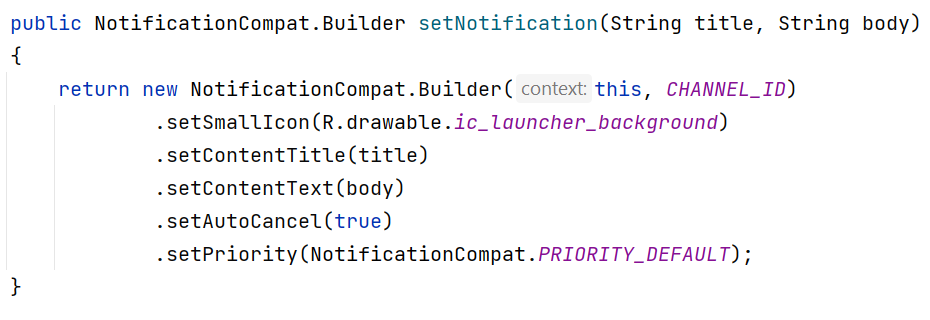
**GoogleApiClient**

* GoogleApiClient là thư viện được Google hỗ trợ để tích hợp các dịch vụ của Google. GoogleApiClient cung cấp một điểm kết nối chung đến các dịch vụ của Google Play để quản lý các kết nối giữa các thiết bị người dùng với từng dịch vụ của Google Play.
* Để bắt đầu, trước tiên phải cài đặt thư viện dịch vụ của Google Play cho Android SDK.
* Trong ứng dụng này nhóm sử dụng phương thức kết nối thủ công (**manually managed connection):**
  + Đầu tiên phải thêm các interfaces callbacks: ConnectionCallbacks và OnConnectionFailedListener, các interfaces này sẽ nhận được lệnh callbacks để phản hồi lại phương thức kết nối không đồng bộ (asynchronous connect()) khi kết nối với các Google Play Services thành công, không thành công hoặc bị tạm ngừng.
  + Khi thực hiện cần phải đặt các hàm **connect()** và **disconnect()** vào đúng chỗ trong chương trình. Để đảm bảo thì phương thức **connect()** cần phải để hàm **onStart()** và **disconnect()** nằm trong hàm **onStop()**



* Xử lý các lỗi kết nối:
  + Khi chương trình nhận phản hồi từ hàm **onConnectionFailed()** , thì nên gọi hàm **hasResolution()** của lớp **ConnectionResult** . Nếu nó trả về true, chương trình có thể yêu cầu người dùng thực hiện hành động để giải quyết lỗi ngay lập tức bằng cách gọi hàm **startResolutionForResult()** của lớp **ConnectionResult**. Phương thức **startResolutionForResult()** hoạt động giống như **startActivityForResult()** trong trường hợp này và khởi chạy một hoạt động phù hợp với ngữ cảnh để giúp người dùng giải quyết lỗi (chẳng hạn như một hoạt động giúp người dùng chọn một tài khoản).
  + Nếu **hasResolution()** trả về false, chương trình sẽ gọi hàm **GoogleApiAvailable.getErrorDialog()** và chuyển mã lỗi cho phương thức này. Thao tác này trả về Hộp thoại do các dịch vụ của Google Play cung cấp phù hợp với lỗi. Hộp thoại có thể chỉ cung cấp một thông báo giải thích lỗi hoặc nó cũng có thể cung cấp một hành động để khởi chạy một hoạt động có thể giải quyết lỗi (chẳng hạn như khi người dùng cần cài đặt phiên bản mới hơn của dịch vụ Google Play).

**Notification**

* Notification là tính năng mà Android gửi các thông báo bên ngoài giao diện ứng dụng của bạn để cung cấp cho người dùng lời nhắc, tin nhắn, hay bất kỳ thông tin gì từ ứng dụng. Người dùng có thể nhấn vào notification để mở ứng dụng, hoặc thực hiện một hành động trực tiếp trên notification như gửi tin nhắn.
* Các bước để sử dụng Notification:
  + Đầu tiên cần thiết lập nội dung và kênh của thông báo bằng đối tượng NotificationCompat.Builder
  + Trong đó:
    - **setSmallIcon():** Thiết lập Icon cho thông báo
    - **setContentTitle():** Tiêu đề thông báo
    - **setContentText():** Nội dung thông báo
    - **setAutoCancel():** tự động loại bỏ thông báo khi người dùng chạm vào nếu tham số truyền vào là **true**
    - **setPriority(NotificationCompat.PRIORITY\_DEFAULT):** thiết lập mức độ ưu tiên của thông báo.
  + Tạo kênh thông báo:



**SQLite**

* SQLite là hệ quả trị cơ sở dữ liệu (DBMS) quan hệ tương tự như Mysql, ... Đặc điểm nổi bật của SQLite so với các DBMS khác là gọn, nhẹ, đơn giản, đặt biệt không cần mô hình server-client, không cần cài đặt, cấu hình hay khởi động nên không có khái niệm user, password hay quyền hạn trong SQLite Database. Dữ liệu cũng được lưu ở một file duy nhất.
* SQLite thường không được sử dụng với các hệ thống lớn nhưng với những hệ thống ở quy mô vừa và nhỏ thì SQLite không thua các DBMS khác về chức năng hay tốc độ. Vì không cần cài đặt hay cấu hình nên SQLite được sử dụng nhiều trong việc phát triển, thử nghiệm … vì tránh được những rắc rối trong quá trình cài đặt

**Broadcast**

* Android Broadcast Receiver là một thành phần nơi bạn có thể đăng ký sự kiện của hệ thống hay ứng dụng. Bạn sẽ nhận được thông báo về các sự kiện đã đăng ký trước đó. Việc phát tin broadcast có thể bắt nguồn từ hệ thống hoặc từ các ứng dụng
* Không giống như Activity, Broadcast receiver trong android không hề có giao diện người dùng. Mặc dù nó có thể tạo thông báo trên thanh status bar.
* Các bước để tạo và sử dụng Broadcast receiver trong android
  + Bước 1 : Tạo Broadcast Receiver
  + Bước 2 : Đăng ký để nhận thông báo từ Broadcast Receiver
  + Bước 3 : Gửi Broadcast Event/Intent
* Có 2 kiểu gửi broadcasts trong Android là gửi theo cách thông thường Context.sendBroadcast(), và gửi theo thứ tự sử dụng hàm Context.sendOrderedBroadcast().
* Bên cạnh đó cần phải chú ý về permission khi sử dụng Broadcast
* Cách khai báo permission trong ***Manifest***như sau
  + **<uses-permission android:name="android.permission.RECEIVE\_BOOT\_COMPLETED" />**
* Tất nhiên, đây chỉ là khai báo thôi, từ Android M trở lên, bạn cần phải viết thêm code để yêu cầu người dùng chấp thuận permission này cho ứng dụng
* Broadcast trong ứng dụng được gửi dưới dạng hẹn thời gian, mặc định thông báo sẽ được gửi mỗi ngày vào đúng thời gian đã được thiết lập trong lúc lập trình.

**API**

* API là các phương thức, giao thức kết nối với các thư viện và ứng dụng khác. Nó là viết tắt của **Application Programming Interface**.
* Khái niệm : Khi sử dụng một ứng dụng trên điện thoại di động, ứng dụng kết nối Internet và gửi dữ liệu tới máy chủ. Sau đó, máy chủ lấy ra dữ liệu đó, diễn giải nó, thực hiện các hành động cần thiết và gửi nó trở lại điện thoại. Ứng dụng sau đó sẽ diễn giải dữ liệu đó và trình bày thông tin bạn muốn theo cách có thể đọc được.
* Các API hiện đại ngày nay
  + Các API hiện đại tuân thủ các tiêu chuẩn (thường là HTTP và REST), có tính dễ sử dụng và dễ hiểu và thân thiện với các nhà phát triển.
  + API được xử lý giống như sản phẩm hơn là code. Chúng được thiết kế cho các đối tượng cụ thể (ví dụ: nhà phát triển thiết bị di động).
  + Vì chúng được chuẩn hóa nhiều hơn, nên tính bảo mật và quản trinh mạnh hơi, cũng như được theo dõi và quản lý hiệu suất, quy mô tốt hơn.
  + Như bất kỳ phần mềm sản phẩm nào khác, API hiện đại có chu kỳ phát triển phần mềm (SDLC) riêng của nó về thiết kế, thử nghiệm, xây dựng, quản lý.
* Ứng dụng sử dụng API của <https://openweathermap.org/> trang web chuyên cung cấp miễn phí các API về thời tiết ở khắp nơi trên thế giới.

**JSON**

* JSON là viết tắt của JavaScript Object Notation, là một kiểu định dạng dữ liệu tuân theo một quy luật nhất định mà hầu hết các ngôn ngữ lập trình hiện nay đều có thể đọc được. JSON là một tiêu chuẩn mở để trao đổi dữ liệu trên web.

**Cấu trúc chuỗi JSON**

* Object trong Json được thể hiện bằng dấu ngoặc nhọn {}. Khái niệm Object trong Json cũng khá tương đồng với Object trong Javascript. Tuy nhiên, Object trong Json vẫn có những giới hạn như:
  + Key: phải luôn nằm trong dấu ngoặc kép, không được phép là biến số.
  + Value: Chỉ cho phép các kiểu dữ liệu cơ bản: numbers, String, Boolean, arrays, objects, null. Không cho phép function, date, undefined.
  + Không cho phép dấu phẩy cuối cùng như Object trong Javascript.

**AsyncTask**

* Android là hệ điều hành đa tiến trình, đa luồng. Các ứng dụng Android cũng có thể sử dụng được nhiều thread (luồng) tài nguyên hệ thống.
* Tuy nhiên các thành phần liên quan đến giao diện người dùng đều được nhét chung vào một thread chính, gọi là UI Thread (hay Main Thread).
* Các thao tác xử lý nặng hầu hết đều bị cấm thực thi trên Main Thread do sẽ gây ra trải nghiệm xấu cho người dùng (bấm nút mãi không lên, chạy mãi cứ treo là hỏng rồi, người ta sẽ xóa app ngay).
* Do đó cần đưa các tác vụ này vào một Thread khác, gọi là Worker Thread. Và sau đó là cập nhật vào Main Thread.
* Cách làm này được gọi là thao tác **xử lý bất đồng bộ** (Asynchronous).
* Các bước sử dụng AsyncTask
* Bước 1 : Tạo một class mới, extends từ lớp **AsyncTask**, tạm gọi là **AsyncTask Network**
  + **Vd: public class AsyncTaskNetwork extends AsyncTask<String, Void, Boolean> {}**
* Bước 2 : Viết các hàm quan trọng xử lý tiến trình như onPreExecute, doInBackground, onPostExecute
  + Trong đó :
    - **onPreExecute**: Tác vụ này vẫn chạy ở UI Thread, nó sẽ chạy trước khi bắt đầu tác vụ bất kỳ nào đó trong doInBackground. Thích hợp nhất là đặt ở đây một ProgressDialog (một thông báo nhỏ quay quay) để cho người dùng biết là đang có tác vụ xử lý.
    - **doInBackground**: Các thao tác về mạng sẽ được đặt vô đây. doInBackground là task chạy trên Worker Thread.
    - **onPostExecute**: Tác vụ này chạy ở UI Thread, lấy dữ liệu từ doInBackground về sau khi xử lý.
  + <String, Void, Boolean> trong phần kế thừa AsyncTask có ý nghĩa là
    - **String**: Đây là kiểu dữ liệu được truyền vào doInBackground.
    - **Void**: Đây là kiểu dữ liệu mà ta truyền vào một hàm đặc biệt có tên publishProgress(). Hàm này có sẵn trong lớp AsyncTask nhưng chúng ta không sử dụng đến. Nó dùng để cập nhật xem tác vụ đã chạy đến đâu rồi.
    - **Boolean**: Kiểu dữ liệu trả về sau khi hoàn thành AsyncTask.
* Ứng dụng của nhóm có sử dụng AsyncTask để đọc đối tượng JSON từ máy chủ cung cấp dữ liệu và hiển thị lại dưới dạng các listView, textView, …