PATRÓN MVVM

1. Crear el paquete Model para manejar el modelo y lógica de negocio
2. Los datos y entidades en Data Class. Inicializarlos y usar **val** siempre que sea posible
3. Crear una clase que extienda de ViewModel
4. Usar:
    1. private _variable = MutableLiveData<T> (//Costructor del objeto T) // Mutable (solo puede ser modificada desde el ViewModel)
    2. private variable :LiveData<T> = _variable // LiveData (la vista solo puede observar, no modificar)
5. Desde los composable usar **observeAsState**:
    1. val variable by viewModel.variable.**observeAsState**(T()). //variable sería la variable/ objeto de referencia
    2. val listaVariables by viewModel.listaVariables.**observeAsState**(emptyList()) // para una lista de objetos. Se inicializa como vacía

CASO ESPECIAL. LAS LISTAS.

Las listas no notifican cambios de estado por modificar su contenido por lo que para que esto ocurra debemos tener una nueva referencia de la lista. Podemos adoptar las siguientes soluciones.

1. (Recomendado) val listaActual = _lista.value?.toMutableList() ?: mutableListOf()
   // Cambios en listaActual
   _lista.value = listaActual // Se asigna la nueva lista
2. private val _lista = MutableLiveData<MutableList<T>>(mutableListOf())