MODULO 4 | Desarrollo de Aplicaciones Móviles Android Kotlin | Ignacio Cavallo

https://github.com/cavigna/modulo_desarrollo_de_aplicaciones_moviles_android_kotlin

Clase 92 | 09-11

En la clase de hoy se propuso un ejercicio muy desafiante que consiste en aplicar lo aprendido en hasta el momento. Este consiste en hacer una llamada a una API, y con esa información guardarla de forma interna en una Base de Datos usando ROOM. El principal desafio será combinar fuentes de datos de forma local como remotas, a partir del patrón de vista MODELO VISTA VISTA MODELO. Estas fuentes de datos estarán contenidas en un repositorio. Más allá de algún contratiempo con el Gradle, la aplicación es funcional y busca datos de forma remota y lo almacena en la base de datos local. Usando MVVM, simplemente la vista observa los cambios de la base de datos a través del viewmodel.

> Dejo a continuación una parte del código. En el fragmento principal vemos el accionar del viewmodel, con un listado desde la base de datos(este a su vez recibe los datos desde una API).

CODIGO

MainActivity.kt

```
class HomeFragment : Fragment() {

    private lateinit var binding: FragmentHomeBinding

    private lateinit var application: Application

    private val viewModel : GalacticaViewModel by activityViewModels {
        GalacticaModelFactory((application as GalacticaApplication).repository)
    }



    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        application = requireActivity().application

    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        binding = FragmentHomeBinding.inflate(layoutInflater, container, false)

        val recyclerView = binding.recyclerView
```

```
        val adapter = TerrenoListAdapter()
        recyclerView.adapter = adapter
        recyclerView.layoutManager = GridLayoutManager(requireContext(), 1,
GridLayoutManager.VERTICAL,false)

        viewModel.listaTerrenosDB.observe(viewLifecycleOwner, {
            adapter.submitList(it)
        })


        return binding.root
    }
```