


PAPER • OPEN ACCESS

Technologies and implementation of an integration with geolocation and maps in mobile application development

To cite this article: E R Kalyuzhny *et al* 2021 *J. Phys.: Conf. Ser.* **2094** 032029

View the [article online](#) for updates and enhancements.



The Electrochemical Society
Advancing solid state & electrochemical science & technology

241st ECS Meeting

May 29 – June 2, 2022 Vancouver • BC • Canada
Extended abstract submission deadline: Dec 17, 2021

Connect. Engage. Champion. Empower. Accelerate.
Move science forward



 Submit your abstract

Technologies and implementation of an integration with geolocation and maps in mobile application development

E R Kalyuzhny^{1,2}, V M Krasnousov^{1,2}, L V Bukreev^{1,2}, G A Shpakovsky^{1,2} and N V Zarikovskaya^{1,2}

¹ Tomsk State University of Control Systems and Radioelectronics, 40, Lenina Prospect, Tomsk, 634050, Russia

² AlderaSoft LLC, 9, block 1, Mokrushina Street, of. 3007, Tomsk, 634045, Russia

E-mail: nata.chepko@gmail.com

Abstract. This article describes native and cross-platform technologies used to implement the functionality of geolocation and geographic maps in the development of mobile applications for iOS and Android platforms. The implementation for iOS platform using Swift programming language, for Android platform using Kotlin programming language and cross-platform implementation using Flutter framework are considered in detail.

1. Introduction

Over the last period, the number of types of mobile devices has increased significantly. This contributes to the growth of the number of mobile applications developed for these devices. The following categories of mobile applications are the most popular: business applications, educational applications, specialized applications of individual stores and companies, entertainment applications, etc. Business applications from the user's side are aimed at faster and better receipt of various services, and from the business side are aimed at maximizing profits. The epidemiological situation in the world also contributed to the growth of mobile applications to obtain information and services in a timely, safe, and high-quality manner. One of the industries which received a significant leap in the category of mobile applications is tourism. These applications are aimed at providing users with opportunities for booking tickets, accommodations, tours, etc. A special place among these applications is occupied by applications capable of helping in organizing individual trips without participating in organized groups.

Such applications overwhelmingly use geolocation services to determine the user's location, implement algorithms to build travel itineraries, as well as display travel checkpoints that show the location of attractions on the map.

Now, there are two major mobile platforms in the world, iOS and Android. This work is devoted to the consideration of issues related to the implementation of mobile applications, the functionality of which directly uses geolocation and work with geographic maps for different platforms. Implementation of mobile applications for platforms iOS and Android, which have the above-described functionality, differs dramatically from each other because of different sets of development tools, which has each platform. There are two ways to implement mobile applications: native and cross-platform. Native development tools include Swift and Objective-C programming languages for iOS platform, Kotlin and Java programming languages for Android platform. When implementing cross-platform solutions Dart



programming language with Flutter framework, JavaScript programming language with ReactNative framework and others are used.

This article describes the main approaches used to implement the functionality of mobile applications related to the use of geolocation and geographical maps for different platforms.

2. Implementation for iOS

For Apple's iOS mobile platform, there is a single framework for tracking and handling a user's geolocation: CoreLocation.

CoreLocation is a framework from Apple that provides capabilities to determine the geographic location, elevation, and orientation of a device or its position relative to the nearest iBeacon device [1]. The platform collects data using all available components on the device, including Wi-Fi, GPS, Bluetooth, magnetometer, barometer, and cellular equipment.

To use the geolocation of a mobile device, you need to create an instance of the CLLocationManager class, which gets the coordinates of the mobile device through the didUpdateLocations delegate method.

In the iOS operating system, the mobile application cannot use the user's geolocation without their permission, so by calling the requestAlwaysAuthorization method on an instance of the CLLocationManager class, the operating system will send a request for permission to use the geolocation in this application. The modal window requesting permission for user geolocation is completely system wide.

After the user responds positively to the geolocation request, the mobile application can use the geolocation of the mobile device to track the user's location on the map.

The MapKit [2] framework from the iOS Software Development Kit (SDK) can be used to implement the routing on a geographic map.

To display points on the map, through which the route will pass, you need to programmatically convert the coordinates of points into an array, the elements of which will be of type MKAnnotation. Displaying these points on the map is implemented by calling the standard method showAnnotations from an instance of MKMapView class.

When you implement drawing a route using the MapKit framework, you need to implement a function that will return the required objects for drawing by passing in an array of route points. This will allow you to draw lines on the map every two waypoints.

To get information about the coordinates of points on all sections of the route we need to use the MKDirections.Request class, into which we need to pass the starting point, the end point, and the type of movement (automobile, walking or transit). Type of movement automobile implies movement by vehicle, type of movement walking implies movement on foot and type of movement transit implies movement by public transport. After passing the parameters, the calculate method must be called to calculate the route and get the MKRoute objects that contain information about the route between the two points. If you calculate the route between the two points sequentially, you can draw lines between them, and the user will get information about the length of the route.

To draw lines, you need to iterate through the MKRoute elements and get MKPolyline objects from them and pass them to the addOverlays method of the MKMapView class instance. To customize lines between points, you need to override the renderFor method of the MKMapView class instance.

The renderFor method is responsible for the customization of objects added to the map, such as MKPolyline, MKCircle, MKPolygon and so on.

By setting the desired line color and width using the renderFor method, lines between points with the specified parameters are rendered on the map. An example implementation for the iOS platform using apple's MapKit framework is shown below in figure 1.



Figure 1. Result of the iOS implementation.

3. Implementation for Android

Let's look at the implementation of geolocation and geographic map functionality for the Android platform. The Android mobile platform allows developers to use tools to determine the location of the user, which are already included in the standard Android SDK [3]. Existing standard tools allow the mobile application to get the location of the device to within a few meters.

Let's take a closer look at the implementation of the functionality of mobile applications related to the use of geolocation and geographical map for the Android platform.

Initially, when working with geolocation in Android, the application must request permission to use geolocation services from the device user. This is implemented through the Manifest file, which carries all the important information about the application, where you need to specify the permissions that may be required in the application [4].

Also, when the geolocation interface is directly invoked, permission should be requested from the user of the device. To use geolocation on Android, you need to create an instance of the FusedLocationProviderClient class, and call the standard requestLocationUpdates method, passing the following parameters to it:

- LocationRequest instance that includes location query parameters, such as detection accuracy and update rate.
- LocationCallback.onLocationResult callback function to get the result of the device location request.
- Looper object, whose message queue will be used to implement the callback mechanism.

After passing the above parameters to the method for requesting geolocation, the application can get the current geolocation, which will be passed to the callback function.

To implement the functionality of drawing routes on the map, the Android platform has a ready-made Maps SDK, which adds geographic maps to Android mobile applications, as well as the associated functionality to draw elements directly on the map provided.

The functionality of drawing a route line on the map, based on the use of the `addPolyline` method from an instance of the `GoogleMap` class. You need to pass an initialized `PolylineOptions` object to this method, which must include points attached to a specific location on the map. These points are `LatLng` object that includes latitude and longitude parameters. In addition, it is possible to set a click handler for a particular line by implementing the `GoogleMap.OnPolylineClickListener` interface and by specifying the clickable parameter to an instance of the `Polyline` class to be able to click on the line.

Drawing points on the map is done in a relatively similar script. To add a point to the map, you must create an instance of the `MarkerOptions` class, which includes many methods for configuring how to display markers on the map. For minimal work and to display a marker on the map, you can use only one method `addPosition`, which passes a `LatLng` object that includes location parameters. The existing SDK allows you to handle both line and marker clicks using the `OnMarkerClickListener` object. The `setOnMarkerClickListener` method is used to add click handling to the map. The result of the implementation of the functionality related to geolocation and geographic maps on the Android platform is shown in figure 2 below.

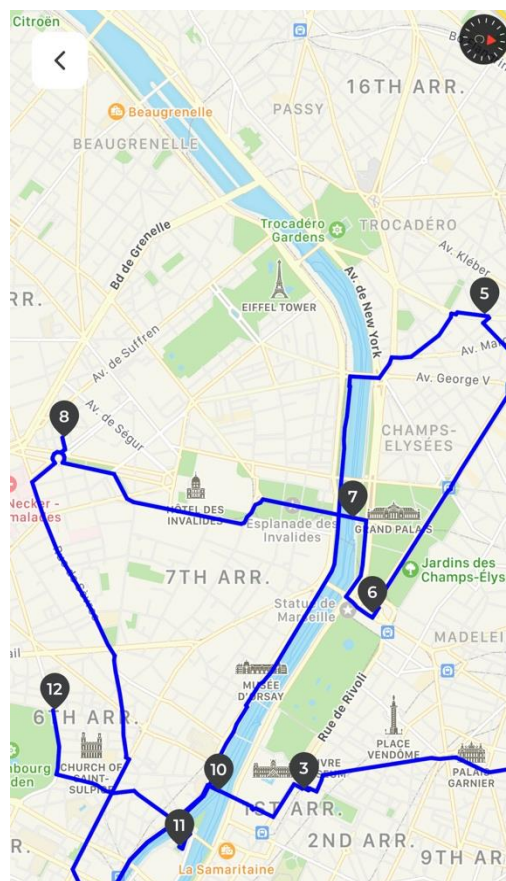


Figure 2. Result of the Android implementation.

4. Cross-platform implementation

For cross-platform implementation of mobile applications, there are many technologies. The most common among them are Flutter and ReactNative. Let's consider the implementation of functionality related to geolocation and geographical maps using the Flutter framework and Dart programming language, since this framework fully reflects all the features of the cross-platform development approach.

Flutter is a cross-platform framework developed by Google for creating applications for platforms such as iOS, Android, Web, Desktop [5]. Since Flutter is a cross-platform framework, the interaction with the native code of the platform is through plugins, which contain the native code of the platforms for which the application is created. Flutter is very actively developing and has a large number of plugins written by Google itself. In Flutter each element of the user interface, whether it's a button, text, or an entire screen is a visual component called a widget [7].

Consider the implementation of the functionality to display the user's geolocation on a geographical map and display the constructed route using the Flutter technology.

To implement this functionality, you need to include two new plugins in the project: GoogleMapsFlutter [4] and Location [5]. GoogleMapsFlutter allows you to display a geographical map on the screen, with the ability to draw lines, display points and geolocation of the user. Location will allow to request permission to use the user's geolocation in the mobile application, as well as to track it.

During direct implementation, you need to check whether geolocation services are enabled on this device using the `serviceEnable` method of the `Location` class instance, which returns true if it supports and false if it does not. Next, you must request the activation of these services using the `requestService` method, which returns true if the services are successfully activated and false if not activated. After activating geolocation services, you must request permission to use geolocation from the user using the `hasPermission` method, because geolocation in a mobile application can only be used with the user's permission. The `hasPermission` method returns the permission status, which can take the following values:

- `granted` - the user has granted permission to use geolocation with high accuracy;
- `grantedLimited` - the user has granted permission to use geolocation, but with low accuracy (relevant only for iOS version 14 and higher);
- `denied` - the user denied permission to use geolocation;
- `deniedForever` - the user permanently denied permission to use geolocation.

After a positive response (`granted` or `grantedLimited`) to the user's request for permission to use the geolocation, you can start working with it. To get the user's current geolocation, call the `getLocation` method, and to track the geolocation, subscribe to the `onLocationChanged` method, which will return all the information about the user's location.

When displaying the geographical map on the screen, we need to place the `GoogleMap` widget in the widget tree of the screen and create an instance of the `GoogleMapController` class, which will be responsible for actions on the map. After creating an instance of the `GoogleMapController` class you can start tracking the user's geolocation and center the map on it, to do this, you need to call an instance of the `Location` class instance in the `onLocationChanged` method of `GoogleMapController` class `animateCamera` method, which should pass the return value from the `CameraUpdate.newCameraPosition` method, which should pass user coordinates and level of proximity to this point.

When you implement the display points on the map you must create an array of `Marker` objects, which contain a unique marker ID (`markerId`), coordinates (`position`) and icon, and pass this array to the `GoogleMap` widget in the parameter `markers`.

If you implement drawing of lines through the points on the map you must create an array of objects `Polyline`, which contain a unique identifier of the line (`polylineId`), the visibility of the line (`visible`), the coordinates of beginning and end of the line (`points`) and the color of the line (`color`) and pass this array to the `GoogleMap` widget in the parameter `polylines`. All the above-described implementation allows you to get in a mobile application a screen with a map, which has a constructed route in the form of points connected by lines and tracking of the user's geolocation directly on the map. The result of implementing the functionality related to geolocation and geographic maps using the Flutter framework is shown in figure 3 below.

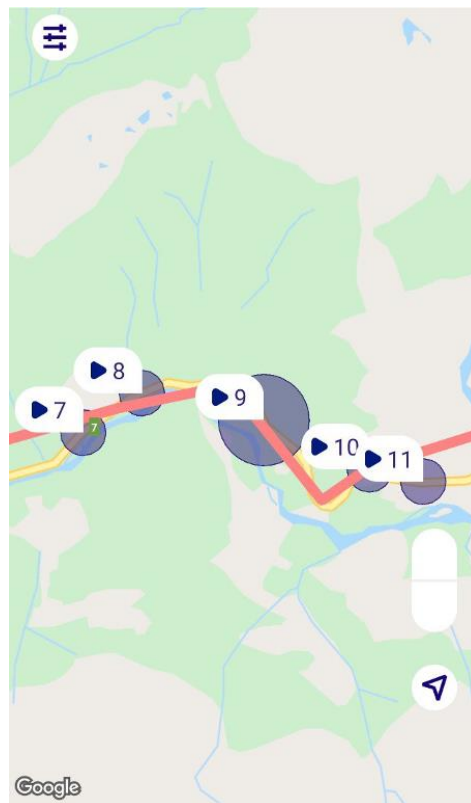


Figure 3. Result of the Flutter implementation.

5. Conclusion

The implementation of the main functionality of any mobile application is quite trivial and well represented in various articles and specialized forums. This article was devoted to the ways to implement one of the most interesting functions of modern mobile applications, such as working with geolocation and displaying routes on geographical maps. Different approaches to the implementation of these applications were considered, namely native and cross-platform.

References

- [1] Zdziarski J 2010 *iPhone SDK Application Development* (St. Petersburg: BHV-Petersburg) 506
- [2] Neuburg M 2020 *iOS 14 Programming Fundamentals with Swift: Swift Xcode and Cocoa Basics 1* (Sebastopol: O'Reilly) 708
- [3] Phillips B, Stewart C and Marsicano C 2017 *Android Programming: The Big Nerd Ranch Guide* (St. Petersburg: Piter) 688
- [4] Leiva A 2016 *Kotlin for Android Developers: Learn Kotlin the easy way while developing an Android App* (California: CreateSpace) 240
- [5] Katz M, Moore K D and Ngo V 2021 *Flutter Apprentice (First Edition): Learn to Build Cross-Platform Apps* (McGaheysville: Razeware LLC) 615
- [6] Kayfitz B 2021 *Flutter Cookbook: Over 100 proven techniques and solutions for app development with Flutter 2.2 and Dart* (Birmingham: Packt Publishing) 646
- [7] Petroustos E 2014 *Google Maps: Power Tools for Maximizing the API 1* (New York: McGraw-Hill Education) 646