FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN

PROGRAMACIÓN DE DISPOSITIVOS MÓVILES

2025-1

Nombre del alumno: **David Alejandro Galicia Cárdenas.**

Núm. de cuenta: **422173025**

Grupo: **9691**

Mtro. Cristian Cardoso A.

Unidad 6

Objetivo de la actividad

-Investiga que es un REST web service y menciona qué relación tiene con un aplicativos Android

-Realiza un documento ilustrativo que refleja un panorama general sobre el consumo de los servicios web en Android

-Desarrolla una app sencilla donde se consuma un servicio rest desde https://api.publicapis.org - /Get /Entries

-La app deberá desplegar en pantalla el resultado de la operación http get

Desarrollo

¿Qué es un REST Web Service?

REST es el servicio web que se encarga de conectar varios sistemas basados en HTTP. Es usado para obtener, generar datos y operaciones en distintos formatos como XML o JSON. REST usa métodos como GET, POST, PUT y DELETE. Los servicios RESTful al ser independientes de la plataforma, es posible que cualquier aplicación haga uso de peticiones HTTP para usar sus servicios, importando poco el lenguaje de programación.

Relación con Aplicaciones Android

Es usado principalmente para obtener o enviar toda la información necesaria a un servidor a través de Request y Response de manera eficaz, pues como es común que en el desarrollo de aplicaciones Android sean necesarios el uso de datos de servidores externos o bases de datos remotas.

Panorama General del Consumo de Servicios Web en Android

1- Primero debemos de agregar las dependencias build.gradle al archivo con el que estamos trabajando y podemos usar Retrofit o Gson.
2- Debemos de otorgar a la app la capacidad de acceder a internet en el AndroidManifest.xml con esta línea de código: <uses-permission android:name="android.permission.INTERNET" />

3- Después vamos a definir los modelos que representarán los datos que vamos a recibir, dicho modelo puede variar dependiendo con lo que trabajemos.
4- Es importante crear una interfaz que defina los endpoints de la API y los métodos HTTP que se van a usar.
5- Configuramos el retrofit para que pueda usar la interfaz de la API.
6- Después hacemos las peticiones para obtener la información que deseamos tener.

Código fuente

Este es el Main:

```
package com.example.api


import android.os.Bundle
import android.util.Log
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import okhttp3.OkHttpClient
import okhttp3.logging.HttpLoggingInterceptor
import retrofit2.*
import retrofit2.converter.gson.GsonConverterFactory


class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
```

```
        val textView: TextView = findViewById(R.id.textView)


        val logging = HttpLoggingInterceptor().apply {
            level = HttpLoggingInterceptor.Level.BODY
        }
        val httpClient = OkHttpClient.Builder()
            .addInterceptor(logging)
            .build()
        val retrofit = Retrofit.Builder()
            .baseUrl("https://catfact.ninja/")
            .addConverterFactory(GsonConverterFactory.create())
            .client(httpClient)
            .build()
        val apiService = retrofit.create(ApiServicio::class.java)
        apiService.getEntries().enqueue(object : Callback<CatFact> {
            override fun onResponse(call: Call<CatFact>, response:
Response<CatFact>) {
                if (response.isSuccessful) {
                    val catFact = response.body()
                    textView.text = catFact?.fact ?: "No se pudo obtener el dato"
                } else {
                    textView.text = "error en la respuesta: ${response.code()}"
```

```
            }
        }
        override fun onFailure(call: Call<CatFact>, t: Throwable) {
            Log.e("MainActivity", "error en la API", t)
            textView.text = "error al obtener datos: ${t.message}"
        }
    })
  }
}
```

Este es el AndroidManifest.XML

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools">

  <uses-permission android:name="android.permission.INTERNET" />
  <application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.AppCompat.Light"
    android:usesCleartextTraffic="true"
    tools:targetApi="31">
```

```
    <activity
        android:name=".MainActivity"
        android:exported="true"
        android:theme="@style/Theme.AppCompat.Light">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
</manifest>
```

Este solo es una parte del gradle:

```
dependencies {


    implementation (libs.retrofit.v2110)

    implementation(libs.converter.gson.v2110)

    implementation (libs.logging.interceptor)



    implementation(libs.androidx.core.ktx)

    implementation(libs.androidx.lifecycle.runtime.ktx)

    implementation(libs.androidx.activity.compose)

    implementation(platform(libs.androidx.compose.bom))

    implementation(libs.androidx.ui)

    implementation(libs.androidx.ui.graphics)
```

```
    implementation(libs.androidx.ui.tooling.preview)
    implementation(libs.androidx.material3)
    implementation(libs.androidx.appcompat)
    testImplementation(libs.junit)
    androidTestImplementation(libs.androidx.junit)
    androidTestImplementation(libs.androidx.espresso.core)
    androidTestImplementation(platform(libs.androidx.compose.bom))
    androidTestImplementation(libs.androidx.ui.test.junit4)
    debugImplementation(libs.androidx.ui.tooling)
    debugImplementation(libs.androidx.ui.test.manifest)
}
```

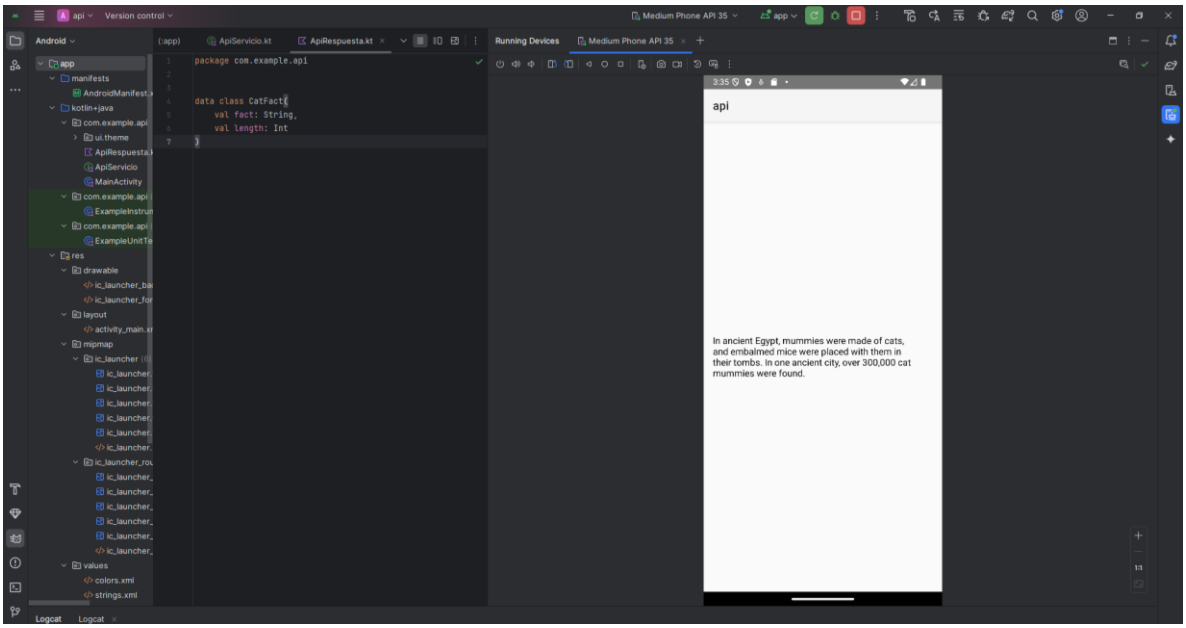
```
package com.example.api
import retrofit2.Call
import retrofit2.http.GET

interface ApiServicio {
    @GET("fact")
    fun getEntries(): Call<CatFact>
}
```

```
package com.example.api
```

```
data class CatFact(
    val fact: String,
    val length: Int
)
```

Capturas de pantalla



Conclusión

Aunque quería usar el enlace que me proporciono, no le entendí a la página y usé otra página que tuviera APIs públicas, esa fue la parte más compleja pues no sabia como hacer uso del servicio REST hasta que vi distintos videos.

Bibliografía:

-Neumann, A., Laranjeiro, N., & Bernardino, J. (2018). An analysis of public REST web service APIs. IEEE Transactions on Services Computing, 14(4), 957-970.

-José, M. R. M. (2018, 17 de Mayo). Qué es REST: Conoce su potencia. Openwebinars. https://openwebinars.net/blog/que-es-rest-conoce-su-potencia/

-Tihomirovs, J., & Grabis, J. (2016). Comparison of soap and rest based web services using software evaluation metrics. Information technology and management science, 19(1), 92-97.

-Barbaglia, G., Murzilli, S., & Cudini, S. (2017). Definition of REST web services with JSON schema. Software: Practice and Experience, 47(6), 907-920.

-Admin Sys. (2021, 17 de Mayo). Cómo consumir una API desde una aplicación Android. Coderslink. https://coderslink.com/talento/blog/como-consumir-una-api-desde-una-aplicacion-android/

-