THE UNIVERSITY OF MELBOURNE

COMP90018

Mobile Computing Systems Programming

WEEK 6 – Firebase and Connectivity

Atiq Shaikh,

atiq.shaikh@unimelb.edu.au

1 Firebase

2 Connectivity

Intended Learning Outcomes

Firebase

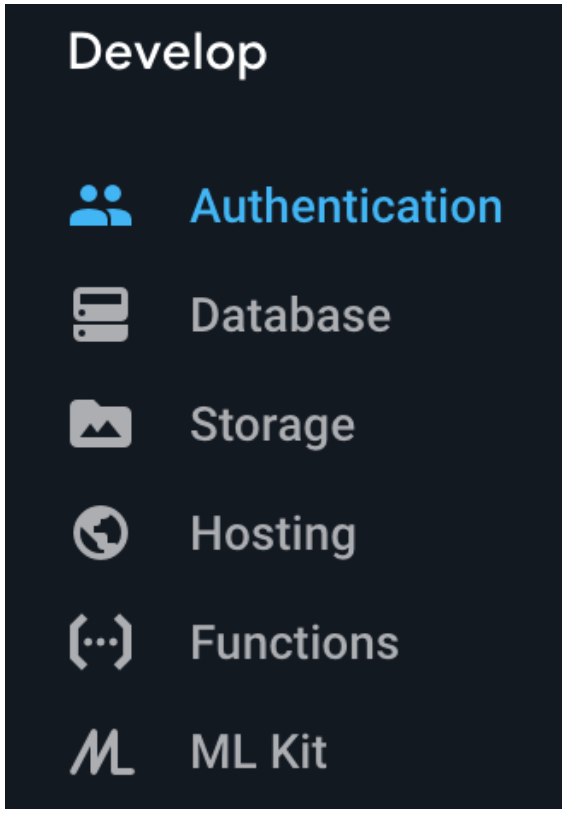
Firebase

- Firebase is a ***mobile and web application development platform*** developed by Firebase, Inc. in 2011, then acquired by Google in 2014.
- As of October 2018, the Firebase platform has 18 products, which are used by 1.5 million apps.
- https://firebase.google.com/

Firebase

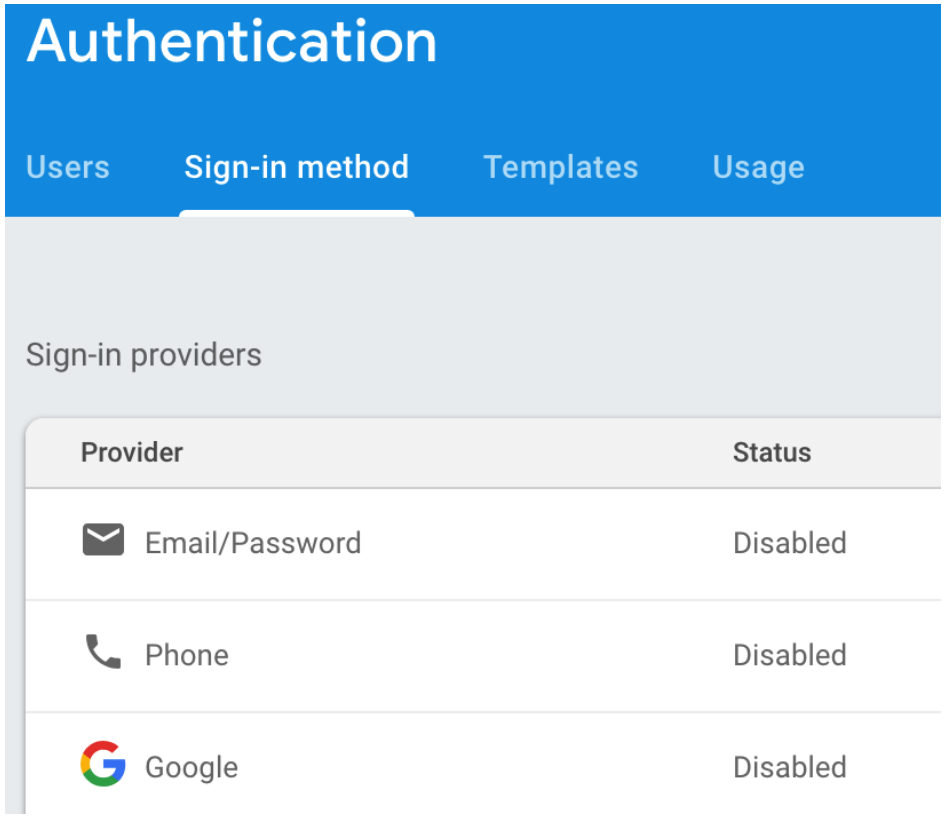
- *Functionality*

Firebase – Develop



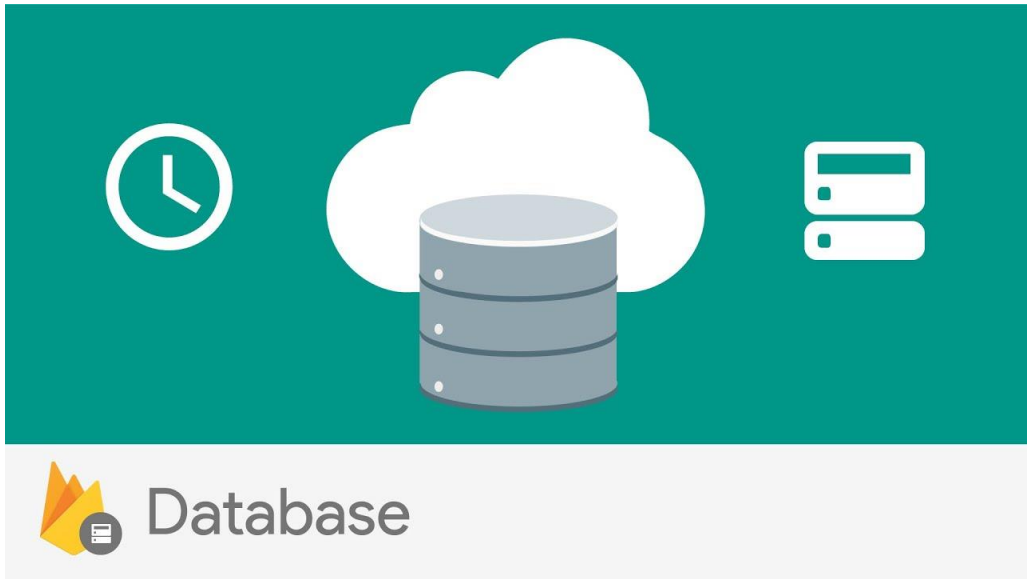
- *Fast mobile app development*

Firebase – Authentication



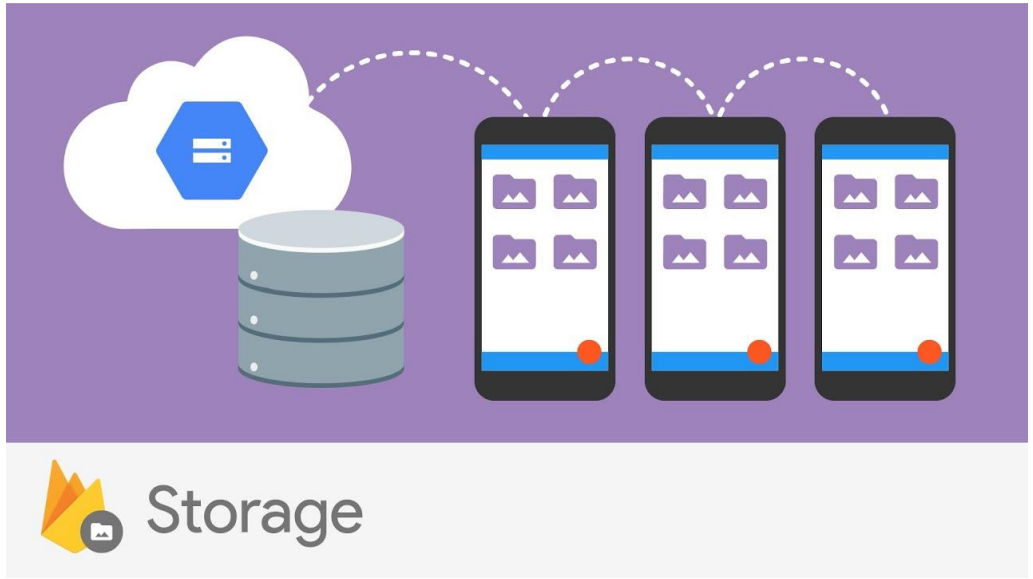
- *Authentication:*
- *Identify users without writing server code*

Firebase – Database



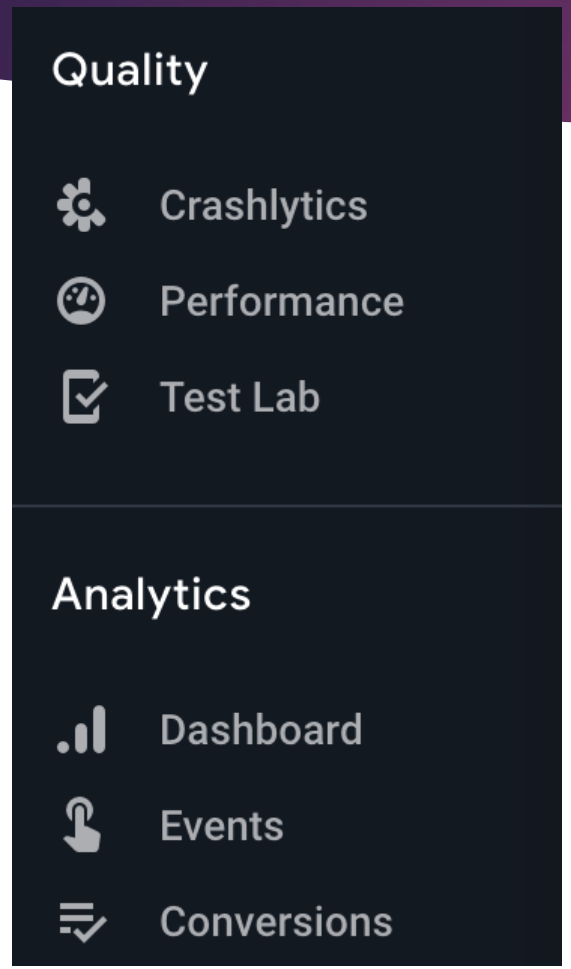
Backend Database:
synchronize data with mobile apps

Firebase – Storage



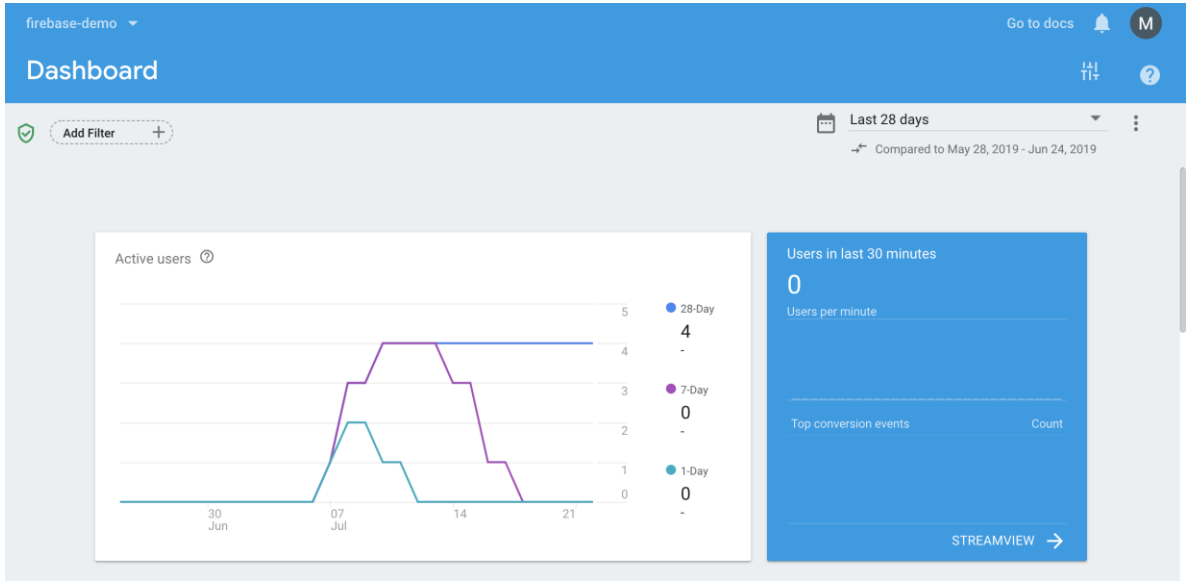
Backend Storage:
manage media files uploaded by mobile apps

Firebase – Test/Monitor Performance



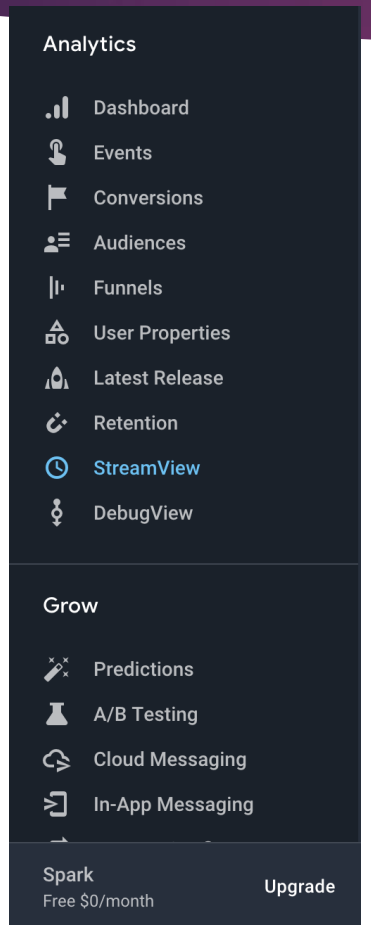
- *Test/monitor mobile app*

Firebase – Monitor Performance



Monitor mobile app

Firebase – More Functions



More Functions …

Firebase

- *First Firebase Application*

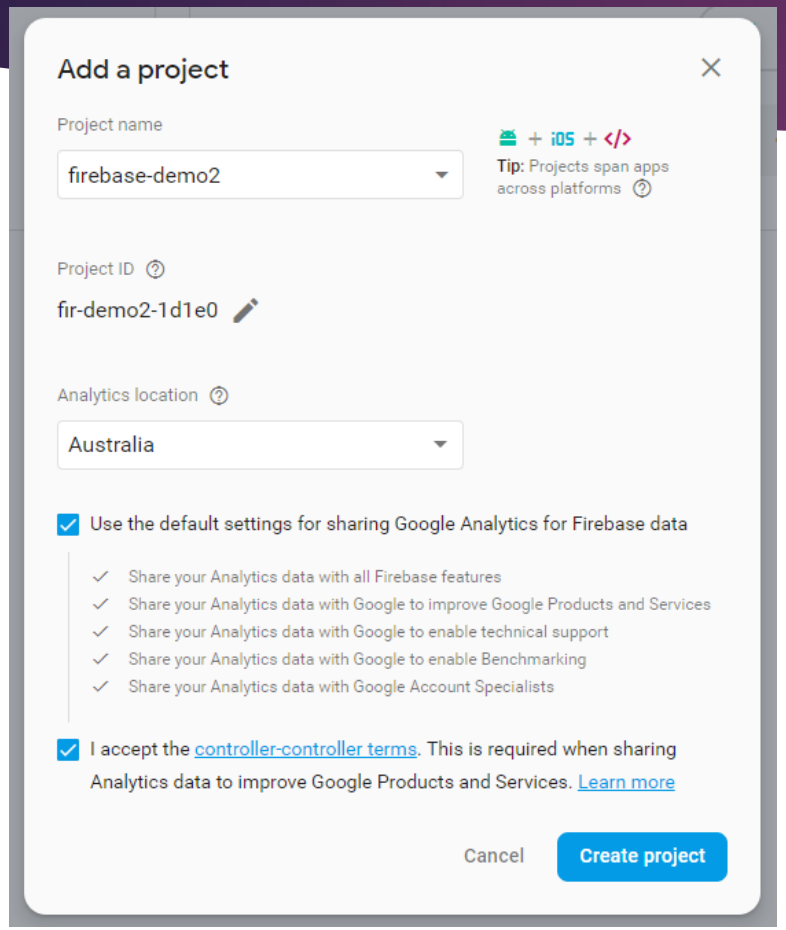
Firebase – Realtime Database

Store and sync data with Firebase ***NoSQL*** *cloud database.*

Data is synced across all clients in realtime, and remains available when your app goes offline.

https://firebase.google.com/docs/database

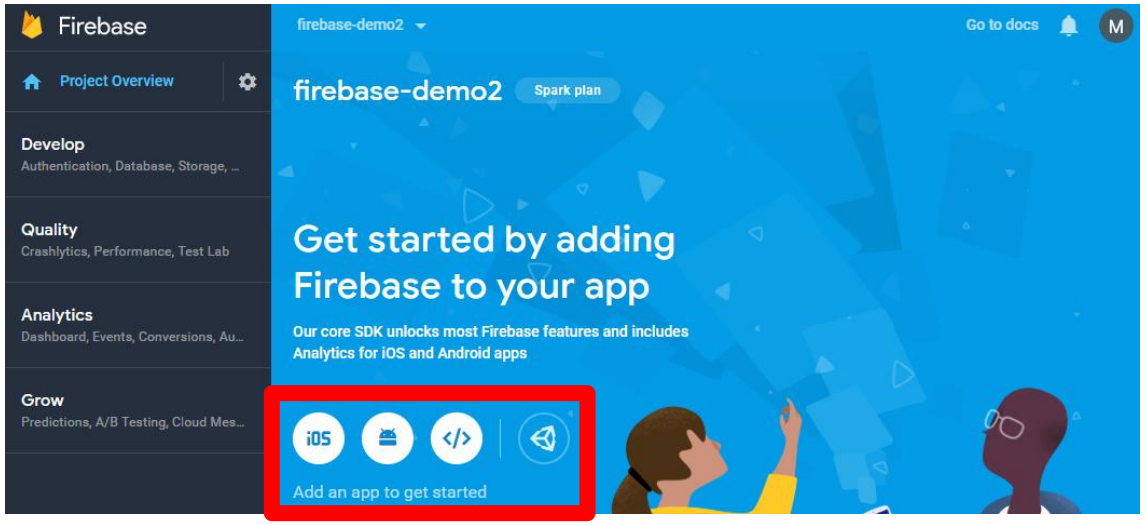
Create a Firebase Application



Step 1: Create a project at Firebase Console

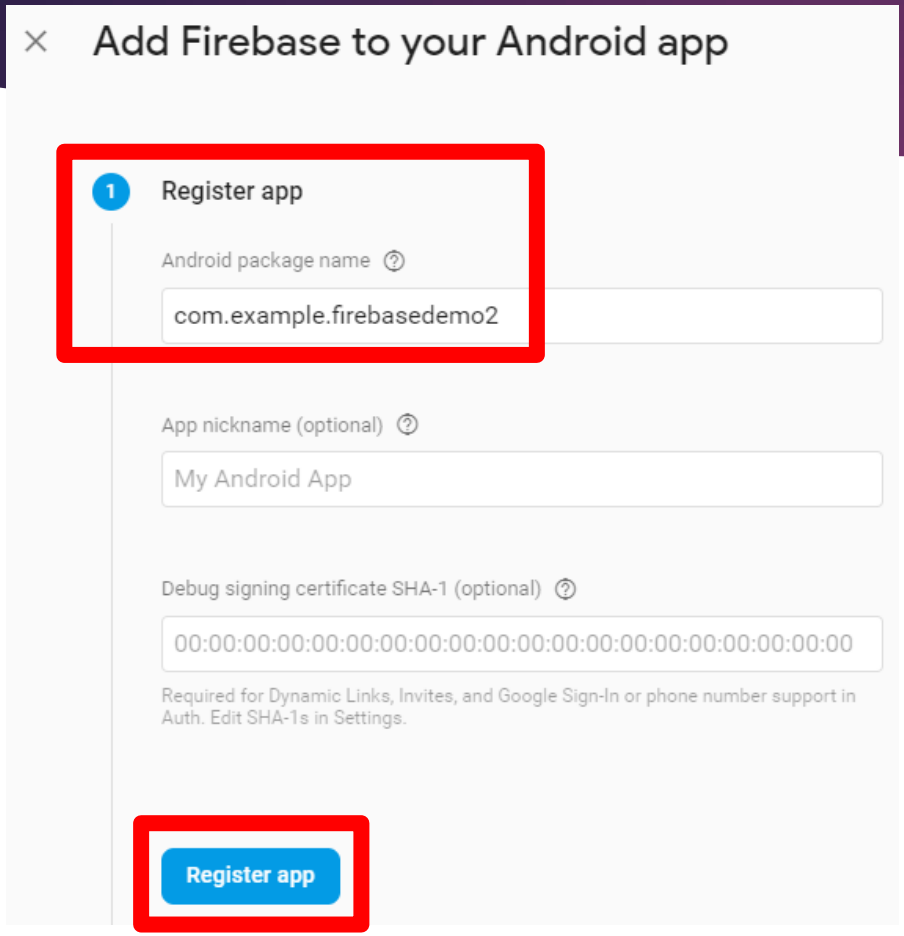
https://console.firebase.google.com

Create a Firebase Application



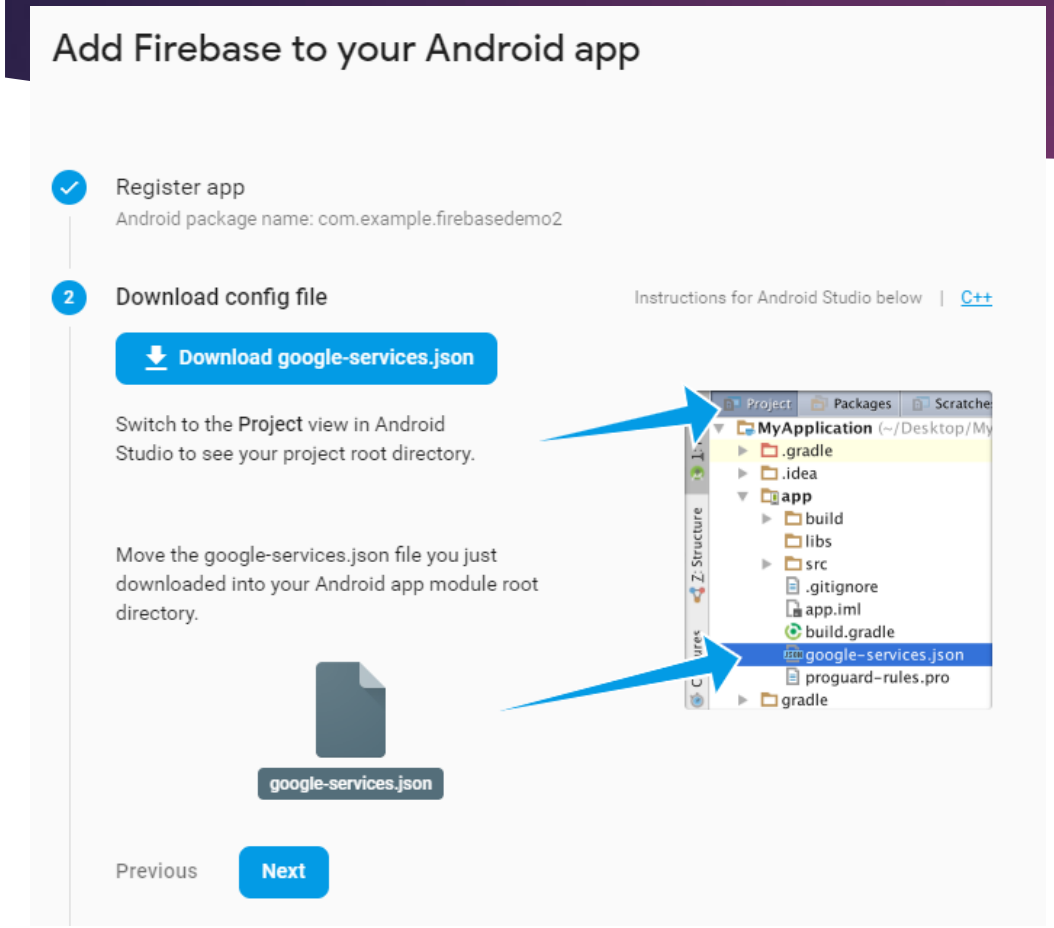
Step 2: Add an app to your project

Create a Firebase Application



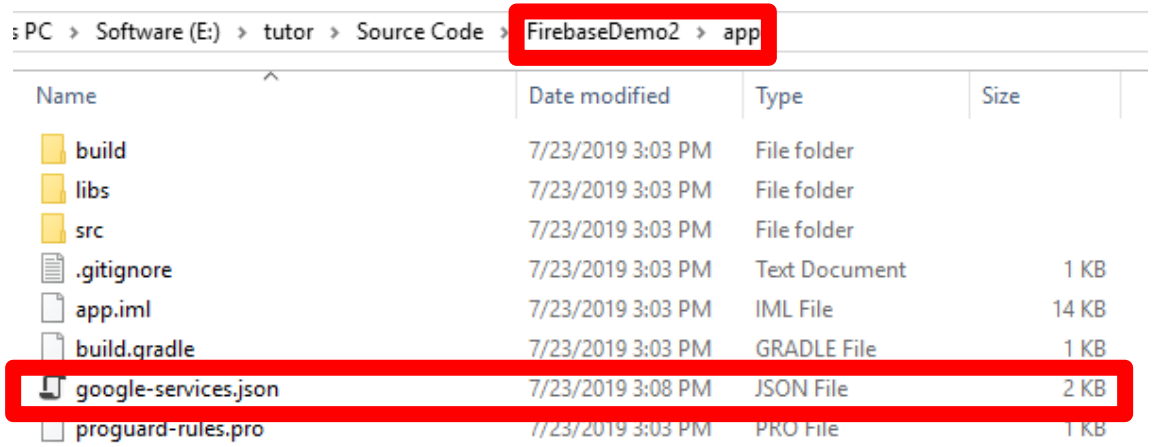
Step 3: Create an Android app and input the package name

Create a Firebase Application



Step 4: Download the config file and put the file into the Android project

Create a Firebase Application



Step 4: Download the config file and put the file into the Android project

Create a Firebase Application

3 Add Firebase SDK

Instructions for Gradle | C++

The Google services plugin for Gradle loads the `google-services.json` file you just downloaded. Modify your build.gradle files to use the plugin.

Project-level build.gradle (`<project>/build.gradle`):

```
buildscript {
  dependencies {
    // Add this line
    classpath 'com.google.gms:google-services:4.2.0'
  }
}
```

App-level build.gradle (`<project>/<app-module>/build.gradle`):

```
dependencies {
  // Add this line
  implementation 'com.google.firebase:firebase-core:17.0.0'
}
...
// Add to the bottom of the file
apply plugin: 'com.google.gms.google-services'
```

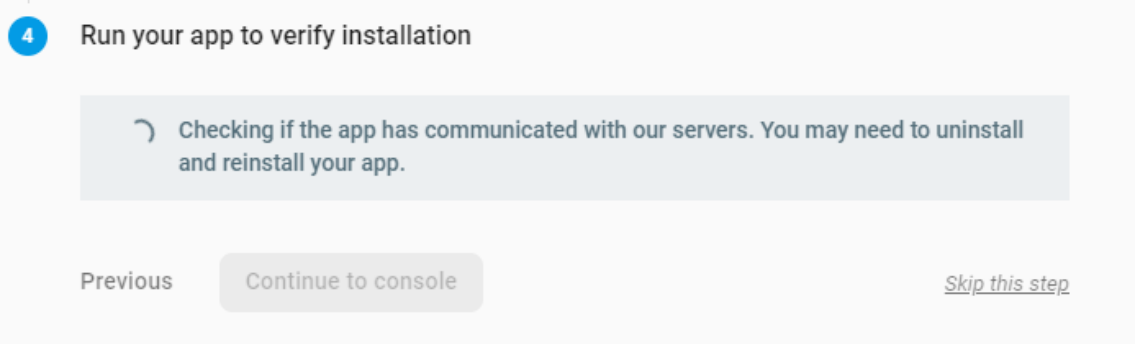
Finally, press "Sync now" in the bar that appears in the IDE:

Gradle files have changed sir | Sync now

Previous | Next

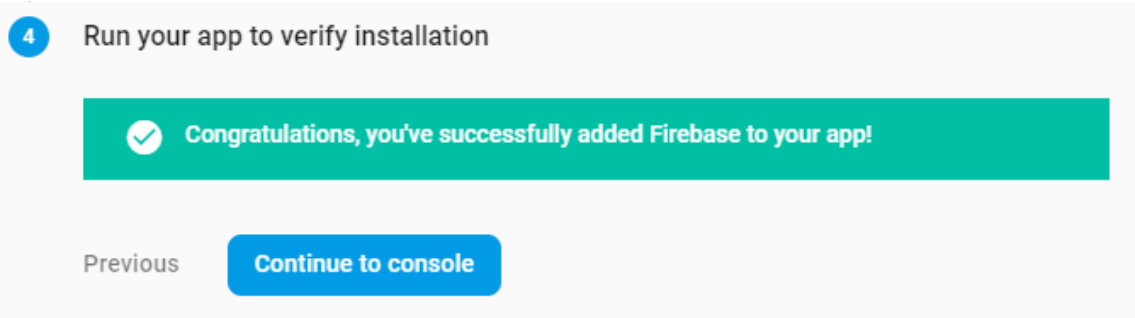
*Step 4: Modify the build.gradle in both **project** level and **app** level*

Create a Firebase Application



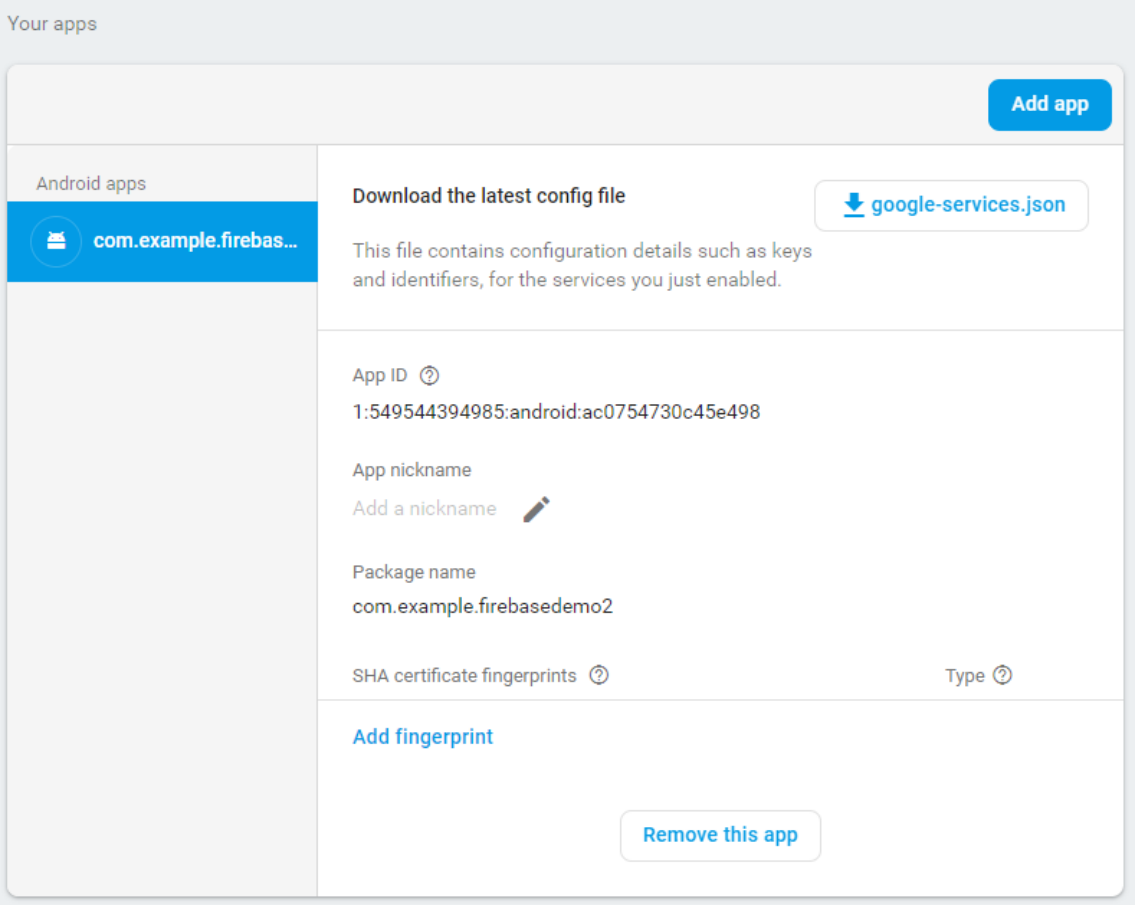
Step 5: Run your app to verify installation

Create a Firebase Application



Step 5: Run your app to verify installation

Create a Firebase Application



Finalize the installation

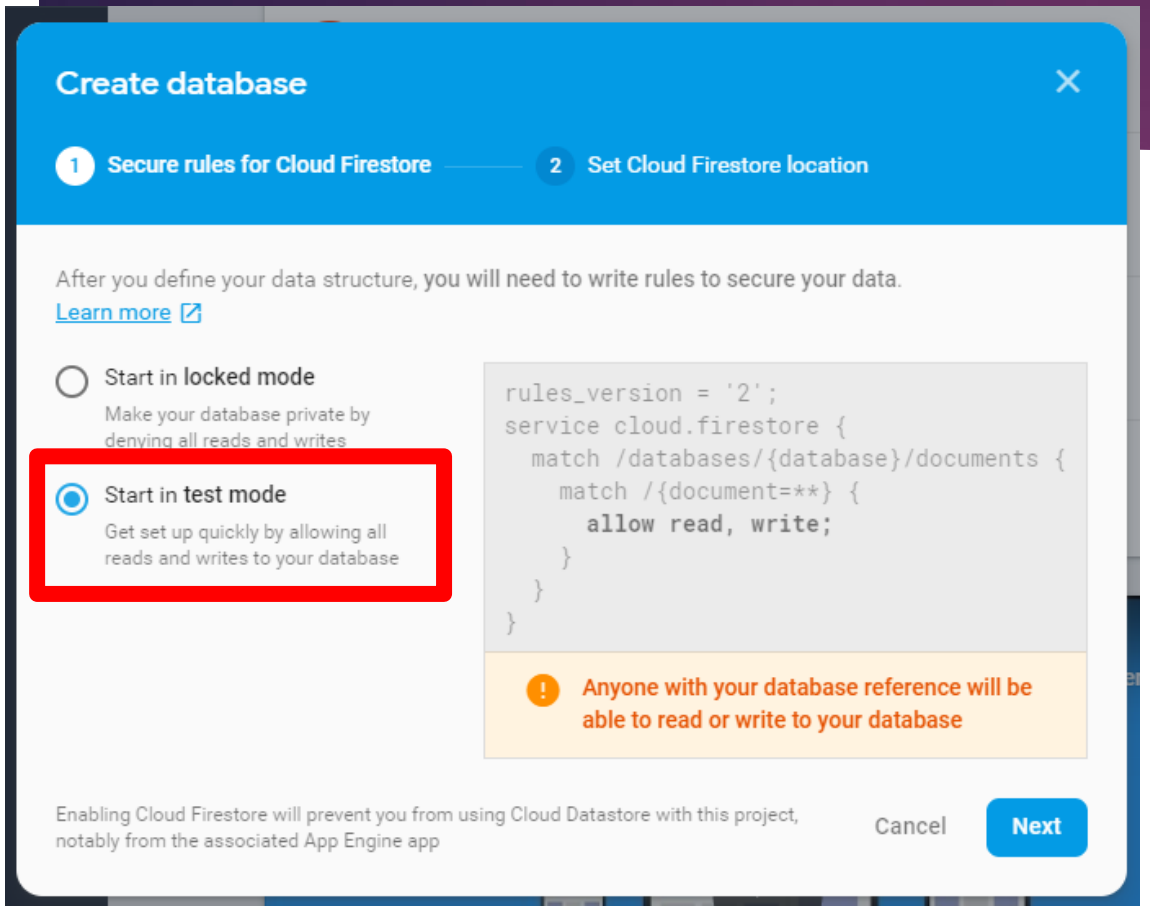
Firebase

- *Integrate Realtime Database into your application*

Integrate Realtime Database

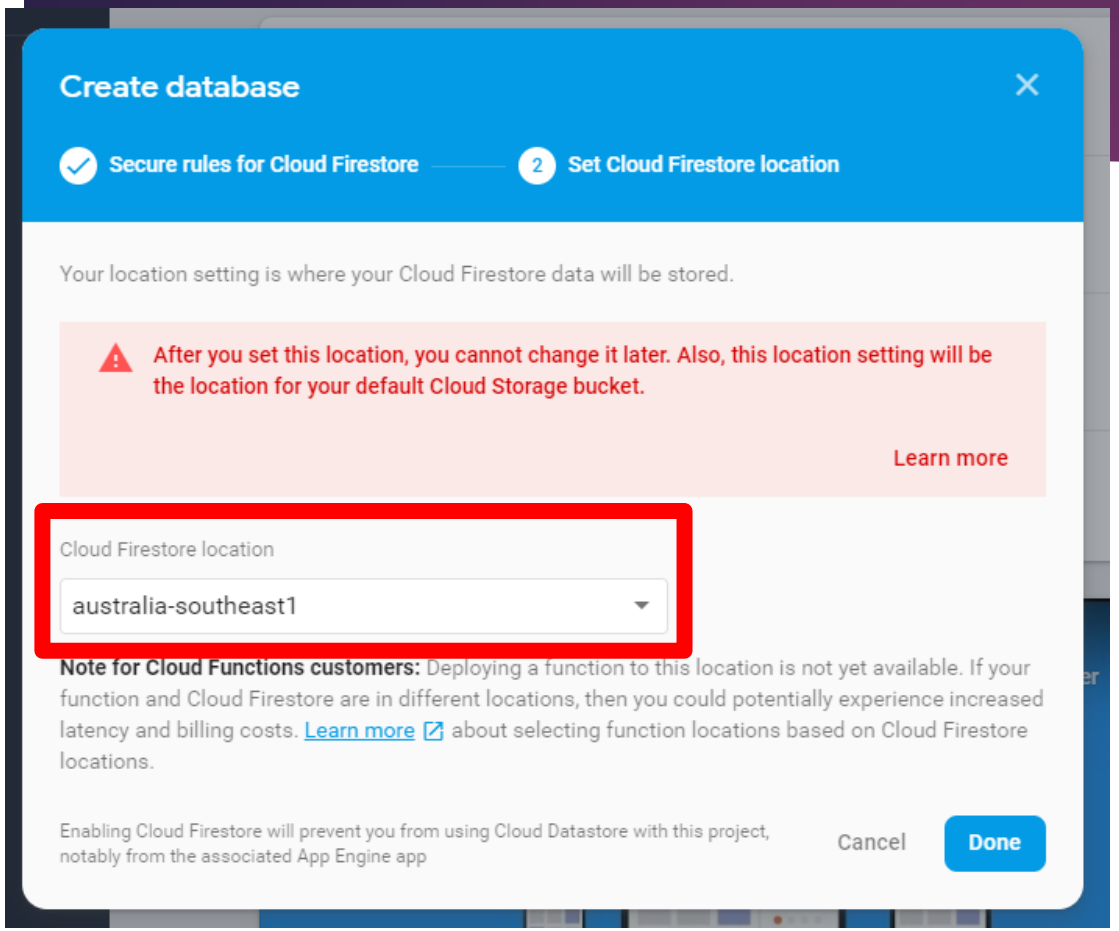
Step 1: Create a real time database from Firebase

Integrate Realtime Database



Step 2: Select the test mode

Integrate Realtime Database



Step 2: Select the location to Australia

or anywhere you want to deploy your database

Integrate Realtime Database

Step 3: Change the Rules of Realtime Database to public (for debugging your application)

Sample rules

By default, your database rules require Firebase Authentication and grant full read and write permissions only to authenticated users. The default rules ensure your database isn't accessible by just anyone before you get a chance to configure it. Once you're set up, you can customize your rules to your needs. Here are some common examples:

Default | Public | User

During development, you can use the public rules in place of the default rules to set your files publicly readable and writable. This can be useful for prototyping, as you can get started without setting up Authentication. **This level of access means anyone can read or write to your database. You should configure more secure rules before launching your app.**

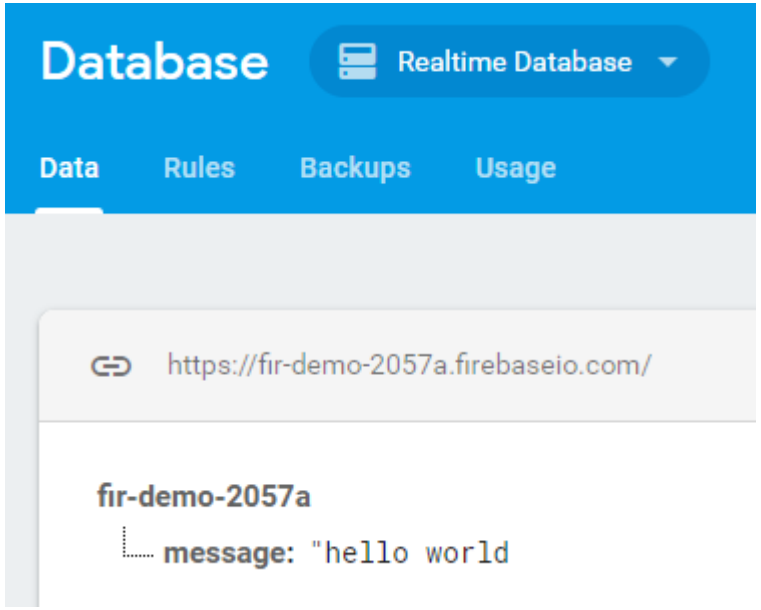
```
// These rules give anyone, even people who are not users of your app,
// read and write access to your database
{
  "rules": {
    ".read": true,
    ".write": true
  }
}
```

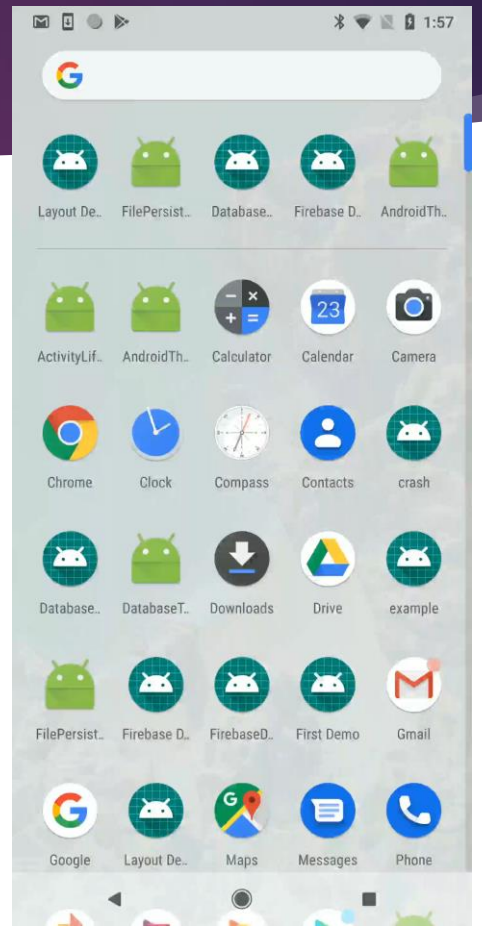
Integrate Realtime Database


```java
private void initialFirebaseRef(String reference) {
    // Write a message to the database
    FirebaseDatabase database = FirebaseDatabase.getInstance();
    myRef = database.getReference(reference);

    myRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            // This method is called once with the initial value and again
            // whenever data at this location is updated.
            String value = dataSnapshot.getValue(String.class);
            displayField.setText(value);
        }

        @Override
        public void onCancelled(DatabaseError error) {
            // Failed to read value
            Log.w(TAG, msg: "Failed to read value.", error.toException());
        }
    });
}

@OnClick({R.id.message_button})
public void sendMessage() {
    String message = inputField.getText().toString();
    myRef.setValue(message);
}
```

Step 4: Modify the Android Application to monitor the data from Realtime database.

Integrate Realtime Database





Connectivity

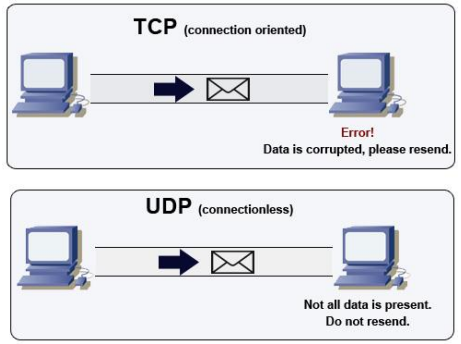
Connectivity

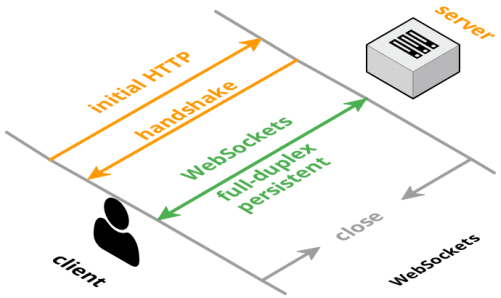
Android provides APIs to let your app connect ***standard network***, and interact other devices with protocols such as ***Bluetooth, NFC***, etc.

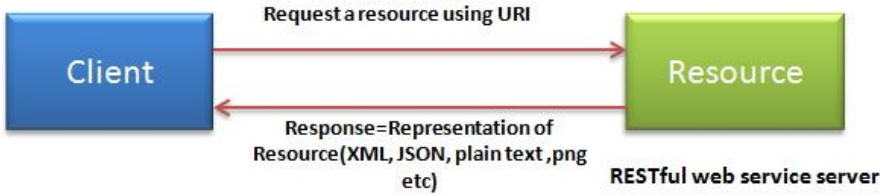
https://developer.android.com/guide/topics/connectivity

Connectivity

Bluetooth®







In this tutorial, we will focus on:

- *Bluetooth*
- *Socket, HTTP, WebSocket*

Connectivity

- *Bluetooth*

Bluetooth

Bluetooth is a wireless technology standard for ***exchanging data*** between fixed and mobile devices over short distances using short-wavelength UHF radio waves.

https://en.wikipedia.org/wiki/Bluetooth

Bluetooth 4.0

Bluetooth Low Energy (Bluetooth LE, colloquially BLE, formerly marketed as Bluetooth Smart) is intended to provide considerably ***reduced power consumption*** and cost while maintaining a similar communication range.

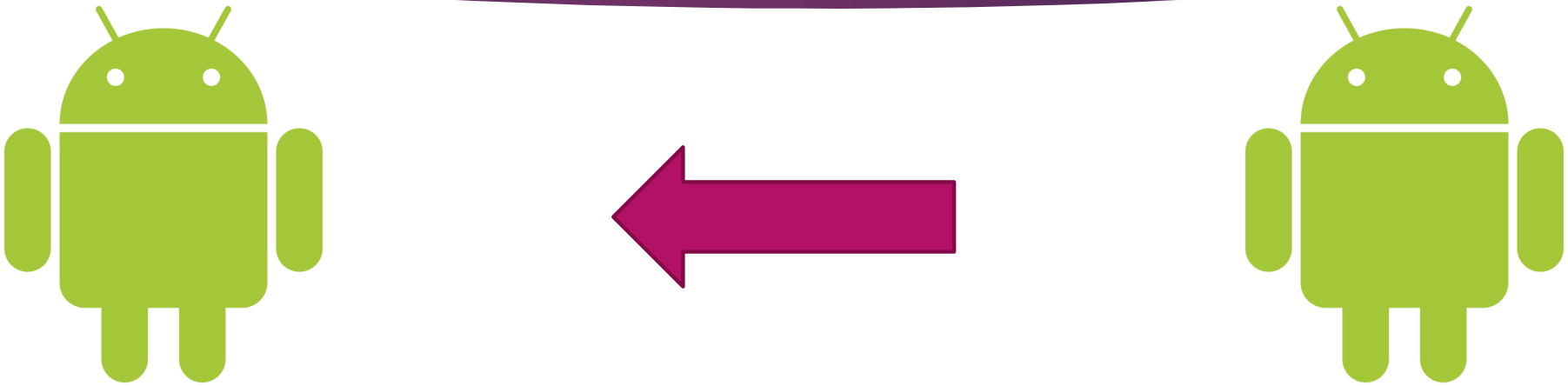
https://en.wikipedia.org/wiki/Bluetooth_Low_Energy

Bluetooth in Android

The ***Android*** platform includes support for the *Bluetooth network stack*, which allows a device to ***wirelessly exchange data*** with other Bluetooth devices.

https://developer.android.com/guide/topics/connectivity/bluetooth

Bluetooth – Build Connection



One device must be server, while another will be the client.

Bluetooth – Build Connection

Users can decide which is server

Bluetooth – Build Connection

Server

Server

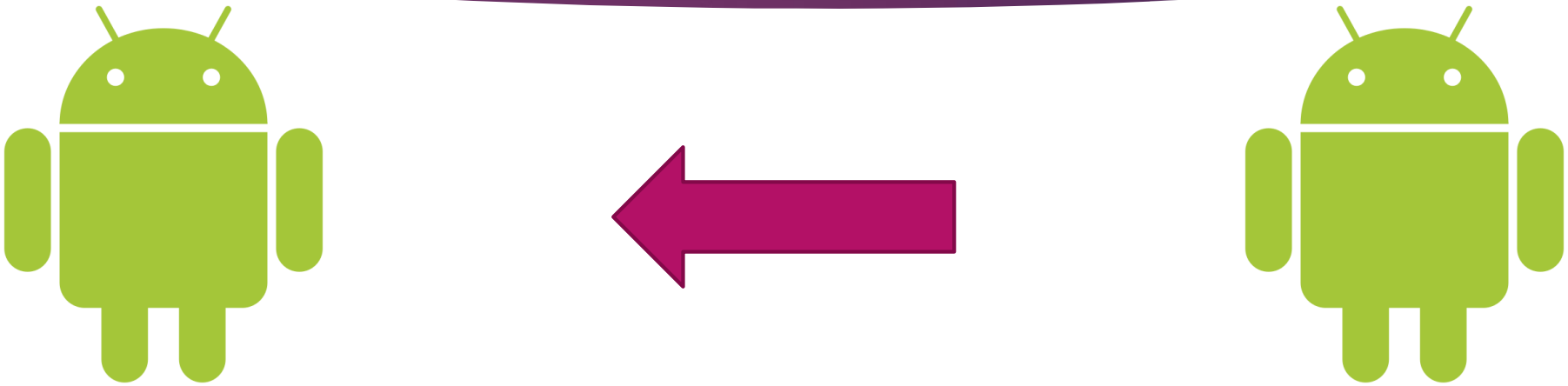
Both be server first

Bluetooth – Build Connection

Server **Request** **Server / Client**

Either requests as client

Bluetooth – Build Connection

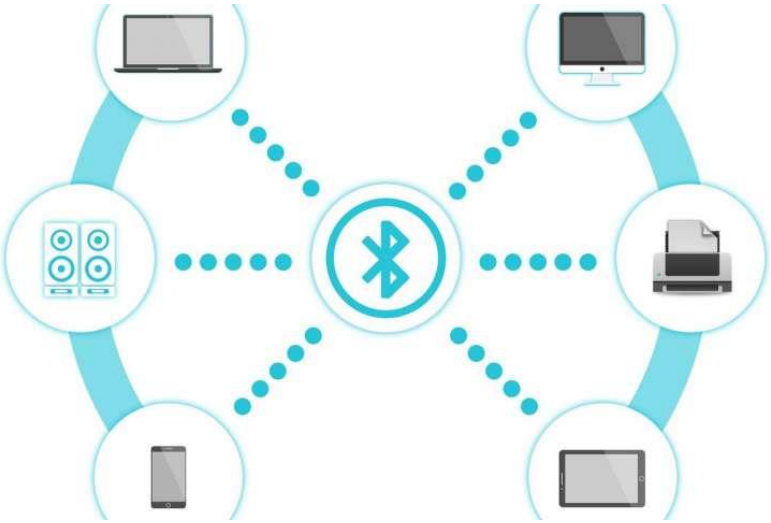


Then, you have a connection

Android Bluetooth APIs

The application framework provides access to the Bluetooth functionality through the ***Android Bluetooth APIs***.

Android Bluetooth APIs



Using the Bluetooth APIs, an ***Android application*** can ***perform***:

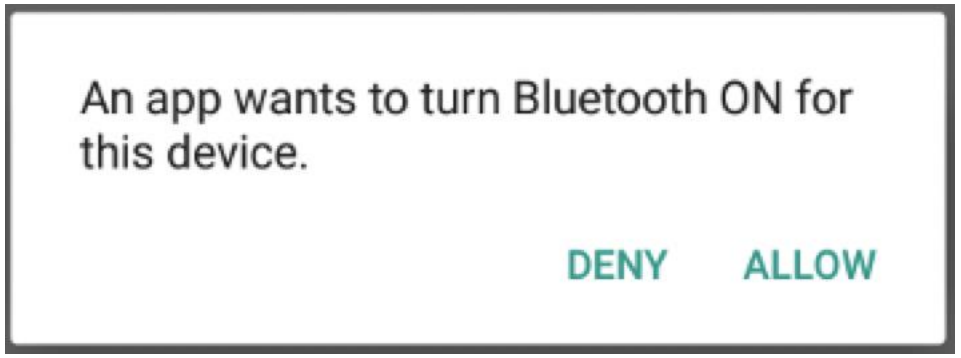
- ***Scan*** for other Bluetooth devices
- ***Query*** the local Bluetooth adapter for paired Bluetooth devices
- ***Establish*** RFCOMM channels
- ***Connect*** to other devices through service discovery
- ***Transfer*** data to and from other devices
- ***Manage*** multiple connections

Build Connection Demonstration

Step 1: Get Permissions

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

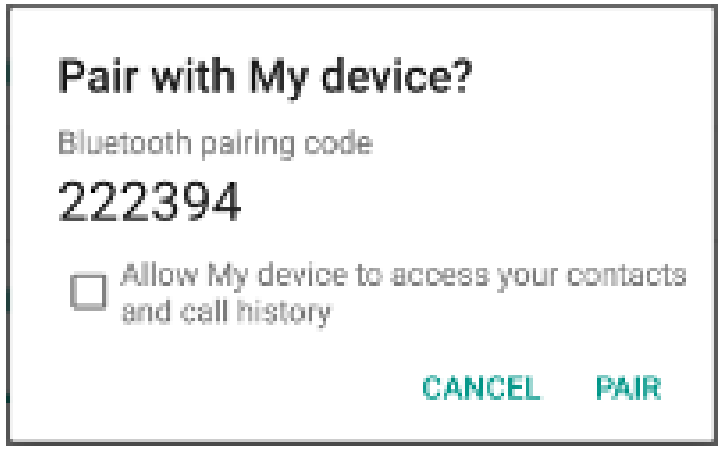
Build Connection Demonstration



Step 2: Get ***BluetoothAdapter***

- *Checks the* ***hardware availability***
- *Helps users* ***switch it on*** *if it is off*

Build Connection Demonstration



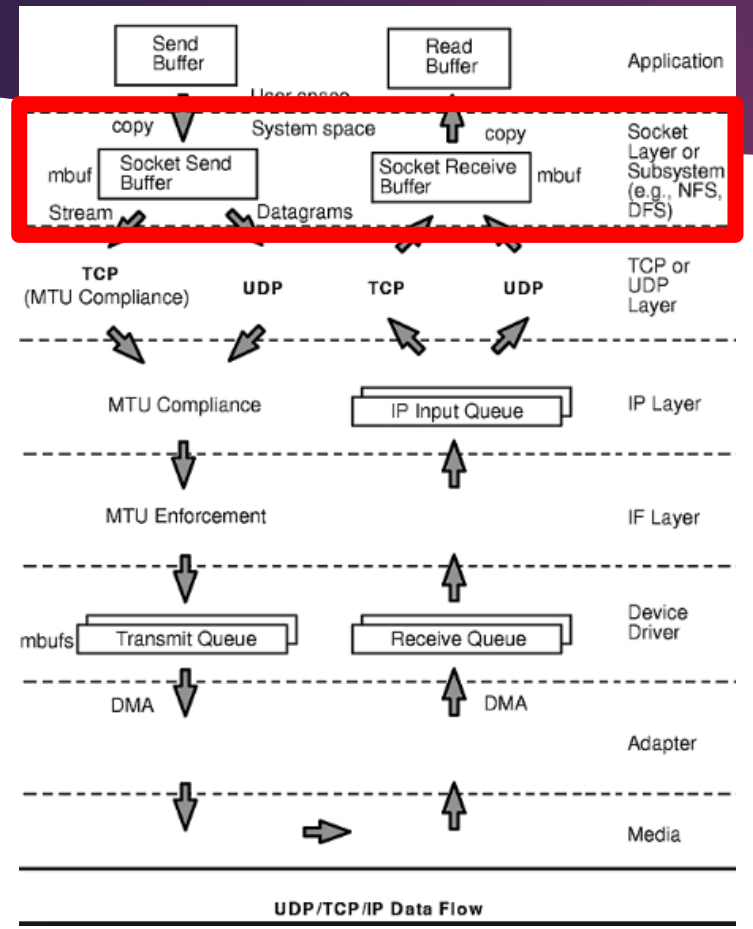
Step 3: Pair other Devices

- ***Find*** *the other nearby device (must be discoverable)*
- ***Build*** *connection using MAC address*
- ***Send/Receive*** *data*

Connectivity

- *Socket*

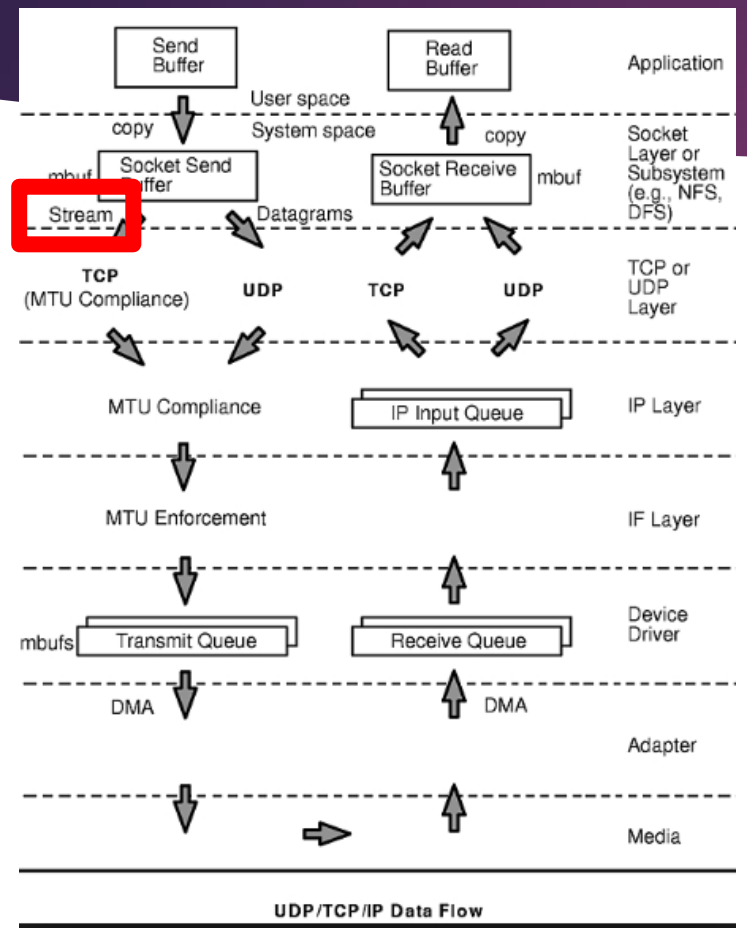
Socket – TCP/UDP



A network socket is ***an internal endpoint*** for ***sending or receiving*** data within a node on a computer network.

https://en.wikipedia.org/wiki/Network_socket

Socket – Stream Socket

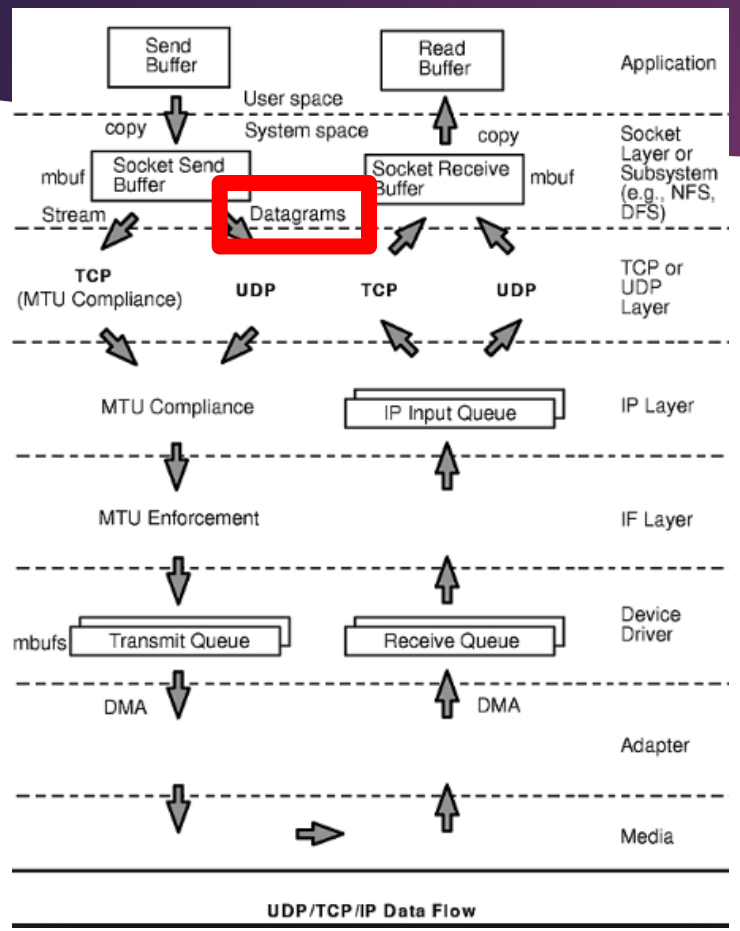


Stream sockets, also known as ***connection-oriented sockets***, which use Transmission Control Protocol (***TCP***), Stream Control Transmission Protocol (***SCTP***) ....

https://developer.android.com/reference/java/net/Socket

https://en.wikipedia.org/wiki/Transmission_Control_Protocol

Socket – Datagram Socket



Datagram sockets, also known as connectionless sockets, which use ***User Datagram Protocol (UDP)***.

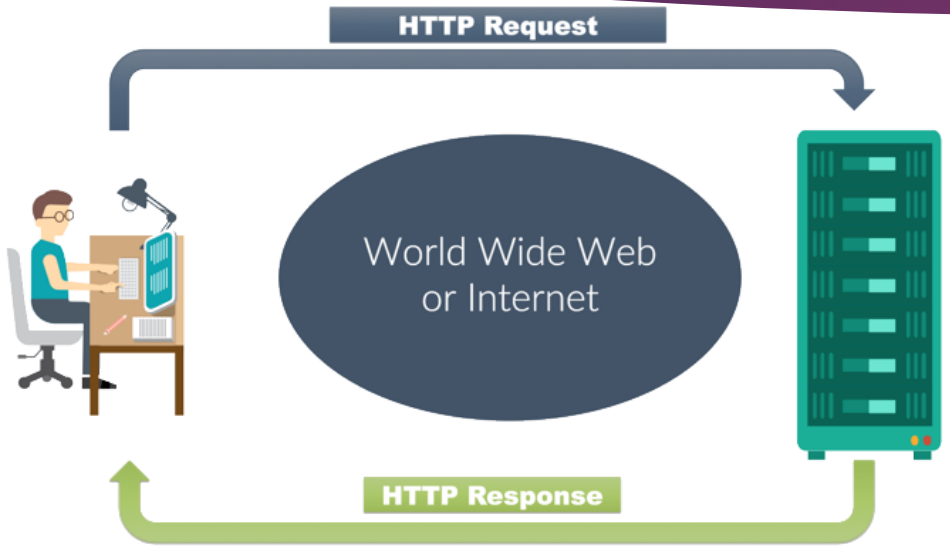
https://developer.android.com/reference/java/net/DatagramSocket

https://en.wikipedia.org/wiki/User_Datagram_Protocol

Connectivity

- *HTTP/HTTPS*

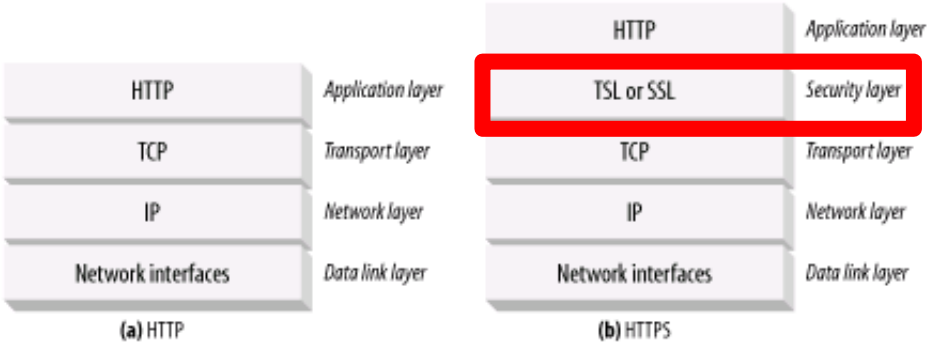
Hypertext Transfer Protocol (HTTP)



The ***Hypertext Transfer Protocol (HTTP)*** is an application protocol for distributed, collaborative, hypermedia information systems.

https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

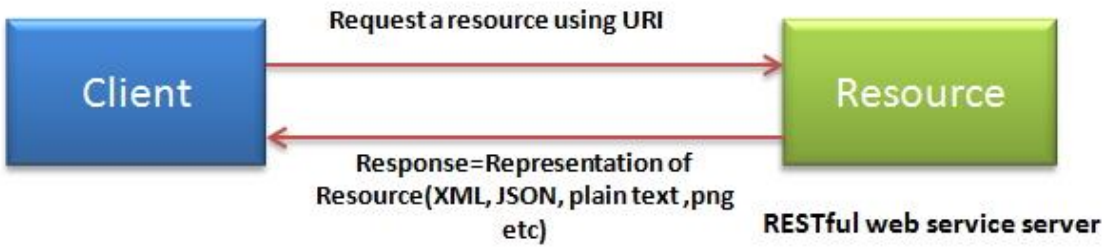
Hypertext Transfer Protocol Secure (HTTPS)



Hypertext Transfer Protocol Secure (HTTPS) is an extension of the Hypertext Transfer Protocol (HTTP). It is used for ***secure communication*** over a computer network and is widely used on the Internet.

https://en.wikipedia.org/wiki/HTTPS

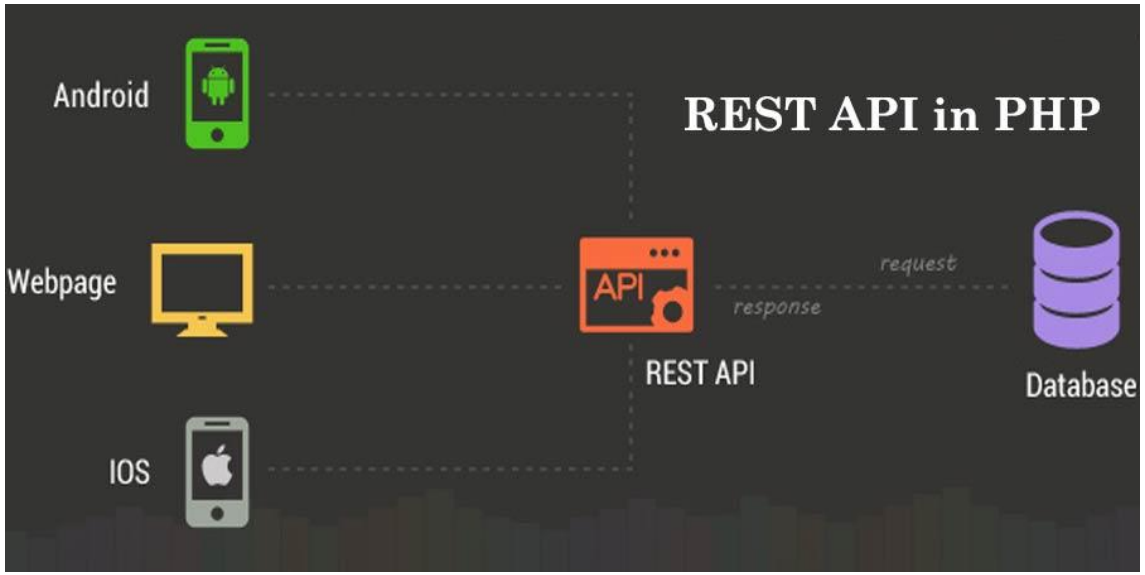
Representational state transfer (REST)



Representational State Transfer (REST) is a software architectural style that defines a set of ***constraints to be used for creating Web services***.

https://en.wikipedia.org/wiki/Representational_state_transfer

Representational state transfer (REST)

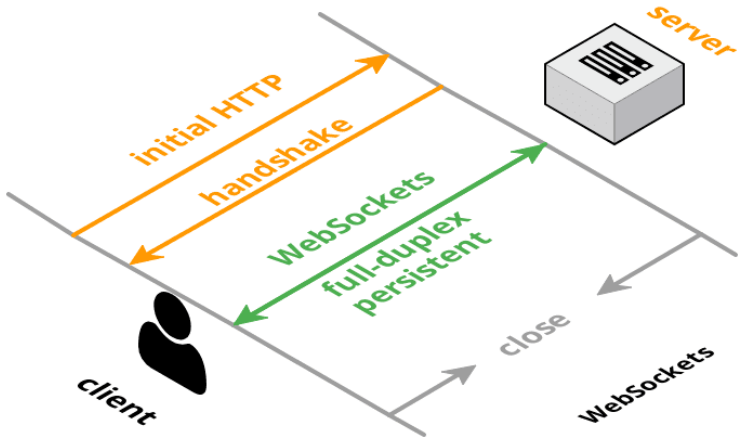


When HTTP is used, as is most common, the operations (HTTP methods) available are ***GET, HEAD, POST, PUT, PATCH, DELETE, CONNECT, OPTIONS and TRACE***.

Connectivity

- *WebSocket*

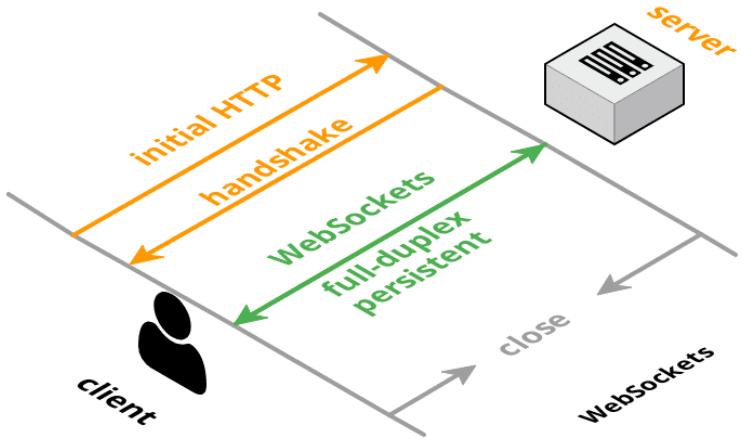
WebSocket



WebSocket is a computer communications protocol, providing ***full-duplex communication channels over a single TCP connection***.

https://en.wikipedia.org/wiki/WebSocket

WebSocket



The ***WebSocket*** protocol enables interaction between a ***web browser*** (or other ***client application***) and ***a web server*** with ***lower overhead than half-duplex alternatives*** such as HTTP polling, facilitating ***real-time data transfer*** from and to the server.

Connectivity

- *Framework*

OKHttp

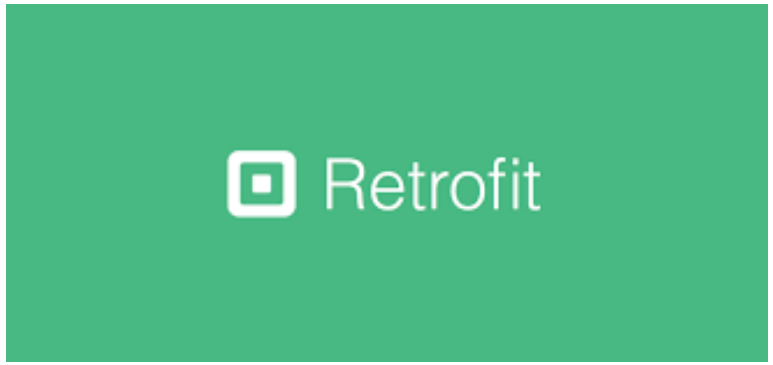
OkHttp

OkHttp is an ***HTTP client*** that's efficient by default:

- HTTP/2 support allows all requests to the same host to ***share a socket***.
- Connection pooling ***reduces request latency*** (if HTTP/2 isn't available).
- Transparent GZIP shrinks ***download sizes***.
- Response caching ***avoids*** the network completely for ***repeat requests***.

https://square.github.io/okhttp/

Retrofit



A type-safe ***HTTP client*** for Android and Java based on ***OKHttp***, *which is able to manipulate URL, Request, loading, caching, etc.*

https://square.github.io/retrofit/

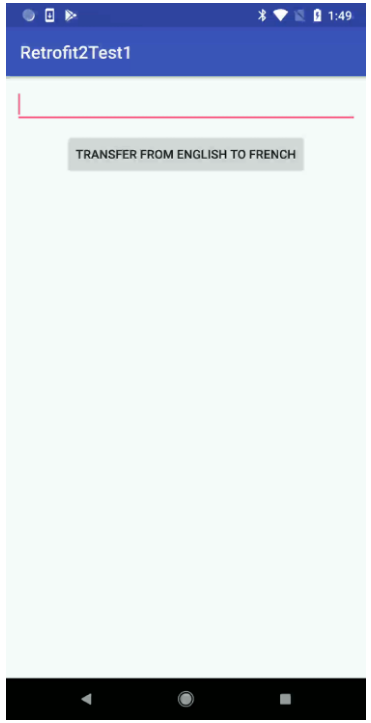
https://github.com/square/retrofit

RxJava – An alternative way of threading

RxJava is a Java VM implementation of Reactive Extensions: a library for composing ***asynchronous*** and ***event-based programs*** by ***using observable sequences***.

https://github.com/ReactiveX/RxJava

Retrofit – Demonstration



THE UNIVERSITY OF MELBOURNE

Thank you