

Internet y uso de APIs



Pasos previos

Librerías

Durante los siguientes ejemplos se usarán las siguientes librería que habrá que importar en gradle.

`implementation 'com.squareup.okhttp3:okhttp:4.8.0'` Para gestionar las conexiones HTTPS.

`implementation 'com.google.code.gson:gson:2.8.6'` Para gestionar los JSON que se reciben de la API.

`implementation "org.jetbrains.kotlinx:kotlinx-serialization-runtime:0.9.1"` Para transformar los datos recibidos eficientemente.

`implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-android:1.3.9'` Para gestionar donde se ejecutarán los distintos elementos que componen la App.

Permisos

Será necesario añadir los siguientes permisos a tu AndroidManifest:

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

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



Uso eficiente de APIs

API de ejemplo

<https://swapi.dev/>

<https://swapi.dev/documentation>

Try it now!

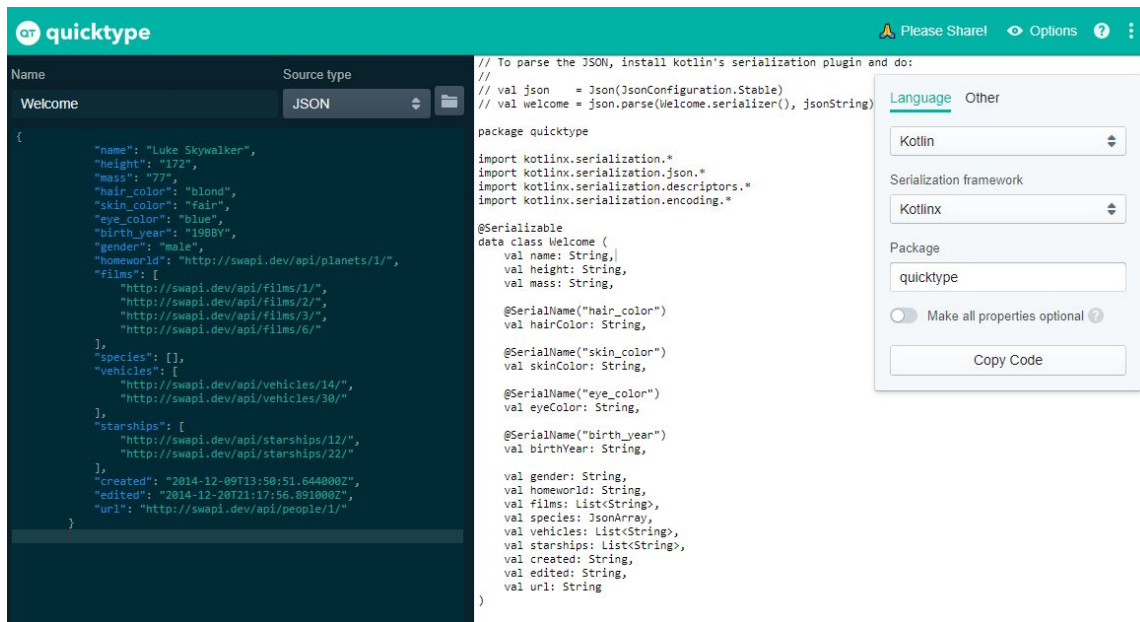
Need a hint? try *people/1/* or *planets/3/* or *starships/9/*

Result:

```
{
  "count": 6,
  "next": null,
  "previous": null,
  "results": [
    {
      "title": "A New Hope",
      "episode_id": 4,
      "opening_crawl": "It is a period of civil war.\r\nRebel",
      "director": "George Lucas",
      "producer": "Gary Kurtz, Rick McCallum",
      "release_date": "1977-05-25",
      "characters": [
        "http://swapi.dev/api/people/1/",
        "http://swapi.dev/api/people/2/",
        "http://swapi.dev/api/people/3/",
        "http://swapi.dev/api/people/4/",
        "http://swapi.dev/api/people/5/"
      ]
    }
  ]
}
```

Transformación automática

(Opcional) Puedes utilizar esta página para realizar la transformación de una manera sencilla.



The screenshot displays the quicktype web application interface. On the left, a JSON object for a character named Luke Skywalker is shown. The 'Source type' is set to 'JSON'. On the right, the generated Kotlin code is displayed, including imports for the kotlinx.serialization library and a data class named 'Welcome' with properties corresponding to the JSON fields. A settings panel on the far right allows selecting 'Kotlin' as the language and 'Kotlinx' as the serialization framework, with a 'Copy Code' button at the bottom.

```
// To parse the JSON, install kotlin's serialization plugin and do:
// val json = Json(JsonConfiguration.Stable)
// val welcome = json.parse(Welcome.serializer(), jsonString)

package quicktype

import kotlinx.serialization.*
import kotlinx.serialization.json.*
import kotlinx.serialization.descriptors.*
import kotlinx.serialization.encoding.*

@Serializable
data class Welcome (
    val name: String,
    val height: String,
    val mass: String,
    @SerializedName("hair_color")
    val hairColor: String,
    @SerializedName("skin_color")
    val skinColor: String,
    @SerializedName("eye_color")
    val eyeColor: String,
    @SerializedName("birth_year")
    val birthYear: String,
    val gender: String,
    val homeworld: String,
    val films: List<String>,
    val species: JsonArray,
    val vehicles: List<String>,
    val starships: List<String>,
    val created: String,
    val edited: String,
    val url: String
)
```

Resultado

De JSON

```
{ "title": "A New Hope", "episode_id": 4, "opening_crawl": "It is a period of civil war.\r\nRebel spaceships, striking\r\nfrom a hidden base, have won\r\ntheir first victory against\r\nthe evil Galactic Empire.\r\n\r\nDuring the battle, Rebel\r\nspies managed to steal secret\r\nplans to the Empire's\r\nultimate weapon, the DEATH\r\nSTAR, an armored space\r\nstation with enough power\r\nto destroy an entire planet.\r\n\r\nPursued by the Empire's\r\nsinister agents, Princess\r\nLeia races home aboard her\r\nstarship, custodian of the\r\nstolen plans that can save her\r\npeople and restore\r\nfreedom to the galaxy....", "director": "George Lucas", ....
```

A Kotlin

```
data class Film(var title: String, @SerializedName("episode_id")var episodeId : Int,
@SerializedName("opening_crawl")var openingCrawl: String, var director: String, var producer: String, var
url: String, var created: String, var edited: String, @SerializedName("species") val speciesUrls:
List<String>?, @SerializedName("starships") var starshipsUrls: List<String>?, @SerializedName("vehicles")
var vehiclesUrls: List<String>?, @SerializedName("planets")var planetsUrls: List<String>?,
@SerializedName("characters")var charactersUrls: List<String>?)
```




Llamadas a Internet

Llamada con clousures

```
val client = OkHttpClient()
```

```
val url = "https://swapi.dev/api/films/"
```

```
val request = Request.Builder().url(url).build()
```

```
val call = client.newCall(request)
```

```
call.enqueue(object : Callback {
```

```
    override fun onFailure(call: Call, e: IOException) { ... }
```

```
    override fun onResponse(call: Call, response: Response) { ... }
```

```
}
```

Atención, el Método 1 y el Método 2 son alternativos

Llamada con corrutinas

```
val client = OkHttpClient()

val url = "https://swapi.dev/api/films/"

val request = Request.Builder().url(url).build()

val call = client.newCall(request)

return withContext(Dispatchers.IO) {

    val response = call.execute()

    // Tu código aquí

}
```

Atención, el Método 1 y el Método 2 son alternativos

Análisis de la respuesta

```
val JsonObject = JSONObject(bodyInString)
```

```
val results = JsonObject.optJSONArray("results")
```

```
results?.let {
```

```
    val gson = Gson()
```

```
    val itemType = object : TypeToken<List<Film>>() {}.type
```

```
    val list = gson.fromJson<List<Film>>(results.toString(), itemType)
```

```
}
```

Válido para el Método 1 y el Método 2

Transformación a nuestro modelo de datos

La función **map** función nos permite, por cada uno de los elementos de la lista, aplicarle una transformación de manera sencilla

```
list
```

```
list ?.map {
```

```
}
```

Válido para el Método 1 y el Método 2

Ejercicio Resumen

Proyecto en GitHub



GitHub

Fin

