

Universidad Nacional Autónoma de México.

Facultad de Contaduría y Administración.



Desarrollo de aplicaciones móviles.

Actividad-Optativa-02.

David Jerónimo Rojas Avalos.

Fecha: -----.

¿Qué es un Servicio Web REST?

REST, que significa *Representational State Transfer* (Transferencia de Estado Representacional), es un estilo arquitectónico para diseñar servicios web que permiten la comunicación entre sistemas a través del protocolo HTTP, uno de los protocolos más utilizados en Internet.

Un servicio web REST expone recursos (datos o funcionalidades) que pueden ser accedidos y manipulados mediante operaciones HTTP estándar como:

- GET: para obtener datos.
- POST: para crear nuevos datos.
- PUT: para actualizar datos existentes.
- DELETE: para eliminar datos.

Estos servicios suelen devolver la información en formatos estructurados, siendo JSON (JavaScript Object Notation) el más común, por su facilidad de lectura tanto para máquinas como para humanos.

Relación entre Servicios Web REST y Aplicaciones Android

Las aplicaciones Android frecuentemente necesitan interactuar con servidores para obtener o enviar información, como datos de usuario, contenido dinámico o resultados de operaciones. Los servicios web REST son ideales para esta comunicación porque:

- Son ligeros y basados en HTTP, protocolo nativo y soportado en Android.
- Permiten intercambio de datos en formatos estándar como JSON, que Android maneja fácilmente.
- Facilitan la integración con sistemas backend existentes, permitiendo que una app móvil utilice funcionalidades o datos de un sistema web o en la nube.
- Ofrecen métodos claros para operaciones CRUD (Crear, Leer, Actualizar, Eliminar), fundamentales para aplicaciones que manejan datos.

Por ejemplo, una app Android puede consumir una API REST para mostrar una lista de productos, enviar pedidos o actualizar información del usuario.

Consumo de Servicios Web REST en Android: Panorama General

Para consumir un servicio web REST en Android, se siguen generalmente estos pasos:

1. Establecer la conexión HTTP: La app realiza solicitudes HTTP al servidor, usando métodos como GET, POST, PUT o DELETE, según la operación deseada.
2. Enviar y recibir datos en formato JSON: La app envía datos en JSON y recibe respuestas en el mismo formato, facilitando el procesamiento y la integración con la interfaz de usuario.
3. Procesar la respuesta: La app interpreta la respuesta JSON para actualizar su estado o mostrar información al usuario.
4. Manejo en segundo plano: Para no bloquear la interfaz, las peticiones a la API REST se realizan en hilos de fondo o mediante tareas asíncronas, asegurando una experiencia fluida.
5. Uso de librerías especializadas: Herramientas como Retrofit simplifican la comunicación con APIs REST, generando código para manejar las solicitudes y respuestas de forma eficiente y mantenible.
6. Optimización y buenas prácticas: Para operaciones que involucran gran cantidad de datos, se recomienda usar técnicas como operaciones batch para minimizar el consumo de batería y mejorar el rendimiento.

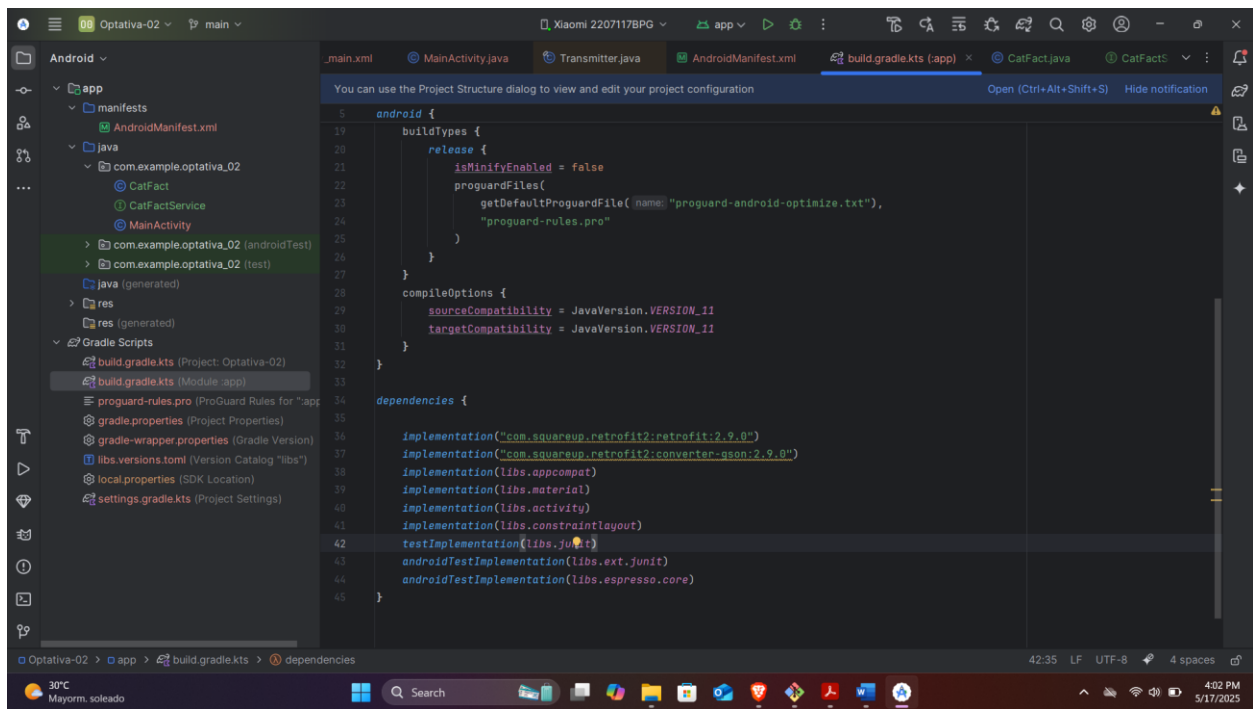
Ventajas de Usar Servicios Web REST en Android

- Portabilidad y escalabilidad: Permiten que la app funcione con diversos sistemas backend sin depender de tecnologías específicas.
- Separación de responsabilidades: El backend se encarga de la lógica y almacenamiento, mientras la app se centra en la experiencia de usuario.
- Facilidad de mantenimiento y evolución: Las APIs REST pueden evolucionar sin afectar directamente a las apps clientes si se diseñan adecuadamente.

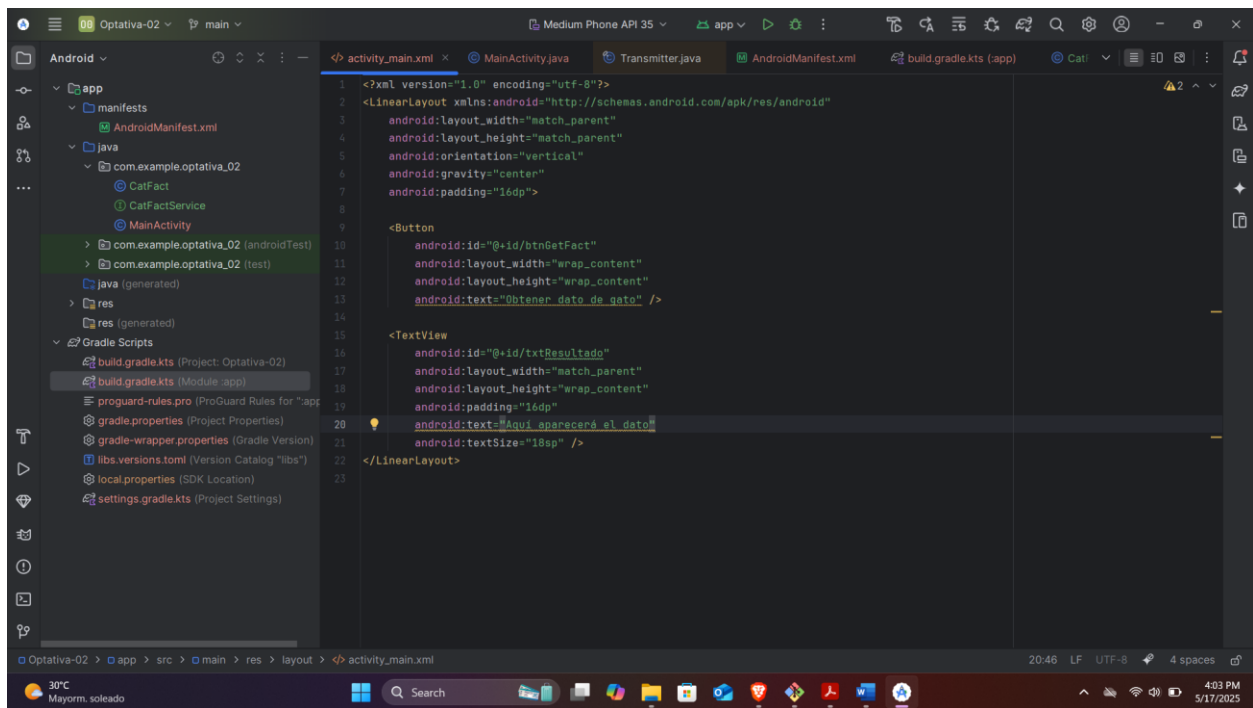
Aspecto	Descripción
Definición REST	Estilo arquitectónico para servicios web que usan HTTP para comunicación entre sistemas

Aspecto	Descripción
Operaciones HTTP principales	GET, POST, PUT, DELETE para manipular recursos
Formato de datos	JSON (principal), XML y otros formatos estructurados
Relación con Android	Comunicación entre app y backend para intercambio de datos y funcionalidades
Herramientas comunes	Retrofit, AsyncTask, HttpURLConnection para gestionar peticiones y respuestas
Buenas prácticas	Realizar peticiones en segundo plano, usar operaciones batch para optimizar el consumo de recursos
Ventajas	Portabilidad, escalabilidad, separación de responsabilidades, facilidad de mantenimiento

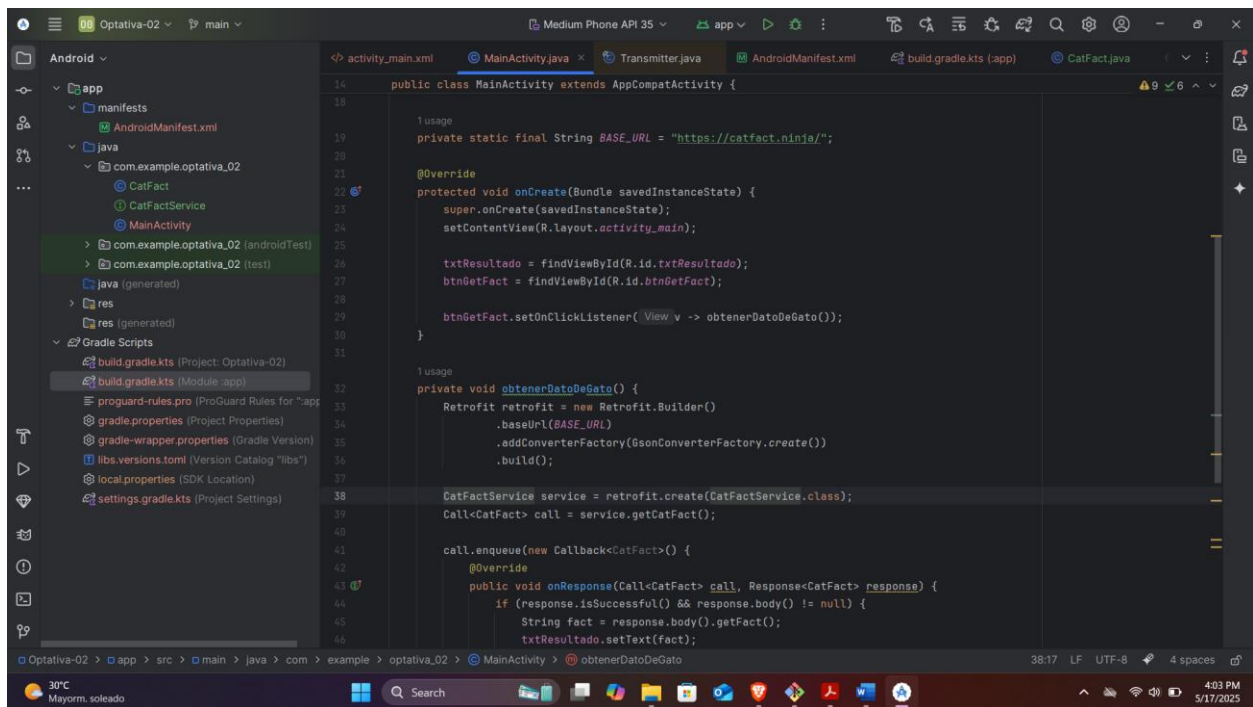
Para la aplicación busque otras APIs porque el enlace no sirve, elegí Cat Facts para que sea la API que consuma en mi aplicación Android, así que agregamos la dependencia necesaria con su versión



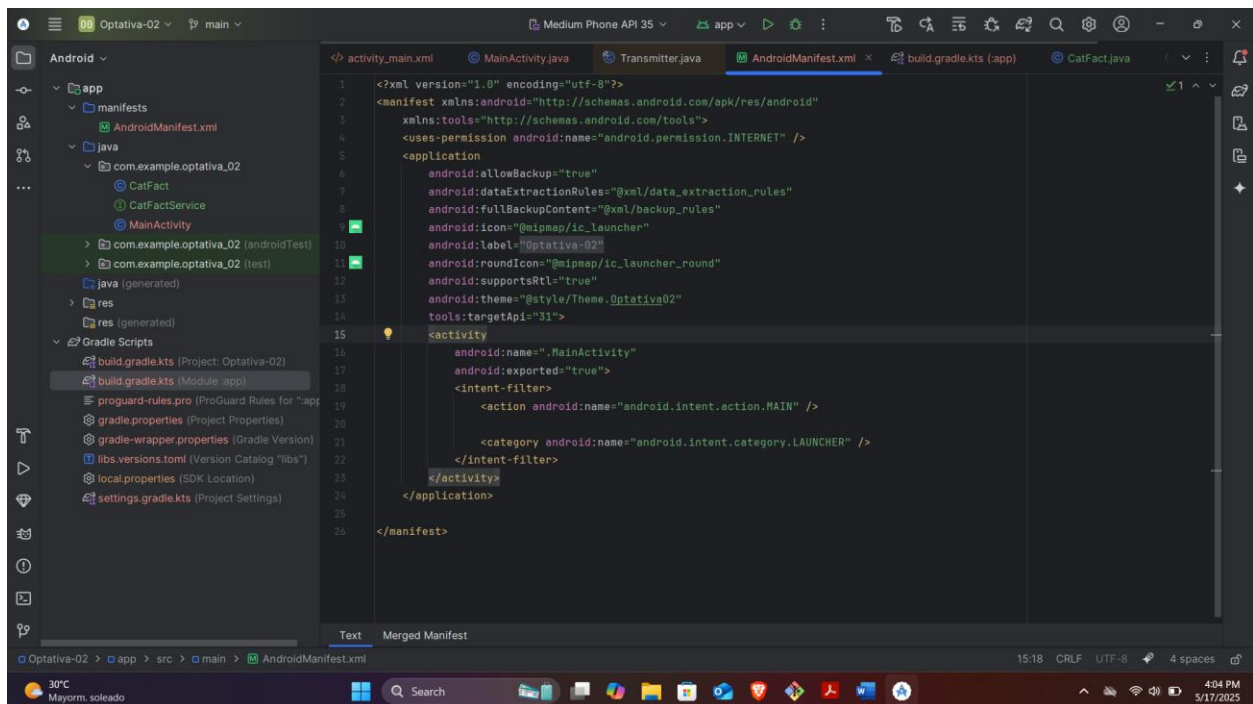
Creamos nuestra vista

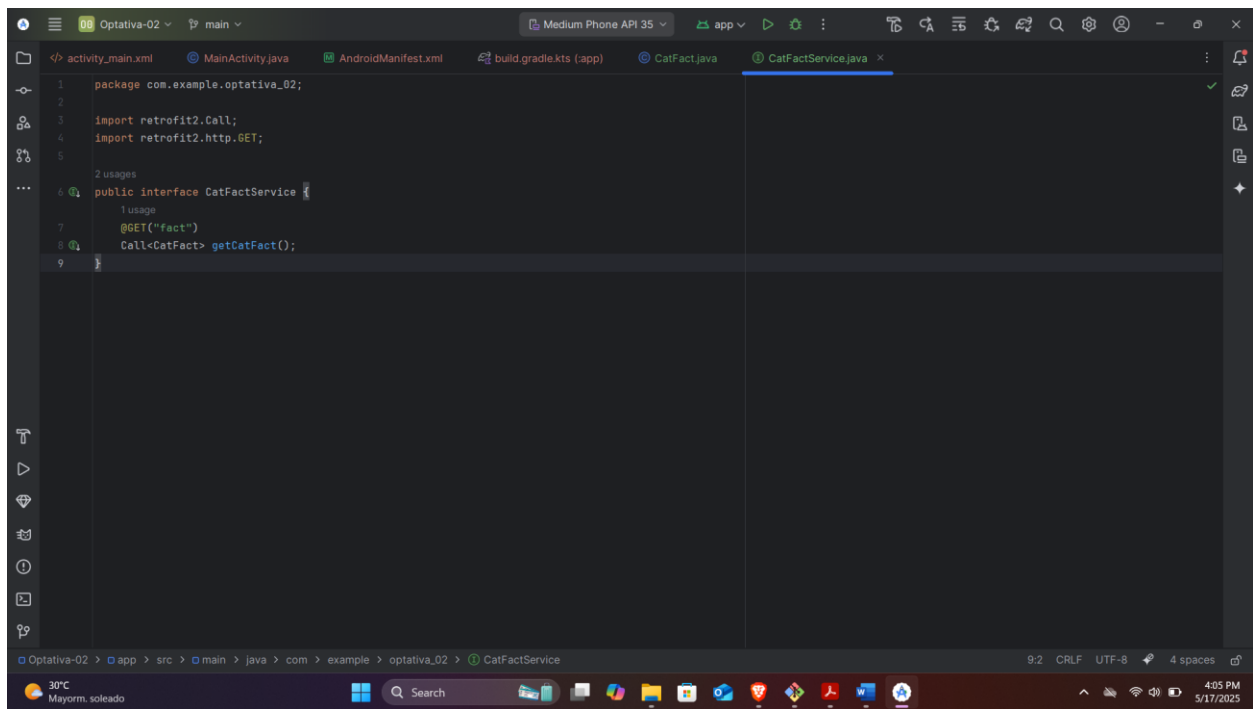


Y la lógica



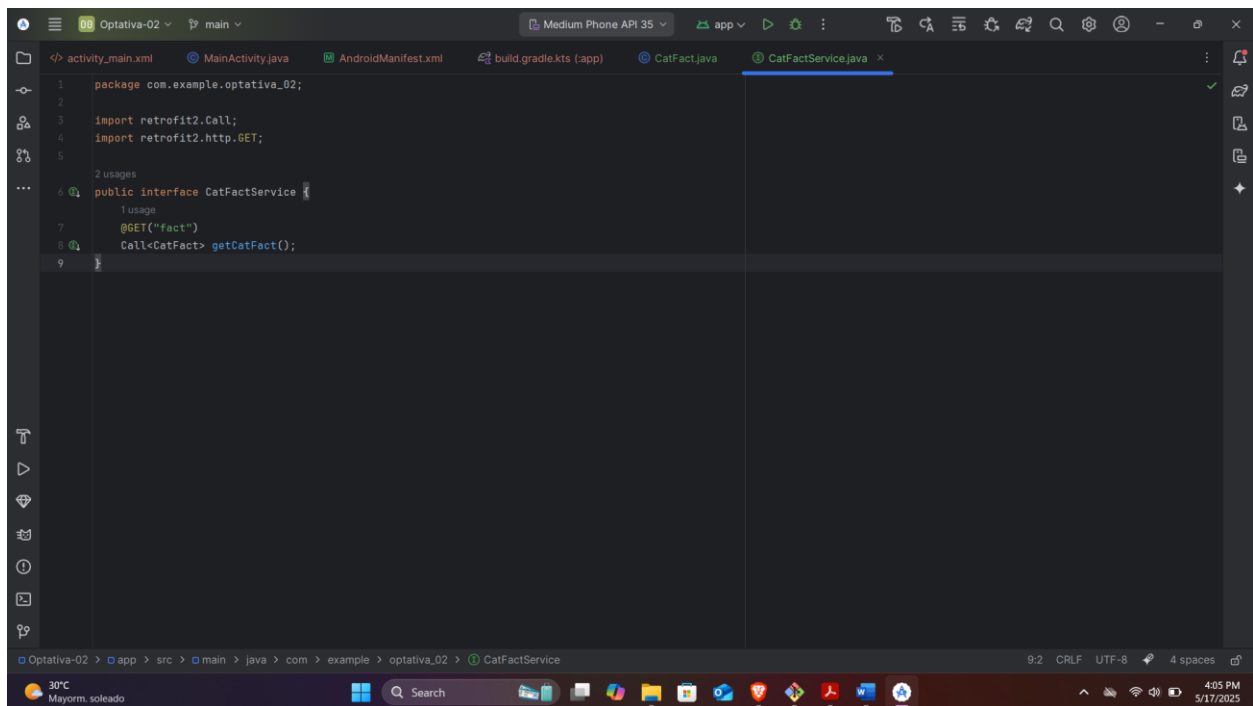
Debido a que es una api que realizara peticiones, la aplicación necesita tener el permiso para usar internet





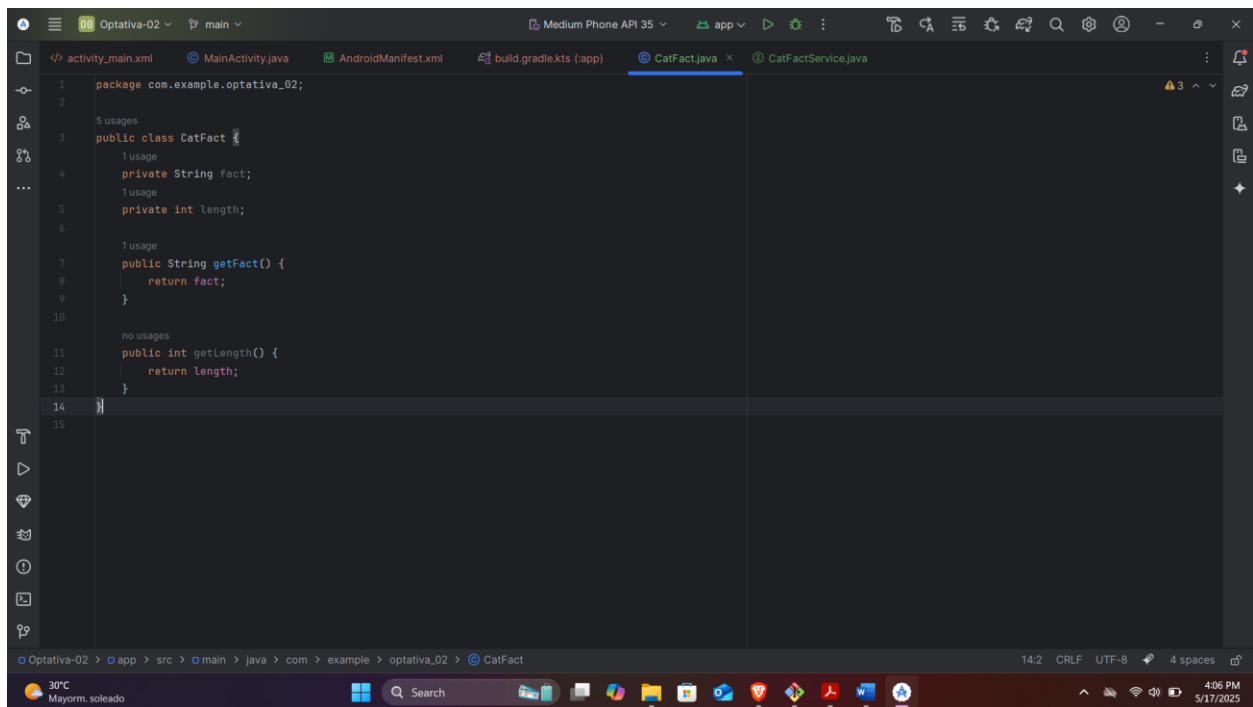
```
1 package com.example.optativa_02;
2
3 import retrofit2.Call;
4 import retrofit2.http.GET;
5
6 2 usages
7 public interface CatFactService {
8     1 usage
9     @GET("fact")
10    Call<CatFact> getCatFact();
11 }
```

creamos la interfaz de la API



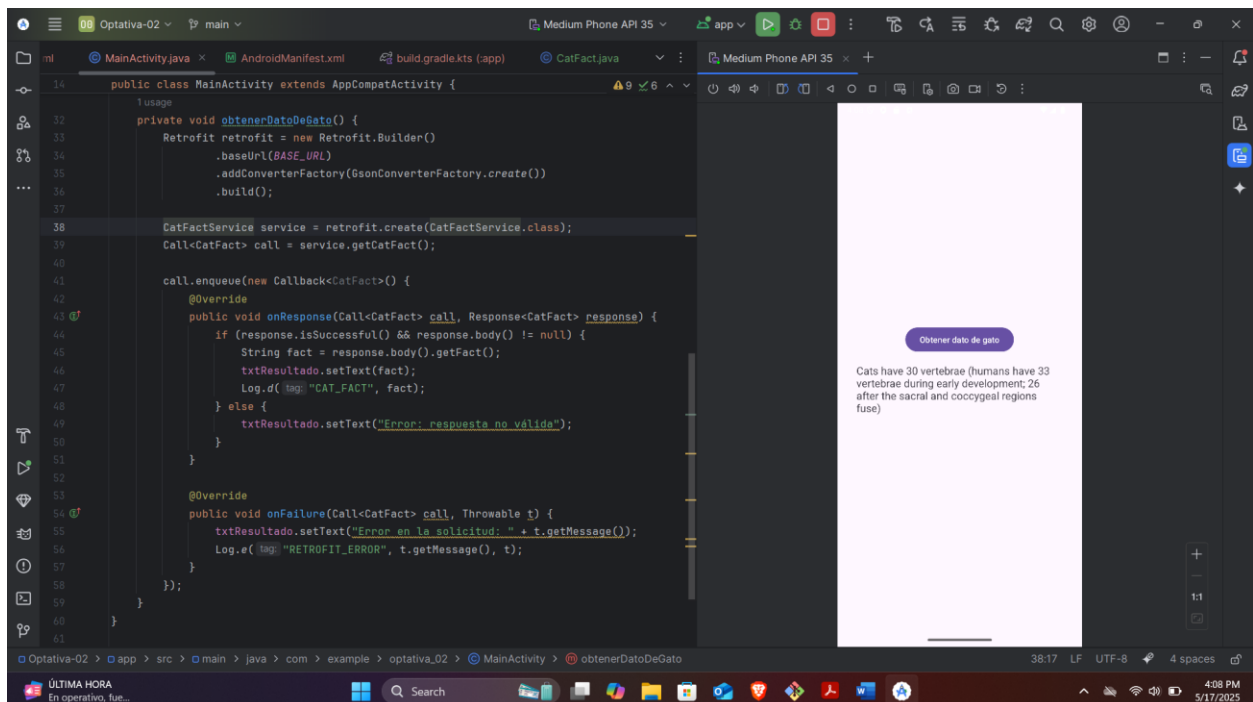
```
1 package com.example.optativa_02;
2
3 import retrofit2.Call;
4 import retrofit2.http.GET;
5
6 2 usages
7 public interface CatFactService {
8     1 usage
9     @GET("fact")
10    Call<CatFact> getCatFact();
11 }
```

Y el modelo

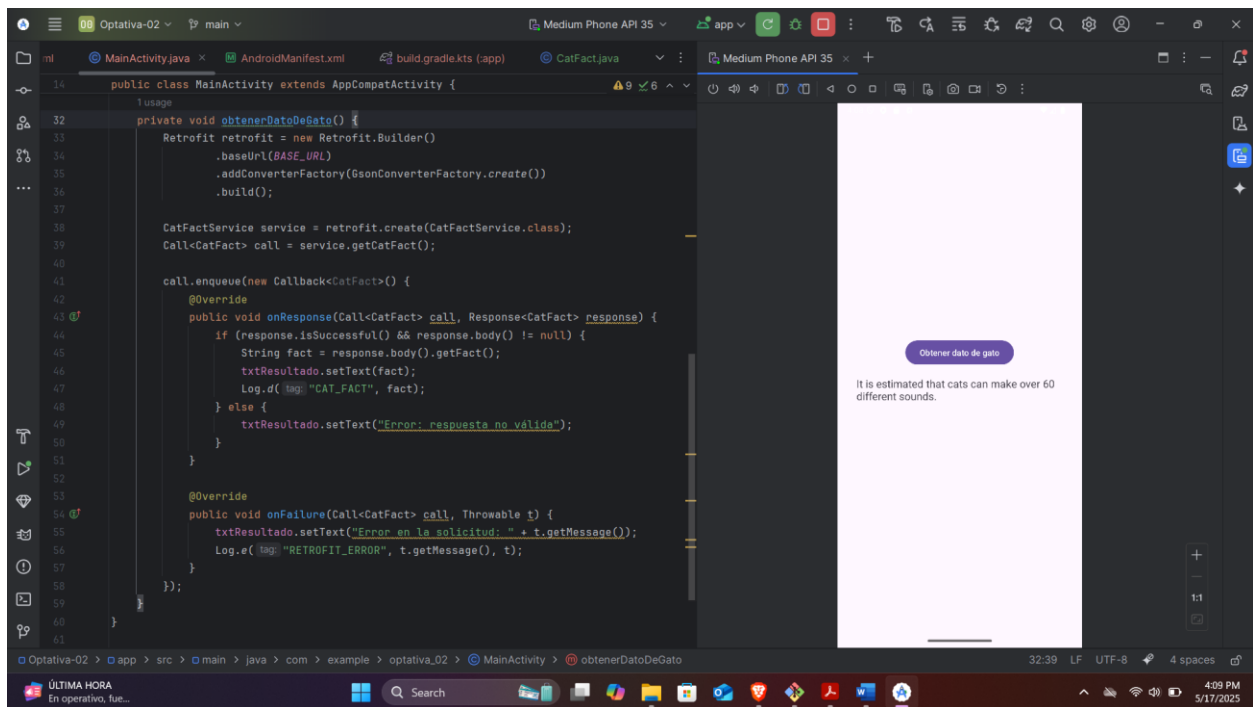


```
1 package com.example.optativa_02;
2
3 5 usages
4 public class CatFact {
5     1 usage
6     private String fact;
7     1 usage
8     private int length;
9
10    1 usage
11    public String getFact() {
12        return fact;
13    }
14
15    no usages
16    public int getLength() {
17        return length;
18    }
19 }
```

Ahora tenemos una aplicación en Android que nos cuenta datos interesantes de los gatos con solo dar click en un botón:



```
14 public class MainActivity extends AppCompatActivity {
15     1 usage
16     private void obtenerDatoDeGato() {
17         Retrofit retrofit = new Retrofit.Builder()
18             .baseUrl(BASE_URL)
19             .addConverterFactory(GsonConverterFactory.create())
20             .build();
21
22         CatFactService service = retrofit.create(CatFactService.class);
23         Call<CatFact> call = service.getCatFact();
24
25         call.enqueue(new Callback<CatFact>() {
26             @Override
27             public void onResponse(Call<CatFact> call, Response<CatFact> response) {
28                 if (response.isSuccessful() && response.body() != null) {
29                     String fact = response.body().getFact();
30                     txtResultado.setText(fact);
31                     Log.d(tag, "CAT_FACT", fact);
32                 } else {
33                     txtResultado.setText("Error: respuesta no valida");
34                 }
35             }
36
37             @Override
38             public void onFailure(Call<CatFact> call, Throwable t) {
39                 txtResultado.setText("Error en la solicitud: " + t.getMessage());
40                 Log.e(tag, "RETROFIT_ERROR", t.getMessage(), t);
41             }
42         });
43     }
44 }
```

La conclusión de esta actividad es que hoy por hoy existen APIs que nos pueden ayudar a implementar distintas funcionalidades a nuestras aplicaciones, claro que no todas son gratuitas y normalmente se tiene que pagar si quieres usar una API que te brinde una solución bastante completa mediante peticiones, así que es fundamental saber que existen este tipo de soluciones y el como podemos consumir APIs.

Fuentes

- OpenWebinars. (s.f.). *Qué es REST: Conoce su potencia*. Recuperado de <https://openwebinars.net/blog/que-es-rest-conoce-su-potencia/>
- GaussWebApp. (s.f.). *Ejemplo de conexión entre API REST Android*. Recuperado de <https://gausswebapp.com/ejemplo-conectar-api-rest-android.html>
- Develou. (s.f.). *Consumir un servicio web REST desde Android*. Recuperado de <https://www.develou.com/consumir-un-servicio-web-rest-desde-android/>
- Amazon Web Services. (s.f.). *¿Qué es una API de RESTful?*. Recuperado de <https://aws.amazon.com/es/what-is/restful-api/>
- sgoliver.net. (s.f.). *Acceso a servicios web REST en Android*. Recuperado de <https://www.sgoliver.net/blog/acceso-a-servicios-web-rest-en-android-22/>
- Red Hat. (2023, julio 31). *¿Qué es una API de REST?*. Recuperado de <https://www.redhat.com/es/topics/api/what-is-a-rest-api>
- Android Developers. (2024, mayo 10). *Cómo obtener datos de Internet*. Recuperado de <https://developer.android.com/codelabs/basic-android-kotlin-compose-getting-data-internet>