Programación Móvil

Nombre: ***Adrian Eduardo Reynosa Flores***

#2010455

Infografía: Volley vs Retrofit

1. Introducción

- **Propósito**: Ambas bibliotecas facilitan las solicitudes HTTP en aplicaciones Android.
- **Desarrolladores**: Principalmente utilizadas por desarrolladores de aplicaciones móviles.

2. Volley

- **Desarrollador**: Google.
- **Fecha de lanzamiento**: 2013.
- **Uso**: Ideal para solicitudes HTTP simples y rápidas.
- **Ventajas**:
    - Fácil de configurar y usar.
    - Integración con la cola de peticiones para gestión automática de solicitudes.
    - Admite caché de respuestas.
    - Soporte para múltiples tipos de solicitudes (GET, POST, etc.).

3. Retrofit

- **Desarrollador**: Square.
- **Fecha de lanzamiento**: 2013.
- **Uso**: Mejor para API RESTful, estructuradas y complejas.
- **Ventajas**:
    - Generación automática de API REST a través de interfaces Java.
    - Soporte para tipos de datos complejos (JSON, XML, etc.).
    - Integración con librerías de serialización (Gson, Jackson).
    - Gestión sencilla de autenticación y manejo de errores.

4. Comparación

- **Performance**:
    - **Volley**: Rápido para solicitudes simples.
    - **Retrofit**: Mejor rendimiento para API RESTful complejas.
- **Configuración**:
    - **Volley**: Configuración básica y rápida.
    - **Retrofit**: Requiere definición de interfaces y configuración adicional para tipos de datos.

- **Flexibilidad**:
  - **Volley**: Más adecuado para casos simples y rápidos.
  - **Retrofit**: Mejor estructuración y gestión de solicitudes complejas y API RESTful.
- **Documentación**:
  - **Volley**: Documentación oficial de Android.
  - **Retrofit**: Documentación completa y soporte de la comunidad.

5. Ejemplos de Código

- **Volley**:

```
String url = "https://api.example.com/data";
RequestQueue queue = Volley.newRequestQueue(context);
StringRequest stringRequest = new StringRequest(Request.Method.GET,
url,
        response -> {
        },
        error -> {
        });
queue.add(stringRequest);
```

- **Retrofit**:

```
public interface ApiService {
    @GET("data")
    Call<DataResponse> getData();
}

Retrofit retrofit = new Retrofit.Builder()
        .baseUrl("https://api.example.com/")
        .addConverterFactory(GsonConverterFactory.create())
        .build();

ApiService apiService = retrofit.create(ApiService.class);
Call<DataResponse> call = apiService.getData();
call.enqueue(new Callback<DataResponse>() {
    @Override
    public void onResponse(Call<DataResponse> call,
Response<DataResponse> response) {
    }

    @Override
    public void onFailure(Call<DataResponse> call, Throwable t) {
    }
});
```

6. Conclusión

- **Elección**: Elegir según las necesidades del proyecto.
- **Compatibilidad**: Ambas bibliotecas son compatibles con Android y pueden integrarse con otras bibliotecas de Jetpack y arquitecturas.