Accessing Remote Recourses

EAST WEST UNIVERSITY
Department of
Computer Science & Engineering

Available Techniques

- Android offers several ways to leverage Internet resources
  - Using WebView
  - Using Client-side SDKs
    - Google or any other third party
  - Using Own Processing Technique

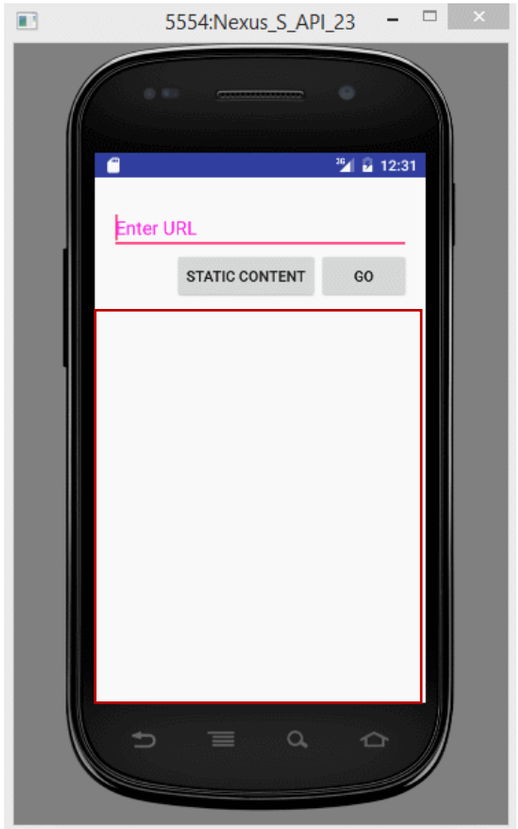

Image Source: thedroidlady.com

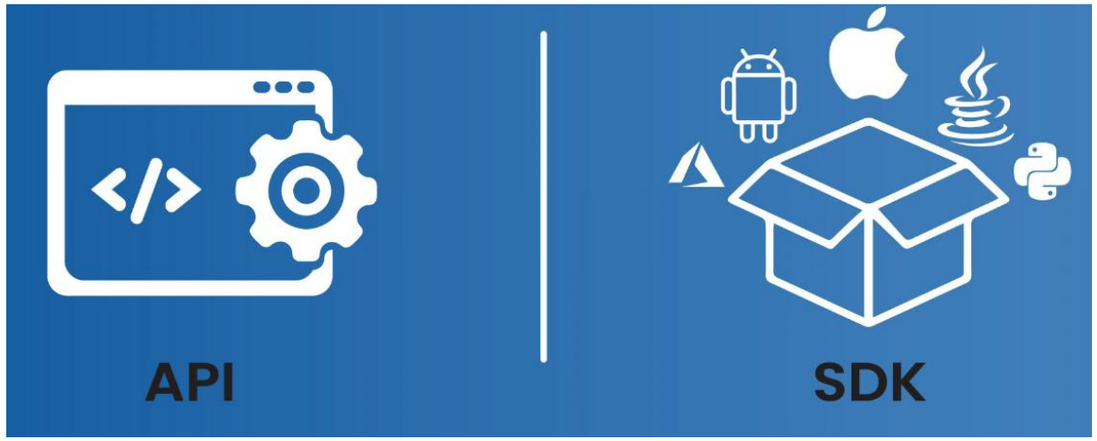

Image Source: alphabravodevelopment.com

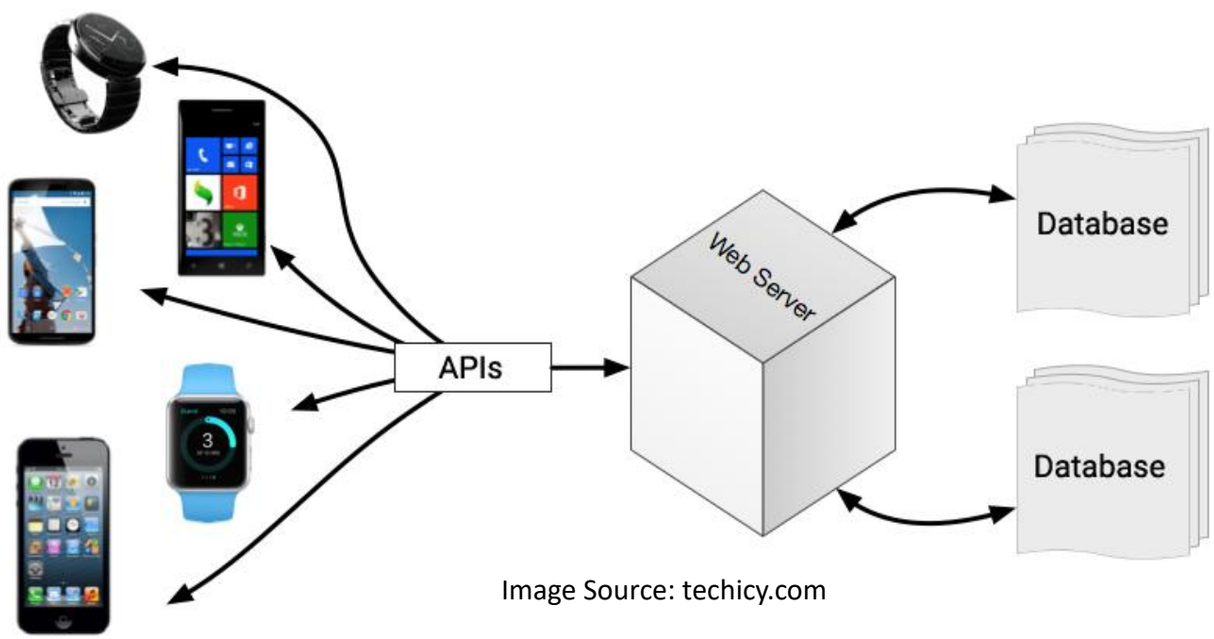

Image Source: techicy.com

WebView

- Includes a WebKit-based browser within an Activity

```
<WebView
        id=”@+id/webview"
            layout_height="..."
            layout_width="..."
            src="http://www.edubd.edu"/>

WebView webview = findViewById(R.id. webview)
```

Image Source: thedroidlady.com

- Works like a web browser
  - Renders and displays HTML contents
  - Load HTML local data
    - webview.loadData(htmlData, mimeType, encoding)
    - webview.loadDataBaseURL(baseURL, htmlData, mimeType, encoding, historyURL)
  - Load HTML remote data
  - webview.loadUrl(url)

WebView: Benefits

- No need to handle remote connection, sending or receiving data from the connection
- Developing a single website (albeit responsive) for any kind of devices (computer/mobile/tab) is enough

Image Source: thedroidlady.com

Client-side SDKs

- SDKs feature pre-written lines of code that can be incorporate in your own project
- Interact directly with remote server/database processes through SDK
- **Examples**
  - Google Firebase APIs
  - Google Firebase Cloud Messaging API
  - Maps API
  - Billing API
  - Payment APIs
  - Many others…

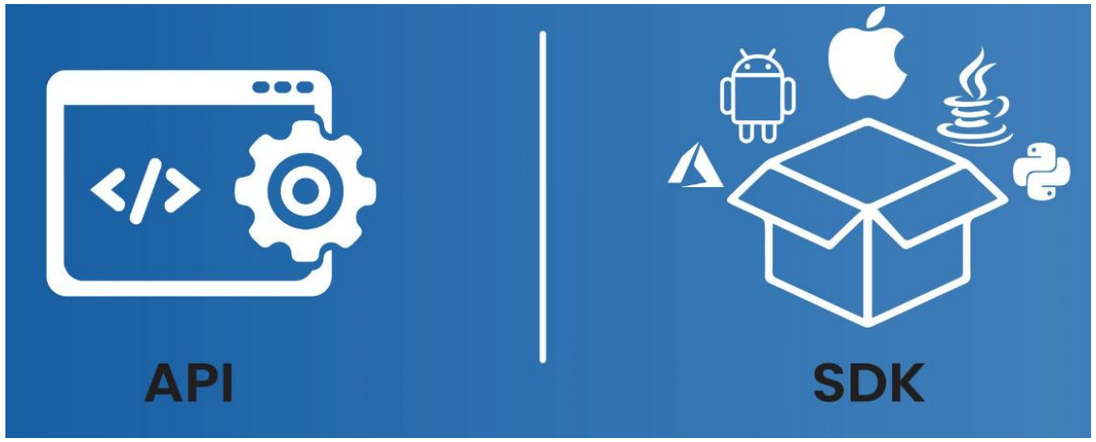

Image Source: alphabravodevelopment.com

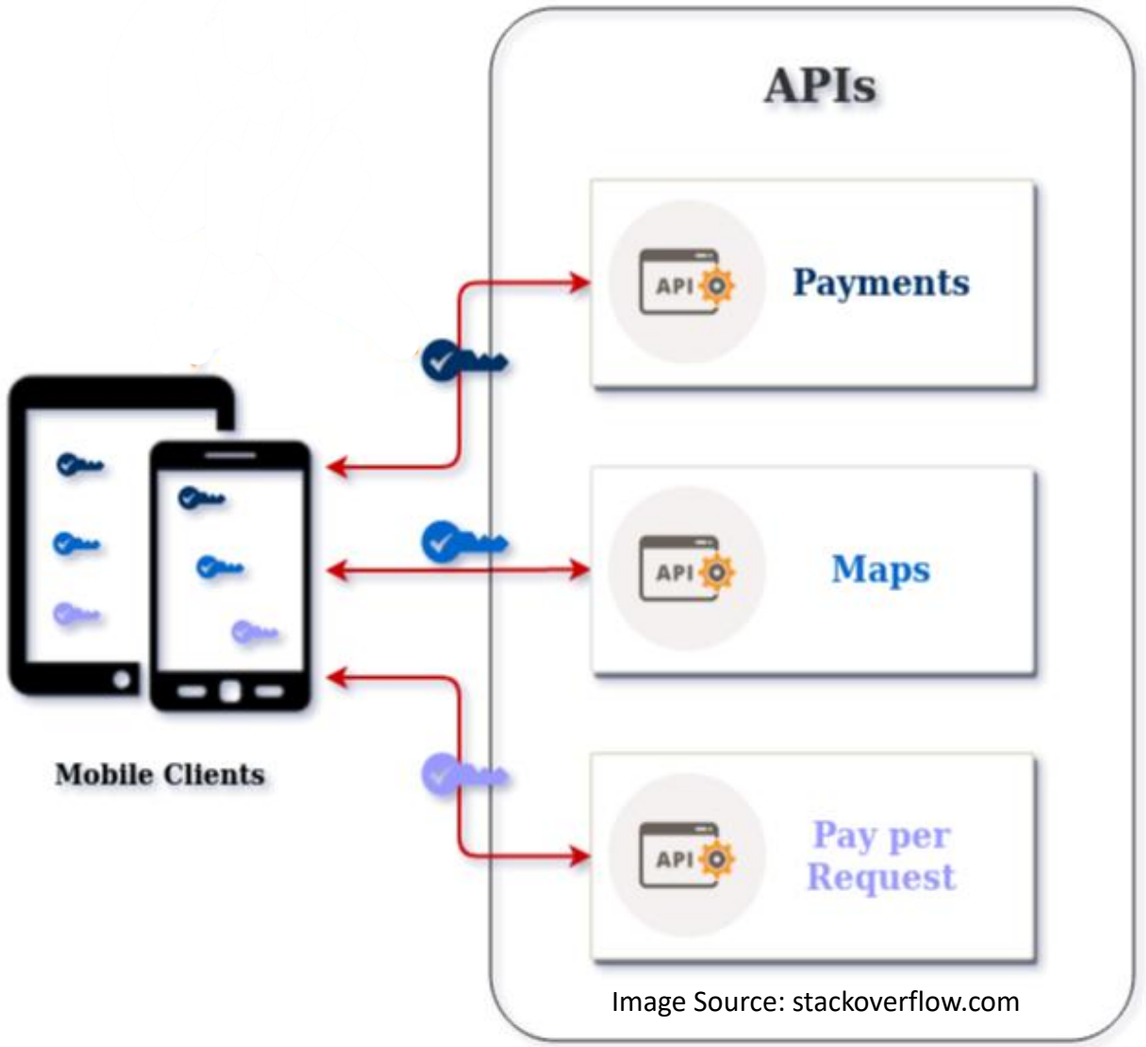

Image Source: stackoverflow.com

Client-side SDKs: Benefits

- No need to handle remote connection, sending or receiving data from the connection by the developer

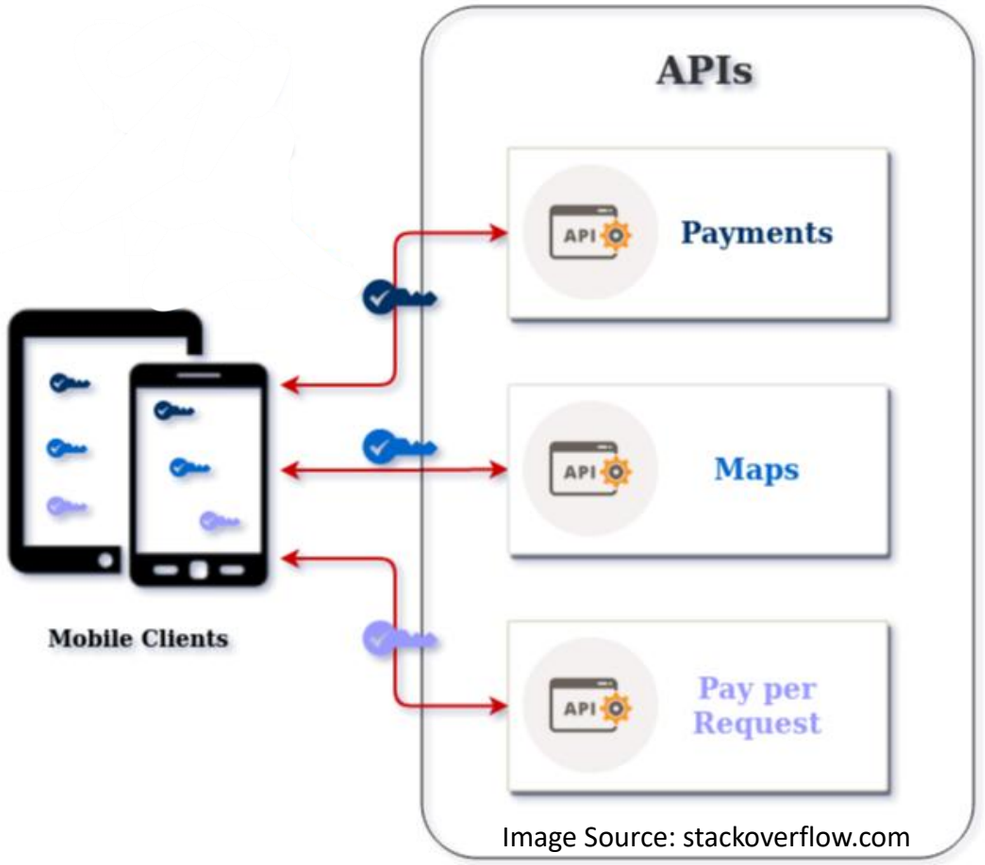

Image Source: stackoverflow.com

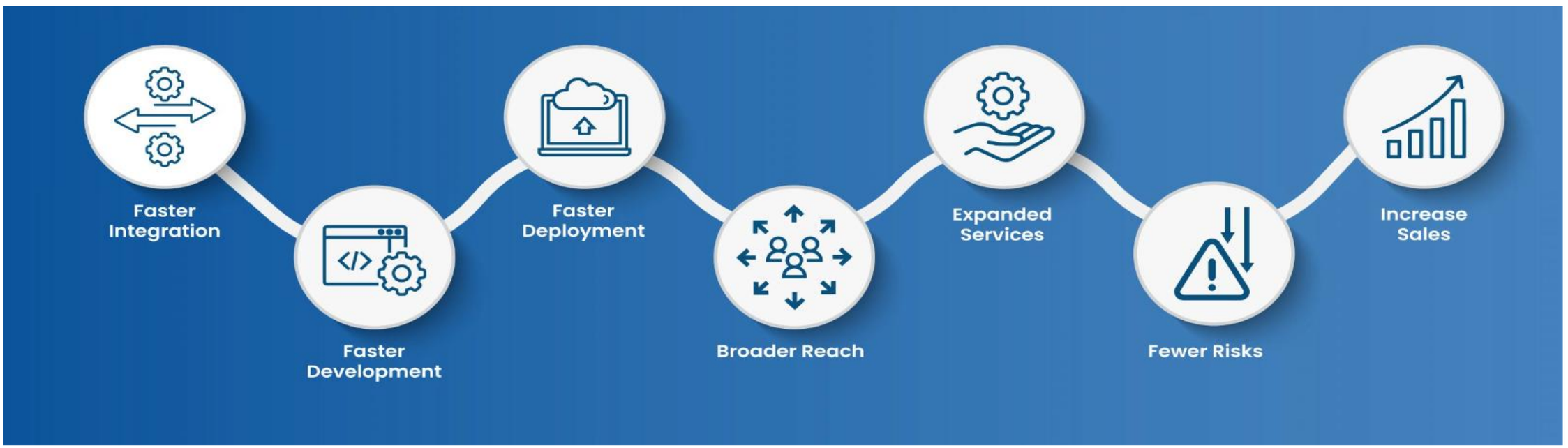

Image Source: alphabravodevelopment.com

Client-side SDKs: Issues

- There may always someone waiting to ambush you

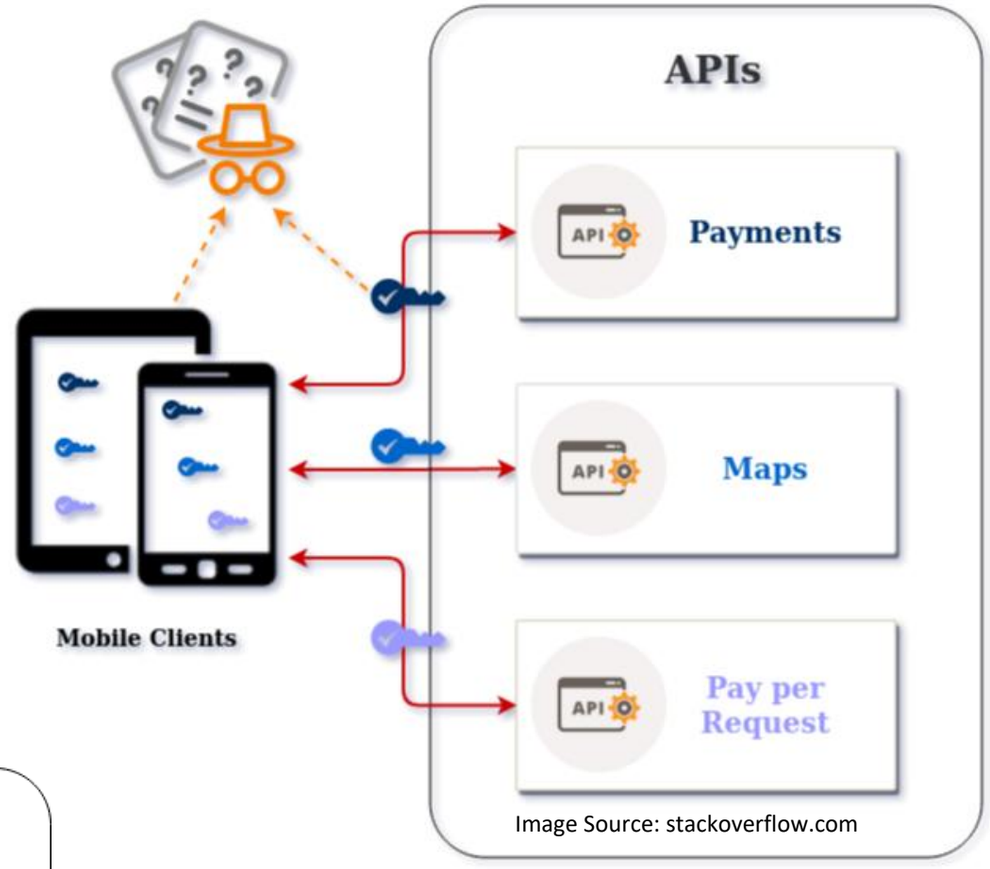

Image Source: stackoverflow.com

Better Approach

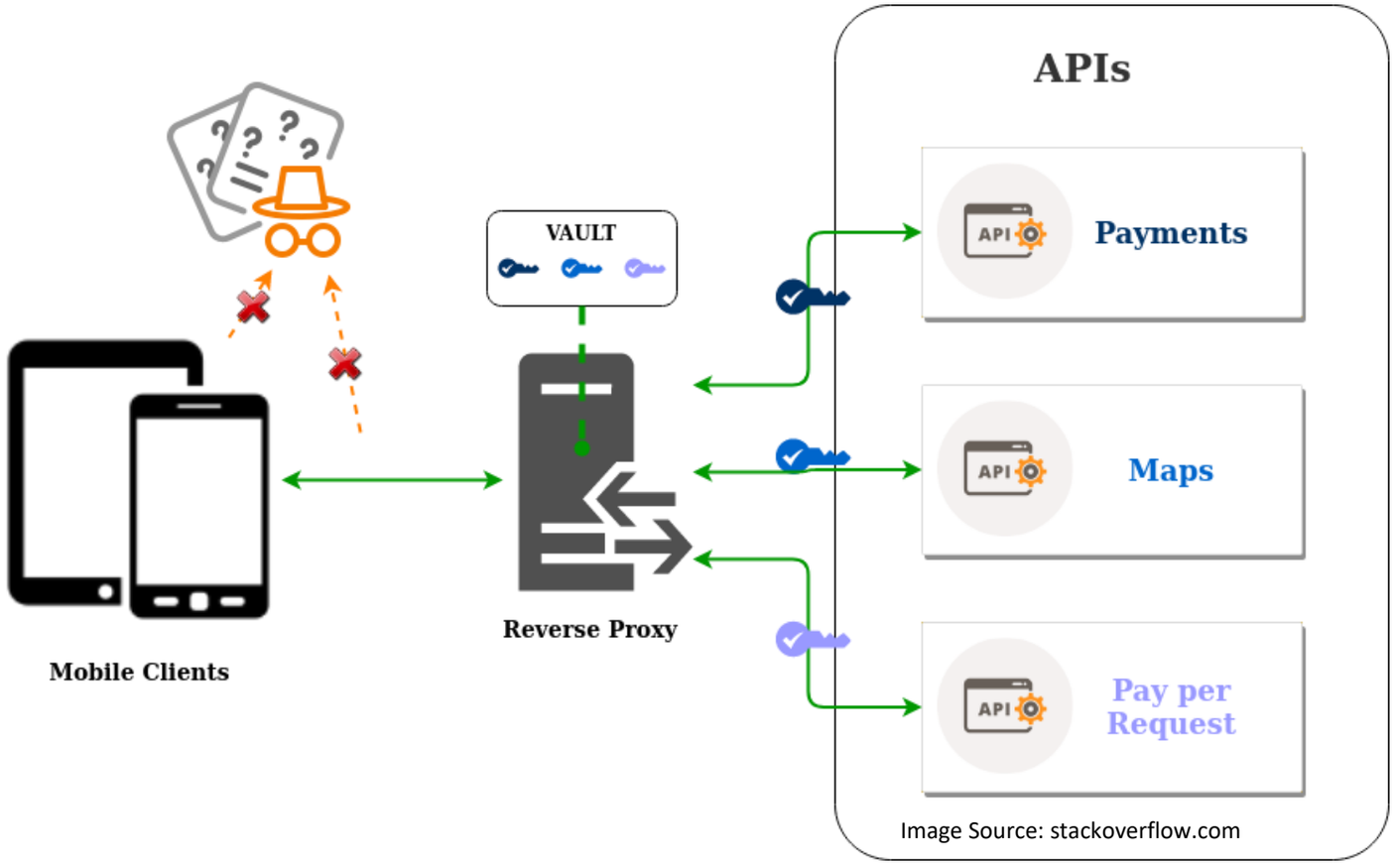

Image Source: stackoverflow.com

Own Processing Technique

- Processes remote XML/HTML feeds to extract and process data using a Java-based XML parser
  - Such as SAX or the XML Pull Parser
- Or processes remote text/json formatted data to extract and process data using a Java-based Json parser
- Handles RAW data

- App requests to web services, which are implemented using java/php/python/nodejs/ etc

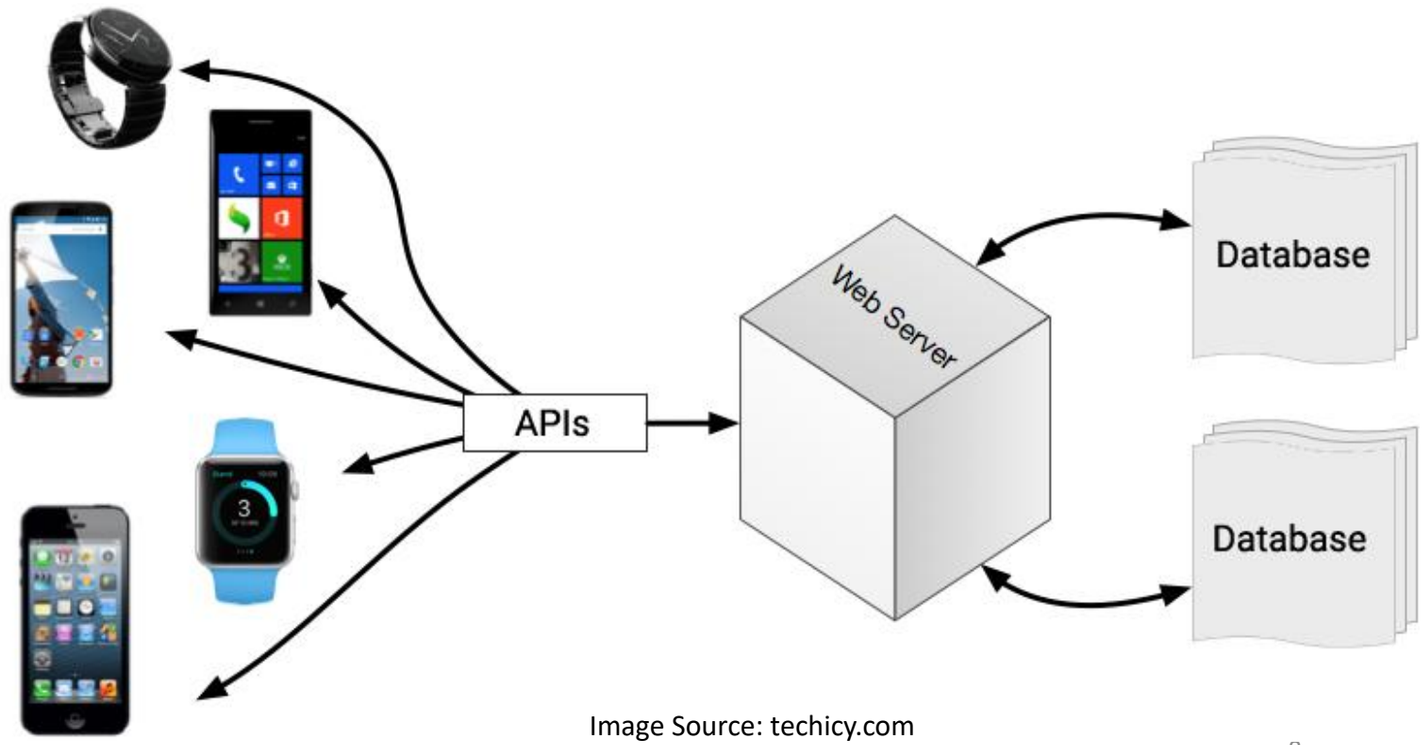

Image Source: techicy.com

How does Own Processing Technique Work?

- Open http(s) connection
- Read streaming data from the remote source
  - Remote source can be your own private server or others
    - i.e. geolocation → geoname
- Accumulate stream data
  - Extract and process the accumulated data

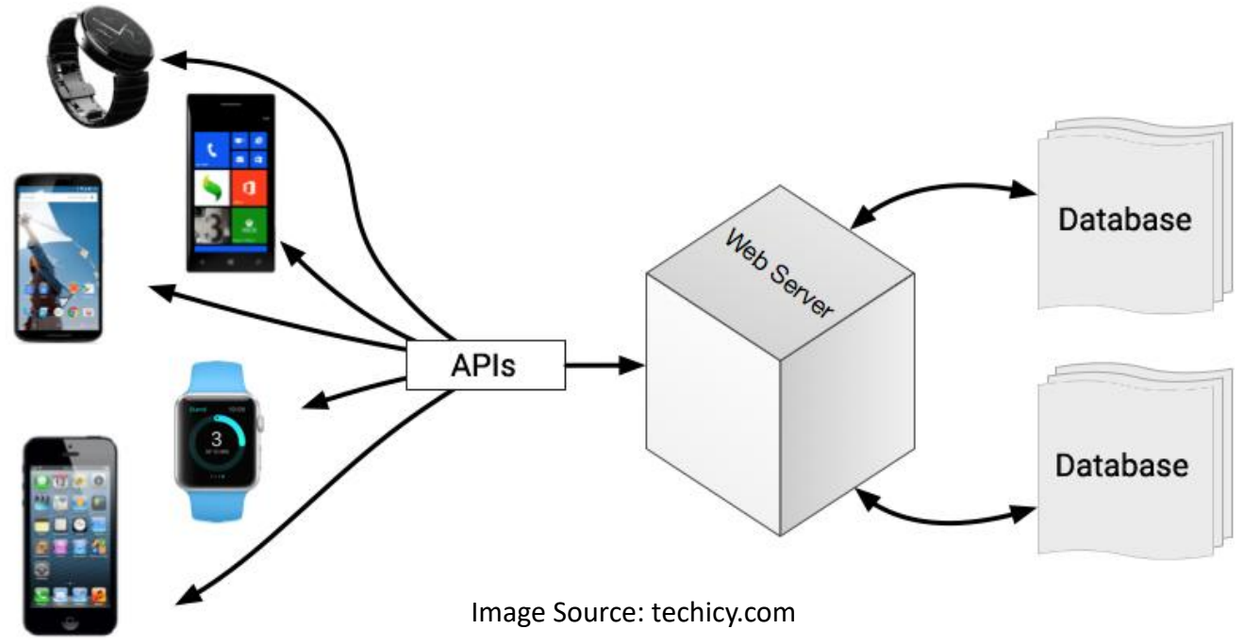

Image Source: techicy.com

- **Concerns**
  - Data format/encoding must be known
  - Caching or buffering is developer's concern
  - Implementing end-to-end processing
  - Maintenance of codes in both client app and server

Why do we need Own Processing Technique?

- With Internet connectivity and a WebKit browser,
  - Is there any reason to create native Internet-based applications when you could make a web-based version instead

Cost Optimization

- There are a number of benefits to creating thick- and thin-client native applications rather than relying on entirely web-based solutions:
- **Bandwidth** — Static resources such as images, videos, and sounds can be expensive on devices with bandwidth restraints
  - By creating a native application, you can limit the bandwidth requirements to changed data only
- **Reducing battery drain** — Each time your application opens a connection to a server, the wireless radio will be turned on (or kept on)
  - A native application can bundle its connections, minimizing the number of connections initiated
  - The longer the period between network requests, the longer the wireless radio can be left off

Ensuring Richer User Experience

- **Caching** — With a browser-based solution, an irregular Internet connection can result in discontinuous application
  - A native application can cache data and user actions to provide as much functionality as possible without a live connection and synchronize with the cloud when a connection is reestablished
  - Optimize the user experience by limiting the quantity of data transmitted and ensure that your application is robust enough to handle network outages and bandwidth limitations
- **Buffering** — Example scenario??
- **Native features** — Android devices are more than simple platforms for running a browser
  - They include location-based services, Notifications, widgets, camera hardware, background Services, and hardware sensors
  - By creating a native application, you can combine the data available online with the hardware features available on the device to provide a richer user experience

Alternatives for accessing the Internet

- Modern mobile devices offer a number of alternatives for accessing the Internet
  - **Mobile Internet** — GPRS, EDGE, 3G, 4G, and LTE Internet access is available through carriers that offer mobile data.
  - **WiFi** — Wi-F receivers and mobile hotspots are becoming increasingly common.

- If you use Internet resources in your application, remember that your users' data connections are dependent on the communications technology available to them
  - EDGE and GSM connections are notoriously low-bandwidth, whereas a Wi-Fi connection may be unreliable in a mobile setting

Opening an Internet Data Stream

- Whatever you do such as downloading an image/audio/video or text file or loading a web page, you need to open an internet data stream

```
String myFeed = getString(R.string.my_feed);
try {
  URL url = new URL(myFeed);

  // Create a new HTTP URL connection
  URLConnection connection = url.openConnection();
  HttpURLConnection httpConnection = (HttpURLConnection)connection;

  int responseCode = httpConnection.getResponseCode();
  if (responseCode == HttpURLConnection.HTTP_OK) {
    InputStream in = httpConnection.getInputStream();
    processStream(in);
  }
}
catch (MalformedURLException e) {
  Log.d(TAG, "Malformed URL Exception.");
}
catch (IOException e) {
  Log.d(TAG, "IO Exception.");
}
```

Connecting to Internet

- First thing first
  - Before you can access Internet resources, you need to add an INTERNET uses-permission node to your application manifest, as shown in the following XML snippet:

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

<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
```

Problem Discussion

- **Two devices**
  - Both are connected to the same WiFi router
  - Router is not connected to the internet (cable is unplugged)
  - Both devices are not capable to access internet via network data
- Can these two devices communicate with each other?

Thanks!