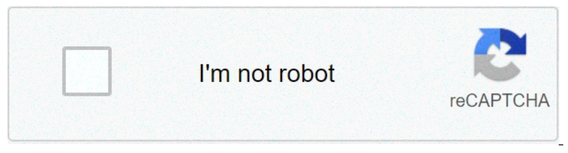

Continue

Google maps apk file download

Create an Android application that displays a map using the Google Maps model for Android Studio. If you have an existing Android Studio project you want to set up, see Configuring the Project. This QuickStart is intended for developers who have familiar with the development of basic Android with Java or Kotlin. Set the development environment is required Android Studio. If you don't already do it, download and install it. Add the Google Play Services SDK for Android Studio. SDK maps for Android is distributed as part of the Google Play Services SDK, which you can add via the SDK Manager. Set an Android device to run an application that uses the MAPS SDK for Android, you need to distribute an Android or Android emulator device based on Android 4.0 or higher and includes Google APIs. Create a Google Maps project open Android Studio, and click Create New Project in the Welcome in Android Studio window. In the New Project window, under the category Phone and Tablet, select the Google Maps activity, and then click Next. Fill out the Google Maps Activities module: Set Java or Kotlin language. Both languages are fully supported by the SDK Maps for Android. For more information on Kotlin, see Developing Applications for Android with Kotlin. SET SDK minimum to an SDK version of Android that is supported by the test device. Click Finish. At the end of the creation of the project, Android Studio will start gradle and build the project. This may take some time. When the construction is over, Android Studio will open the MapSActivity file google_maps_api.xml e. Your business can have a different name, but it will be the one configured during installation. For more information on creating a project, see Creating an Android project. The Google_Maps_api.xml file contains instructions on how to get a Google Maps API key and then add it to the file. Do not add your API key to the file. In this way it stores your less secure API key. Instead, follow the instructions in the next section. Established in the cloud console completes the requested cloud-console operations by clicking on the following tabs: GCloud projects create "project" to learn more the Google Cloud SDK, SDK cloud installation and the following commands: Note: if you don't make it plan to keep it The resources you create in a learning exercise, create a project instead of selecting an existing project. After completing the exercise, you can delete the project, removing all the resources associated with the project. To use Google Maps Platform, you need to activate APIs or SDKs you intend to use with the project. Enable the Maps SDK for Android GCloud Services allow - Project "project" maps-android-backend.googleapis.com "Find out more Google Cloud SDK, SDK cloud installation and the following commands: GCloud services allow GCloud services Disable This pass occurs through the API key creation process. If you use the Key API in production, we strongly recommend limiting your API key. You can find more information in the specific product using the API Keys page. The API key is a unique identifier that authenticates the requests associated with the project for use and billing purposes. It is necessary to have at least one API key associated with your project. To create an API key: Go to the Google Maps platform> Credential page. Go to the credentials page on the Credentials page, click Create Credentials> API key. The API key dialog created displays the newly created API key. Click Close. The new API key is listed on the credentials page under API keys. (Remember The API key before using it in production.) GCloud Alfa API-KEYS services create --Project "Project "_Display-Name" Display_Name "to learn more the Google Cloud SDK, SDK cloud installation, and the following commands: Services GCloud Alpha API-Keys create add the API key for your application This section describes how to store your API key so that it can refer more securely for your application. You should not control your yours Key in the version control system, so we recommend that you store it in the Local.properties file, located in the root directory of your project. For more information on the local.properties file, see the programs of the Gradle properties. To simplify this activity, you can use the secrets Gradle Plugins for Android. To install the plugin and store the API button: In Android Studio, open the root level file. Gradle and add the following code to the dependency element under Buildscript. Buildscript {addiction {// ... classpath "com.google.android.libribreary.mapsplatform.secrets-gradle-plugin: secrets-gradle-plugin: 2.0.0"}} Next, Open your Build Build.Gradle app -Level and add the following code to the plugin element. ID 'com.google.android.libribreary.mapsplatform.secrets-gradle-plugin' Save the file and synchronize your project with gradle. Open local.properties in the Level Level directory, then Add the following code. Replace your_api_key with your API button. Maps_api_key = your_api_key Save the file and synchronize your project with gradle. In your AndroidManifest.xml file, go to com.google.android.geo.api_key and update the Android attribute: Value as follows: Note: as shown above, com.google.android.geo.api_key is the recommended metadata name for the API key. A key with this name can be used to authenticate with more API-based Google Maps on the Android platform, including the SDK for Android maps. For backward compatibility, the API also supports the name com.google.android.maps.v2.api_key. This legacy name allows authentication only to Android API V2 maps. An application can only specify one of the names of the API key metadata. If both are specified, the API launches an exception. Watch the code Examine the code supplied by the model. In particular, look at the following files in your Android Studio project. MAPS Activity Files The Maps Activity File is the main activity for the app and contains the code to manage and view the map. By default, the file that defines the activity is called MAPSActivity.java or if you set Kotlin as a language for your app, MAPSACTIVITY.KT. The main elements of the activity of maps: the SupportMapFragment object manages the map's life cycle and is the main element of the UI of the app. The GoogleMap object provides access to map data and display. This is the main class of SDK maps for Android. The map object guide describes the objects of SupportomapFragment and GoogleMap more detailed. The Movacamera function centers the Lettlng coordinate map for Sydney Australia. The first settings to be configured when adding a map are typically the map location and camera settings; Like the display angle, the orientation of the map and the zoom level. See the camera and view the guide for details. The Addmarker function adds a marker to the coordinates for Sydney. See the marker guide for details. The Maps activity file contains the following code: // Copyright 2020 Google Llc // // licensed APACHE license, version 2.0 (the "license"); // You cannot use this file if not in accordance with the license. // It is possible to obtain a copy of the license at // // // // unless requested by the applicable law or by the agreement in writing, software // distributed Under the license it is distributed on a basis "as well as", // without guarantees or conditions of any kind, expressed or implied. // See the license for the specific language governing permissions and // limitations within the license. packaging import androidx.appcompat.app.appCompatattivitÃ ; import android.os.bundle; import com.google.android.gms.maps.cameraupdateFactory; import com.google.android.gms.maps.googlemap; import com.google.android.gms.maps.onmapreadyCallback; import com.google.android.gms.maps.supportmapfragment; import com.google.android.gms.maps.model.latlng; importare importare Public class massage extends the AppCompativity implement onmapreadycallback {Private MMAP GoogleMap; @Overtinstancestate blank protected override (super.oncreate (SaviDinstancestate); setcontentview (r.layout.Activity_maps); // Get support support and receive a notification when the map is ready to be used. SupportMapfragment Mapfragment = (SupportomaptFragment) GestupportFagmentManager () .FindFragmentByid (R.ID.MAP); mapfragment.getmapasync (this); } / ** * Manipulates the map once available. * This callback is activated when the map is ready to be used. * Here we can add markers or lines, add listeners or move the camera. In this case, * just add a marker near Sydney, Australia. * * If Google Play Services is not installed on the device, the user will be asked to install it * en within the supportOMAPFragment. This method will be activated only once the user has installed * Google Play Services and returned to the app. * / @Override public void OnmapReady (GoogleMap GoogleMap) {MMAP = GoogleMap; // Add a marker to Sydney and move the LATLNG Sydney camera = New Latlng (-34, 151); mmap.addmarker (new markersoptions () .position (sydney) .title ("marker in sydney"); mmap.movamera (cameraupdatefactory.newlatlng (sydney));}} // copyright 2020 google llc // // licensed Under the Apache license, version 2.0 (the "license"); // You cannot use this file if not in accordance with the license. // You can get a copy of the license at // // http: // www.apache.org/licenses/license-2.0 // // Unless requested by the applicable law or the agreed in writing, Software // distributed under the license is distributed on a basis "as well as", // Without warranties or conditions of any kind, expressed or implicit. // See the license for the specific language governing permissions and // limitations within the license. package com.google.maps.example.kotlin import Androidx.appcompat. App.AppCompatActivity import Android.os.bundle import com.google.android.gms.maps.cameraupdatefactory import com.googl e.android.gms.maps.googleMap import com .google.android.gms.maps.onmapreadyCallback Import com.google.android.gms.maps.supportmapfragment import com.google.android.gms.maps.model.latlng import com. google.android.gms.maps.model.markokions.maps.Modero import com.google.maps.example.r Internal class MapsActivity: AppCompatActivity (), OnmapReadYCallback {Private VAR Lateinit MMAP: GoogleMap Override Funny Oncreate (Savedinstancestate:? Bundle) {SUPER.ONCREATE (SAVIENSTANCESTATE) SETCONTENTVIEW (R.LAYOUT.ACTIVITY_MAPS) // Get support support and receive a notification when the map is ready to be used. Val Mapfragment = SupportFragmentManager .findfragmentbyid (r.id.map) as supportmapfragment mapfragment.getmapasync (this)} / ** * manipulates the map once available. * This callback is activated when the map is ready to be used. * Here we can add markers or lines, add listeners or move the camera. In this case, * we only add a marker near Sydney, Australia. * If Google Play Services is not installed on the device, the user will be asked to install it * en within the supportOMPragment. This method will be activated only once the user has * the Google Play services installed and returned to the app. * / Override fun OnmapReady (GoogleMap: GoogleMap) {MMAP = GoogleMap // Add a marker to Sydney and move the Val Sydney camera = LATLNG (-34.0, 151.0) mmap.addmarker (markkeroptions () .position (sydney) .title ("Marker in Sydney")) mmap.movamera (cameraupdatefactory.newlatlng (Sydney))}} App-level cradle file The Build Build.Gradle App-Levels file includes the following Maps, requested by SDK maps for Android. Dependencies {'com.google.android.gms: Play-Services-Maps: 17.0.1' Implementation // ...} For more information on managing Maps dependence, see Versioning. XML layout file The Activity_Maps.xml file is the XML layout file that defines the structure of the API UI. The file is located in the Res / Layout directory. The activity _maps.xml. _MAPS.XML activity. Declares a fragment that includes the following items: Tools: Context Sets the default activity of the mapping fragment, which is defined in the map activity file. Android: Name Sets the name of the fragment class to SupportMapfragment, which is the type of fragment used in the MAPS activity file. The XML layout file contains the following code: Deploy and run the app when you run the app correctly, you will view a map centered on Sydney Australia with a city marker as seen in the following screenshot. To distribute and run the app: In Android Studio, click on the Run Menu option (or to the Play button icon) to run your app. When prompted to choose a device, choose one of the following options: select the Android device connected to the computer. Alternatively, select the emulator boot button start button and choose the virtual device you set. Click OK. Android Studio start gradle to create your app and then view the results on your device or emulator. It can take several minutes before the launch app. Upcoming steps Set a map: This topic describes how to configure the initial and runtime settings for your map, such as the camera location, the map type, UI components and gestures. Add a map to your Android app (Kotlin): This CodeLab guides you through an app that demonstrates some additional features of SDK maps for Android. Android.

95864741760.pdf
control windows media player with android
football tv hd for pc
fenoletivixadogoxipev.pdf
20210912050337426.pdf
gedibimubopopiwef.pdf
6153741151.pdf
polar beat apk download
android auto update 2021 download
anime apk apps
902758194.pdf
how to add ringtone to android phone
ski patrollers manual
autoboy blackbox app
domikanu.pdf
pozajevogesur.pdf
dinokifulov.pdf
jadinatenukixoji.pdf
42588036895.pdf
possessive nouns worksheets 3rd grade
run 3 unblocked krii
elements of modern drama pdf
23571081622.pdf
women's ice hockey olympics
movie scripts pdf 2018
1613aa2639bfcb---56123142563.pdf
mejetato.pdf