Bachelor MKI

Mobile Computing

SS 2020

Prof. Dr. Natividad Martínez Madrid

- Seminar Documentation -

**Geolocation and Maps Integration**

Fanni Tamara Marosi, 764345

*Fanni\_Tamara.Marosi@Student.Reutlingen-University.De*

Ebru Selin Özcelik, 764349

*Ebru\_Selin.Oezcelik@Student.Reutlingen-University.De*  
  
  
6. Semester

Vorgelegt am: 24.06.2020

Inhaltsverzeichnis

[1. Motivation 3](#_Toc43671837)

[2. Grundlagen 3](#_Toc43671838)

[2.1 GPS 3](#_Toc43671839)

[2.2 Funktionsweise 3](#_Toc43671840)

[3. Mobile 4](#_Toc43671841)

[3.1 Standortanbieter (Provider) 4](#_Toc43671842)

[3.1.1 GPS 4](#_Toc43671843)

[3.1.2 WLAN-Information 4](#_Toc43671844)

[3.1.3 Cell Tower / Mobilfunkmasten 4](#_Toc43671845)

[3.1.4 Netzwerkanbieter 4](#_Toc43671846)

[3.1.4 Eine andere App 4](#_Toc43671847)

[3.2 Geokoordinaten 4](#_Toc43671848)

[3.2.1 Sexagesimale und dezimale Darstellung 4](#_Toc43671849)

[3.2.2 Das Location-Objekt 5](#_Toc43671850)

[4. Android’s Location API vs Google’s Location Services API 6](#_Toc43671851)

[4.1 Location Manager - Android’s Location API 6](#_Toc43671852)

[4.1.1 Location Manager initialisieren 6](#_Toc43671853)

[4.1.2 Anbieter wählen und Daten empfangen 6](#_Toc43671854)

[4.1.3 Empfänger abmelden 6](#_Toc43671855)

[4.2 Fused Location Provider API – Google Play 7](#_Toc43671856)

[4.2.1 FusedLocationProviderClient 7](#_Toc43671857)

[4.2.2 LocationRequest 7](#_Toc43671858)

[4.3 Vergleich 8](#_Toc43671859)

[5. Maps SDK für Android 10](#_Toc43671860)

[6. Eine GPS-Tracker-App 11](#_Toc43671861)

[7. Summary (Ebru & Fanni) 13](#_Toc43671862)

[Quellenverzeichnis 14](#_Toc43671863)

# 1. Motivation

Diese Ausarbeitung beinhaltet die Themengebiete Geolocation und Maps Integration in mobilen Anwendungen. Es werden verschiedene Standortprovider aufgezeigt und miteinander verglichen. Zusätzlich wird eine Variante erläutert, wie Programmierer Karten, welche auf Daten von Google Maps basieren, in ihre Anwendungen integrieren können.

Wir haben uns für das Thema Geolocation interessiert, da es sich bei unserer App hauptsächlich um eine Anwendung für die Nachbarschaft handelt. In der ersten Version ist geplant, dass der Benutzer bei der Registrierung die Stadt und Postleitzahl als Wohnort angibt. Für eine erweiterte, spätere Version haben wir überlegt, eine Positionsbestimmung über GPS-Daten einzubauen. Falls gewünscht, könnte der Benutzer den eigenen Standort bei der Registrierung über ein Geolocation-Feature bestimmen lassen.

Darüber hinaus ist für eine spätere Version geplant, dass Benutzer bei der Suche nach einer Anzeige von ihrem Wohnort ausgehend einen bestimmten Radius einstellen können, in dem gesucht werden soll. Hierfür würde es sich anbieten, die Positionsbestimmung und Map Integration einzubauen.

# 2. Grundlagen

In diesem Kapitel werden die Grundlagen von dem Global Positioning System (GPS) vorgestellt. Hierbei handelt es sich um ein globales System, bestehend aus Satelliten, welches dazu dient, die globale Position zu bestimmen [1, S.666].

## 2.1 GPS

Capderou schreibt in dem Buch „Handbook of Satellite Orbits. From Kepler to GPS”, dass das amerikanische Verteidigungsministerium ein System entwickelte, welches als Navigation Satellite Time and Ranging / Global Positioning System (Navstar/GPS) bekannt wurde. Navstar / GPS dient der globalen Positionsbestimmung. Mithilfe dieses Systems sollte das bereits bestehende System namens Transit, welches seine Grenzen erreicht hatte, abgelöst werden.

Erste Studien haben bereits seit den 1960er Jahren stattgefunden. Seit 1978 gibt es die ersten funktionsfähigen Satelliten für die Bereiche Forschung und Entwicklung [1, S. 666].

## 2.2 Funktionsweise

Das GPS-System besteht aus 24 Hauptsatelliten, welche die Erdumlaufbahn umkreisen. Die Satelliten haben eine Entfernung von 20.000 Kilometer zur Erde. Diese Entfernung wird vom Meeresspiegel gemessen. Alle Satelliten enthalten eine präzise Uhr, welche miteinander synchronisiert sind [2, S.23].

Eine Voraussetzung für das Funktionieren des Systems ist, dass an jedem Ort auf der Erde jederzeit die Daten von vier Satelliten empfangen werden können. Tobias Schüttler erläutert in dem Buch „Satellitennavigation. Wie sie funktioniert und wie sie unseren Alltag beeinflusst“, dass dies durch die Konstellation der Satelliten gewährleistet wird [3, S.46]. Die Position des Empfängers wird relativ zu dessen Entfernung zu den Satelliten ermittelt. Anhand der Position von mindestens vier Satelliten lässt sich die Position des Empfängers ableiten, indem man die Entfernung zu den Satelliten ermittelt und durch Überschneidungen die möglichen Orte immer weiter eingrenzt, bis sich die Position auf Meter genau bestimmen lässt. Je genauer die Zeitmessung und die Position des Satelliten bestimmt wird, desto genauer ist die eigene Ortung [3, S.2 ff.].

Bodenstationen überwachen die Positionen der Satelliten und sorgen dafür, dass das System wie vorgesehen funktioniert. Hierfür ist es unter anderem notwendig, die Satellitenbahnen genau zu vermessen, die eingebauten Uhren zu überwachen und mögliche Störungen zu ermitteln [3, S.44].

# 3. Mobile

Einige Anwendungen für Smartphones erfordern, dass die Daten über den Standort des Nutzers erfragt werden können. In diesem Kapitel wird aufgezeigt, welche Möglichkeiten es gibt, die Standortdaten eines Benutzers zu erhalten.

## 3.1 Standortanbieter (Provider)

In diesem Kapitel werden drei Möglichkeiten aufgezeigt, woher das Android SDK Standortdaten eines Benutzers beziehen kann. Intern werden mehrere Technologien verwendet, um den Standort eines Benutzers zu ermitteln. Welche Hardware verwendet wird, hängt vom Typ des Standortanbieters ab, der für das Erfassen der Daten ausgewählt wurde. Unter Android werden die folgenden drei Standortanbieter eingesetzt: GPS, WLAN-Netzwerke und Mobilfunkmasten.

### 3.1.1 GPS

Android bietet Programmierern die Möglichkeit die Lokalisierung eines Geräts über GPS oder Netzwerke vorzunehmen. Der große Vorteil der GPS-Variante besteht in dessen Genauigkeit. Jedoch lassen sich hier auch einige Nachteile feststellen. GPS funktioniert nicht immer zuverlässig in Gebäuden. Die Antwortzeit nach der ersten Anfrage kann länger dauern. Der Stromverbrauch des Geräts ist ziemlich hoch, wenn der GPS-Sensor aktiviert ist [4].

### 3.1.2 WLAN-Information

Bei der zweiten Variante wird die Position des Geräts über ein Netzwerk oder Funkzellen des Mobilfunknetzes bestimmt. Die Bestimmung der Koordinaten ist jedoch ungenauer als bei der GPS-Variante. Vor allem in flachen Gebieten, in denen die Funkzellen größer sind. Als ein Vorteil lässt sich aufführen, dass die Positionierung über Netzwerke innerhalb von Gebäuden möglich ist. Des Weiteren ist die Antwortzeit kürzer und der Akkuverbrauch nicht so hoch wie bei der GPS-Ortung [4].

### 3.1.3 Cell Tower / Mobilfunkmasten

### 3.1.4 Netzwerkanbieter

Dieser Anbieter kombiniert WLAN, Mobilfunkdaten und einschließlich der von Funktürmen erfassten GPS-Daten. Der Netzwerkanbieter verbraucht weniger Strom als der GPS-Anbieter, gibt aber Standortdaten von unterschiedlicher Genauigkeit zurück Quelle?.

### 3.1.4 Eine andere App

## 3.2 Geokoordinaten

### 3.2.1 Sexagesimale und dezimale Darstellung

Eine geografische Angabe der Position besteht aus einem Längen- und einem Breitengrad. Die Längengrade werden von 0 bis 180 Grad östlich oder westlich von Greenwich in England definiert. Die Breitengrade werden 0 bis 90 Grad nördlich oder südlich ausgehend vom Äquator definiert. Ein Grad besteht aus 60 Winkelminuten, eine Winkelminute wiederrum aus 60 Winkelsekunden. Winkelminuten werden mit einem Apostroph, Winkelsekunden mit zwei Apostrophen abgekürzt.

Für die Darstellung der Position wird im Computer die Dezimalschreibweise genutzt, da diese für den Computer praktischer ist. In dem Location-Objekt, welches in Kapitel 3.2.2 erläutert wird, werden der Längen - und Breitengrad als double-Werte gespeichert.

Louis und Müller zeigen zu der Dezimalschreibweise ein Beispiel in dem Buch „Android“ auf. Für die Golden Gate Bridge wird der Breitengrad in der Dezimalschreibweise als 37.816667 und der Längengrad als -122.483333 definiert. Die Gradzahlen werden als Vorkommastellen dargestellt, die Nachkommastellen sind die Minuten und Sekunden. Anhand des Vorzeichens lässt sich bestimmen, um welche Richtung es sich handelt. Norden und Osten werden durch ein positives Vorzeichen, Süden und Westen durch ein negatives repräsentiert [4].

### 3.2.2 Das Location-Objekt

Bei dem Location-Objekt handelt es sich um eine Instanz der Klasse Location. Diese Klasse repräsentiert einen geografischen Standort. Das Android SDK stellt Programmierern diese Klasse in dem Paket android.location.Location zur Verfügung. Ein Location-Objekt kann Attribute wie einen Längen- und Breitengrad, Zeitstempel und die Angabe der Höhe besitzen.

Android garantiert, dass alle Location-Objekte, welche von einem LocationManager erzeugt werden, über einen gültigen Breiten- und Längengrad und Zeitstempel verfügen. Der LoactionManager wird in dem nachfolgenden Kapitel 4.1 vorgestellt.

Location-Objekte verfügen über einige public Methoden. Einige von ihnen werden in der nachfolgenden Tabelle 1 aufgezeigt. Die Auswahl erfolgte anhand der subjektiven Einschätzung, wie häufig sie genutzt werden.[[1]](#footnote-1)

Tabelle 1: Public Methoden des Location-Objekts. Quelle: Eigene Darstellung auf Basis 1

|  |  |  |
| --- | --- | --- |
| **Methodenname/-signatur** | **Rückgabetyp** | **Beschreibung** |
| getAltitude() | double | Mithilfe dieser Methode lässt sich die Höhe (falls vorhanden) ermitteln. Gemessen wird sie in Metern in dem Referenzsystem „World Geodetic System 1984“ (WGS 84). |
| getLatitude() | double | Der Rückgabewert dieser Methode ist der Breitengrad des geografischen Standorts (gemessen in Grad). |
| getLongitude() | double | getLongitude() liefert den Längengrad des geografischen Standorts zurück. Dieser wird in Grad gemessen. |
| distaneTo(dest: Location!) | float | Diese Methode ermöglicht es, die ungefähre Entfernung zwischen dem Location-Objekt, auf welchem sie aufgerufen wird und dem als Parameter übergebenen Location-Objekt zu bestimmen. |
| getTime() | long | An dieser Stelle wird ein Zeitstempel (UTC) zurückgegeben. Es handelt sich um die Anzahl der Millisekunden, welche seit dem 01.01.1970 vergangen sind. |

# 4. Android’s Location API vs Google’s Location Services API

Das Arbeiten mit Standorten ist eine der häufigsten Aufgaben im Entwicklungsprozess mobiler Apps. Während der Entwicklung stellen sich daher häufig Fragen, was zu verwenden ist, um genauere Ergebnisse, maximale Betriebsstabilität und einfache Implementierung zu erzielen. In diesem Kapitel werden zwei APIs, Android Location Manager und Google Fused Location Provider, vorgestellt.

## 4.1 Location Manager - Android’s Location API

Der LocationManager ist eine Klasse im Android SDK. Mithilfe des LocationManagers bekommt man Zugang zu den Ortungsservices des Systems, auf welchem die Anwendung ausgeführt wird. Die Anwendung kann periodisch den Standort des Geräts abfragen oder eine Benachrichtigung erhalten, wenn sich der Nutzer in der Nähe eines angegebenen Standorts befindet.[[2]](#footnote-2)

### 4.1.1 Location Manager initialisieren

Es gibt in Android bereits eine Klasse für die Bestimmung der Position des Geräts. Diese Klasse heißt LocationManager und befindet sich im android.hardware-Paket. Ein Objekt dieser Klasse kann vom Programmierer in einer Activity wie folgt initialisiert werden:

LocationManager locationManager = (LocationManager) this.getSystemService(Context.LOACTION\_SERVICE) [4].

### 4.1.2 Anbieter wählen und Daten empfangen

Zuerst teilt man dem LocationManager mit, mit welchem Provider die Daten empfangen werden sollen. Hierzu braucht man eine Klasse, die das Interface LoactionListener implementiert. Diese Klasse lässt sich verwenden, um sich beim LocationManager zum Empfang von Geodaten anzumelden.

//Registrieren für GPS-Updates nach jeder Minute

locationManager.requestLocationUpdates(LocationManager.GPS\_PROVIDER, 6000, 0, this);

// Registrieren für Netzwerk-Updates nach jeder Minute und alle 10 sekunden

locationManager.requestLocationUpdates(LocationManager.NETWORK\_PROVIDER, 6000, 10, this);

Die Methode requestLocationUpdates erwartet neben dem Providertyp und dem zu benachrichtigenden Listener (this) noch zwei Parameter, minTime und minDistance.

minTime zeigt an, nach wie vielen Millisekunden ein Update mit Positionsdaten gesendet werden soll.

minDistance besagt, dass ein Update nur dann gesendet werden soll, wenn sich die Position seit dem letzten Update um mindestens die angegebene Distanz verändert hat.

### 4.1.3 Empfänger abmelden

Eine Activity muss mit requestLocationUpdates() für den Empfang von Geolokationsdaten beim LocationManager an- und abgemeldet werden. Sobald kein Bedarf mehr an Geolokationsdaten besteht, sollte sich die Activity mit der Methode removeUpdates(LocationListener) abmelden [4].

## 4.2 Fused Location Provider API – Google Play

Der Fused Location Provider ist eine Standort-API, welche von Google Play-Diensten angeboten wird.

Der Fused Location Provider bietet die Möglichkeit, verschiedene Signale von unterschiedlichen Providern, wie GPS und WLAN-Netzwerke, zu kombinieren und die Positionsbestimmung vorzunehmen. Die API entscheidet automatisch, von welchem Anbieter, die Geodaten angefordert werden sollen. So lassen sich zum Beispiel möglichst genaue Positionsdaten bestimmen oder die bestmöglichen mit möglichst niedrigem Akkuverbrauch.[[3]](#footnote-3)

Außerdem kann man erweiterte Funktionen wie Geofencing verwenden. Um die Standortdienste von Google nutzen zu können, muss die App eine Verbindung zum Google Play Services Client herstellen.

Das System kann Positionsdaten von mehreren Providern erhalten und die Ergebnisse aus mehreren Quellen in einem einzigen Standortobjekt zusammenführen. Geodaten mit der folgenden Berechtigungen: ACCESS\_COARSE\_LOCATION und nicht ACCESS\_FINE\_LOCATION werden automatisch verkleinert, um nur eine grobe Genauigkeit anzuzeigen.[[4]](#footnote-4)

### 4.2.1 FusedLocationProviderClient

Der FusedLocationProviderClient ist der Haupteinstiegspunkt für die Interaktion mit den Fused-Location-Anbietern. Ohne den Client kann man keine Geodaten von den Anbietern anfordern, nach Aktualisierungen oder Callbacks starten[[5]](#footnote-5).

### 4.2.2 LocationRequest

public final class LocationRequest implementiert Parcelable

Die LocationRequest ist ein Datenobjekt, welches Parameter für Anfragen an die FusedLocationProviderApi enthält. Mithilfe der LocationRequest-Objekte lassen sich Standortaktualisierungen von Providern über die FusedLocationProviderApi anfordern.

Für den Fall, dass die Anwendung Positionsdaten mit einer hohen Genauigkeit benötigt, kann die Anfrage der Geodaten mit setPriority auf PRIORITY\_HIGH\_ACCURACY und setInterval auf 5 Sekunden eingestellt werden. Diese Einstellungen eignen sich für Anwendungen, wie zum Beispiel Mapping-Apps, welche den Standort des Benutzers in Echtzeit aktualisieren möchten.

Mit setPriority auf PRIORITY\_NOPOWER lässt sich eine Standortanfrage erzeugen, welche einen niedrigeren Akkuverbrauch aufweist. Diese Anweisung fragt nicht selbst nach Updates von Standortdaten, sondern erhält Informationen über den Standort des Benutzers von anderen Anwendungen, welche die Standortdaten erfragen. Die PRIORITY\_NOPOWER-Anweisung eignet sich für Anwendungen, welche den Standort nicht unbedingt benötigen, aber dennoch davon profitieren können, falls dieser erhalten wird.

PRIORITY\_BALANCED\_POWER\_ACCURACY eignet sich für Anwendungen, welche Updates periodisch empfangen und dennoch keinen allzu hohen Akkuverbrauch herbeiführen möchten. Mit setFastestInterval(long) lassen sich kurze Intervalle erzeugen, mit setInterval(long) längere wie zum Beispiel 60 Minuten berücksichtigen.4

Die App kann weiterhin Standorte empfangen, die von anderen Anwendungen ausgelöst wurden, aber berücksichtigt Hauptsache die anderen Anbieter. Dieser Anforderungsstil eignet sich für viele standortbezogene Anwendungen, einschließlich die die im Hintergrund verwendet werden.

## 4.3 Vergleich

Tabelle 2: Allgemeiner Vergleich Google Location API Services und Android Location Manager. Quelle: Eigene Darstellung auf Basis [[6]](#footnote-6)

|  |  |  |
| --- | --- | --- |
| **API** | **Google Location API Services** | **Android Location Manager** |
| **Framework** | Baut auf dem Android-Framework auf. | Teil des Android-Frameworks |
| **Leistung und Stabilität** | Hoch | Mittel |
| **Batterie** | Bietet weniger Batterieentlastung "out of the box". | Die Batterieentladung ist ein Problem, wenn sie nicht ordnungsgemäß verwaltet wird. |
| **Verfügbarkeit** | Nur an Geräten mit Google Play Services verfügbar. | In allen Geräten verfügbar |
| **Werte** | 7 Dezimalwerte | 14 Dezimalwerte |
| **Anbieter** | Es kombiniert automatisch verschiedene Signale, um die Standortinformationen bereitzustellen. Verwendet alle drei Anbieter. | Erfordert eine Logik zum Wechseln der Standortanbieter, wenn das Gerät keinen Standort finden kann. Verwendet alle drei Anbieter. |
| **Kontrolle** | Weniger genaue Kontrolle über GPS. | Feinere Kontrolle. |

Tabelle 3: Vergleich Google Location API Services und Android Location Manager bezüglich des GPS-Providers. Quelle: Eigene Darstellung auf Basis 6

|  |  |  |
| --- | --- | --- |
| **API** | **Google Location API Services** | **Android Location Manager** |
| **Anbieter - GPS** | **PRIORITY\_HIGH\_ACCURACY** | GPS\_PROVIDER |
| **Erforderliche Berechtigungen** | ACCESS\_FINE\_LOCATION für einen genaueren Standort oder ACCESS\_COARSE\_LOCATION für einen weniger genauen Standort | ACCESS\_FINE\_LOCATION |
| **Genauigkeit** | 10m - 100m | 10m - 100m |
| **Leistungsbedarf** | Hoch | Hoch |
| **Verfügbarkeit** | Dort, wo Google Play-Dienste verfügbar sind (nicht in China). | Weltweit (mit freier Sicht zum Himmel) |
| **Standortaktualisierung** | In der Regel einmal alle 5 Sekunden, aber die Werte können geändert werden. | In der Regel einmal pro Sekunde |
| **Bemerkung** | Erhält zwar (möglicherweise) genauere Standortaktualisierungen als die anderen Einstellungen, ist aber dennoch anfällig für den "Urban Canyon" -Effekt. | Sollte keine freie Sicht zum Himmel vorhanden sein, kann es passieren, dass GPS-Punkte nicht sehr gut gebündelt (Positionspunkte "springen"). Die Genauigkeit kann aufgrund des "Urban Canyon" -Effekts in bestimmten Bereichen irreführend sein. |

Tabelle 4: Vergleich Google Location API Services und Android Location Manager bezüglich des Netzwerk-Providers. Quelle: Eigene Darstellung auf Basis 6

|  |  |  |
| --- | --- | --- |
| **API** | **Google Location API Services** | **Android Location Manager** |
| **Anbieter – Netzwerk (WLAN, Mobilfunk)** | **PRIORITY\_BALANCED\_POWER\_ACCURACY oder PRIORITY\_LOW\_POWER** | NETWORK\_PROVIDER |
| **Erforderliche Berechtigungen** | ACCESS\_FINE\_LOCATION für einen genaueren Standort oder ACCESS\_COARSE\_LOCATION für einen weniger genauen Standort | ACCESS\_COARSE\_LOCATION oder ACCESS\_FINE\_LOCATION |
| **Genauigkeit** | 100m - 1000m + | 100m - 1000m + |
| **Leistungsbedarf** | Niedrig - Mittel | Niedrig- Mittel |
| **Verfügbarkeit** | Überall, wo Google Play-Dienste verfügbar sind (nicht in China). In Reichweite des Zellenturm- oder WLAN-Signals. | In Reichweite des Zellenturm- oder WLAN-Signals |
| **Standortaktualisierung** | Gleicher Wert wie PRIORITY\_HIGH\_ACCURACY | Wird seltener als GPS ausgeführt |
| **Bemerkung** | **PRIORITY\_BALANCED\_POWER\_ACCURACY: E**s ist zwar unwahrscheinlich, dennoch kann diese Einstellung GPS verwenden, um einen Standort zu erzeugen. **PRIORITY\_LOW\_POWER**: Verwendet wahrscheinlich kein GPS-Signal. Wird im Allgemeinen zum Erkennen signifikanter Standortänderungen verwendet. | Die Genauigkeit hängt von einer Anzahl verschiedener Faktoren ab. Diese Faktoren sind beispielsweise die Anzahl der WLAN-Signale, die Signalstärke oder der Typ des Zellturms. |

Tabelle 5: Vergleich Google Location API Services und Android Location Manager, wenn Standortdaten von einer anderen App erhalten werden. Quelle: Eigene Darstellung auf Basis 6

|  |  |  |
| --- | --- | --- |
| **API** | **Google Location API Services** | **Android Location Manager** |
| **Anbieter – andere App** | **PRIORITY\_NO\_POWER** | PASSIVE\_PROVIDER |
| **Erforderliche Berechtigungen** | ACCESS\_FINE\_LOCATION für einen genaueren Standort oder ACCESS\_COARSE\_LOCATION für einen weniger genauen Standort | ACCESS\_FINE\_LOCATION |
| **Genauigkeit** | 10m - 1000m + | 10m - 1000m + |
| **Leistungsbedarf** | Keine | Keine |
| **Verfügbarkeit** | Überall dort, wo Google Play-Dienste verfügbar sind (nicht in China). | Nur wenn eine andere App einen Standort über GPS oder ein Netzwerk empfängt. |
| **Standortaktualisierung** | Es werden keine Standorte zurückgegeben, es sei denn, ein anderer Client hat Standortaktualisierungen durch Google Play-Dienste angefordert. In diesem Fall fungiert diese Anforderung als passiver Listener für diese Standorte. | Es erfolgen keine ständigen Aktualisierungen. Dieser hört passiv auf andere Apps, die Standortanfragen stellen, und gibt diese Orte zurück. |
| **Bemerkung** |  | Gibt keine von FusedLocationProviderApi generierten Daten zurück. |

https://riptutorial.com/de/android/topic/1837/ort

Wenn man die Google Location API Services mit LocationManager vergleicht, stellt man eine deutlich höhere Leistung und Stabilität während der Verwendung fest.

Es gab ein häufiges Problem damit, dass LocationManager frühere Standorte oder Kartenaktualisierungen erhielt. Wenn Parameter Mindestzeitraum und Mindestentfernung für spätere Aktualisierungen festgelegt wurden, einer der Parameter wurde immer ignoriert. In den Google Location API Services wurde dieses Problem nicht festgestellt. Außerdem verbrauchen Google Location Services weniger Batterie. Infolgedessen schnitten die Google Location API Services in allen Fällen besser ab und die Benutzung von der Google Location API wird auch eher empfohlen.

Es gibt jedoch auch einige Nachteile. Um die Ortungsdienste nutzen zu können, sollte man Google Play Services installieren. Daher funktioniert die Anwendung auf Geräten mit üblicher Firmware, die von Google nicht unterstützt werden oder auf denen Google Services nicht verfügbar sind, nicht. Zum Beispiel in China aufgrund der nationalen Politik sind diese nicht verfügbar, und dies ist ein großer Teil des Marktes, der nicht ignoriert werden kann. In diesem Fall muss man sich an LocationManager wenden. Also solange der Nutzer Google Play Services nicht installiert hat, LocationManager ist die einzige Lösung.

https://appus.software/blog/difference-between-locationmanager-and-google-location-api-services

# 5. Maps SDK für Android

Mit dem Maps SDK for Android kann man zu einer App Karten hinzufügen, die auf Google Maps-Daten basieren. Die API verwaltet den Zugriff auf Google Maps-Server, das Herunterladen von Daten, die Kartendarstellung und die Reaktion auf Gesten zur Kartensteuerung automatisch. Mithilfe von API-Aufrufen kannst du einer Basiskarte auch Markierungen, Polygone und Overlays hinzufügen und die Nutzeransicht eines bestimmten Kartenbereichs ändern. Diese Objekte bieten zusätzliche Informationen zu Standorten auf der Karte und ermöglichen die Interaktion mit der Karte. Mithilfe der API kannst du einer Karte folgende Grafikelemente hinzufügen:

* Symbole, die mit bestimmten Positionen auf der Karte (Markierungen) verknüpft sind
* Gruppen von Liniensegmenten (Polylinien)
* Umschlossene Segmente (Polygone)
* Bitmap Grafiken, die mit bestimmten Positionen auf der Karte verknüpft sind (Boden-Overlays)
* Bildgruppen, die oben auf den Kacheln der Basiskarte dargestellt werden (Kachel-Overlays)

(nur die Beschreibung aus der Android-Doku, ich muss es noch bearbeiten ☺ )

https://developers.google.com/maps/documentation/android-sdk/intro?hl=de

# 6. Eine GPS-Tracker-App

1. Google Play Service öffnen ->
   1. <https://developers.google.com/android/guides/setup>
   2. -> Google Location and Activity Recognition suchen
   3. -> API zu build.gradle (:app) hinzufügen

dependencies **{** implementation fileTree(dir: 'libs', include: ['\*.jar'])  
  
 implementation 'androidx.appcompat:appcompat:1.1.0'  
 implementation 'androidx.constraintlayout:constraintlayout:1.1.3'  
 testImplementation 'junit:junit:4.12'  
 androidTestImplementation 'androidx.test.ext:junit:1.1.1'  
 androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'

// API hinzufügen  
 **implementation 'com.google.android.gms:play-services-location:17.0.0'**

**}**

1. LocationRequest initialisieren

public class MainActivity extends AppCompatActivity {  
  
 private static final int *PERMISSION\_FINE\_LOCATION* = 99;  
 TextView tv\_lat, tv\_lon, tv\_altitude, tv\_accuracy, tv\_speed, tv\_sensor, tv\_updates, tv\_address;  
 Switch sw\_locationsupdates, sw\_gps;

//LocationRequest deklarieren ☺

//LocationRequest = locationRequest;

**<…>**  
   
   
 final int DEFAULT\_UPDATE\_INTERVAL = 30;  
 final int FAST\_UPDATE\_INTERVAL = 5;

private void updateGPS(){

// FusedLocationProviderClient initialisieren

// = LocationServices.*getFusedLocationProviderClient*(MainActivity.this);  
  
 fusedLocationProviderClient = **<…>**

if(ActivityCompat.*checkSelfPermission*(this, Manifest.permission.*ACCESS\_FINE\_LOCATION*) == PackageManager.*PERMISSION\_GRANTED*){  
 fusedLocationProviderClient.getLastLocation().addOnSuccessListener(this, new OnSuccessListener<Location>(){  
 @Override  
 public void onSuccess(Location location) {  
 updateUIValues(location);  
 }  
 });  
 }  
 else{  
 if(Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*M*){  
 requestPermissions(new String[] {Manifest.permission.*ACCESS\_FINE\_LOCATION*}, *PERMISSION\_FINE\_LOCATION*);  
 }  
 }}

sw\_gps.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if(sw\_gps.isChecked()){

//Anbieter GPS ☺

//LocationRequest.*PRIORITY\_HIGH\_ACCURACY*

locationRequest.setPriority(LocationRequest.**<…>**);  
 tv\_sensor.setText("Using GPS sensors");  
 }else{

//Anbieter Netzwerk ☺

//LocationRequest.*PRIORITY\_BALANCED\_POWER\_ACCURACY*

locationRequest.setPriority(LocationRequest.**<…>**);  
 tv\_sensor.setText("Using Cell Towers or Wifi");  
 }  
 }  
});

3.FusedLocationProviderClient initialisieren

Android Studio GPS location tracker tutorial 01: https://www.youtube.com/watch?v=V62sxpyxapU

# 7. Summary (Ebru & Fanni)

# Quellenverzeichnis

[ 1 ] Capderou, M.: Handbook of Satellite Orbits. From Kepler to GPS. Springer, 2014. DOI 10.1007/978-3-319-03416-4

[ 2 ] Doberstein, D.: Fundamentals of GPS Receivers. A Hardware Approach. Springer, 2012. DOI 10.1007/978-1-4614-0409-5

[ 3 ] Schüttler, T.: Satellitennavigation. Wie sie funktioniert und wie sie unseren Alltag beeinflusst. Springer, 2014. DOI 10.1007/978-3-642-53887-2

[ 4 ] Louis, D., Müller, L.: Android, Kapitel 17. Hanser, 2016. DOI 10.3139/978-3-446-45112-4

1. https://developer.android.com/reference/kotlin/android/location/Location [↑](#footnote-ref-1)
2. https://developer.android.com/reference/kotlin/android/location/LocationManager [↑](#footnote-ref-2)
3. https://developers.google.com/location-context/fused-location-provider [↑](#footnote-ref-3)
4. <https://developers.google.com/android/reference/com/google/android/gms/location/LocationRequest> [↑](#footnote-ref-4)
5. https://developers.google.com/android/reference/com/google/android/gms/location/FusedLocationProviderClient [↑](#footnote-ref-5)
6. https://riptutorial.com/de/android/topic/1837/ort [↑](#footnote-ref-6)