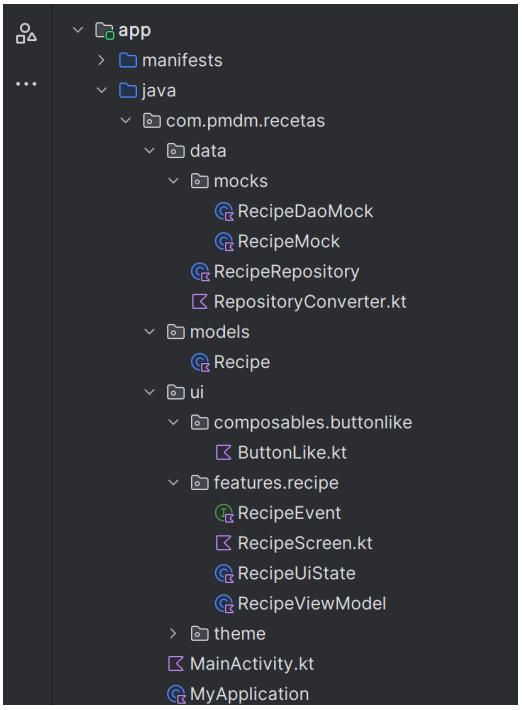
Definiendo ButtonLike recetas

Descargar estos apuntes

Ejercicio

Vamos a añadir al ejercicio `Card` donde se visualiza la información de una receta la lógica para elevar el estado al `ViewModel`. En el *preview* se muestra el card siguiendo los convenios de Material Design 3 (M3).

Siguiendo la arquitectura propuesta para nuestras aplicaciones, la estructura de paquetes y contenido de los mismos con sus correspondientes archivos `.kt` es la siguiente:



Paso 1: Definiendo el `model`

En el paquete **`com.pmdm.recetas`** vamos a crear el paquete **`models`** donde según la arquitectura propuesta en el curso, se definirán los modelos de datos de nuestra **`app`**. Dentro crearemos el fichero **`Recipe.kt`**

```
data class Recipe(
    val name: String,
    val description: String,
    val chef: String,
    val photo: ImageBitmap?,
    val likes: Int,
    val isLiked: Boolean
)
```

Estamos simplificando bastante la solución del ejercicio, por eso vamos a suponer que el atributo `isLiked` pertenece al objeto receta. En una aplicación más desarrollada este atributo pertenecería a una relación existente entre el usuario concreto y la receta.

Paso 2: Definiendo el paquete `data.mocks`

En el paquete **`com.pmdm.recetas.data.mocks`** vamos a crear dos archivos: **`RecipeDaoMock.kt`** (una `class`) y **`RecipeMock.kt`** (un `data class`).

La clase **`RecipeDaoMock.kt`** solo contendrá una objeto de tipo receta con los métodos que me permitirán obtener la eceta y actualizar la receta (cuando se modifique a través de la acción del `ButtonLike`):

```
class RecipeDaoMock {
    //solo tengo una receta
    private var recipe =
        RecipeMock(
            name = "Magdalenas de la abuela",
            description = "Fabulosas magdalenas con pepitas de chocolate y un suave sabor a n
            chef = "Carlos Arguiñano",
            photo = null,
            likes = 8,
            isLiked = false
        )

    fun get(): RecipeMock =recipe
    fun updateRecipe(recipeRemote:RecipeMock){
        recipe=recipeRemote
    }
}
```

El 'data class' `RecipeMock.kt` es similar en su definición a la clase de nuestro modelo `Recipe.kt`.

Dentro del paquete `data`, pero no dentro de `data.mocks`, definimos los archivos `RecipeRepository.kt` (una `class`) y `RepositoruCoverter.kt`.

Con los conocimientos adquiridos en ejercicios previos debéis poder definir estos dos elementos.

Paso 3: Definiendo el estado de la vista

Para manejar el estado de nuestra vista definiremos el archivo `RecipeUiState.kt`:

```
data class RecipeUiState(
    val recipeName: String,
    val recipeDescription: String,
    val recipeChef: String,
    var recipeFoto: ImageBitmap?,
    val numberOfLikes: Int,
    val iLike: Boolean
)
```

Con todos estos elementos definidos es el momento de que definái el `View Model`.

A tener en cuenta en esta definición es que a través de `RecipeEvent` vamos a gestionar todos los eventos sobre el `Card`. En nuestro caso solo será uno, pero mantenemos esta estructura de funcionamiento, que ya hemos utilizado en ejemplos anteriores, para fijar en nuestra mente esta forma de procesar eventos.