Evidencia día 02 - Semana 15

**Cristian Maturana Ibarra**

**Ejercicio:**



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

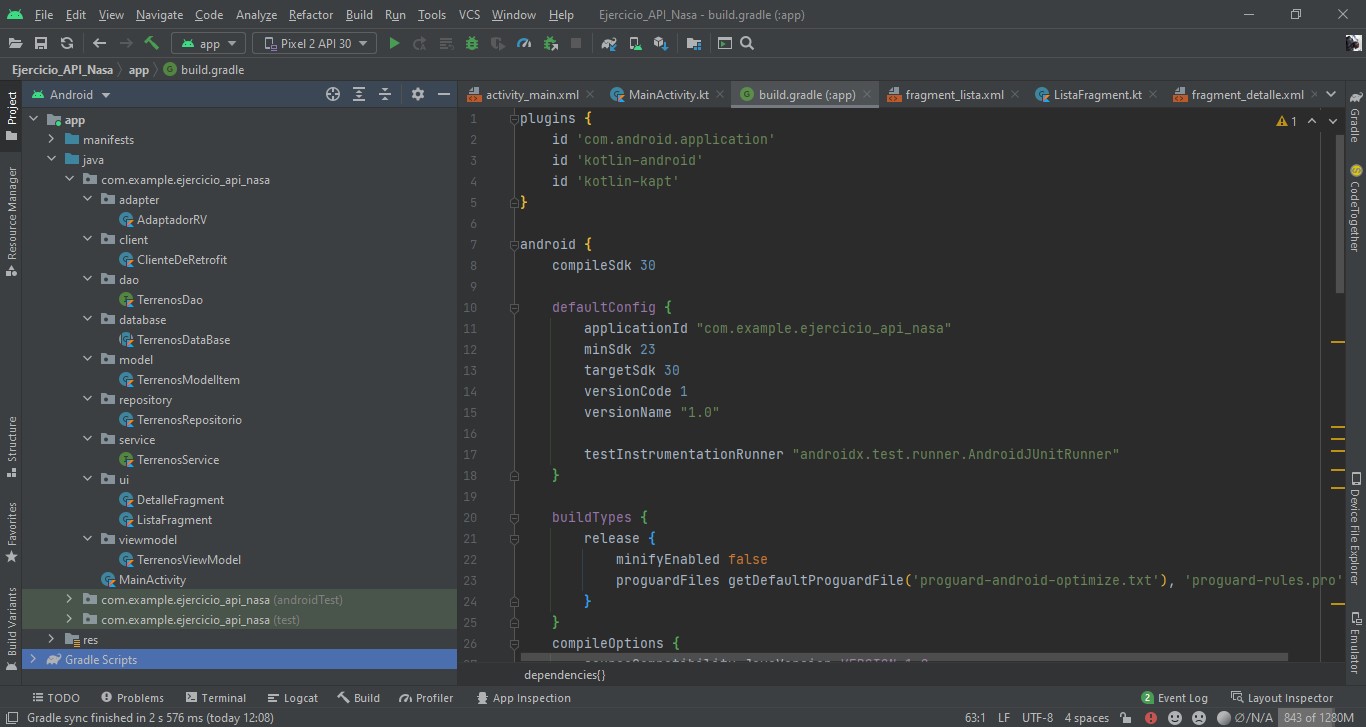
Luego creo los packages y clases/interfaces necesarias para implementar Room y la conexión a la API.

@Dao

interface

TerrenosDao

{



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:**

@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>

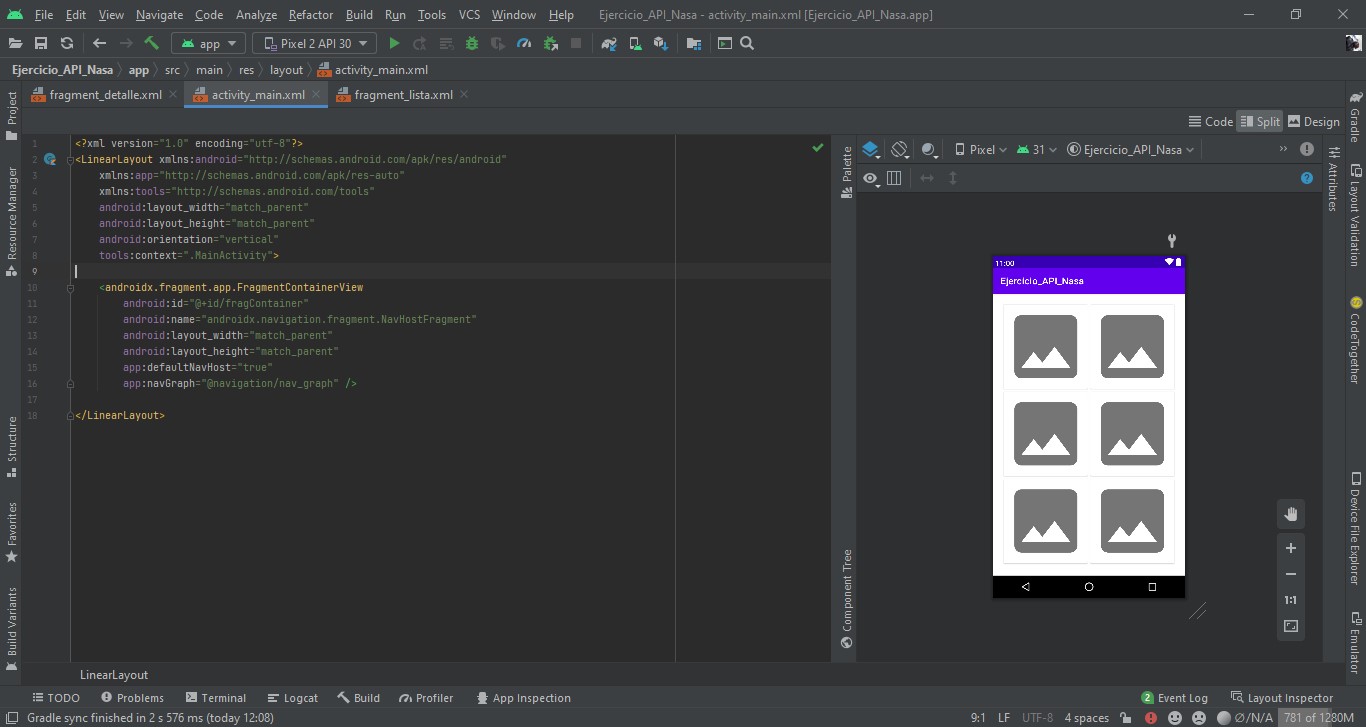
notifyDataSetChanged()

}

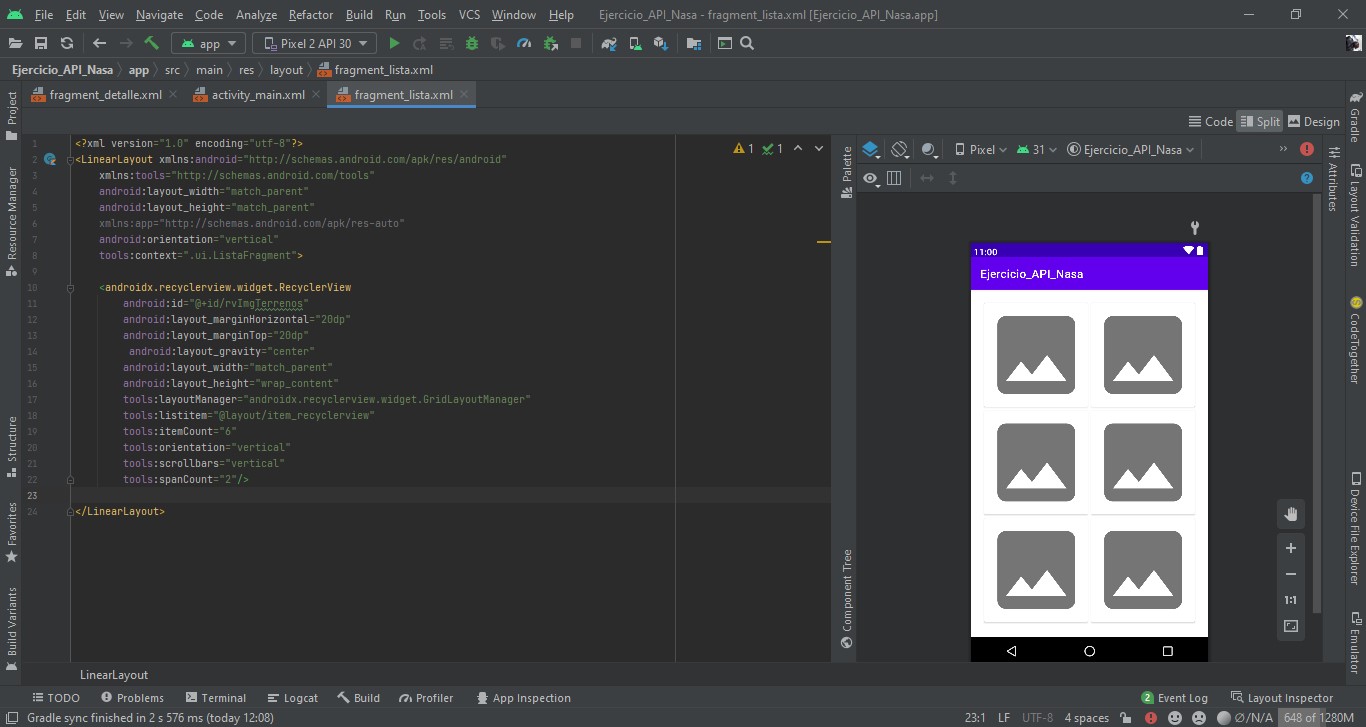
}

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

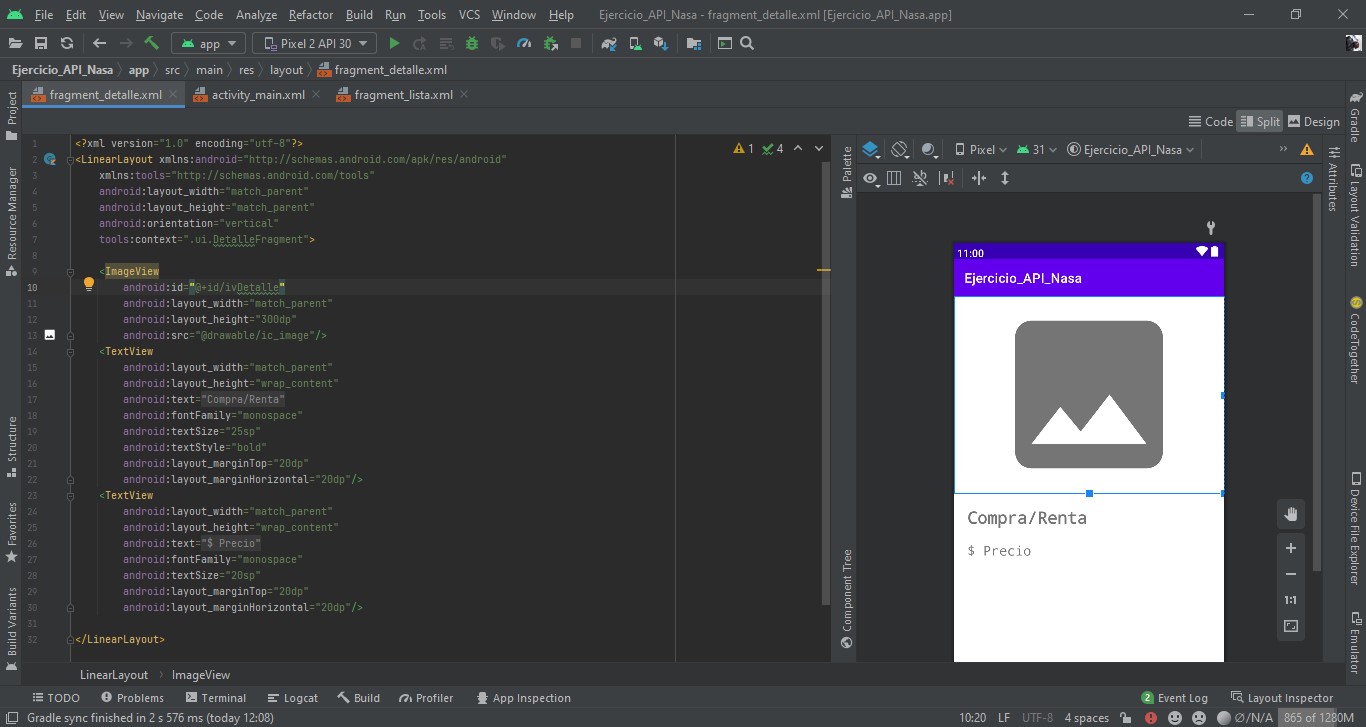
**MainActivity**



**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.

**Reflexión:**

Estos ejercicios son bastante complejos, cuesta de primera aprender los pasos pero con mucha practica voy aprendiendo la forma de llevarlos a cabo, espero seguir aprendiendo más.