Retrofit y JobQueue

1.

Añadir dependencias

“

Debemos añadir *Retrofit* y *OkHttp3* a nuestro fichero *build.gradle*

```
compile "com.squareup.okhttp3:okhttp:3.9.0"
compile "com.squareup.okhttp3:logging-interceptor:3.9.0"

compile ("com.squareup.retrofit2:retrofit:2.3.0"){
    exclude module: 'okhttp'
}
compile "com.squareup.retrofit2:converter-gson:2.3.0"
```

2.

Nuestra primera interfaz

“

Tendrá un método que se encargará de hacer una llamada a un servicio

```
interface ResourceService {

    @GET("resource/hma6-9xbg.json")
    fun requestResourceList(@Query("category") category: String,
                            @Query("item") item: String): Call<List<ItemDTO>>
}
```

3.

Crear instancias

Clase que contiene un *companion object* para inicializar *Retrofit* y *OkHttp*

```
class ApiUtils {

    companion object {
        private fun generateOkHttpBuilder(): OkHttpClient {
            return OkHttpClient().newBuilder()
                    .build()
        }

        fun generateRetrofitInstance(): Retrofit {
            return Retrofit.Builder()
                    .baseUrl(AppConstants.ENDPOINT)
                    .client(generateOkHttpBuilder())
                    .addConverterFactory(GsonConverterFactory.create())
                    .build()
        }
    }

}
```

4.

Modelado

ItemDto. Objeto de transferencia de datos

```
class ItemDto(
        @SerializedName("item") val item: String,
        @SerializedName("business") val business: String,
        @SerializedName("farmer_id") val farmerId: String,
        @SerializedName("category") val category: String,
        @SerializedName("l") val l: String,
        @SerializedName("farm_name") val farmName: String,
        @SerializedName("phone1") val phone1: String
)
```

“

ItemModel. Objetos que usará nuestro adapter

```
class ItemModel(
        val item: String?,
        val business: String?,
        val farmerId: String?,
        val category: String?,
        val l: String?,
        val farmName: String?,
        val phone1: String?
)
```

6.

Mapper

ItemMapper. DTO → Model

```
class ItemMapper {
    fun transform(items: List<ItemDTO>): List<ItemModel> {
        return items.map { transform(it) }
    }

    fun transform(item: ItemDTO): ItemModel {
        return ItemModel(item.item,
                item.business,
                item.farmerId,
                item.category,
                item.l,
                item.farmName,
                item.phone1)
    }
}
```

7.

Dependencia y configuración de JobQueue

Pasos

1. Añadir dependencia

```
compile 'com.birbit:android-priority-jobqueue:2.0.0'
```

2. Inicializamos la interfaz

```
val resourceService = ApiUtils
        .generateRetrofitInstance()
        .create(ResourceService::class.java)
```

3. Realizamos la llamada

```
val call = resourceService.requestResourceList("Fruit", "Peaches")
```

Pasos

4. Después de ejecutar la llamada transformamos los DTOs a objetos de dominio

```
val result = call.execute().body()
val items = ModelMapper().transform(result!!)
```

Pasos

5. Creamos el manager que controlará la ejecución del *job*

```
val builder = Configuration.Builder(this)
        .minConsumerCount(1)
        .maxConsumerCount(3)
        .loadFactor(3)
        .consumerKeepAlive(120)

val jobManager: JobManager = JobManager(builder.build())
val serviceJob: GetResourceListJob = GetResourceListJob(Params(50).requireNetwork(), this)
jobManager.addJobInBackground(serviceJob)
jobManager.start()
```

8.

Crear Job - onRun

“

Constructor

```
class GetResourceListJob(params: Params?, val view: MainView) : Job(params) {
```

Interfaz *MainView*

```
interface MainView {
    fun setDataSet(items: List<ItemModel>)
}
```

“

Lanzar resultado en primer plano

```
val uiHandler = Handler(Looper.getMainLooper())
val runnable = Runnable {
    view.setDataSet(items)
}
uiHandler.post(runnable)
```

Implementar método de la interfaz

```
override fun setDataSet(items: List<ItemModel>) {
    mainRecycler.layoutManager = LinearLayoutManager( context: this)
    mainRecycler.adapter = ItemAdapter(items) {
        toast(String.format("Click en %s", it.farmerId))
    }
}
```