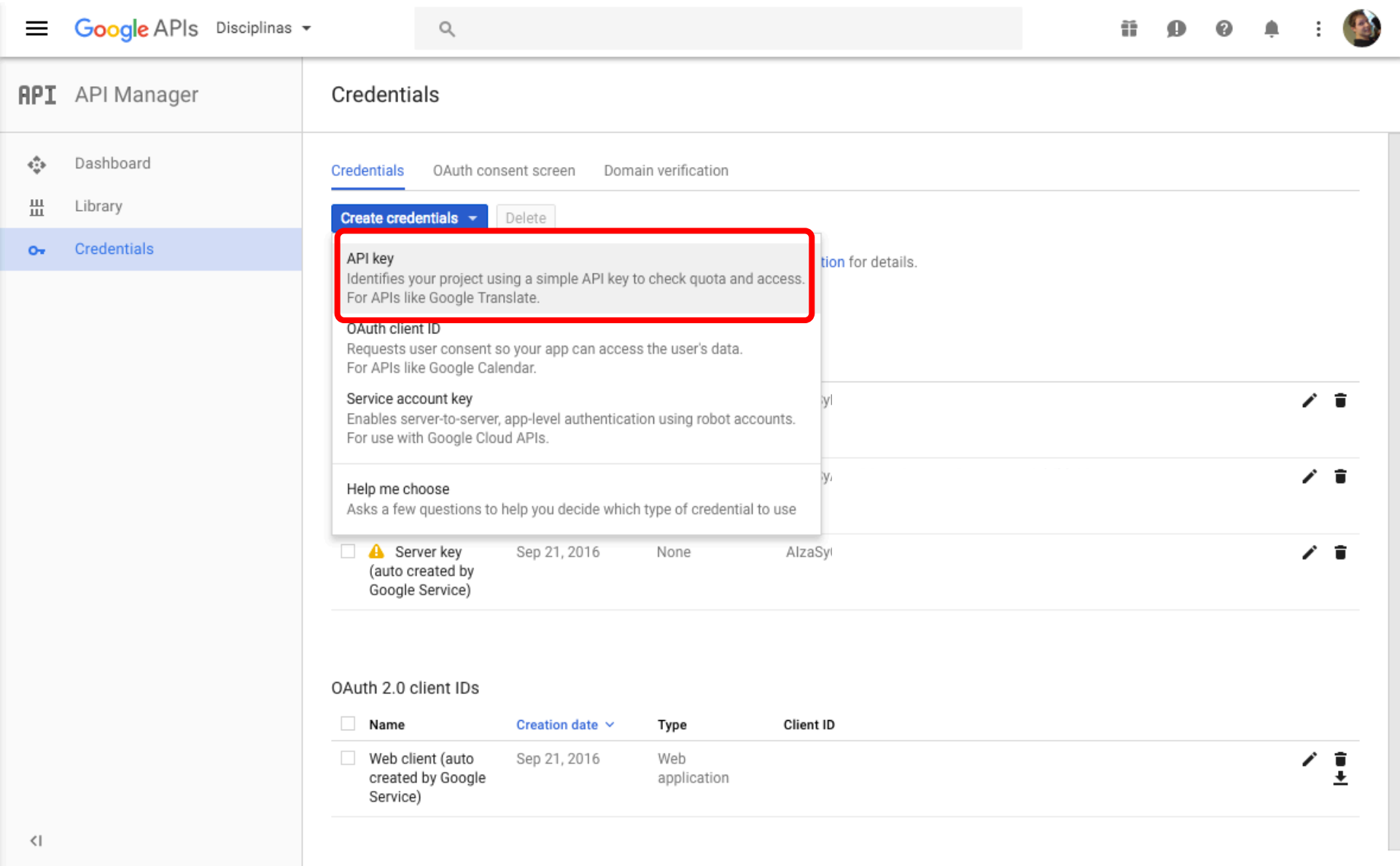
Clique em habilitar a API



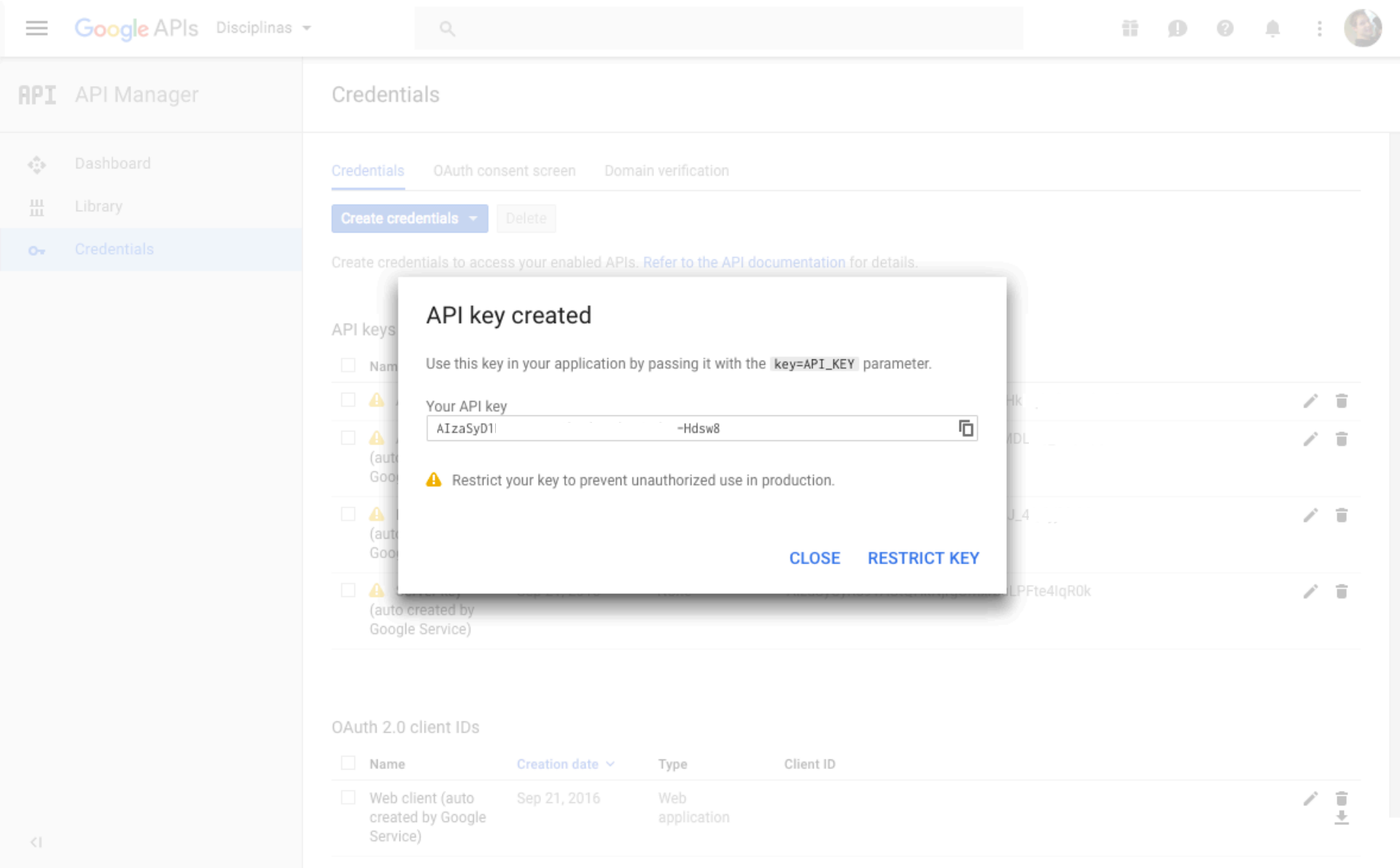
Então, clique em Credenciais e em Criar Credencial



Então selecione a opção ”API Key"



Então copie a API Key e cole no index.html



```
26  <body>
27    <ion-app></ion-app>
28
29    <script async defer
30      src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&callback=initMap">
31    </script>
32
33    <!-- cordova.js required for cordova apps -->
34    <script src="cordova.js"></script>
35    <!-- Polyfill needed for platforms without Promise and Collection support -->
36    <script src="build/js/es6-shim.min.js"></script>
37    <!-- Zone.js and Reflect-metadata  -->
38    <script src="build/js/Reflect.js"></script>
39    <script src="build/js/zone.js"></script>
```

Então você deve inserir o script do Google Maps no index.html, adicionando sua API Key.

No home.html, adicionamos um botão para chamar a função localizar( )

```
      Geolocation
    </ion-title>
    <ion-buttons end>
      <button ion-button (click)="localizar()">
        <ion-icon name="pin"></ion-icon>
      </button>
    </ion-buttons>
  </ion-toolbar>
</ion-header>

<ion-content>
  <div id="mapaView"></div>
</ion-content>
```

Também precisamos criar um div com id específico, no qual iremos inserir o mapa.

```
      Geolocation
    </ion-title>
    <ion-buttons end>
      <button ion-button (click)="localizar()">
        <ion-icon name="pin"></ion-icon>
      </button>
    </ion-buttons>
  </ion-toolbar>
</ion-header>

<ion-content>
  <div id="mapaView"></div>
</ion-content>
```

No CSS precisamos adicionar o comprimento

e a altura como 100% (requisito)

```
#mapaView {
    height: 100%;
    width: 100%;
}
.scroll {
    height: 100%;
}

```

Adicionando o Mapa

```
1   import { Component } from '@angular/core';
2   import { AlertController } from 'ionic-angular';
3   import { Geolocation } from 'ionic-native';
4
5   declare var google: any;
6
7   @Component({
8       templateUrl: 'build/pages/home/home.html'
9   })
10  export class HomePage {
11      public mapa;
12      public latitude;
13      public longitude;
14      public coordenadas;
```

Primeiro vamos declarar a variável google para que o TypeScript consiga validar o script

```
29          this.coordenadas = new google.maps.LatLng(this.latitude, this.longitude);
```

```
4
5   declare var google: any;
6
7   @Component({
8       templateUrl: 'build/pages/home/home.html'
9   })
10  export class HomePage {
11      public mapa;
12      public latitude;
13      public longitude;
14      public coordenadas;
15      public endereco;
16
17      constructor(private alert: AlertController) {…
26
27
28
29
30
31
32
33              zoom: 16,
```

Então vamos declarar algumas variáveis para auxiliar no armazenamento das informações de localização.

Podemos criar o método abrir, que é executado após a captura da localização no construtor.

```
templateUrl: 'build/pages/home/home.html'

public endereco;

constructor(private alert: AlertController) {…

abrir() {
    // Coordenadas
    this.coordenadas = new google.maps.LatLng(this.latitude, this.longitude);
    // Vamos criar o mapa e definir algumas opções
    this.mapa = new google.maps.Map(document.getElementById('mapaView'), {
        center: this.coordenadas,
        zoom: 16,
        mapTypeId: google.maps.MapTypeId.ROADMAP,
        disableDefaultUI: true
    });
}
```

Primeiro, devemos criar um objeto do tipo LatLng com as coordenadas de referência para o mapa.

```
templateUrl: 'build/pages/home/home.html'

public endereco;

constructor(private alert: AlertController) {…

abrir() {
    // Coordenadas
    this.coordenadas = new google.maps.LatLng(this.latitude, this.longitude);
    // Vamos criar o mapa e definir algumas opções
    this.mapa = new google.maps.Map(document.getElementById('mapaView'), {
        center: this.coordenadas,
        zoom: 16,
        mapTypeId: google.maps.MapTypeId.ROADMAP,
        disableDefaultUI: true
    });
}
```

Em seguida criamos o mapa, passando dois parâmetros para a função google.maps.Map

```
templateUrl: 'build/pages/home/home.html'

public endereco;

constructor(private alert: AlertController) {…

abrir() {
    // Coordenadas
    this.coordenadas = new google.maps.LatLng(this.latitude, this.longitude);
    // Vamos criar o mapa e definir algumas opções
    this.mapa = new google.maps.Map(document.getElementById('mapaView'), {
        center: this.coordenadas,
        zoom: 16,
        mapTypeId: google.maps.MapTypeId.ROADMAP,
        disableDefaultUI: true
    });
}
```

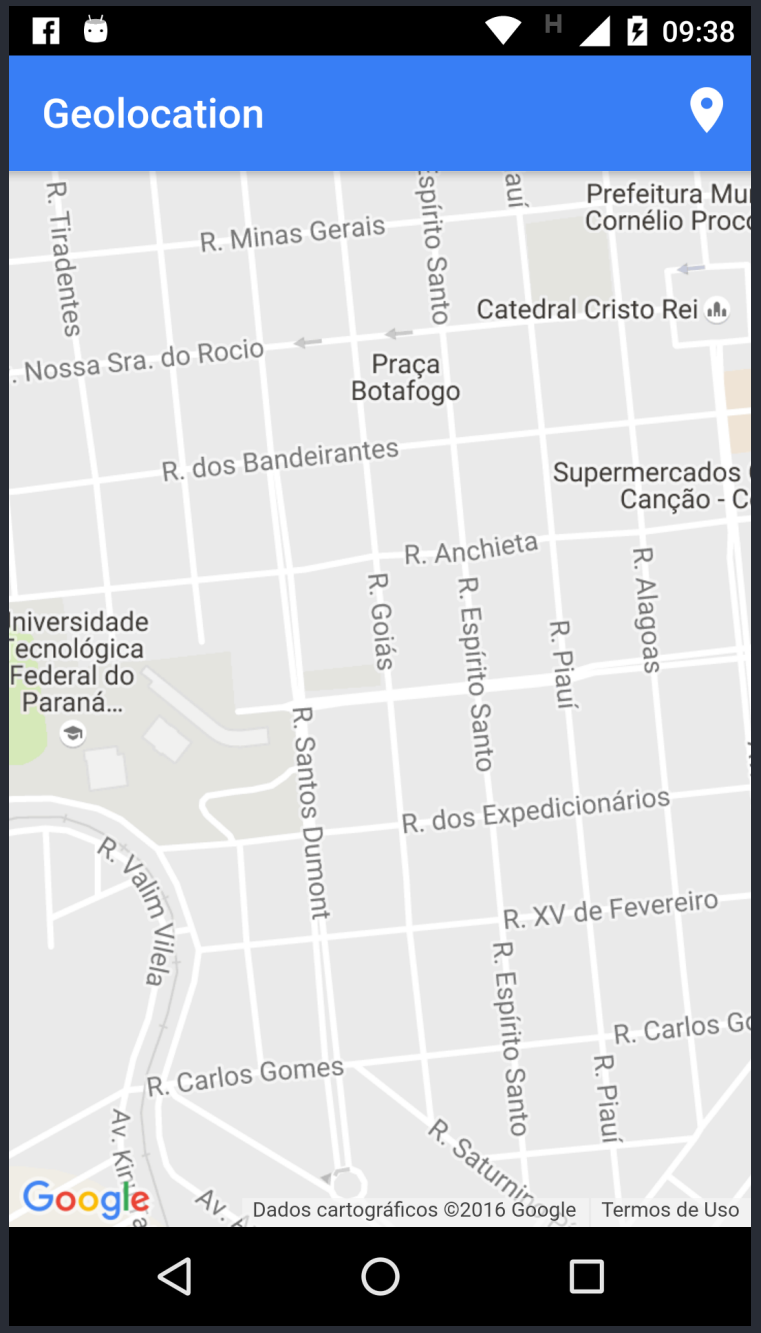
O primeiro deve ser o elemento HTML no qual o mapa será renderizado, ou seja, o elemento mapaView

```
templateUrl: 'build/pages/home/home.html'

public endereco;

constructor(private alert: AlertController) {

abrir() {
    // Coordenadas
    this.coordenadas = new google.maps.LatLng(this.latitude, this.longitude);
    // Vamos criar o mapa e definir algumas opções
    this.mapa = new google.maps.Map(document.getElementById('mapaView'), {
        center: this.coordenadas,
        zoom: 16,
        mapTypeId: google.maps.MapTypeId.ROADMAP,
        disableDefaultUI: true
    });
}
```

O segundo é um array de opções, como center, zoom, tipo de mapa e uma opção para desabilitar os controles.

```
templateUrl: 'build/pages/home/home.html'

public endereco;

constructor(private alert: AlertController) {…

abrir() {
    // Coordenadas
    this.coordenadas = new google.maps.LatLng(this.latitude, this.longitude);
    // Vamos criar o mapa e definir algumas opções
    this.mapa = new google.maps.Map(document.getElementById('mapaView'), {
        center: this.coordenadas,
        zoom: 16,
        mapTypeId: google.maps.MapTypeId.ROADMAP,
        disableDefaultUI: true
    });
}
```

Você pode acessar a API do Google para saber mais informações e métodos disponíveis para os mapas



Adicionando um Marcador

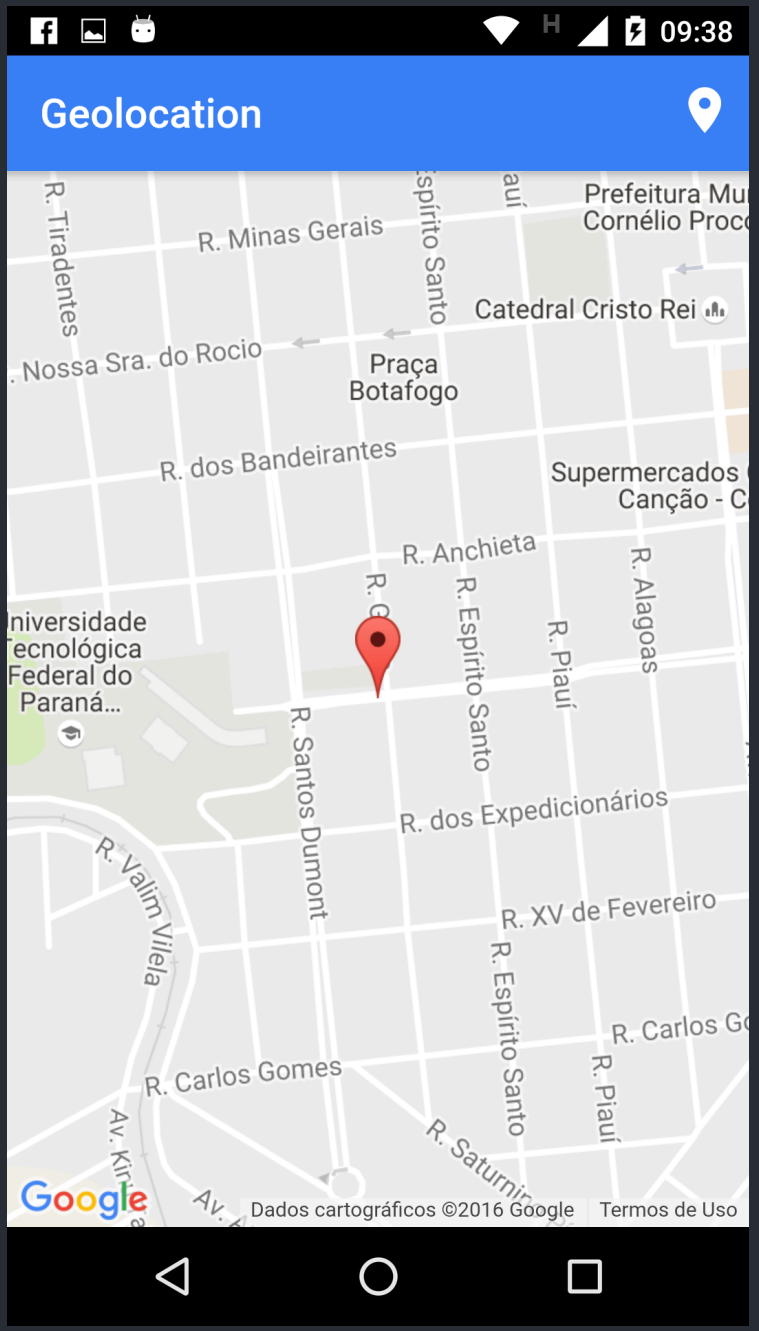
Ao chamar a função localizar, podemos criar um novo marker para o mapa atual.

```
27 ›    abrir() {…
38
39      localizar() {
40          let marker = new google.maps.Marker({
41              map: this.mapa,
42              animation: google.maps.Animation.DROP,
43              position: this.coordenadas
44          });
45          // Adiciona um evento de click
46          marker.addListener('click', () => {
47              this.informacoes();
48          });
49      }
```

Como opções, podemos informar qual o mapa que será mostrado o marker, a animação e a posição.

```
27 ›    abrir() {…
38
39      localizar() {
40          let marker = new google.maps.Marker({
41              map: this.mapa,
42              animation: google.maps.Animation.DROP,
43              position: this.coordenadas
44          });
45          // Adiciona um evento de click
46          marker.addListener('click', () => {
47              this.informacoes();
48          });
49      }
```

Você pode acessar a API do Google para saber como customizar os marcadores com suas imagens.



Eventos

O Google Maps possibilita adicionar diversos eventos aos mapas e marcadores, como click, drag, mouseover, etc.

```
39      localizar() {
40          let marker = new google.maps.Marker({
41              map: this.mapa,
42              animation: google.maps.Animation.DROP,
43              position: this.coordenadas
44          });
45          // Adiciona um evento de click
46          marker.addListener('click', () => {
47              this.informacoes();
48          });
49      }
```

Neste exemplo adicionamos um evento (listener) do tipo click ao marker. Quando clicado, chamamos a função informacoes( )

```
39  localizar() {
40      let marker = new google.maps.Marker({
41          map: this.mapa,
42          animation: google.maps.Animation.DROP,
43          position: this.coordenadas
44      });
45      // Adiciona um evento de click
46      marker.addListener('click', () => {
47          this.informacoes();
48      });
49  }
```

Geocoder

```
39 ›    localizar() {⋯
50
51      informacoes() {
52          let geo = new google.maps.Geocoder;
53          geo.geocode({ location: this.coordenadas }, (res, status) => {
54              if (status === google.maps.GeocoderStatus.OK) {
55                  // Seleciona o endereço formatado
56                  this.endereco = res[0].formatted_address;
57              }
58              else {
59                  this.endereco = "Endereço não encontrado!";
60              }
61
62
63
64
65
66
67      }
```

Ao chamar a função informacoes( ) queremos exibir o endereço do usuário em um alert.

```
39 ›    localizar() {⋯
50
51      informacoes() {
52          let geo = new google.maps.Geocoder;
53          geo.geocode({ location: this.coordenadas }, (res, status) => {
54              if (status === google.maps.GeocoderStatus.OK) {
55                  // Seleciona o endereço formatado
56                  this.endereco = res[0].formatted_address;
57              }
58              else {
59                  this.endereco = "Endereço não encontrado!";
60              }
61
62
63
64
65
66
67      }
```

Para isso, vamos utilizar as funções do

google.maps.Geocoder

```
39 ›    localizar() {⋯
50
51      informacoes() {
52          let geo = new google.maps.Geocoder;
53          geo.geocode({ location: this.coordenadas }, (res, status) => {
54              if (status === google.maps.GeocoderStatus.OK) {
55                  // Seleciona o endereço formatado
56                  this.endereco = res[0].formatted_address;
57              }
58              else {
59                  this.endereco = "Endereço não encontrado!";
60              }
67      }
```

Então podemos chamar a função geocode, passando dois parâmetros.

```
39 ›    localizar() {⋯
50
51      informacoes() {
52          let geo = new google.maps.Geocoder;
53          geo.geocode({ location: this.coordenadas }, (res, status) => {
54              if (status === google.maps.GeocoderStatus.OK) {
55                  // Seleciona o endereço formatado
56                  this.endereco = res[0].formatted_address;
57              }
58              else {
59                  this.endereco = "Endereço não encontrado!";
60
61
62
63
64
65
66
67      }
```

O primeiro consiste de um objeto com a location (no formato LatLng). É possível também passar um endereço em string para retornar a LatLng.

```
39 ›    localizar() {
50
51      informacoes() {
52          let geo = new google.maps.Geocoder;
53          geo.geocode({ location: this.coordenadas }, (res, status) => {
54              if (status === google.maps.GeocoderStatus.OK) {
55                  // Seleciona o endereço formatado
56                  this.endereco = res[0].formatted_address;
57              }
58              else {
59                  this.endereco = "Endereço não encontrado!";
60              }
61
62
63
64
65
66
67      }
```

O segundo parâmetro é uma função, que recebe dois parâmetros (res e status).

```
39 ›    localizar() {⋯
50
51      informacoes() {
52          let geo = new google.maps.Geocoder;
53          geo.geocode({ location: this.coordenadas }, (res, status) => {
54              if (status === google.maps.GeocoderStatus.OK) {
55                  // Seleciona o endereço formatado
56                  this.endereco = res[0].formatted_address;
57              }
58              else {
59                  this.endereco = "Endereço não encontrado!";
60              }
61
62
63
64
65
66
67      }
```

O status serve para verificar qual a situação da busca no Google. Por exemplo, se está OK.

```
39 ›    localizar() {
50
51      informacoes() {
52          let geo = new google.maps.Geocoder;
53          geo.geocode({ location: this.coordenadas }, (res, status) => {
54              if (status === google.maps.GeocoderStatus.OK) {
55                  // Seleciona o endereço formatado
56                  this.endereco = res[0].formatted_address;
57              }
58              else {
59                  this.endereco = "Endereço não encontrado!";
60              }
61
62
63
64
65
66
67      }
```

Se o status está OK, então podemos acessar o objeto res para pegar a resposta com o endereço.

```
39 ›    localizar() {
50
51      informacoes() {
52          let geo = new google.maps.Geocoder;
53          geo.geocode({ location: this.coordenadas }, (res, status) => {
54              if (status === google.maps.GeocoderStatus.OK) {
55                  // Seleciona o endereço formatado
56                  this.endereco = res[0].formatted_address;
57              }
58              else {
59                  this.endereco = "Endereço não encontrado!";
60              }
61
62
63
64
65
66
67      }
```

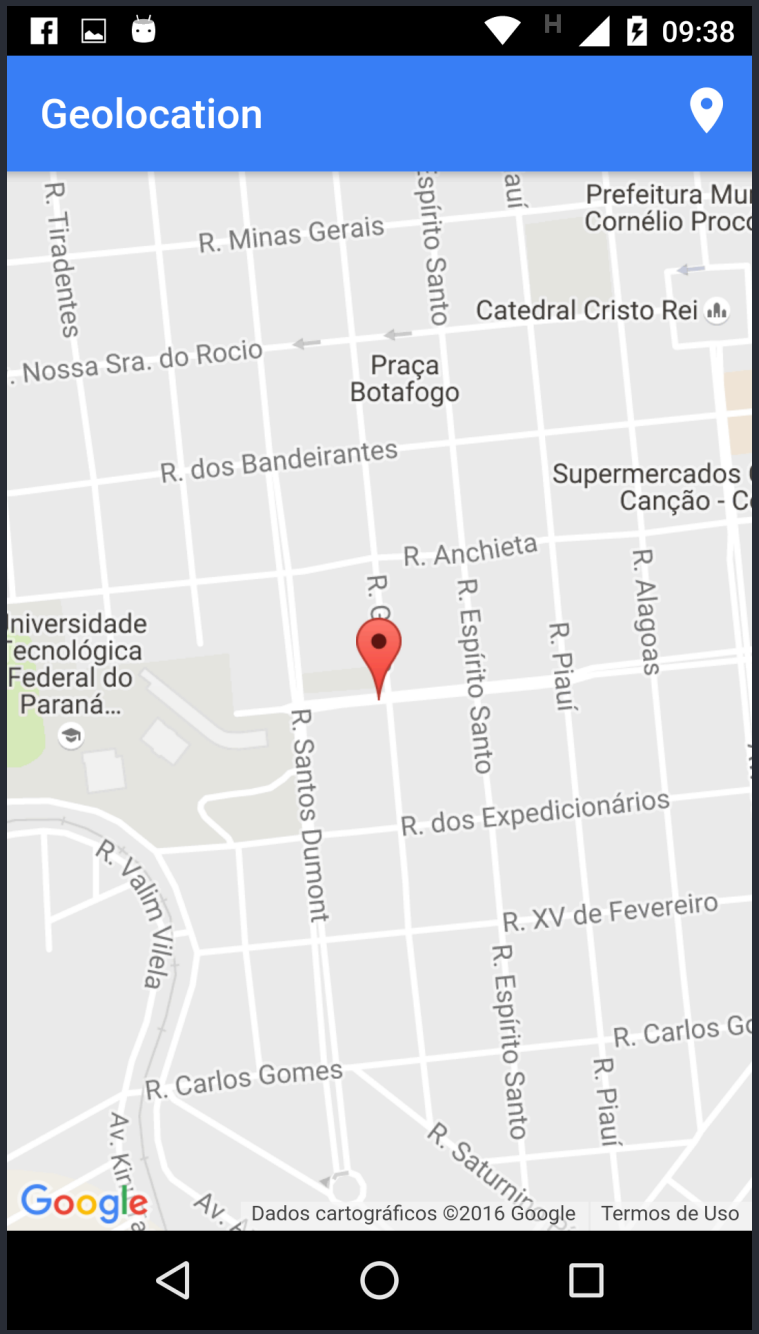
Para isso acessamos a primeira posição, na propriedade formatted_address.

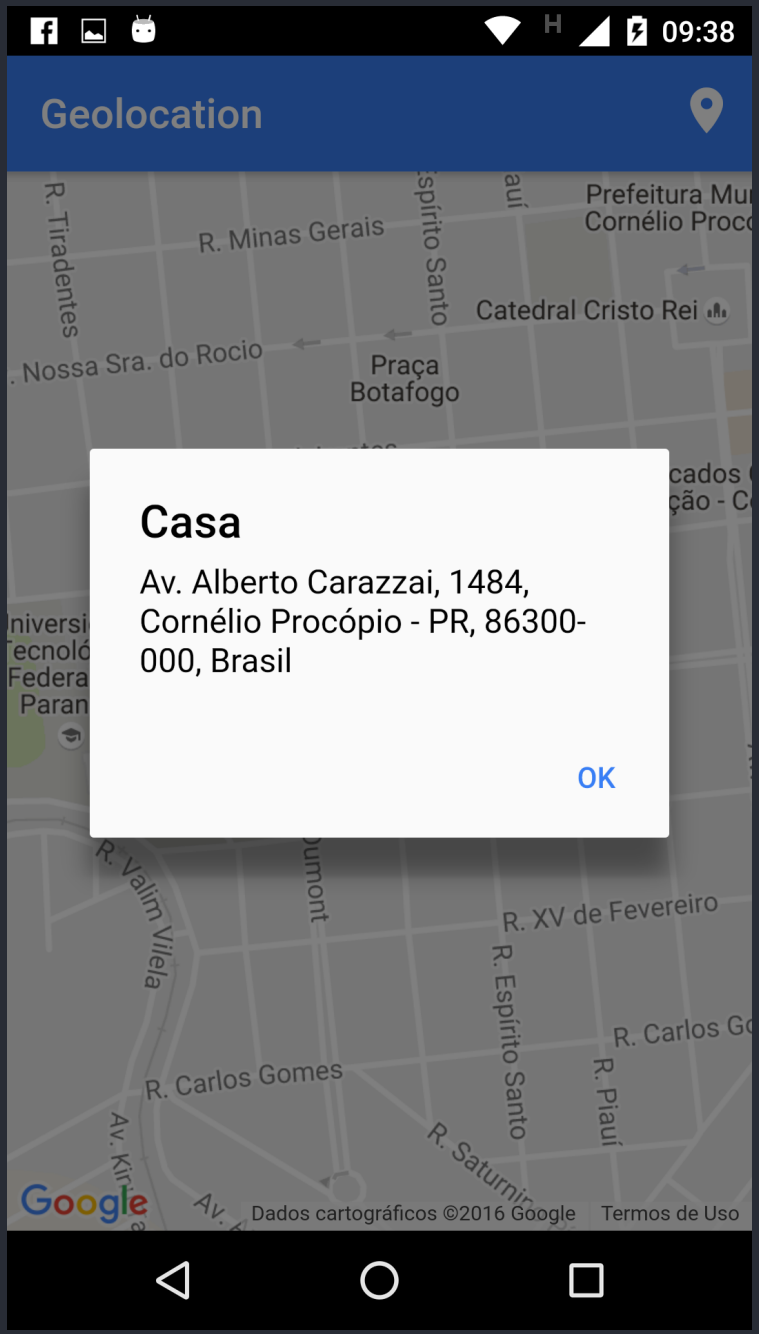
```
39 ›    localizar() {
50
51      informacoes() {
52          let geo = new google.maps.Geocoder;
53          geo.geocode({ location: this.coordenadas }, (res, status) => {
54              if (status === google.maps.GeocoderStatus.OK) {
55                  // Seleciona o endereço formatado
56                  this.endereco = res[0].formatted_address;
57              }
58              else {
59                  this.endereco = "Endereço não encontrado!";
60              }
61
62
63
64
65
66
67      }
```

Na documentação é possível ver quais mais dados são retornados neste objeto res.

Para finalizar, mostramos o endereço formatado em um alert.

```
            // Seleciona o endereço formatado
            this.endereco = res[0].formatted_address;
        }
        else {
            this.endereco = "Endereço não encontrado!";
        }
        this.alert.create({
            title: 'Casa',
            subTitle: this.endereco,
            buttons: ['Ok']
        }).present();
    });
}
```


09:38
Geolocation
R. Minas Gerais
Prefeitura Mu
Cornélio Proc
R. Tiradentes
Catedral Cristo Rei
Nossa Sra. do Rocio
Praça
Botafogo
R. dos Bandeirantes
Supermercados
Canção - C
R. Anchieta
R. Alagoas
R. Piauí
R. Espírito Santo
R. Santos Dumont
R. dos Expedicionários
R. Valim Vilela
R. XV de Fevereiro
R. Carlos Gomes
R. Carlos G
R. Saturni
Google
Dados cartográficos ©2016 Google
Termos de Uso


09:38
Geolocation
R. Minas Gerais
Catedral Cristo Rei
Nossa Sra. do Rocio
Praça
Botafogo
Casa
Av. Alberto Carazzai, 1484, Cornélio Procópio - PR, 86300-000, Brasil
OK
R. Valim Vilela
R. XV de Fevereiro
R. Espírito Santo
R. Carlos Gomes
R. Piauí
Google
Dados cartográficos ©2016 Google
Termos de Uso

Links importantes

developers.google.com/maps/documentation/javascript/markers

developers.google.com/maps/documentation/javascript/events

developers.google.com/maps/documentation/javascript/directions

developers.google.com/maps/documentation/javascript/examples

Exercícios

Exercícios

1. Testar os exemplos e explorar a API de Mapas;
2. Finalizar a Lista de Exercícios 6.
3. Iniciar a Lista de Exercícios 7.