

7

Android Permissions and Google Maps

Activity 7.01 – Creating an app to find the location of a parked car

Solution

The following steps will help you complete the activity:

1. Create an **Empty Activity** app in Android Studio. Name it `Where Is My Car` and name its package `com.example.whereismycar`.
2. Obtain a Google Maps API key for the app from here: <https://developers.google.com/maps/documentation/embed/get-api-key>.
3. Include the secrets plugin in the project by updating the version catalog:

```
[versions]
...
secretsGradlePlugin = "[latest version]"

[libraries]
...
secrets-gradle-plugin = { module =
    "com.google.android.libraries.mapsplatform.secrets-gradle-
        plugin:secrets-gradle-plugin",
    version.ref = "secretsGradlePlugin" }
```

4. Update the buildscript block of the project build.gradle file to include the secrets plugin path:

```
buildscript {  
    dependencies {  
        classpath(libs.secrets.gradle.plugin)  
    }  
}
```

5. In the app's build.gradle file, add the secrets plugin:

```
plugins {  
    ...  
    id("com.google.android.libraries.mapsplatform  
        .secrets-gradle-plugin")  
}
```

6. At the bottom of the app's build.gradle file, configure the secrets plugin:

```
secrets {  
    propertiesFileName = "secrets.properties"  
    defaultPropertiesFileName = "local.defaults.properties"  
}
```

7. Set the passwords plugin up with the key you obtained in the last step by creating a secrets.properties file at the root of the project and updating its contents. Replace (our key) with the actual key that you obtained earlier:

```
MAPS_API_KEY=(our key)
```

8. Create an update to the local.defaults.properties fallback file at the root of the project, with the following content:

```
MAPS_API_KEY=DEFAULT_API_KEY
```

9. Update the AndroidManifest.xml file to read the API key:

```
<meta-data  
    android:name="com.google.android.geo.API_KEY"  
    android:value="${MAPS_API_KEY}" />
```

10. Add the play-services-location and google-maps-compose dependencies:

```
play-services-location = {  
    group = "com.google.android.gms",  
    name = "play-services-location",  
    version.ref = "playServicesLocation"  
}  
  
google-maps-compose = {  
    group = "com.google.maps.android",  
    name = "maps-compose",  
    version.ref = "googleMapsCompose"  
}
```

11. Include a GoogleMap composable in your layout in the MainActivity class:

```
val cameraPositionState = rememberSaveable(  
    saver = CameraPositionState.Saver  
) {  
    CameraPositionState(  
        position = CameraPosition.fromLatLngZoom(LatLng(0.0,  
            0.0), 10f)  
    )  
}  
  
GoogleMap(  
    modifier = Modifier.padding(innerPadding),  
    cameraPositionState = cameraPositionState  
)
```

12. Add a marker to the map. Label it **My Car**:

```
GoogleMap(...) {  
    Marker(  
        state = markerState,  
        title = "My Car"  
    )  
}
```

13. Introduce a button at the bottom of the screen with an **I'm parked here** label:

```
Button(
    modifier = Modifier
        .padding(vertical = 8.dp)
        .fillMaxWidth(),
    onClick = {}
) {
    Text(text = "I'm parked here")
}
```

14. Include the fused location client in the MainActivity class:

```
private val fusedLocationClient by lazy {
    LocationServices.getFusedLocationProviderClient(this)
}
```

15. Add the location permission to AndroidManifest.xml:

```
<uses-permission android:name=
    "android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name=
    "android.permission.ACCESS_FINE_LOCATION" />
```

16. Implement code to read the current user's location:

```
var currentCarLocation by remember {
    mutableStateOf(LatLng(0.0, 0.0))
}

fun setCarLocationToUserLocation() {
    val cancellationTokenSource = CancellationTokenSource()
    lifecycleScope.launch @SuppressLint("MissingPermission") {
        suspendCancellableCoroutine { continuation ->
            fusedLocationClient.getCurrentLocation(
                PRIORITY_HIGH_ACCURACY,
                cancellationTokenSource.token
            ).addOnSuccessListener { location: Location? ->
                if (location != null) {
                    val locationLatLng = LatLng(location.latitude,
                        location.longitude)
                }
            }
            continuation.resumeWithSuccess(locationLatLng)
        }
    }
}
```

```

        currentCarLocation = locationLatLng
        saveLocation(locationLatLng)
    }
}
continuation.invokeOnCancellation {
    cancellationTokenSource.cancel()
}
}
}
}
}
}

```

17. Add logic to request the user's permission to access their location. Present the rationale if needed:

```

var locationPermissionsGranted by remember {
    mutableStateOf(areLocationPermissionsGranted())
}

var shouldShowLocationPermissionRationale by remember {
    mutableStateOf(false)
}

val requestLocationPermissionLauncher =
    rememberLauncherForActivityResult(
        contract = ActivityResultContracts.RequestMultiplePermissions(),
        onResult = { permissions ->
            locationPermissionsGranted = permissions.values.all { it }
            if (locationPermissionsGranted) {
                setCarLocationToUserLocation()
            } else {
                shouldShowLocationPermissionRationale =
                    shouldShowLocationPermissionRationale()
            }
        })

fun requestLocationPermission() {
    requestLocationPermissionLauncher.launch(
        arrayOf(

```

```

        Manifest.permission.ACCESS_FINE_LOCATION,
        Manifest.permission.ACCESS_COARSE_LOCATION
    )
)
}
private fun shouldShowLocationPermissionRationale() =
    shouldShowRequestPermissionRationale(
        Manifest.permission.ACCESS_COARSE_LOCATION
    ) || shouldShowRequestPermissionRationale(
        Manifest.permission.ACCESS_FINE_LOCATION
    )

private fun areLocationPermissionsGranted(): Boolean =
    ContextCompat.checkSelfPermission(
        this,
        Manifest.permission.ACCESS_FINE_LOCATION
    ) == PackageManager.PERMISSION_GRANTED

```

18. When the button is clicked, check the location permission, request it if needed, and move the marker to the user's current location:

```

Button(
    modifier = ...,
    onClick = {
        if (locationPermissionsGranted) {
            setCarLocationToUserLocation()
        } else {
            shouldShowLocationPermissionRationale =
                shouldShowLocationPermissionRationale()
        }
        if (!locationPermissionsGranted &&
            !shouldShowLocationPermissionRationale
        ) {
            requestLocationPermission()
        }
    }
) { ... }

```

19. Add a vector car icon to your project from the Android Studio clip art options.

20. Apply the car icon to the marker:

```
val carIcon by remember {
    mutableStateOf(
        getBitmapDescriptorFromVector(
            R.drawable.car_marker
        )
    )
}
Marker(
    state = markerState,
    title = "My Car",
    icon = carIcon
)
```

21. Add functionality to move the car icon by tapping on the map:

```
GoogleMap(
    modifier = Modifier.padding(innerPadding),
    cameraPositionState = cameraPositionState,
    onMapClick = { latLng ->
        markerState.position = latLng
    }
) { ... }
```

22. Bonus step: Store the selected location in SharedPreferences. The following function, placed in your activity, will help you do this:

```
private fun saveLocation(latLng: LatLng) =
    getPreferences(MODE_PRIVATE)?.edit()?.apply {
        putString(
            "latitude", latLng.latitude.toString()
        )
        putString(
            "longitude", latLng.longitude.toString()
        )
        apply()
    }
```

23. Bonus step: Restore any saved locations from `SharedPreferences`. You can use the following function:

```
val latitude = sharedPreferences
    .getString("latitude", null)
    ?.toDoubleOrNull()
    ?: return null
val longitude = sharedPreferences
    .getString("longitude", null)
    ?.toDoubleOrNull()
    ?: return null
```

With this activity completed, you have demonstrated your understanding of requesting permissions in an Android app. You have also shown that you can present the user with a map and control pins on that map. Finally, you have also demonstrated your knowledge of obtaining the user's current location. Well done!



Important note

The solution to this activity can be found at <https://github.com/PacktPublishing/How-to-Build-Android-Apps-with-Kotlin-Third-Edition/tree/main/Chapter07/Activity07.01>.