



# Programming Android With External API

mederic.hurier@uni.lu



# Agenda

- Introduction
- API Oriented
- Google Sign In API
- OpenStreetMaps API
- Tutorial

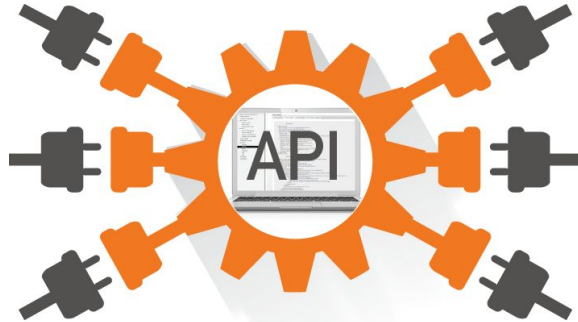


# Introduction

# Definition

An API (Application Programming Interface) is a way of communicating with a particular program

An API can be viewed as a set of rules that defines how a software can request/receive information



Source: <https://dictionary.cambridge.org/dictionary/english/api>



# An abstract concept ...

It seems an API can be ANYTHING (e.g. language, library, operating system, rendering service ...)

Remember that an API is about shape and effect, not about content (think documentation / test)

```
words :: String -> [String]
```

`words` breaks a string up into a list of words, which were delimited by white space.

```
>>> words "Lorem ipsum\dolor"
["Lorem","ipsum","dolor"]
```

```
unlines :: [String] -> String
```

`unlines` is an inverse operation to `lines`. It joins lines, after appending a terminating newline to each.

```
>>> unlines ["Hello", "World", "!"]
"Hello\nWorld\n!\n"
```

```
unwords :: [String] -> String
```

`unwords` is an inverse operation to `words`. It joins words with separating spaces.

```
>>> unwords ["Lorem", "ipsum", "dolor"]
"Lorem ipsum dolor"
```

```
def "HashMap accepts null key"() {
  given:
    def map = new HashMap()

  when:
    map.put(null, "elem")

  then:
    notThrown(NullPointerException)
}
```



# Examples of well known APIs

- **POSIX:** for software compatibility with variants of Unix and other operating systems
- **ODBC:** for accessing database independent of the underlying database/operating systems
- **OpenGL:** for rendering 2D and 3D vector graphics and accessing hardware accelerated rendering
- **Google Maps:** for embedding maps into an external website and provide additional services (GPS)

On Android, you can specify the API level in your project *build.gradle* file (`targetSdkVersion`)

Test, Support, Material, Wearable, Billing, Things, Constraint, Architecture, AndroidX ...

Source: <https://developer.android.com/reference/>

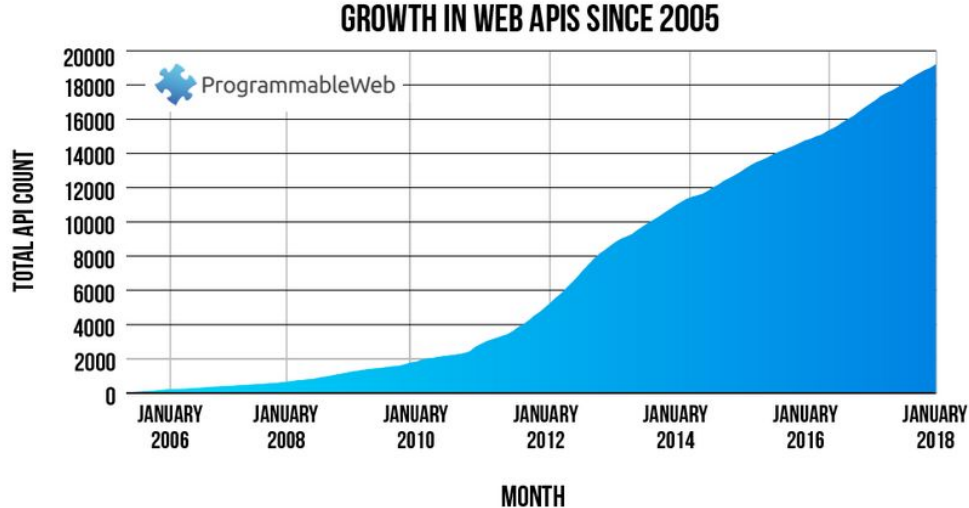


# Internal vs. External vs. Underlying vs. Online

<b>Internal API</b>  An API readily available to your application e.g. Android, Java, JDBC ...	<b>External API</b>  An API that comes from an foreign source e.g. JUnit, Espresso, Mockito ...
<b>Underlying API</b>  An API used under the surface of an other e.g. POSIX, libc, ssl ...	<b>Online API</b>  An API that communicates with the Internet e.g. Google Maps, Twitter, LinkedIn ...

# Where can we find external and online API ?

You can browse online providers (e.g. [github.com](https://github.com)) or look at API directory (e.g. [programmableweb.com](https://programmableweb.com))







# Battle for Java API: Oracle vs. Google (Android)

“A federal appeals court ruled that Google violated Oracle's copyrights when it built a custom version of the Java platform for its Android operating system.”

“The case revolves around what are called application programming interfaces, or APIs.”

“The court sent the case back to a district court to decide how much Google should pay Oracle.”

“Google switched to a fully open source version of Java starting with the Nougat release in 2016.”

“But it could force many software companies to rewrite parts of their products, even if they’re not using Java”

Source: <https://www.wired.com/story/the-case-that-never-ends-oracle-wins-latest-round-vs-google/>



# Benefits of using API

1. You don't have to reinvent the wheel (e.g. how to read a file, how to put a map on a screen)
2. You can focus on the added value of your application (e.g. bike location, cooking recipes ...)
3. API are easy to integrate, either as project dependencies or with direct HTTP requests
4. They can be monetized if you want to provide external access (e.g. API call per month)
5. They allow to create API Oriented System to enable a modular project architectures
6. They provide common building blocks for programmers (e.g. authentication, crypto ...)
7. They create an abstraction over a particular code implementation (e.g. standard library)
8. API makes you a hAPI programmer: <https://github.com/trending> <https://www.w3.org>

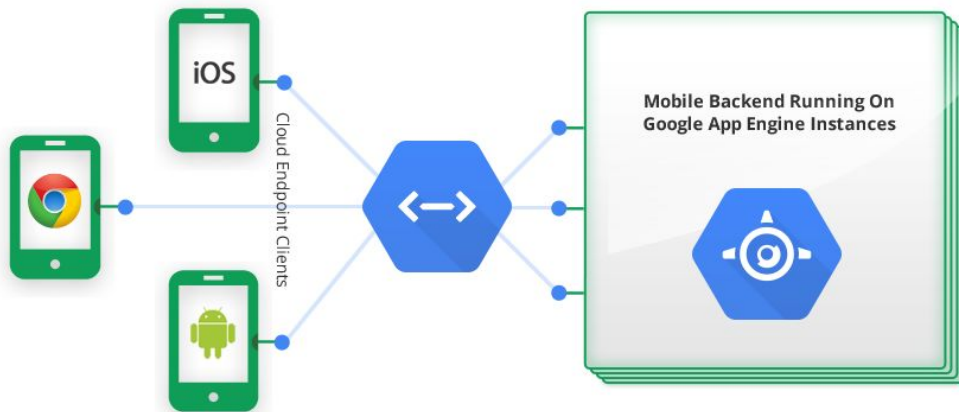


# API Oriented

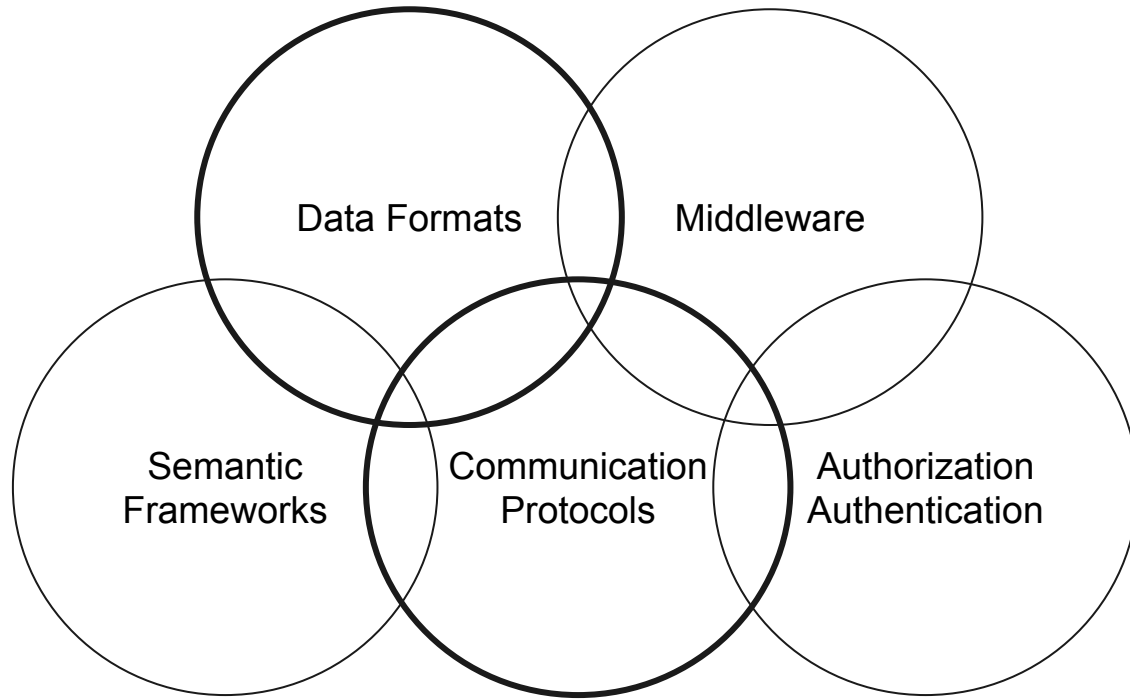
# Introduction

The service we create must be accessible to more and more devices: web, mobile, internet of things ...

In this context, it makes sense to put as much code on the server and let the client handle user interactions



# Main Building Blocks of an API Oriented System





# Data Formats

A data format specifies the syntax and structure of information you want to store or exchange

Common API Formats: JSON, XML, CSV, Protocol Buffer / human readable ? easy to parse ? ...

```
{ "menu": {  
  "id": "file",  
  "value": "File",  
  "popup": {  
    "menuitem": [  
      { "value": "New", "onclick": "CreateNewDoc()" },  
      { "value": "Open", "onclick": "OpenDoc()" },  
      { "value": "Close", "onclick": "CloseDoc()" }  
    ]  
  }  
}}
```

```
<menu id="file" value="File">  
  <popup>  
    <menuitem value="New" onclick="CreateNewDoc()" />  
    <menuitem value="Open" onclick="OpenDoc()" />  
    <menuitem value="Close" onclick="CloseDoc()" />  
  </popup>  
</menu>
```

# Communication Protocols

A communication protocol specifies the method of interaction between two systems (client/server, P2P)





# Example: REST (Representational State Transfer)

RESTful API HTTP methods

Resource	GET	PUT	POST	DELETE
Collection URI, such as <code>http://api.example.com/resources/</code>	<b>List</b> the URIs and perhaps other details of the collection's members.	<b>Replace</b> the entire collection with another collection.	<b>Create</b> a new entry in the collection. The new entry's URI is assigned automatically and is usually returned by the operation. <sup>[10]</sup>	<b>Delete</b> the entire collection.
Element URI, such as <code>http://api.example.com/resources/item17</code>	<b>Retrieve</b> a representation of the addressed member of the collection, expressed in an appropriate Internet media type.	<b>Replace</b> the addressed member of the collection, or if it does not exist, <b>create</b> it.	Not generally used. Treat the addressed member as a collection in its own right and <b>create</b> a new entry in it. <sup>[10]</sup>	<b>Delete</b> the addressed member of the collection.



# Example: GraphQL (create by Facebook, 2015)

A data query/manipulation language for APIs + A runtime for fulfilling queries with existing data  
prevent excessively large amounts of data from being returned (e.g. full user records, N+1 problem ...)





# Semantic Web

The goal of semantic web is to provide a framework for sharing and reusing data across applications

Common Semantic Technologies: JSON-LD, RDF, SPARQL, OWL / familiar ? traction ? tooling ?

```
{
  "@context": {
    "name": "http://xmlns.com/foaf/0.1/name",
    "homepage": {
      "@id": "http://xmlns.com/foaf/0.1
/workplaceHomepage",
      "@type": "@id"
    },
    "Person": "http://xmlns.com/foaf/0.1/Person"
  },
  "@id": "https://me.example.com",
  "@type": "Person",
  "name": "John Smith",
  "homepage": "https://www.example.com/"
}
```

```
<rdf:Description
  rdf:about="http://www.recshop.fake/cd/Hide your heart">
  <cd:artist>Bonnie Tyler</cd:artist>
  <cd:country>UK</cd:country>
  <cd:company>CBS Records</cd:company>
  <cd:price>9.90</cd:price>
  <cd:year>1988</cd:year>
</rdf:Description>
```



# Middleware

Middleware are higher-level functions that add additional functionality (e.g. caching, authorization ...)

e.g. a middleware can be viewed as a function that takes an HTTP request and return an HTTP response


```
(defn what-is-my-ip [request]
  {:status 200
   :headers {"Content-Type" "text/plain"}
   :body (:remote-addr request)})
```

```
(defn wrap-content-type [handler content-type]
  (fn [request]
    (let [response (handler request)]
      (assoc-in response [:headers "Content-Type"] content-type))))
```

```
(def app
  (-> handler
    (wrap-content-type "text/html")
    (wrap-keyword-params)
    (wrap-params)))
```

## ▼ Response Headers [view source](#)

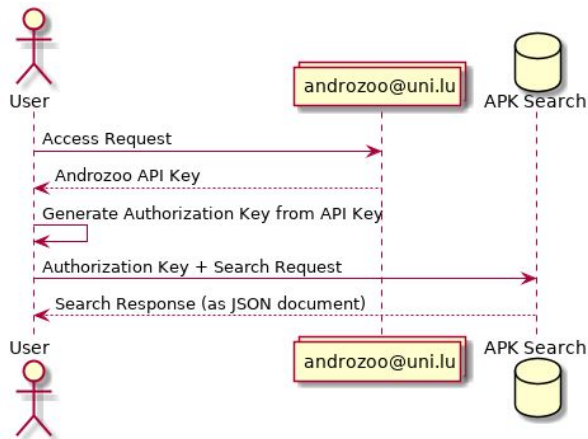
```
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: authorization, if-match, access-control-allow-origin, if-unmodified-since
Access-Control-Allow-Origin: http://localhost:3000
Access-Control-Max-Age: 60
Connection: Keep-Alive
Content-Length: 0
Content-Type: text/html; charset=UTF-8
Date: Sat, 24 Feb 2018 08:52:29 GMT
Server: nginx/1.9.15
Vary: Origin
```



# Example: Usage Monitoring with API Keys

Most API services are NOT free. API providers track API consumers through an API Key

The API keys are send with each API request (e.g. using HTTP header with middleware)



SKU	\$200 MONTHLY CREDIT EQUIVALENT FREE USAGE	MONTHLY VOLUME RANGE (PRICE PER THOUSAND)		
		0–100,000	100,001-500,000	500,001+
<a href="#">Mobile Native Static Maps</a>	Unlimited loads	\$0.00	\$0.00	<a href="#">CONTACT SALES</a> for volume discounts.
<a href="#">Mobile Native Dynamic Maps</a>	Unlimited loads	\$0.00	\$0.00	
<a href="#">Embed</a>	Unlimited loads	\$0.00	\$0.00	
<a href="#">Embed Advanced</a>	Up to 14,000 loads	\$14.00	\$11.20	
<a href="#">Static Maps</a>	Up to 100,000 loads	\$2.00	\$1.60	
<a href="#">Dynamic Maps</a>	Up to 28,000 loads	\$7.00	\$5.60	
<a href="#">Static Street View</a>	Up to 28,000 panos	\$7.00	\$5.60	
<a href="#">Dynamic Street View</a>	Up to 14,000 panos	\$14.00	\$11.20	

# Authentication and Authorization: OAuth 2.0

OAuth is an open standard for authentication and authorization between multiple parties (RFC 6749)

It provides a secure, delegated access to server resources (user) on behalf of a resources owner (google)

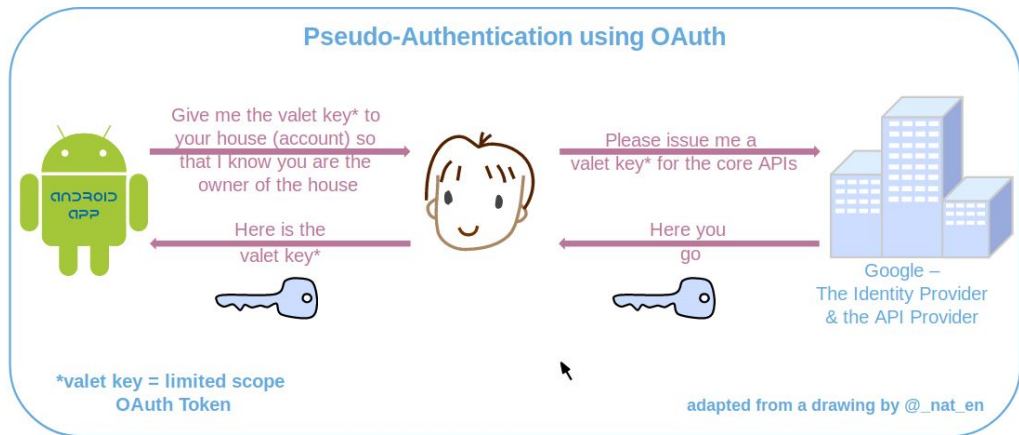
**Sign in**

**Email**

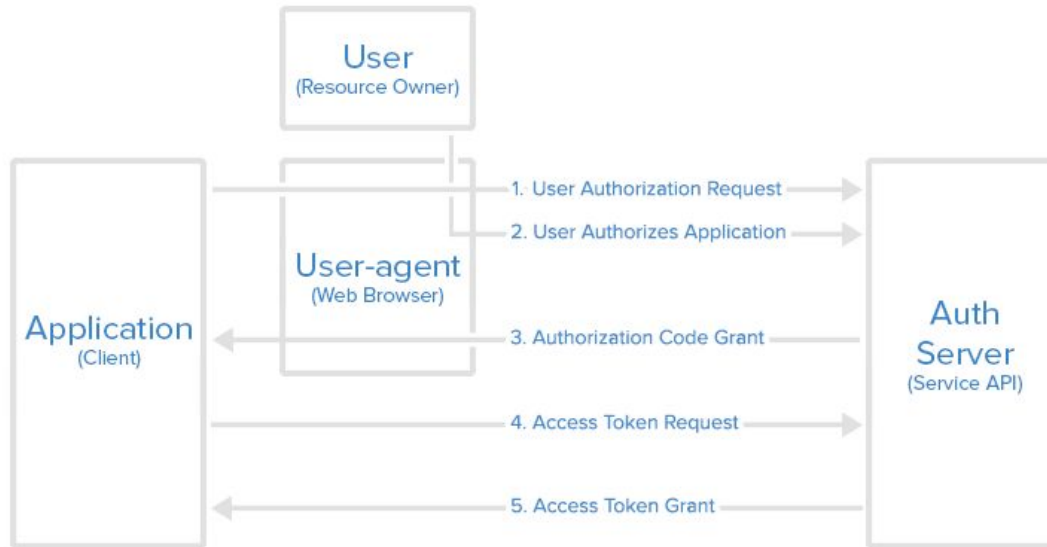
**Password**

[Forgot your password?](#)

or



## Authorization Code Flow



Source: [www.digitalocean.com](http://www.digitalocean.com)



# Difference between API Keys and OAuth Tokens

An API key identifies an application and trace its usage (between API producer and API consumer)

Resource or API Call	Free Quota
Frontend Instances (Automatic Scaling Modules)	28 free instance-hours per day
Backend Instances (Basic and Manual Scaling Modules)	8 free instance-hours per day

An OAuth token grants an application access to some user resource (between API consumer and its a user)

Scope	Meaning
<code>https://www.googleapis.com/auth/calendar</code>	read/write access to Calendars
<code>https://www.googleapis.com/auth/calendar.readonly</code>	read-only access to Calendars



# Google Sign In API





# Introduction

Google Sign In is a secure authentication system that enables users to login with their Google Account

It is also a gateway to connect Google users and services in a secure manner (e.g. with Google Pay)

## Motivation:

- reduce the burden of creating and remembering new credentials for your users
- avoid reinventing the wheel by providing a developer friendly authentication toolkit
- great integration with Google systems: smart lock, firebase, documents, sms retriever ...



# Prerequisites

- An Android device that runs Android version 4.0+ and includes Google Play Store applications
- OR an emulator with Google APIs based on Android 4.2.2+ and Google Play services 15.0.0+
- An Android project configured to compile against Android 4.0 (Ice Cream Sandwich) or newer
- The Google Play services SDK (enabled from Android Studio during application development)

## **To Install Google Play services SDK from Android Studio:**

- In Android Studio, select Tools > Android > SDK Manager
- Scroll to the bottom of the package list and select Extras > Google Repository



## Step 0: Add Google Play Services dependency

This step is required to access Google Play Services packages and resources from Android

Ensure that Google's Maven repository is included in your project *build.gradle*:

```
allprojects {  
    repositories {  
        google()  
    }  
}
```

Add Google Play services as a dependency in your application *build.gradle*:

```
apply plugin: 'com.android.application'  
...  
  
dependencies {  
    compile 'com.google.android.gms:play-services-auth:15.0.1'  
}
```



# Step 1: Configure Google API Console project

This step is required to authorize to application to use Google Play Services

And to charge you based on your API usage (note that Google Sign In is a free service)

Follow the instruction at this link:

[https://developers.google.com/identity/sign-in/android/start-integrating#configure\\_a\\_console\\_name\\_project](https://developers.google.com/identity/sign-in/android/start-integrating#configure_a_console_name_project)

## **Signing-certificate fingerprint**

Add your package name and SHA-1 signing-certificate fingerprint to restrict usage to your Android apps. [Learn more](#)

Get the package name from your AndroidManifest.xml file. Then use the following command to get the fingerprint:

```
keytool -exportcert -keystore path-to-debug-or-production-keystore -list -v
```



## Step 2: Configure Google Sign In objects

This step is required to specify the scope of your request (e.g. user ID and basic profile information)

In *SignInActivity*, add this statement to the *onCreate* method:

```
GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
    .requestEmail()
    .build();
```

You must also add this statement to create a Sign In client with the options you created:

```
mGoogleSignInClient = GoogleSignIn.getClient(this, gso);
```



## Step 3: Check for an existing signed in user

This step is required to check an user already signed in does not have to enter his credentials again

In your *SignInActivity.onStart* method, check if a user has already sign in with this statement:

```
GoogleSignInAccount account = GoogleSignIn.getLastSignedInAccount(this);  
updateUI(account);
```

You should also create an *updateUI* method in your activity to enable/disable components:

- if account is null: disable other activities, terminate background tasks, inform the user log out
- if account is not null: display a message to inform the user, enable other activities, gather profile



## Step 4: Add the Google Sign In button

This step is required to add a button with the same look as other Android applications

In the SignInActivity layout, add the button:

```
<com.google.android.gms.common.SignInButton  
    android:id="@+id/sign_in_button"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content" />
```

You must also register an *OnClickListener* to connect your button with your activity (this):

```
findViewById(R.id.sign_in_button).setOnClickListener(this);
```



## Step 5: Setup the Sign In flow

This step is required to authenticate the user once he/she clicks on the button

In the *SignInActivity* activity, create the following method:

```
private void signIn() {  
    Intent signInIntent = mGoogleSignInClient.getSignInIntent();  
    startActivityForResult(signInIntent, RC_SIGN_IN);  
}
```

This will send an Intent that prompts the user to enter his/her account credentials





Once the user sign in, you can get an account object by defining on *onActivityResult* method

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    // Result returned from launching the Intent from GoogleSignInClient.getSignInIntent(...);
    if (requestCode == RC_SIGN_IN) {
        // The Task returned from this call is always completed, no need to attach
        // a listener.
        Task<GoogleSignInAccount> task = GoogleSignIn.getSignedInAccountFromIntent(data);
        handleSignInResult(task);
    }
}
```

Since these events are **asynchronous**, you must define the method to receive the result of the activity



Once the output of the activity is received, you can handle them with *handleSignInResult* method

```
private void handleSignInResult(Task<GoogleSignInAccount> completedTask) {  
    try {  
        GoogleSignInAccount account = completedTask.getResult(ApiException.class);  
  
        // Signed in successfully, show authenticated UI.  
        updateUI(account);  
    } catch (ApiException e) {  
        // The ApiException status code indicates the detailed failure reason.  
        // Please refer to the GoogleSignInStatusCodes class reference for more information.  
        Log.w(TAG, "signInResult:failed code=" + e.getStatusCode());  
        updateUI(null);  
    }  
}
```

You must handle the case where the user did authenticated (try) and when an error occurred (catch)



## Step 6: Retrieve User Profile Information

This step is required only if you want to retrieve profile information from a signed in user

Use `GoogleSignIn.getLastSignedInAccount` method to request profile information:

```
GoogleSignInAccount acct = GoogleSignIn.getLastSignedInAccount(getActivity())
if (acct != null) {
    String personName = acct.getDisplayName();
    String personGivenName = acct.getGivenName();
    String personFamilyName = acct.getFamilyName();
    String personEmail = acct.getEmail();
    String personId = acct.getId();
    Uri personPhoto = acct.getPhotoUrl();
}
```



## Step 7: Sign Out Users with a button

This step is nice to have if you want to enable your users to disconnect their account from your app

You must add a sign out button in your app that attach an *onClickListener* when the button is clicked:

```
private void signOut() {  
    mGoogleSignInClient.signOut()  
        .addOnCompleteListener(this, new OnCompleteListener<Void>() {  
            @Override  
            public void onComplete(@NonNull Task<Void> task) {  
                // ...  
            }  
        });  
}
```





# OpenStreetMaps API





## Step 0: Add osmdroid dependency

This step is required to access OpenStreetMaps data conveniently from your application

Put this code in your application *build.gradle* (current release version: 6.0.3)

```
repositories {  
    mavenCentral()  
}  
  
dependencies {  
    compile 'org.osmdroid:osmdroid-android:<VERSION>'  
}
```





## Step 1: Add permissions to access location

This step is required to enable your application to access geolocation, store and Internet

Put this elements in your *AndroidManifest.xml* file:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Note: Android 6.0+ devices require to check for dangerous permissions at runtime:

<https://developer.android.com/training/permissions/requesting>



## Step 2: Create fullscreen layout to display the map

This step is required to display the map on an activity will be defined in the next slides

Create a layout file (e.g. src/main/res/layouts/main.xml) and put this content in it:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <org.osmdroid.views.MapView android:id="@+id/map"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" />
</LinearLayout>
```



## Step 3: Create an activity to handle the map

This step is required to manage the map (initialization, resume, pause, edit, view ...) from an Activity

```
MapView map = null;
@Override public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    //handle permissions first, before map is created. not depicted here

    //load/initialize the osmdroid configuration, this can be done
    Context ctx = getApplicationContext();
    Configuration.getInstance().load(ctx, PreferenceManager.getDefaultSharedPreferences(ctx));
    //setting this before the layout is inflated is a good idea
    //it 'should' ensure that the map has a writable location for the map cache, even without permissions
    //if no tiles are displayed, you can try overriding the cache path using Configuration.getInstance().setCachePath
    //see also StorageUtils
    //note, the load method also sets the HTTP User Agent to your application's package name, abusing osm's tile server

    //inflate and create the map
    setContentView(R.layout.activity_main);

    map = (MapView) findViewById(R.id.map);
    map.setTileSource(TileSourceFactory.MAPNIK);
}
```



You must also define *onResume* and *onPause* methods to restore the state of the map between switches

```
public void onResume(){
    super.onResume();
    //this will refresh the osmdroid configuration on resuming.
    //if you make changes to the configuration, use
    //SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(this);
    //Configuration.getInstance().load(this, PreferenceManager.getDefaultSharedPreferences(this));
    map.onResume(); //needed for compass, my location overlays, v6.0.0 and up
}

public void onPause(){
    super.onPause();
    //this will refresh the osmdroid configuration on resuming.
    //if you make changes to the configuration, use
    //SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(this);
    //Configuration.getInstance().save(this, prefs);
    map.onPause(); //needed for compass, my location overlays, v6.0.0 and up
}
```



## Step 4: Add map controls and a default location

This step is nice to have if you want to provide a familiar graphical environment to your user

To enable maps control, add this code to *onCreate* method:

```
map.setBuiltInZoomControls(true);  
map.setMultiTouchControls(true);
```

To move the map to a default view point, add this code to *onCreate* method:

```
IMapController mapController = map.getController();  
mapController.setZoom(9.5);  
GeoPoint startPoint = new GeoPoint(48.8583, 2.2944);  
mapController.setCenter(startPoint);
```

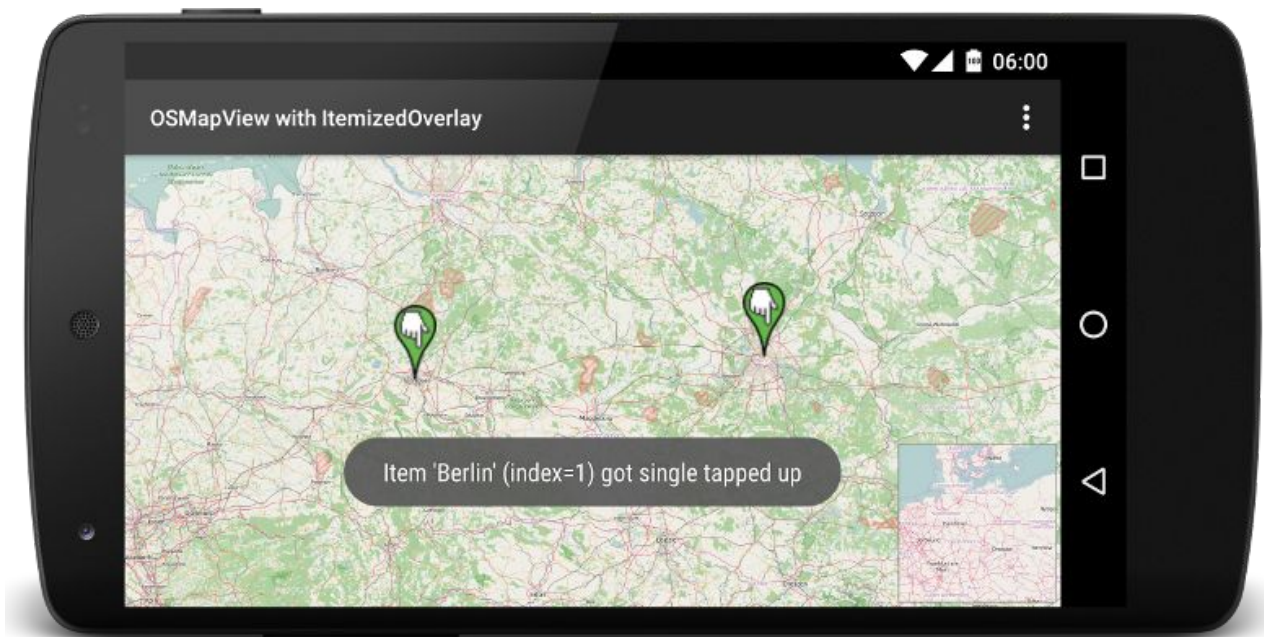


## Step 5: Add markers (overlay) on the map

```
//your items
ArrayList<OverlayItem> items = new ArrayList<OverlayItem>();
items.add(new OverlayItem("Title", "Description", new GeoPoint(0.0d,0.0d))); // Lat/Lon decimal degrees

//the overlay
ItemizedOverlayWithFocus<OverlayItem> mOverlay = new ItemizedOverlayWithFocus<OverlayItem>(items,
    new ItemizedIconOverlay.OnItemGestureListener<OverlayItem>() {
        @Override
        public boolean onItemSingleTapUp(final int index, final OverlayItem item) {
            //do something
            return true;
        }
        @Override
        public boolean onItemLongPress(final int index, final OverlayItem item) {
            return false;
        }
    });
mOverlay.setFocusItemsOnTap(true);

mMapView.getOverlays().add(mOverlay);
```





# Tutorial





# Goal

You have an existing application called “Routes”:

<https://git.fmind.me/fmind/android-routes>

You have to create a login activity to authenticate your users

Your users can then access a map activity to view their position on a map



# Google Sign In API

Open “*app/java/SignInActivity.java*” and fill the parts highlighted by a TODO comment:

- *onCreate*: configure sign in with `DEFAULT_SIGN_IN` and build a sign in client
- *onStart*: check a user already sign in and retrieve its account if it's the case
- *onActivityResult*: handle the result of the sign in activity send with intent
- *HandleSignInResult*: perform the action when the user sign in (or not)
- *handleSignOutResult*: perform the action when the user sign out



# OpenStreetMaps API

Open “*app/java/MapsActivity.java*” and fill the parts highlighted by a TODO comment in *onCreate*:

1. create a layout
2. initialize the map from the layout
3. center the map to Maison du Nombre location
4. add a marker on this position and set the default zoom
5. create an event listener able to add and remove overlay on click