Evidencia dia 02 - Semana 15

Leonardo Rodenas Escobar

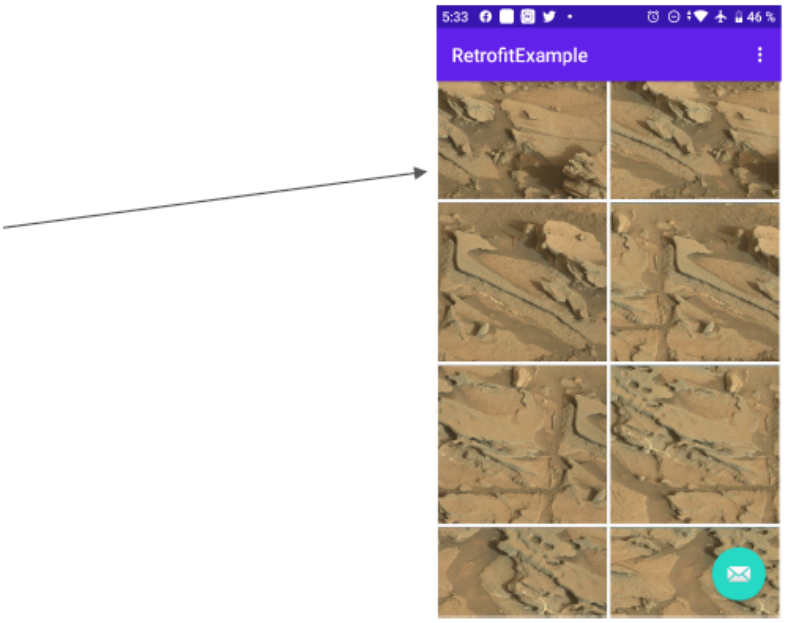
Reflexión:

El curso va bastante bien a pesar del millón de problemas que tuvimos hacias atrás (y que no vale la pena explicar ahora). Hoy fue un día más de reforzar los conocimientos y de consolidar todo hasta ahora en un sólo ejercicio. Por el momento no pude terminar todo pero voy progresando poco a poco sobre ello.

Ejercicio:

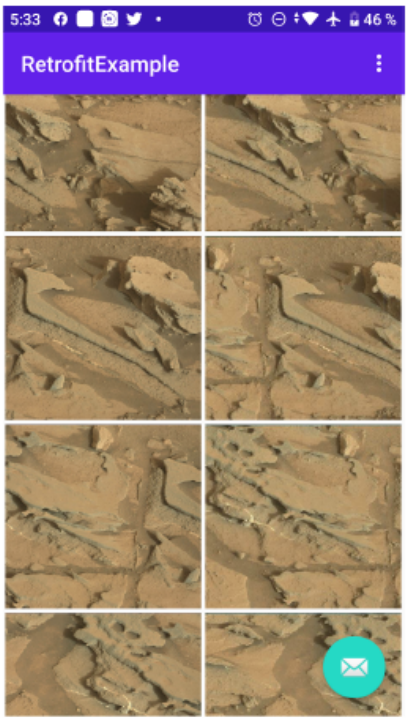
Ejercicio: Integrar Retrofit

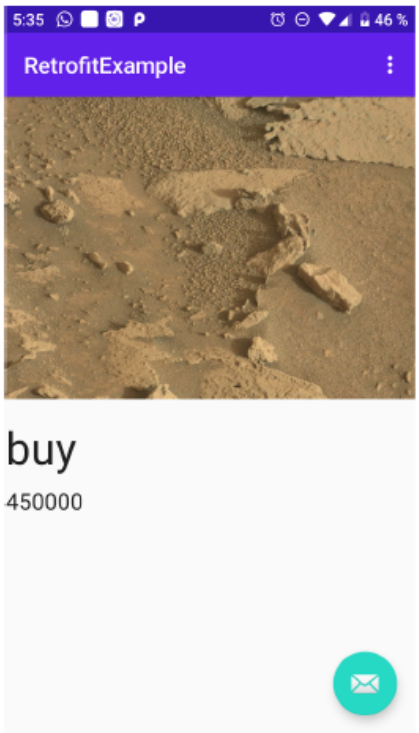
1. Vamos a conectarnos a un servicio web y traernos los datos hacia nuestra app.
2. Retrofit manejara la conexión y a través de Gson se transformarán los datos a objeto.
3. Mostraremos el listado de objetos en un vista.
4. Tenemos un repositorio con el proyecto inicial, donde ya están construidas las vistas.



Una vez construida la app deberia verse asi:

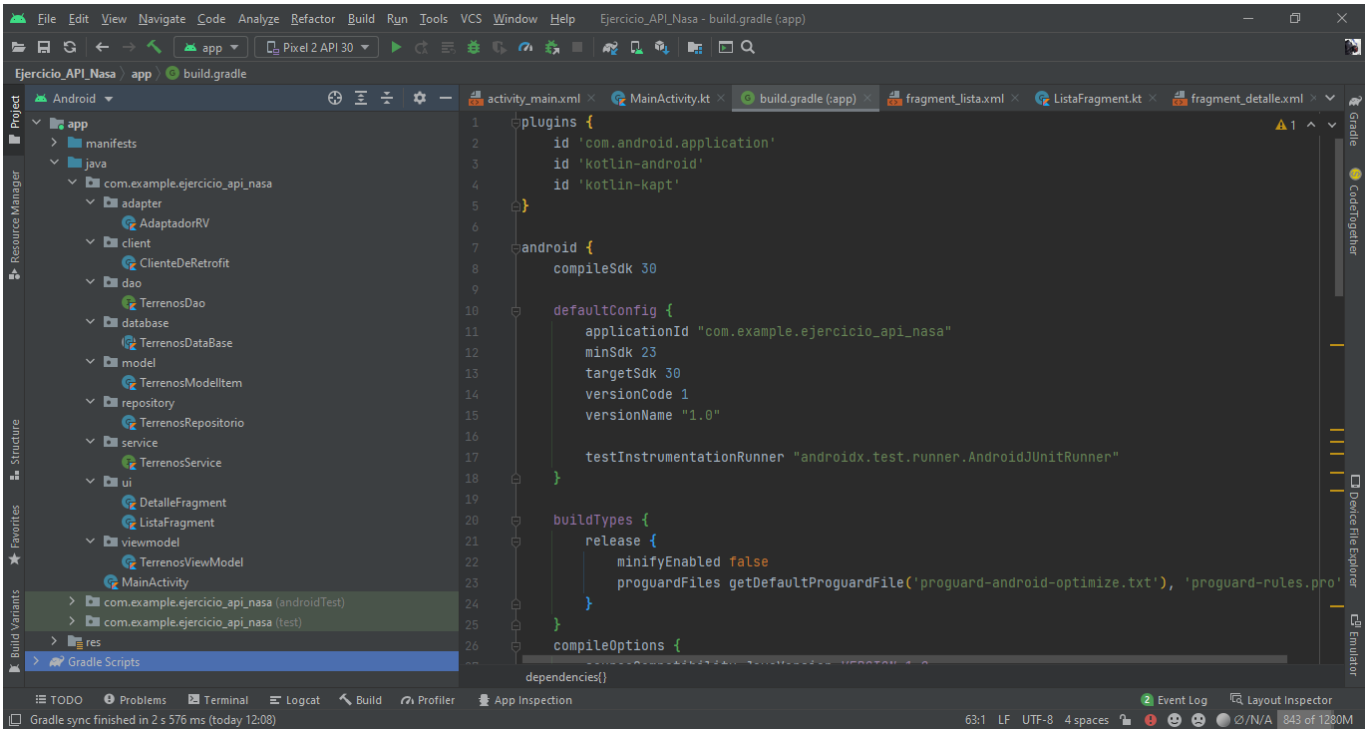
1. Nos conectaremos a una API que nos va a entregar un listado de elementos.
2. Tendremos que analizar los elementos que nos enviaran. TIP: utiliza POSTMAN
   a. https://android-kotlin-fun-mars-server.appspot.com/realestate
3. Descarga el proyecto base desde el siguiente repositorio.





Para iniciar el ejercicio comencé importando todas las dependencias necesarias (retrofit, lyfecycle, corrutinas, room, picasso) e implemento el viewBinding en Gradle.

Luego creo los packages y clases/interfaces necesarias para implementar Room y la conexión a la API.



Dejo acá el código escrito en cada una de ellas:

Entitys(modelo):

```
@Entity(tableName = "tabla_terrenos")
data class TerrenosModelItem(

    @PrimaryKey(autoGenerate = false)
    val id: String,

    @SerializedName("imagen")
    val img_src: String,

    val price: Int, //Long en el ejemplo?

    val type: String
)
```

Dao:

```
@Dao
interface TerrenosDao {
```

```
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertarTodosLosTerrenos(listaDeTerrenos: List<TerrenosModelItem>)

    @Query("SELECT * FROM tabla_terrenos")
    fun obtenerTodosLosTerrenosDeLaBD(): LiveData<List<TerrenosModelItem>>

}
```

Database:

```
@Database(entities = [TerrenosModelItem::class], version = 1)
abstract class TerrenosDataBase : RoomDatabase() {

    abstract fun obtenTerrenosDelDao(): TerrenosDao

    companion object {

        @Volatile
        private var baseDeDatosCreada: TerrenosDataBase? = null

        fun crearDatabase(context: Context): TerrenosDataBase {

            if (baseDeDatosCreada == null) {
                synchronized(this)
                {
                    baseDeDatosCreada = Room.databaseBuilder(
                        context,
                        TerrenosDataBase::class.java,
                        "base_De_Datos_Terrenos"
                    ).build()
                }
            }
            return baseDeDatosCreada!!
        }

    }
}
```

Repositorio:

```
class TerrenosRepositorio(private val terrenosDao: TerrenosDao) {

    private val service = ClienteDeRetrofit.obtenCliente()
    val miLiveData = terrenosDao.obtenerTodosLosTerrenosDeLaBD()
```

```
    fun obtenDataDelServer() {
        val call = service.obtenerTerrenos()
        call.enqueue(object : Callback<List<TerrenosModelItem>> {
            override fun onResponse(
                call: Call<List<TerrenosModelItem>>,
                response: Response<List<TerrenosModelItem>>
            ) {
                CoroutineScope(Dispatchers.IO).launch {
                    response.body()?.let {
                        terrenosDao.insertarTodosLosTerrenos(it)
                    }
                }
            }

            override fun onFailure(call: Call<List<TerrenosModelItem>>, t:
Throwable) {
                call.cancel()
            }

        })
    }
}
```

Service:

```
interface TerrenosService {

    @GET("realstate")
    fun obtenerTerrenos(): Call<List<TerrenosModelItem>>

}
```

Cliente:

```
class ClienteDeRetrofit {

    companion object{
        private val url = "https://android-kotlin-fun-mars-server.appspot.com/"

        fun obtenCliente(): TerrenosService{
            val retrofit = Retrofit.Builder().baseUrl(url).addConverterFactory(
                GsonConverterFactory.create()).build()
            return retrofit.create(TerrenosService::class.java)
        }
```

```
    }
}
```

ViewModel:

```
class TerrenosViewModel(application: Application) : AndroidViewModel(application)
{

    private var repositorio : TerrenosRepositorio

    init {
        //indica funcion que traera el repositorio
        val terrenosDao =
TerrenosDataBase.crearDatabase(application).obtenTerrenosDelDao()
        repositorio = TerrenosRepositorio(terrenosDao)
    }

    fun exponeDatosDelServer():LiveData<List<TerrenosModelItem>> {
        return repositorio.miLiveData
    }
}
```

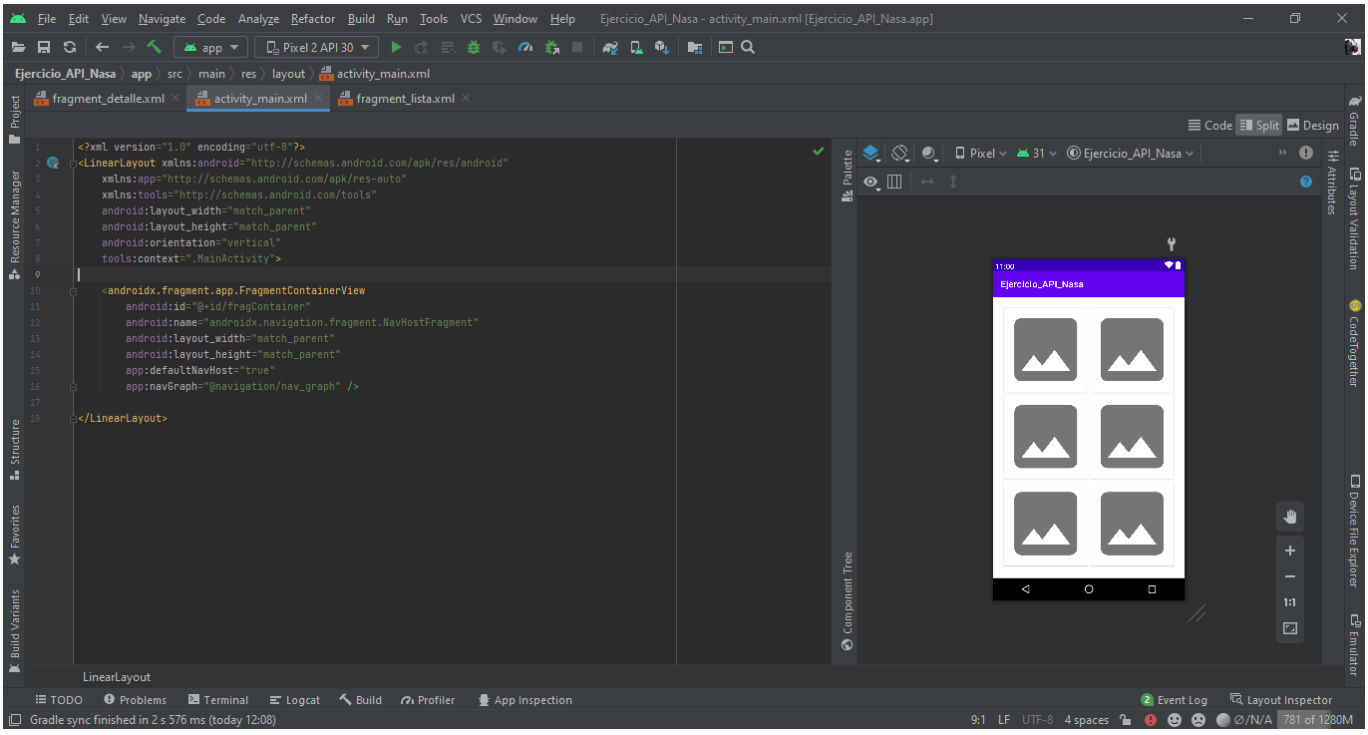
Adaptador del recyclerView:

```
//OJO QUE PARA HACER ESTO, PREVIO SE CREO EL ITEM_RECYCLERVIEW.XML PARA PODER
HACER EL BINDING E IMPLEMENTAR EL ADAPTADOR

class AdaptadorRV : RecyclerView.Adapter<AdaptadorRV.CustomViewHolder>() {

    private var lista: List<TerrenosModelItem> = ArrayList()

    class CustomViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {

        val binding = ItemRecyclerviewBinding.bind(itemView)

        fun bindData(img: TerrenosModelItem) {
            Picasso.get().load(img.img_src).into(binding.ivTerreno)
        }

    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
CustomViewHolder {
        val view:View =
```

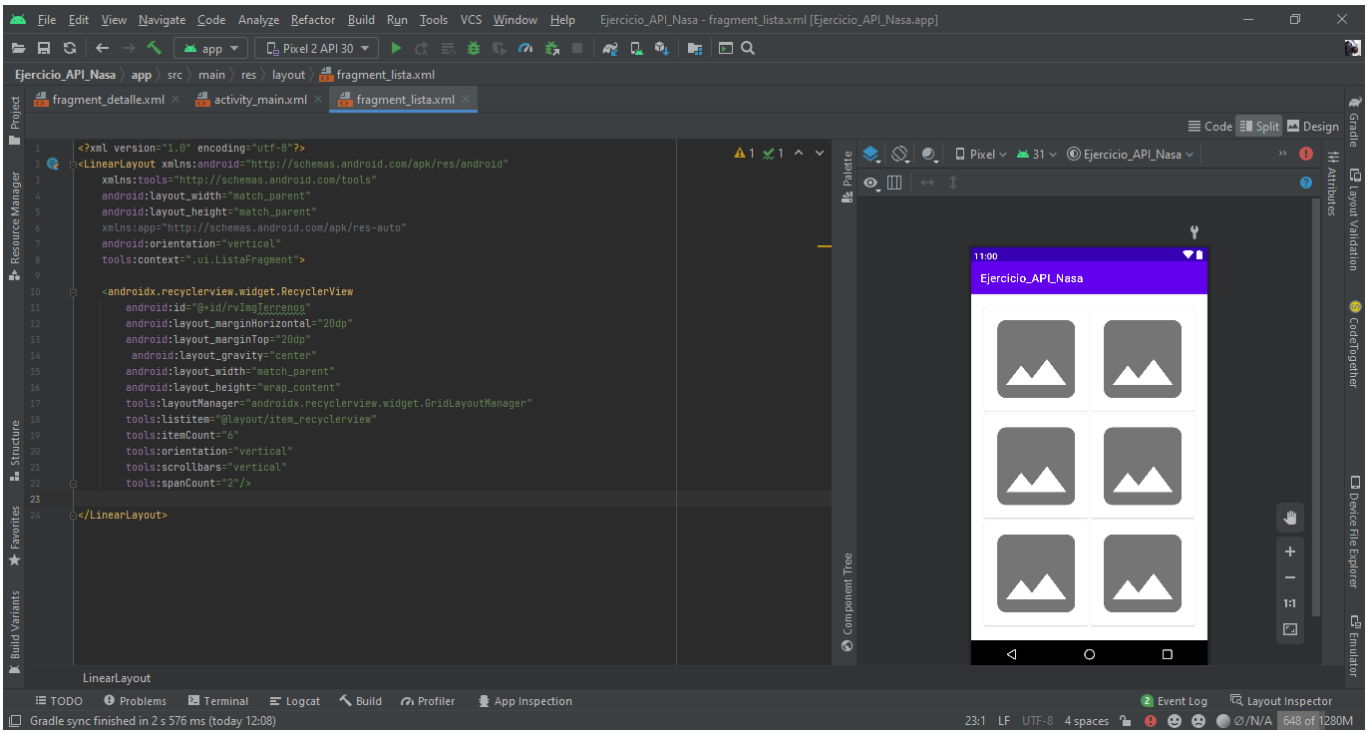
```
LayoutInflater.from(parent.context).inflate(R.layout.item_recyclerview,parent,fals
e)
        return CustomViewHolder(view)
    }

    override fun onBindViewHolder(holder: CustomViewHolder, position: Int) {
        holder.bindData(lista[position])
    }

    override fun getItemCount(): Int {
        return lista.size
    }

    fun setTerrenos(frases: List<TerrenosModelItem>) {
        lista = frases as ArrayList<TerrenosModelItem>
        notifyDataSetChanged()
    }

}
```

Con esto listo, comienzo a hacer las vistas de la aplicación.

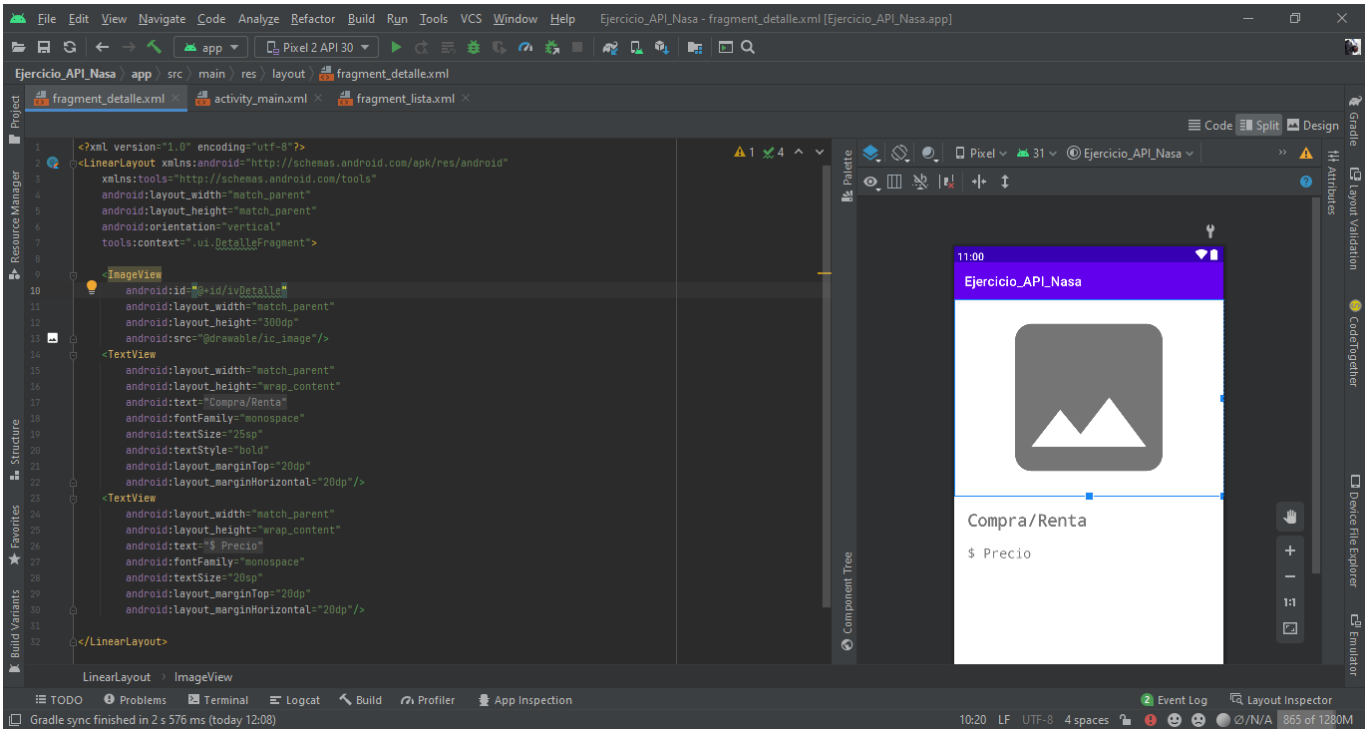
MainActivity



ListaFragment:



DetalleFragment:



Y bueno, hasta acá fue mi progreso durante la clase, a la tarde y mañana trataré de continuar y finalizar el ejercicio, muchas gracias.

Leo Rodenes Escobar 😄