Evidencia dia 03 Semana 15

Leonardo Rodenas Escobar

Reflexión:

El día de hoy se basó más en resolución de consultas por parte de nosotros los alumnos y en tratar de descubrir entre todos los problemas que existian en los proyectos de nuestro compañeros, que en entregar contenido nuevo. Estuvo bastante entretenido y didactico esa modalidad.

Por mi parte continue con el ejercicio de ayer resolviendo un par de dudas referentes a no poder mostrar los datos en el recycler view, lo cual se solucionó sacándome la confusion de lo que hacia un metodo (traer la info del server de la API a la base de datos)y dividiendolo este en esa funcion y en otra que mostrará los datos desde la Base de datos aL Recycler. Esto lo creé tanto en el repositorio como en el ViewModel, para de esta manera poder llamar a estas funciones desde la vista y poder ejecutarlas y observarlas con LiveData. Por ultimo corregí el listener del Adapter para convertir los objetos de la lista en clickeables.

Adjunto código de los cambios.

En 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 {
                        Log.v("logenrepo", response.body().toString())
                        terrenosDao.insertarTodosLosTerrenos(it)
                    }
                }
            }

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

        })
    }
```

```
//acá el mayor cambio, crear este método

    fun exponeDatosDelBaseDeDatos(): LiveData<List<TerrenosModelItem>> {
        return terrenosDao.obtenerTodosLosTerrenosDeLaBD()
    }
}
```

En ViewModel:

```
//Notar los nuevos métodos creados traemeLoDelServer y exponeDatosDeDB

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

    private var repositorio : TerrenosRepositorio

    init {

        val terrenosDao =
TerrenosDataBase.crearDatabase(application).obtenTerrenosDelDao()
        repositorio = TerrenosRepositorio(terrenosDao)

    }

    fun traemeLoDelServer() {

      repositorio.obtenDataDelServer()

    }

    fun exponeDatosDeDB():LiveData<List<TerrenosModelItem>> {
        return repositorio.exponeDatosDelBaseDeDatos()
    }
}
```

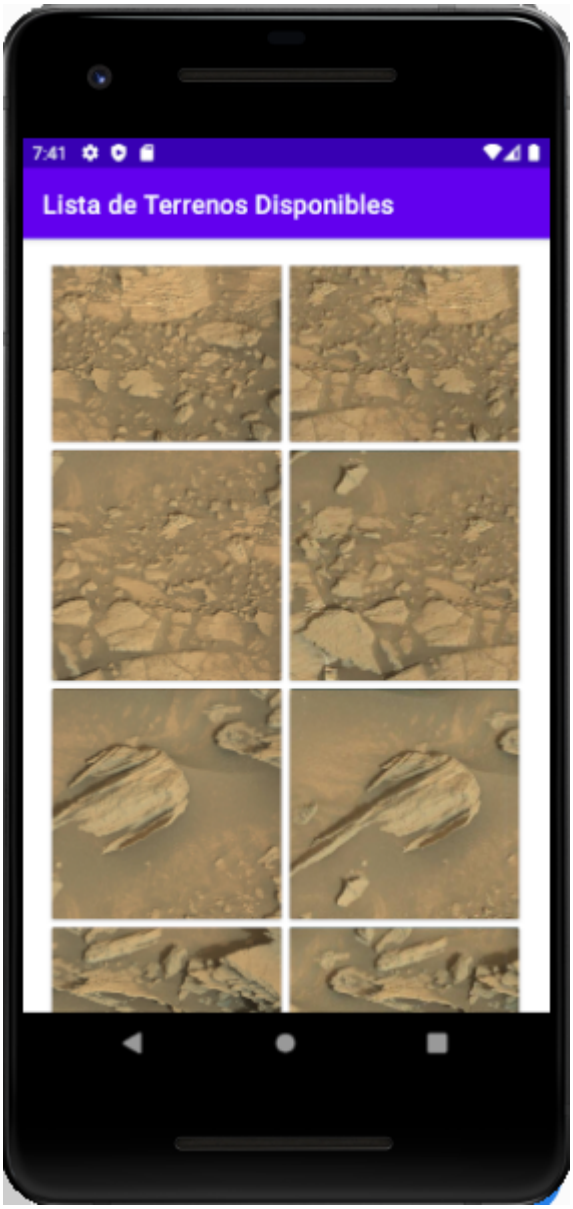
En Adapter:

```
//Notar los cambios en listener con la entrega anterior

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

    private var lista: List<TerrenosModelItem> = ArrayList()
    private lateinit var miListener: alClickearItemRV
```

```
    class CustomViewHolder(
        private val binding: ItemRecyclerviewBinding,
        private val listener: alClickearItemRV
    ) :
        RecyclerView.ViewHolder(binding.root) {

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

            Picasso.get().load(img.img_src).fit().centerCrop()
                .placeholder(R.drawable.user_placeholder)
                .error(R.drawable.user_placeholder_error)
                .into(binding.ivTerreno)
            binding.itemCard.setOnClickListener {

                listener.itemClick(adapterPosition)

            }

        }

    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
CustomViewHolder {
        return CustomViewHolder(
            ItemRecyclerviewBinding.inflate(
                LayoutInflater.from(parent.context),
                parent,
                false
            ), miListener
        )
    }

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

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

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

    interface alClickearItemRV {

        fun itemClick(position: Int)

    }

    fun setearListener(listener: alClickearItemRV) {
```

```
            miListener = listener
        }

    }
```

Con esto la vista de la App queda de la siguiente manera:



Ese sería mi avance y evidencia por el día de hoy, muchas gracias.

Leonardo Rodenas Escobar 😄