UNDERSTAND AND USE EXTERNAL APIS IN YOUR ANDROID APP

MEDERIC.HURIER@UNI.LU

2017

OUTLINE

- What is an API and its use case ?
- Authentication/Authorization
- How-to: Google Sign-In
- How-to: Google Maps
- Tutorial

WHAT IS AN API AND ITS USE CASE ?

DEFINITIONS

API stands for **A**pplication **P**rogramming **I**nterface

A way of communicating with a particular program.

A set of rules that defines how a software program can request and receive information from another software.

Source: dictionary.cambridge.org

FOR US, PROGRAMMERS

A good API makes it easier to develop a program

Each API provides a building block for your program.

The programmers then compose these building blocks.

They can be used to access databases, computer hardware, GUI components (e.g. POSIX, OpenGL, SAX, PEP 249).

They can also support **Web Services** by providing a specification for remote calls (e.g. Google, Twitter, Github).

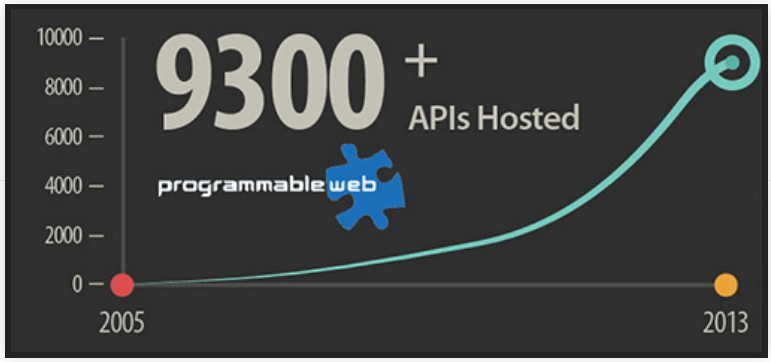
WHERE CAN WE FIND APIS ?

We can find APIs everywhere on the web.

More than 14 000 APIs are available on
programmableweb.com

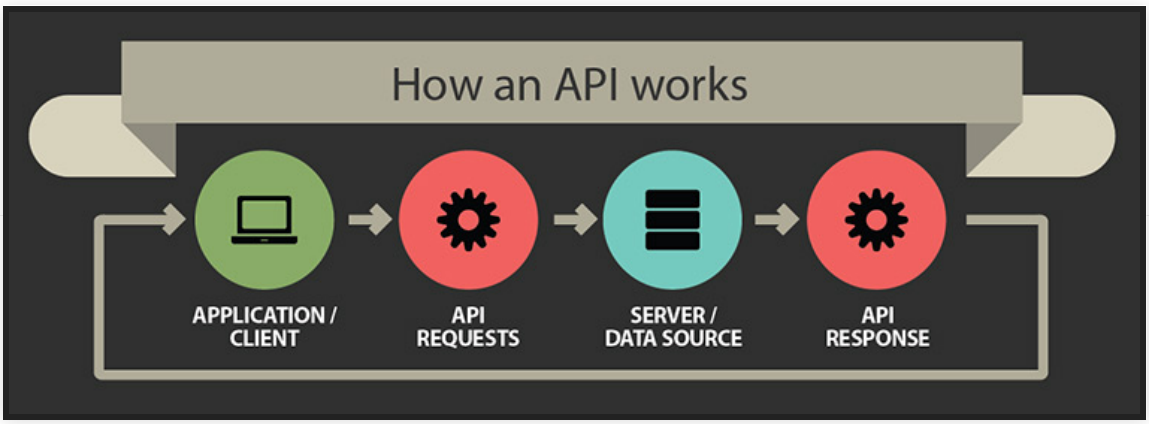
http://www.programmableweb.com/apis/directory

EVOLUTION



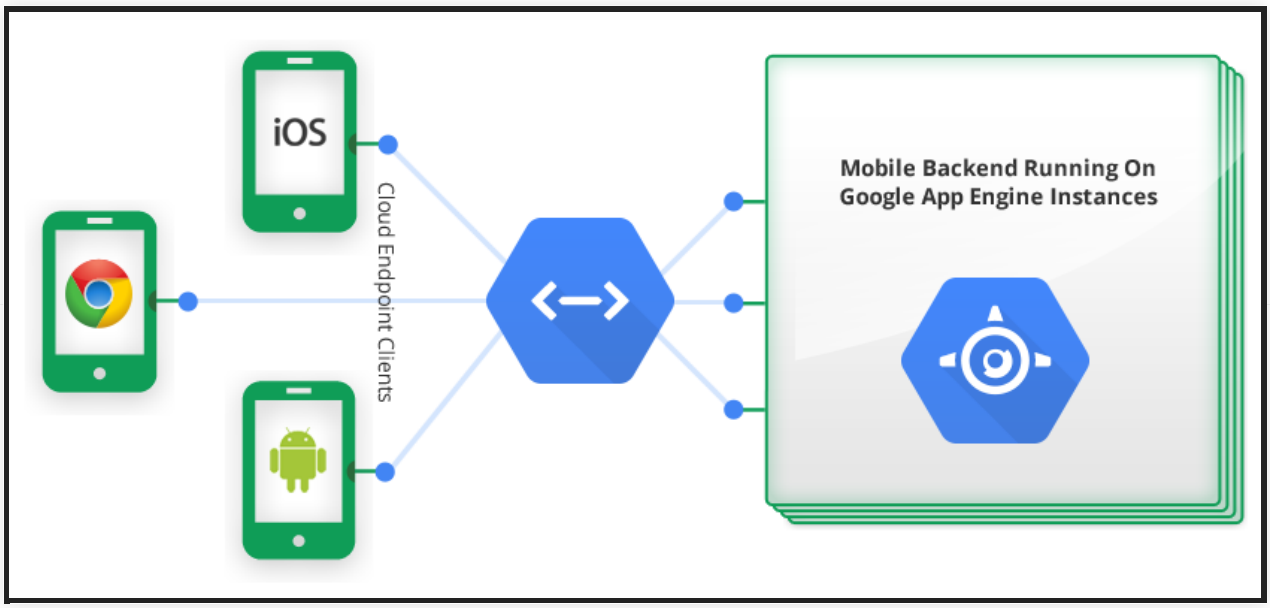
Source: smartfile.com

WORKFLOW



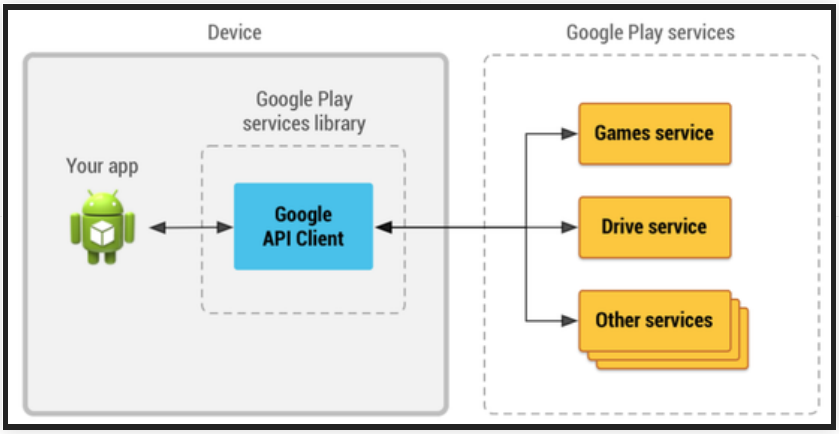
Source: smartfile.com

API ORIENTED



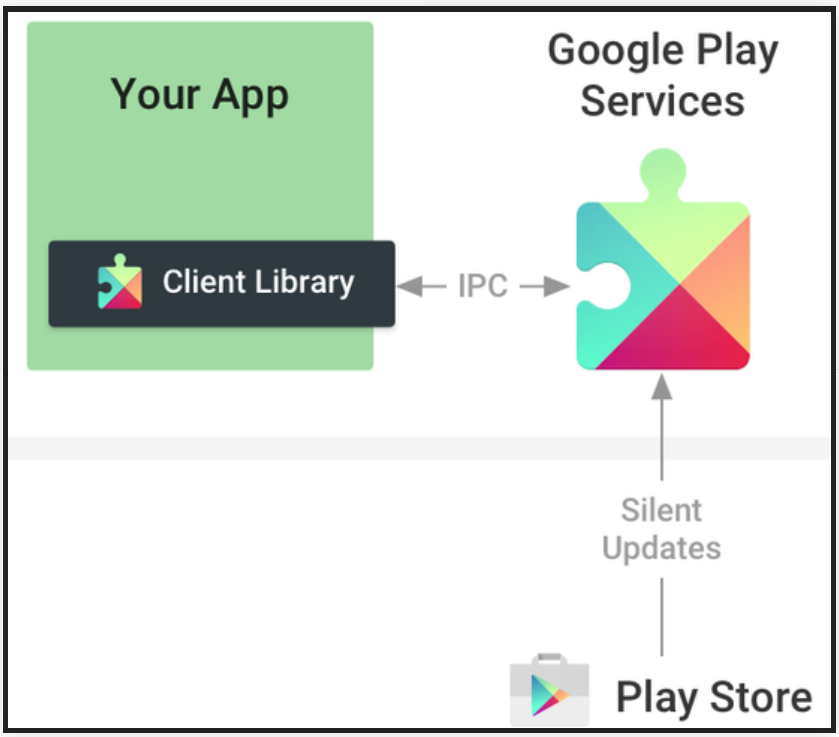
Source: cloud.google.com

ON THE WEB



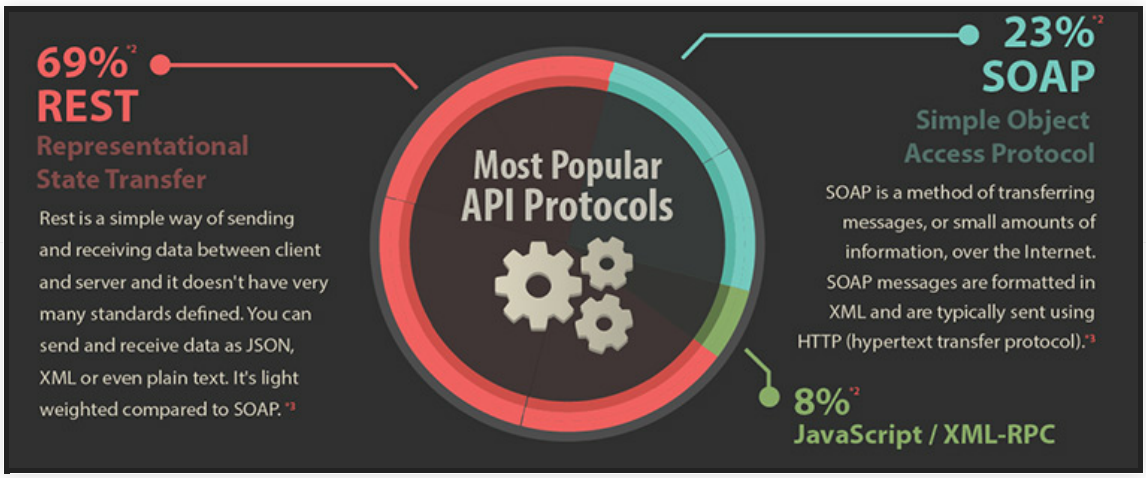
Source: developers.google.com

ON YOUR DEVICE



Source: developers.google.com

REST VS SOAP VS JS

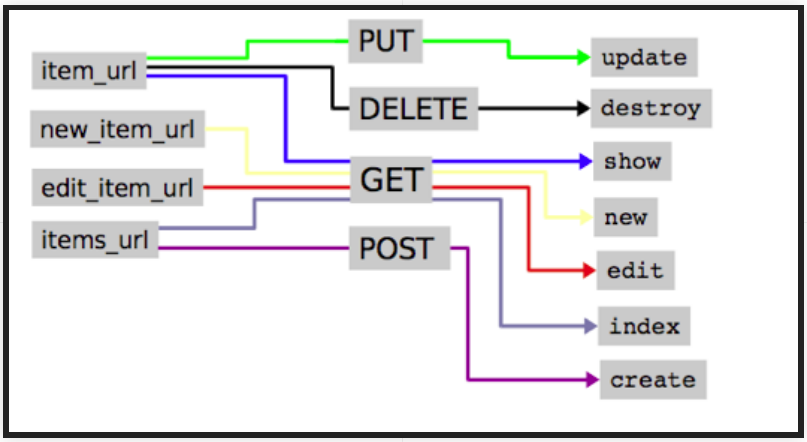


Source: smartfile.com

REST

REST for Representational State Transfer

It is an architectural style and an approach to lightweight communication between producers and consumers.



REST has several constraints (called RESTful):

1. Client-Server should be independent from each other
2. Requests should be unrelated from each other (stateless)
3. Cache can to store responses and save bandwidth
4. Intermediary can be used to improve the communication in term of scalability or security (layered system)
5. There should be an uniform interface to decouple the architecture (more information on the next slide)

Applied to Web Services development:

1. Each resources should have a base URI
   - e.g. http://example.com/resources
2. The communication should rely on a media type
   - e.g. JSON, XML, ATOM, Protocol Buffer
3. HTTP protocol and HTTP methods should be used
   - e.g. GET, PUT, POST, DELETE
4. Links can reference states and related resources

RESTful API HTTP methods

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

OTHER RESOURCES

- **GraphQL**: a query language for your API (Facebook)

http://graphql.org/

- **API Security Checklist** (for API developers)

https://github.com/shieldfy/API-Security-Checklist

- **REST Anti-Patterns** (for API developers)

https://www.infoq.com/articles/rest-anti-patterns

AUTHENTICATION/AUTHORIZATION

API USAGE

Most API services are NOT free of use

but free tiers are available for most providers

You have to send an **API Key** with each request for tracking

CLOUD TRANSLATION API PRICING

Translation API pricing is based on usage. Translation usage is calculated in millions of characters. You are billed per character, so you pay only for what you use.

FEATURE	COST (USD) UP TO 1,000 M CHARACTERS/MONTH
Text Translation	$20 per million characters
Language detection	$20 per million characters

Example: Google Translation API

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** on behalf (e.g. user profile) of a **resources owner** (e.g. Google).

Sign in

Email

email

Password

password

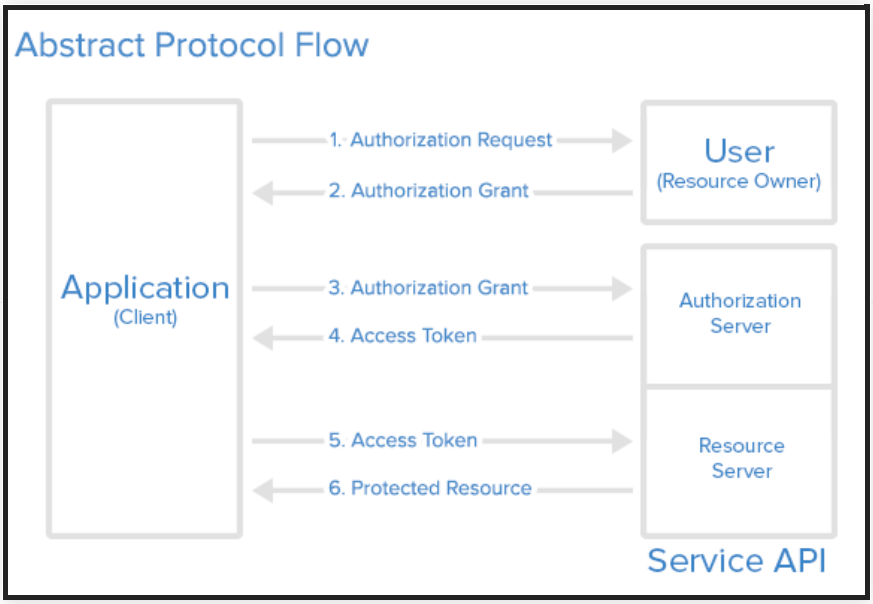
Forgot your password?

Sign In

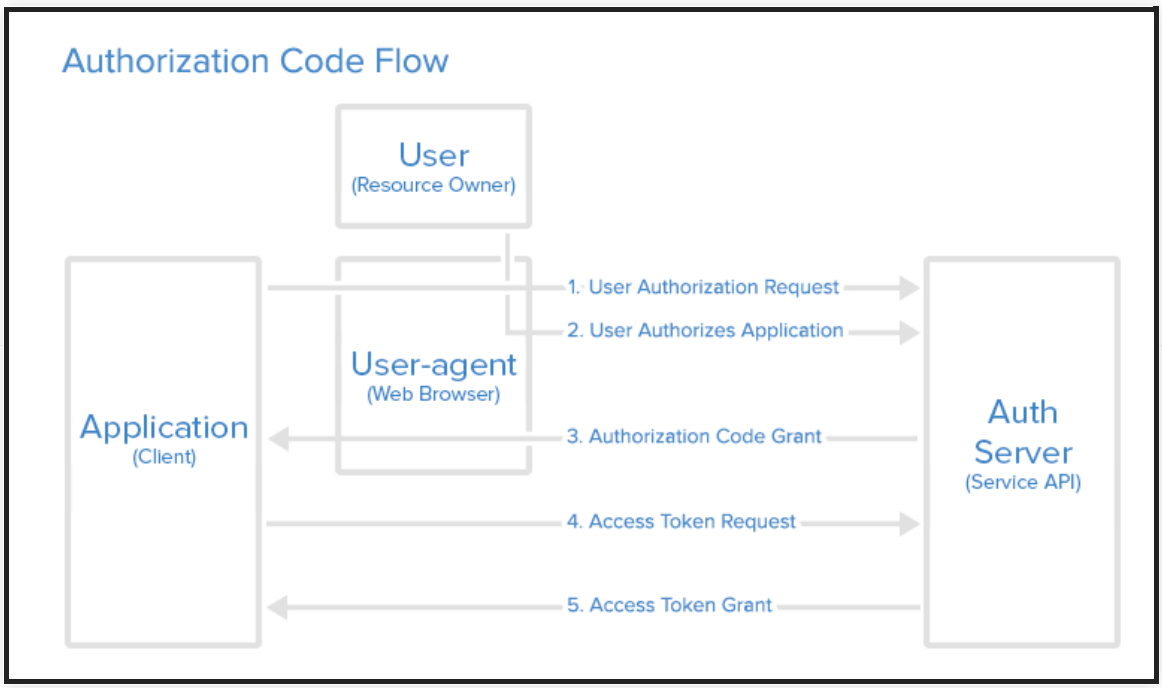
or

Sign in with Facebook

Sign in with Google



Source: www.digitalocean.com



Source: www.digitalocean.com

KEYS AND SECRETS

- **An API key** identifies your application and traces its usage (between the API producer and the 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

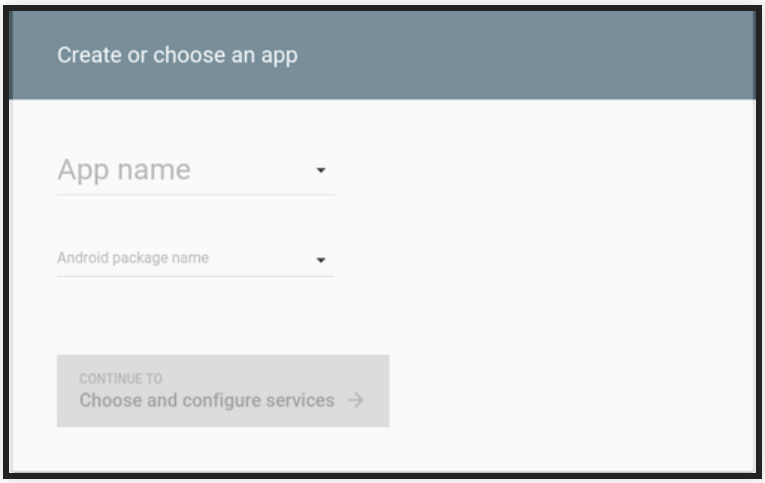
- **An OAuth Token** grants your application access to the user resources (between the API consumer and the user)

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

HOW-TO: GOOGLE SIGN-IN

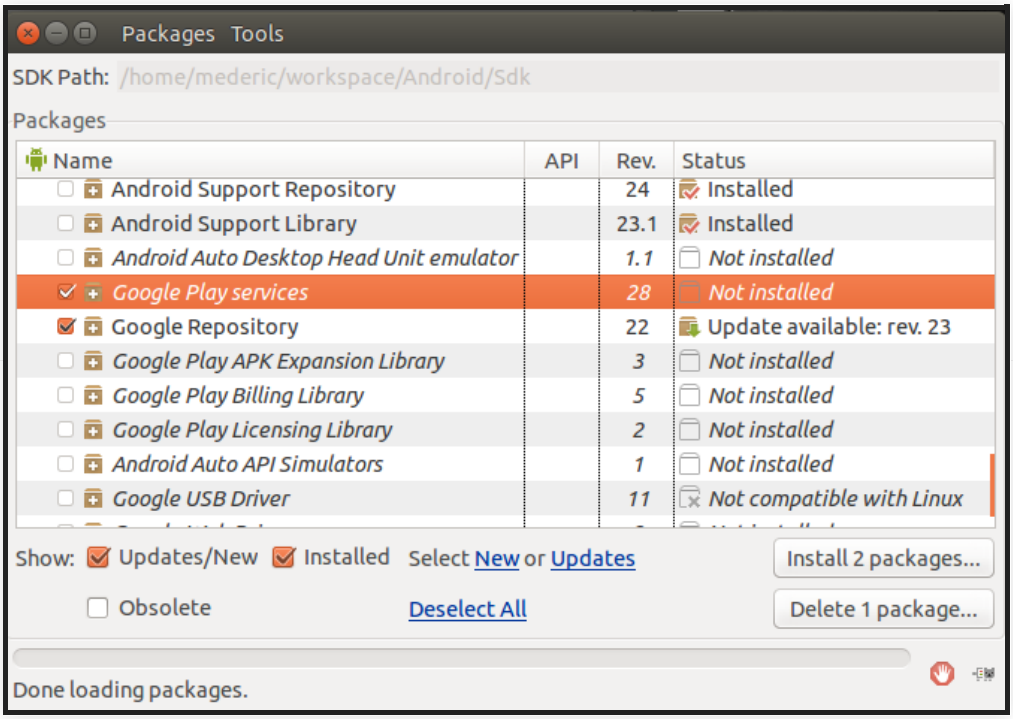
0. Get a configuration file and move it in app/

https://developers.google.com/mobile/add?platform=android&cntapi=signin



(Behind the scene, Google creates an API key and OAuth ID)

1. Install **Google Play Service** from SDK Manager



(You must have Android Studio and Android SDK installed)

2. Include this line in your project/build.gradle

```
buildscript {
    ...
    dependencies {
        ...
        classpath 'com.google.gms:google-services:1.5.0-beta2'
    }

}
```

(Don't worry about the beta version ;))

3. And these line in your app/build.gradle

```
...
apply plugin: 'com.google.gms.google-services'

dependencies {
    ...
    compile 'com.google.android.gms:play-services-auth:8.3.0'
}
```

NOTE: your device needs to have the same version !

4. Create a new activity for your application package/SignInActivity.xml

```
public class SignInActivity extends AppCompatActivity
    implements GoogleApiClient.OnConnectionFailedListener {
    private static final int RC_SIGN_IN = 9001;
    private GoogleApiClient client;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
        GoogleSignInOptions gso = new GoogleSignInOptions
                .Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
                .build();
        client = new GoogleApiClient.Builder(this)
                .enableAutoManage(this, this)
                .addApi(Auth.GOOGLE_SIGN_IN_API, gso)
                .build();
    }
```

```
        @Override
        public void onConnectionFailed(
            ConnectionResult connectionResult) {

        }

        @Override
        public void onClick(View v) {
            switch (v.getId()) {
                case R.id.button_sign_in:
                    signIn();
                    break;
            }
        }
```

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

@Override
public void onActivityResult(int requestCode,
        int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == RC_SIGN_IN) {
        GoogleSignInResult result = Auth.GoogleSignInApi
            .getSignInResultFromIntent(data);
        handleSignInResult(result);
    }
}
```

```
    private void handleSignInResult(GoogleSignInResult result) {
            Log.i(TAG, "handleSignInResult:" + result.isSuccess());

            if (result.isSuccess()) {
                // display a message
                Toast toast = Toast.makeText(
                    getApplicationContext(),
                    "Sign-in OK !", Toast.LENGTH_LONG);
                toast.show();
            }
        }
```

5. Add this button in your activity layout res/layout /content_sign_in.xml

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

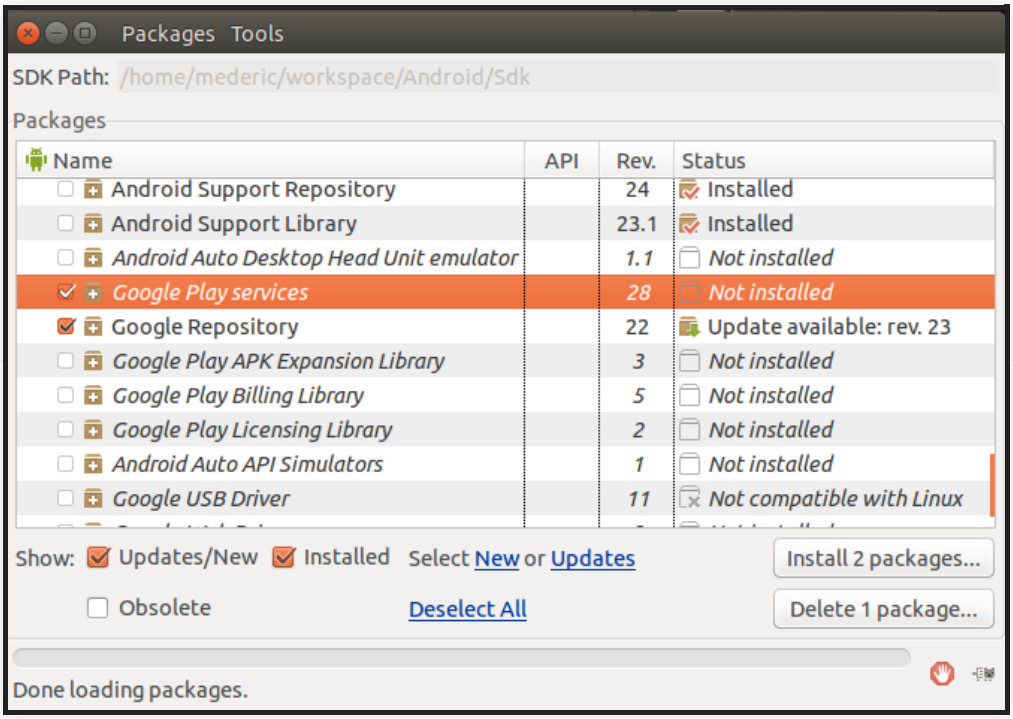
6. Don't forget to add a Listener in package/SignInActivity.java

```
    protected void onCreate(Bundle savedInstanceState) {
            ...
            findViewById(R.id.button_sign_in)
                .setOnClickListener(this);
        }
```

7. Test your app ;)

HOW-TO: GOOGLE MAPS

1. Install **Google Play Service** from SDK Manager



(You must have Android Studio and Android SDK installed)

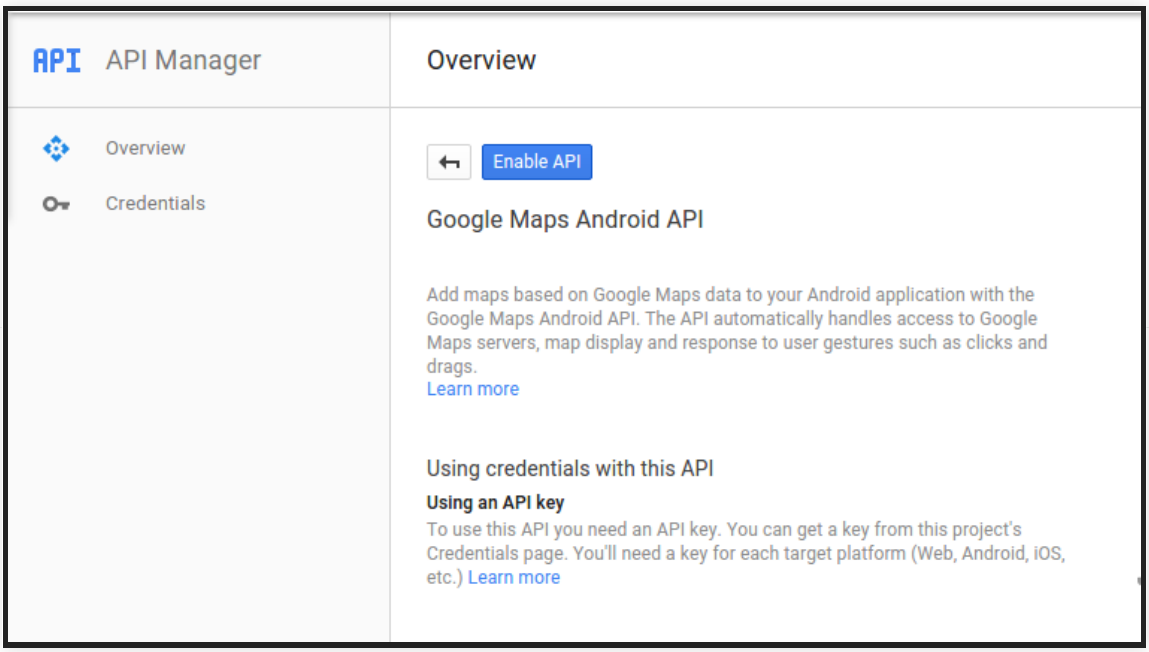
2. Add the API library to your app/build.gradle

```
dependencies {
    ...
    // Google Maps
    compile 'com.google.android.gms:play-services-maps:8.3.0'
}
```

(Android Studio should ask you to synchronize your project)

3. Get an Android API Key and enable Google Maps API (instruction below)

https://developers.google.com/maps/documentation/android-api/signup



4. Set the library version and the API key in AndroidManifest.xml

```
<manifest>
    ...
    <meta-data
      android:name="com.google.android.gms.version"
      android:value="@integer/google_play_services_version" />

    <meta-data
      android:name="com.google.android.geo.API_KEY"
      android:value="YOUR_API_KEY" />
</manifest>
```

(The API key is a long string which start with 'AIza')

5. (Optional) Add additional permissions in AndroidManifest.xml

- **ACCESS_COARSE_LOCATION**: allows an app to access approximate location.
- **ACCESS_FINE_LOCATION**: allows an app to access precise location.

```
<manifest>
    ...
    <uses-permission
      android:name="android.permission.YOUR_PERMISSION" />
</manifest>
```

(In Android 6.0, permissions will be asked during runtime)

6. Create a new activity for your application: package/MapsActivity.java

```
public class MapsActivity extends FragmentActivity
    implements OnMapReadyCallback {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);

        SupportMapFragment mapFragment =
            (SupportMapFragment) getSupportFragmentManager()
            .findFragmentById(R.id.map);

        mapFragment.getMapAsync(this);
    }
```

(Remember to add an activity entry in AndroidManifest.xml)

7. Edit the layout file as follow: res/layout /activity_maps.xml

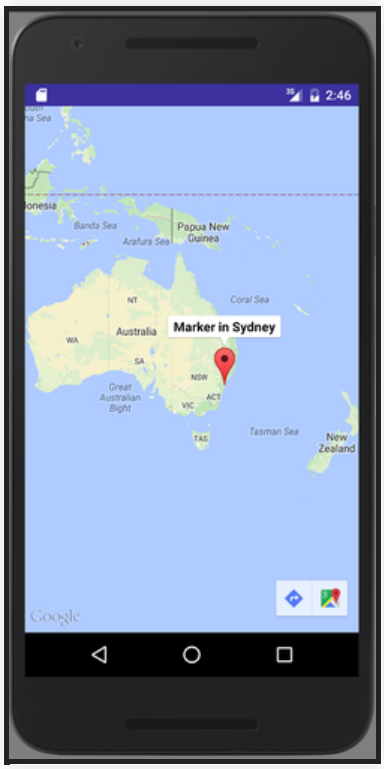
```
<fragment
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/map"
    tools:context=".MapsActivity"
    android:name="com.google.android.gms.maps.SupportMapFragment"
  />
```

```
@Override
public void onMapReady(GoogleMap map) {
    LatLng sydney = new LatLng(-34, 151);

    map.addMarker(new MarkerOptions()
        .position(sydney).title("Marker in Sydney"));

    map.moveCamera(CameraUpdateFactory.newLatLng(sydney));
}
```

8. Test your app :)



TUTORIAL

GET INSPIRATION

Import these projects from Google to find more examples

Maps: https://github.com/googlesamples/google-services

Sign-In: https://github.com/googlemaps/android-samples

(you will need to setup your own API key and Clients ID)

GOOGLE SIGN-IN

- Display the user's email address once he authenticates (you need to request the permission using GoogleSignInOptions)
- Prevent MapsActivity from launching if the user is not login (you can use a static boolean in MainActivity)
- Add a disconnect button (that truly disconnect the user)

GOOGLE MAPS

- Center the map on Campus Kirchberg with a zoom of 15 (use Google Maps in your browser to find the coordinates and the newLatLngZoom method instead of newLatLng)
- Create a destination marker after a long click on a location (implements OnMapLongClickListener, there should be only 1 destination marker at a time !)
- Display the route, the duration and the distance between the points (have a look at Google Direction API)
  - you may want to use this library instead ;)

TIPS

- Use hardware acceleration to improve the emulator speed https://developer.android.com/tools/devices/emulator.html
- To zoom in/out with your mouse: double click, don't release the second click and then move your mouse up and down
- Remember that you can use mocks during your devs !

All the APIness for your Assignment :)