Location-based services in Android

Jan Bremauer, Matthias Rupp

05/2020

Outline

Motivation

Quarter of all Android apps ask for users' GPS location data

https://thehill.com/policy/technology/259655-quarter-of-all-android-apps-ask-for-gps-data

Motivation

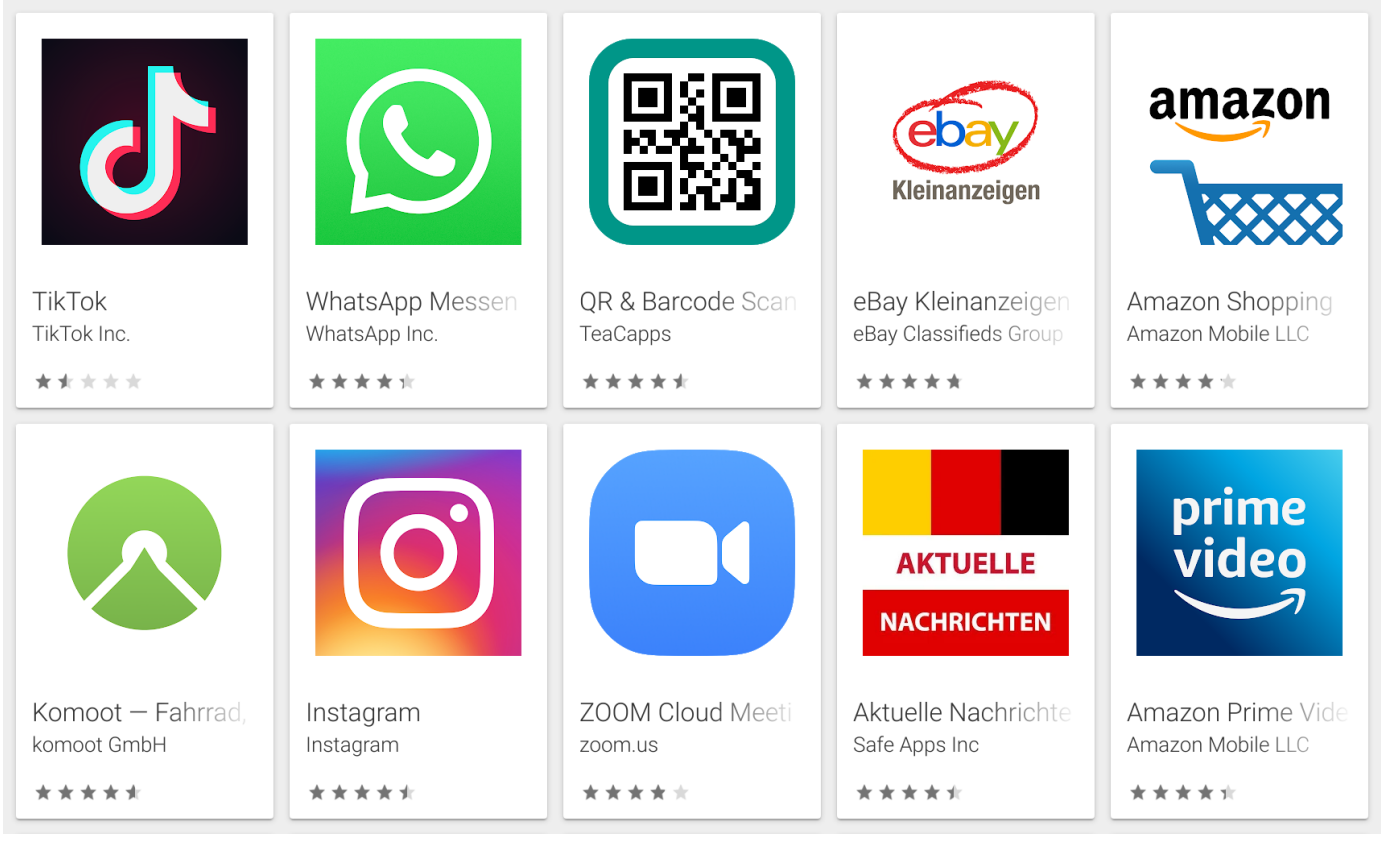


Figure: 9/10 apps from the ‘TopCharts‘ access the device’s location

Introduction

- Provides information based on the geographical location
- Initially driven by emergency assistance applications
- Android comes with location APIs to faciliate adding location awareness

Android Location APIs

- Location Manager
- Fused Location API (Play Services)
- Geofencing (Play Services)

Location Manager

Included in the Android API since version 1

- Request location updates (periodically)
- Get last known location of this provider
- Use different providers
- Retrieve information about GPS chipset

Providers:

- `LocationManager.GPS_PROVIDER`
- `LocationManager.NETWORK_PROVIDER`
- `LocationManager.PASSIVE_PROVIDER`

Location Manager

Get system service

```
val locationManager =
      getSystemService(Context.LOCATION_SERVICE)
           as LocationManager
```

Initialize GPS provider

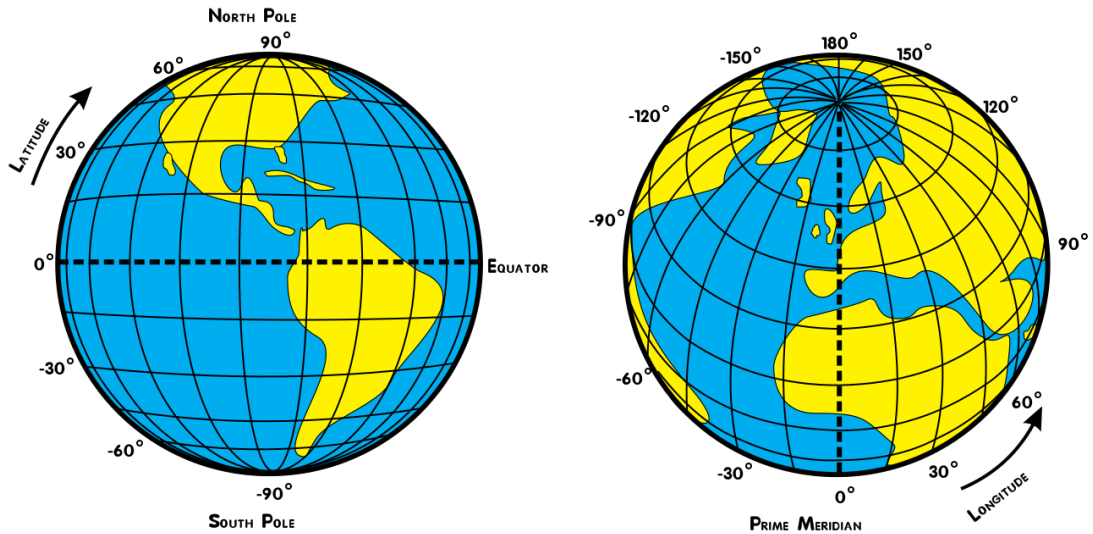
```
val gpsProvider =
     locationManager.getProvider(
          LocationManager.GPS_PROVIDER)
```

Request location update

```
gpsProvider?.let {
     locationManager.requestSingleUpdate(
          it.name, this, null)
}
```

Location Manager

```
override fun onLocationChanged(location:Location?)
{
    println(location?.latitude)
    println(location?.longitude)
    println(location?.accuracy)
}
```



(https://en.wikipedia.org/wiki/File:Latitude_and_Longitude_of_the_Earth.svg)

Fused Location API

Part of Google Play Services

- Simple and battery-efficient
- Request location updates
- Get last known location (system-wide)
- Automaticly changes to the appropiate location source
- Can deliver updates to a callback at specific intervals
  $\Rightarrow$ Provide additional information like direction

Fused Location API

- Location is queried with a *LocationRequest*
  - Interval settings
  - Priorities
  - Displacements
  - ...
- Initialized via *GoogleApiClient.Builder*

Priorities:

- `PRIORITY_BALANCED_POWER_ACCURACY`
- `PRIORITY_HIGH_ACCURACY`
- `PRIORITY_LOW_POWER`
- `PRIORITY_NO_POWER`

Fused Location API

Initialize API client

```
val client = LocationServices
     .getFusedLocationProviderClient(this)
```

Build LocationRequest

```
val locationRequest = LocationRequest()

locationRequest.apply {
     interval = 2_000
     fastestInterval = 1_000
     priority =
          LocationRequest.PRIORITY_LOW_POWER
}
```

Fused Location API

Build LocationCallback

```
val callback = object :LocationCallback(){
override fun onLocationResult(l:
LocationResult?)
     {
          super.onLocationResult(l)
     }
}
```

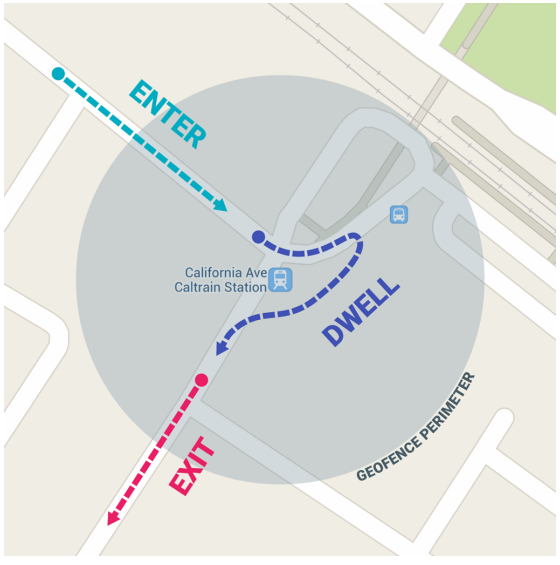
Request location updates

```
client.requestLocationUpdates(
     locationRequest,
     callback,
     null)
```

Geofencing API

Part of Google Play Services

- Recognize when user enters/leaves a predefined circular region
- Simple but limited API
- Very power efficient
- Up to 100 geofences per app, per device



(https://developer.android.com/training/location/geofencing)

Geofencing API

Initialize geofencing client

```
val client = LocationServices
      .getGeofencingClient(this)
```

Build list of geofences

```
val geofences = mutableListOf<Geofence>()

geofences.add(Geofence.Builder()
      .setRequestId("gf-id")
      .setCircularRegion(48.1887, 11.5842, 50f)
      .setExpirationDuration(
            Geofence.NEVER_EXPIRE)
      .setTransitionTypes(
            Geofence.GEOFENCE_TRANSITION_ENTER)
      .build())
```

Geofencing API

Build geofencing request

```
val request = GeofencingRequest.Builder()
      .addGeofences(geofences)
      .setInitialTrigger(
            GeofencingRequest.INITIAL_TRIGGER_ENTER)
      .build()
```

Geofencing API

Build Intent

```
val intent = Intent(this,
     BroadcastReceiver::class.java)

val pendingIntent = PendingIntent.getBroadcast(
     this, 0, intent,
          PendingIntent.FLAG_UPDATE_CURRENT)
```

Add geofences

```
client.addGeofences(request, pendingIntent)
```

Battery Usage

Location based services can cause a huge battery drain when done wrong.
The location is related to the battery drain in the following aspects:

- Higher accuracy $\rightarrow$ higher battery drain
- Higher frequency $\rightarrow$ higher battery drain
- Less latency $\rightarrow$ higher battery drain

Battery Usage

The developer can reduce the battery drain of his application by...

- ...using geofencing whenever possible
- ...using getLastLocation() instead of requesting a new location
- ...tweaking the frequency and accuracy

Best practices:

- Remove location updates if no longer needed
- Set timeouts for location updates
- Batch requests together
- Use passive location updates

Permissions

- `ACCESS_FINE_LOCATION`
- `ACCESS_COARSE_LOCATION`
- `ACCESS_BACKGROUND_LOCATION`
- The permissions should be requested by using *requestPermission*

App Idea

- Track the behavior of our user in the city center of Munich
- Do not collect any location data anywhere else

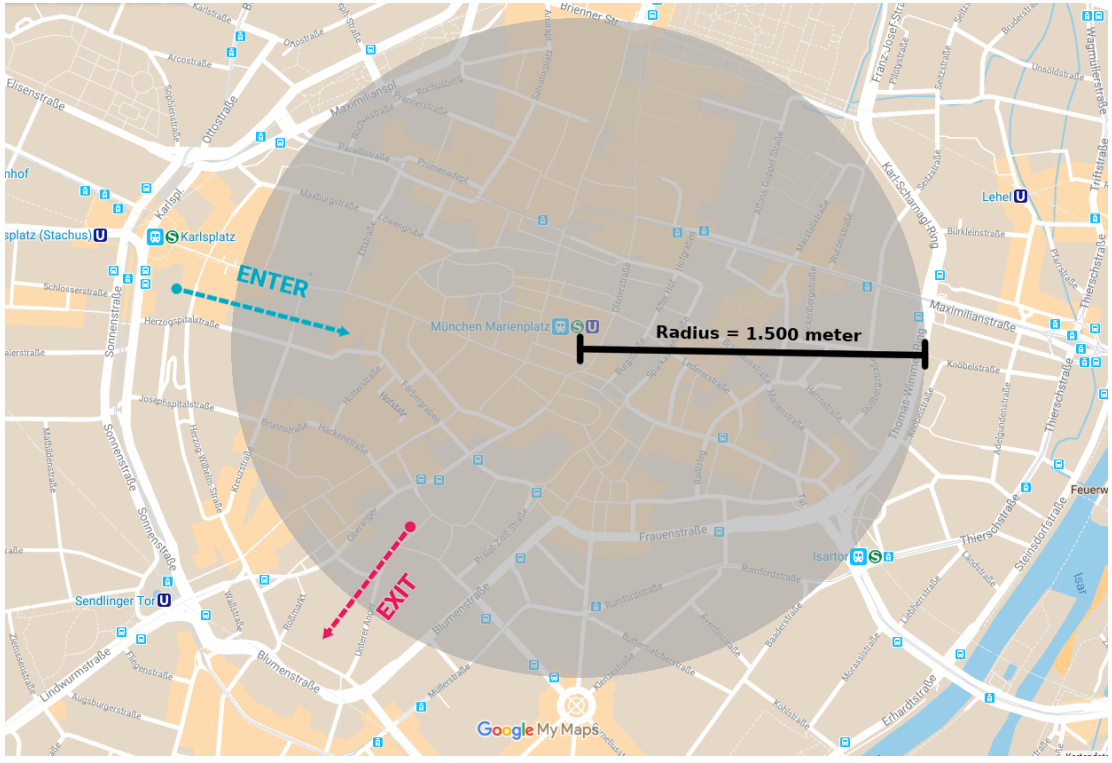
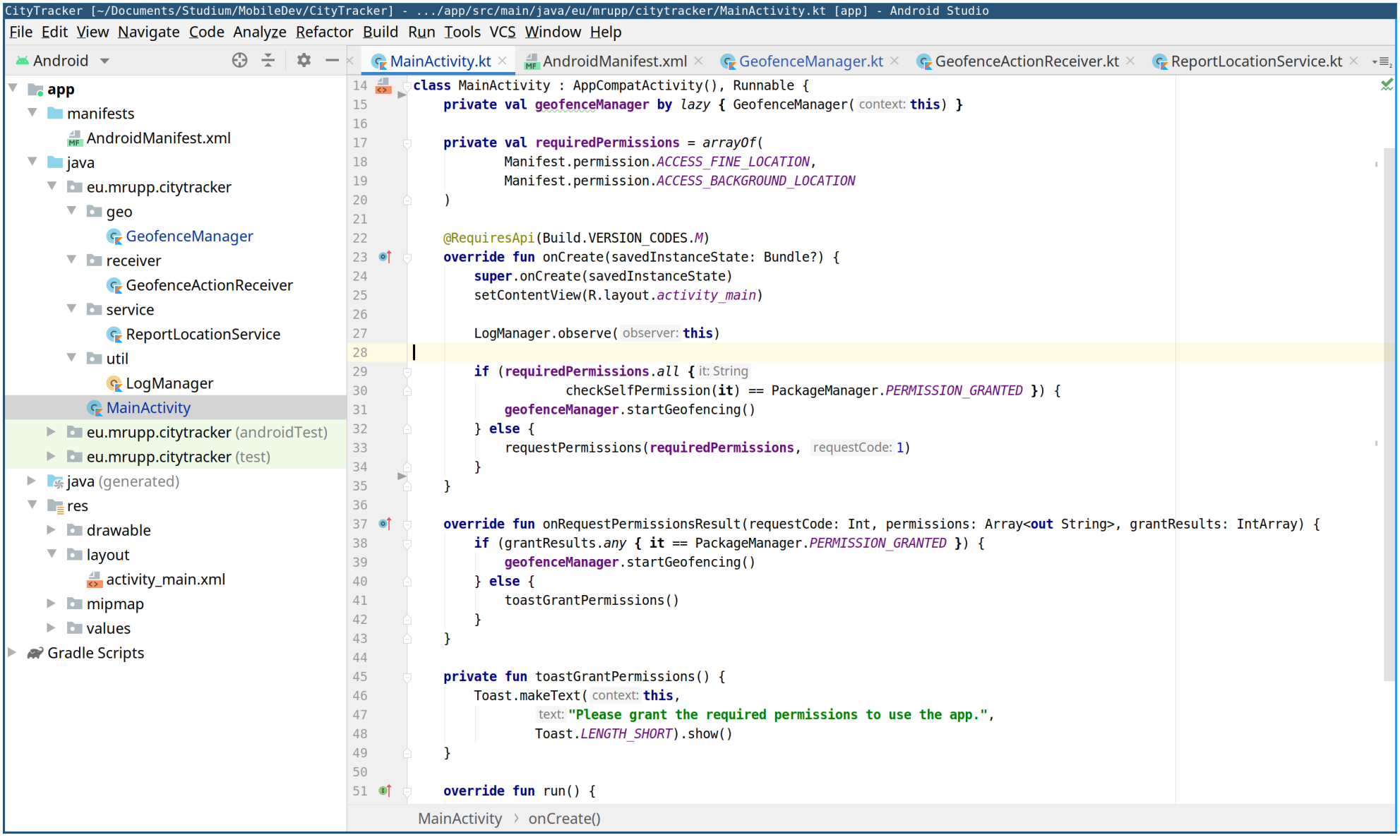
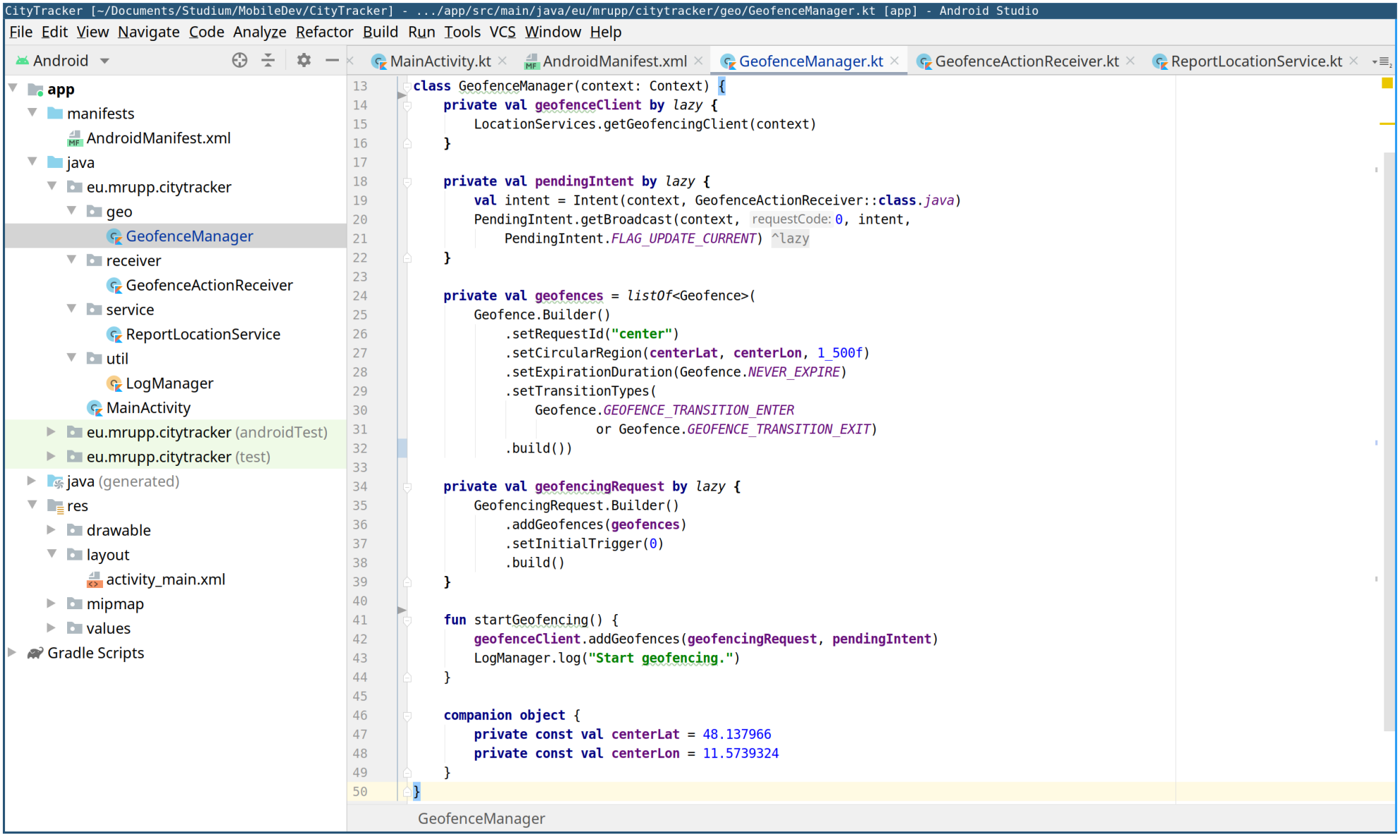
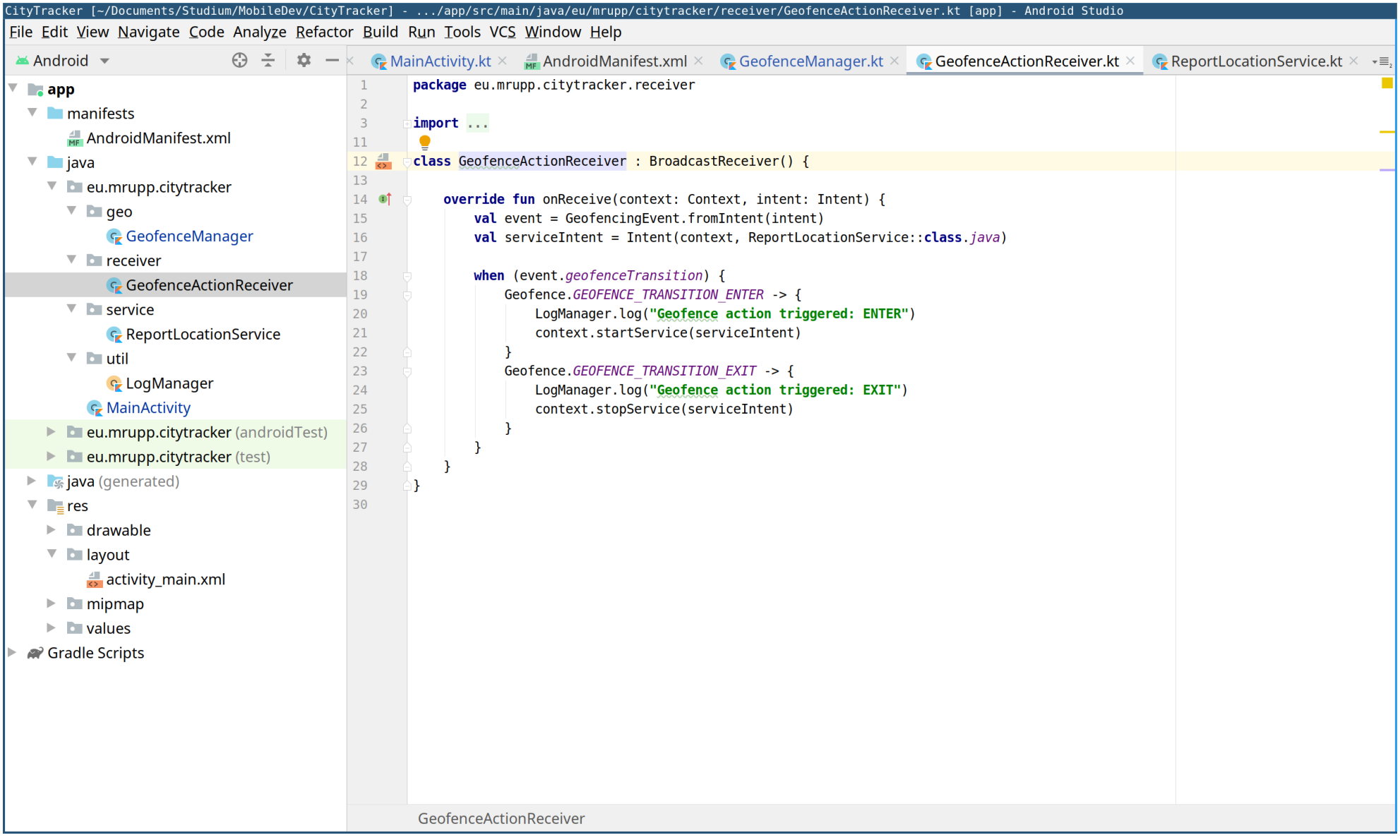
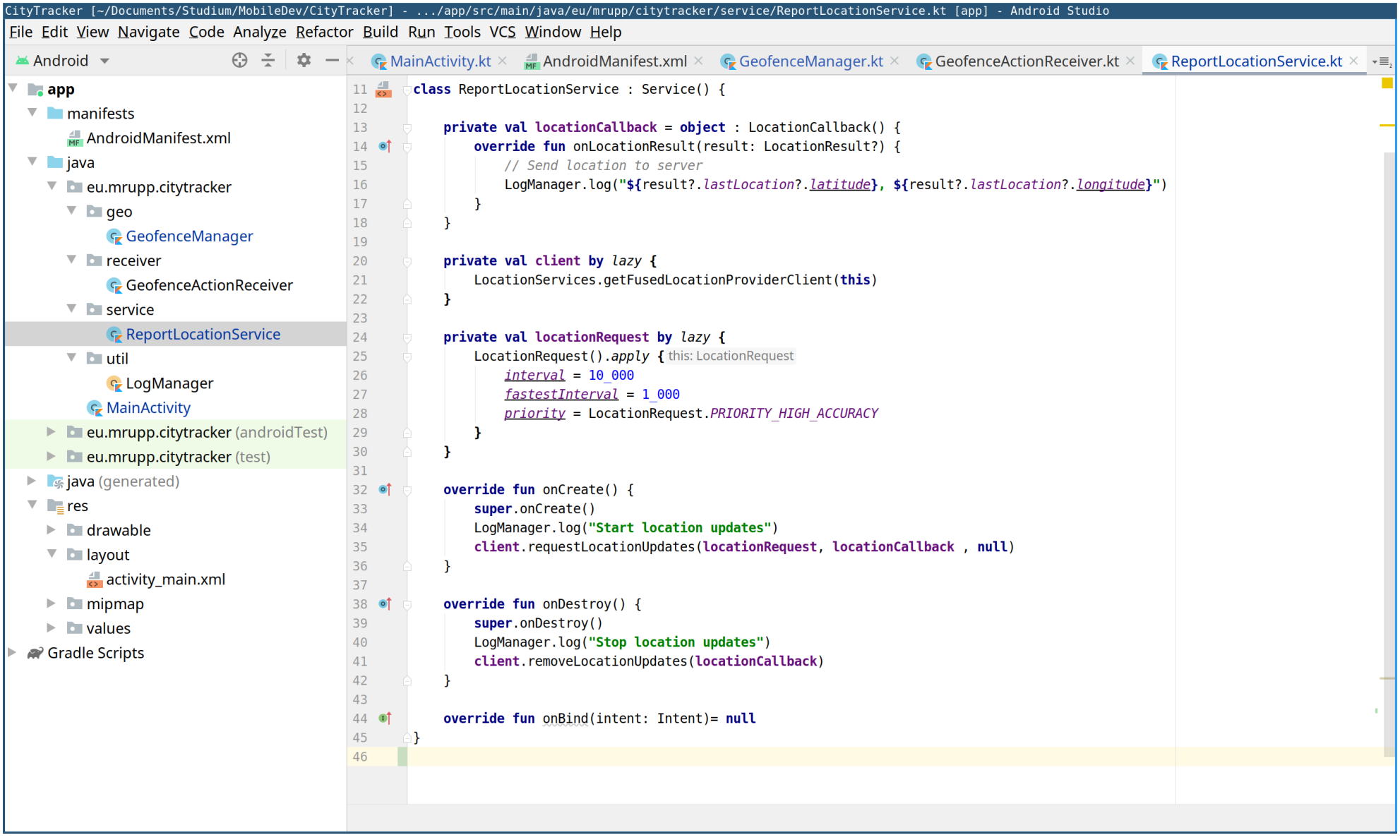


Figure: Our Geofence

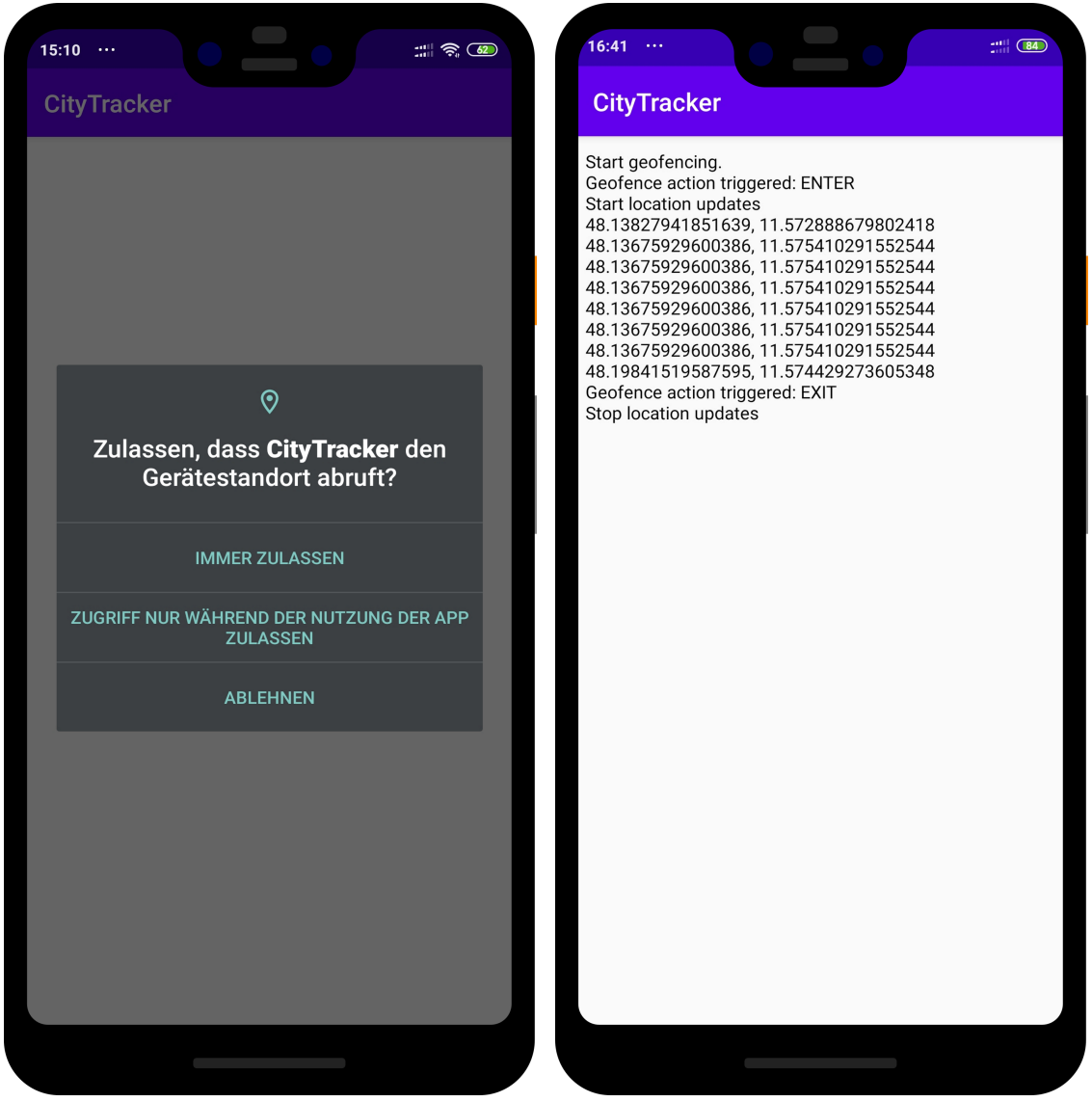

CityTracker [~/Documents/Studium/MobileDev/CityTracker] - .../app/src/main/java/eu/mrupp/citytracker/MainActivity.kt [app] - Android Studio
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Android
app
manifests
AndroidManifest.xml
java
eu.mrupp.citytracker
geo
GeofenceManager
receiver
GeofenceActionReceiver
service
ReportLocationService
util
LogManager
MainActivity
eu.mrupp.citytracker (androidTest)
eu.mrupp.citytracker (test)
java (generated)
res
drawable
layout
activity_main.xml
mipmap
values
Gradle Scripts
MainActivity.kt
AndroidManifest.xml
GeofenceManager.kt
GeofenceActionReceiver.kt
ReportLocationService.kt
class MainActivity : AppCompatActivity(), Runnable {
private val geofenceManager by lazy { GeofenceManager(context: this) }
private val requiredPermissions = arrayOf(
Manifest.permission.ACCESS_FINE_LOCATION,
Manifest.permission.ACCESS_BACKGROUND_LOCATION
)
@RequiresApi(Build.VERSION_CODES.M)
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_main)
LogManager.observe(observer: this)
if (requiredPermissions.all { it: String
checkSelfPermission(it) == PackageManager.PERMISSION_GRANTED }) {
geofenceManager.startGeofencing()
} else {
requestPermissions(requiredPermissions, requestCode: 1)
}
}
override fun onRequestPermissionsResult(requestCode: Int, permissions: Array<out String>, grantResults: IntArray) {
if (grantResults.any { it == PackageManager.PERMISSION_GRANTED }) {
geofenceManager.startGeofencing()
} else {
toastGrantPermissions()
}
}
private fun toastGrantPermissions() {
Toast.makeText(context: this,
text: "Please grant the required permissions to use the app.",
Toast.LENGTH_SHORT).show()
}
override fun run() {
MainActivity › onCreate()


CityTracker [~/Documents/Studium/MobileDev/CityTracker] - .../app/src/main/java/eu/mrupp/citytracker/geo/GeofenceManager.kt [app] - Android Studio
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Android
app
manifests
AndroidManifest.xml
java
eu.mrupp.citytracker
geo
GeofenceManager
receiver
GeofenceActionReceiver
service
ReportLocationService
util
LogManager
MainActivity
eu.mrupp.citytracker (androidTest)
eu.mrupp.citytracker (test)
java (generated)
res
drawable
layout
activity_main.xml
mipmap
values
Gradle Scripts
MainActivity.kt
AndroidManifest.xml
GeofenceManager.kt
GeofenceActionReceiver.kt
ReportLocationService.kt
class GeofenceManager(context: Context) {
private val geofenceClient by lazy {
LocationServices.getGeofencingClient(context)
}
private val pendingIntent by lazy {
val intent = Intent(context, GeofenceActionReceiver::class.java)
PendingIntent.getBroadcast(context, requestCode: 0, intent,
PendingIntent.FLAG_UPDATE_CURRENT) ^lazy
}
private val geofences = listOf<Geofence>(
Geofence.Builder()
.setRequestId("center")
.setCircularRegion(centerLat, centerLon, 1_500f)
.setExpirationDuration(Geofence.NEVER_EXPIRE)
.setTransitionTypes(
Geofence.GEOFENCE_TRANSITION_ENTER
or Geofence.GEOFENCE_TRANSITION_EXIT)
.build())
private val geofencingRequest by lazy {
GeofencingRequest.Builder()
.addGeofences(geofences)
.setInitialTrigger(0)
.build()
}
fun startGeofencing() {
geofenceClient.addGeofences(geofencingRequest, pendingIntent)
LogManager.log("Start geofencing.")
}
companion object {
private const val centerLat = 48.137966
private const val centerLon = 11.5739324
}
}
GeofenceManager


CityTracker [~/Documents/Studium/MobileDev/CityTracker] - .../app/src/main/java/eu/mrupp/citytracker/receiver/GeofenceActionReceiver.kt [app] - Android Studio
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Android
app
manifests
AndroidManifest.xml
java
eu.mrupp.citytracker
geo
GeofenceManager
receiver
GeofenceActionReceiver
service
ReportLocationService
util
LogManager
MainActivity
eu.mrupp.citytracker (androidTest)
eu.mrupp.citytracker (test)
java (generated)
res
drawable
layout
activity_main.xml
mipmap
values
Gradle Scripts
MainActivity.kt
AndroidManifest.xml
GeofenceManager.kt
GeofenceActionReceiver.kt
ReportLocationService.kt
1 package eu.mrupp.citytracker.receiver
2
3 import ...
11
12 class GeofenceActionReceiver : BroadcastReceiver() {
13
14 override fun onReceive(context: Context, intent: Intent) {
15 val event = GeofencingEvent.fromIntent(intent)
16 val serviceIntent = Intent(context, ReportLocationService::class.java)
17
18 when (event.geofenceTransition) {
19 Geofence.GEOFENCE_TRANSITION_ENTER -> {
20 LogManager.log("Geofence action triggered: ENTER")
21 context.startService(serviceIntent)
22 }
23 Geofence.GEOFENCE_TRANSITION_EXIT -> {
24 LogManager.log("Geofence action triggered: EXIT")
25 context.stopService(serviceIntent)
26 }
27 }
28 }
29 }
30
GeofenceActionReceiver


CityTracker [~/Documents/Studium/MobileDev/CityTracker] - .../app/src/main/java/eu/mrupp/citytracker/service/ReportLocationService.kt [app] - Android Studio
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Android
app
manifests
AndroidManifest.xml
java
eu.mrupp.citytracker
geo
GeofenceManager
receiver
GeofenceActionReceiver
service
ReportLocationService
util
LogManager
MainActivity
eu.mrupp.citytracker (androidTest)
eu.mrupp.citytracker (test)
java (generated)
res
drawable
layout
activity_main.xml
mipmap
values
Gradle Scripts
MainActivity.kt
AndroidManifest.xml
GeofenceManager.kt
GeofenceActionReceiver.kt
ReportLocationService.kt
class ReportLocationService : Service() {
private val locationCallback = object : LocationCallback() {
override fun onLocationResult(result: LocationResult?) {
// Send location to server
LogManager.log("${result?.lastLocation?.latitude}, ${result?.lastLocation?.longitude}")
}
}
private val client by lazy {
LocationServices.getFusedLocationProviderClient(this)
}
private val locationRequest by lazy {
LocationRequest().apply { this: LocationRequest
interval = 10_000
fastestInterval = 1_000
priority = LocationRequest.PRIORITY_HIGH_ACCURACY
}
}
override fun onCreate() {
super.onCreate()
LogManager.log("Start location updates")
client.requestLocationUpdates(locationRequest, locationCallback , null)
}
override fun onDestroy() {
super.onDestroy()
LogManager.log("Stop location updates")
client.removeLocationUpdates(locationCallback)
}
override fun onBind(intent: Intent)= null
}

App Example



(a) (b)

App

https://github.com/mobileappdevhm20/CityTracker

Summary

- Important feature for android development
- Android provides many different possibilities to implement location awareness
- Keep the battery usage in mind and use the best practices
- Pay attention to the access guidelines

References

- https://developer.android.com/training/location
- https://developers.google.com/location-context/fused-location-provider
- https://thehill.com/policy/technology/259655-quarter-of-all-android-apps-ask-for-gps-data
- https://developers.google.com/maps/documentation/android-sdk/location
- https://developer.android.com/training/location/geofencing