PROGRAMACIÓN MULTIMEDIA Y DISPOSITIVOS MÓVILES

Práctica Evaluable:
Descargar el repositorio de Dogs en la rama testroom y realizar cambios.
REPOSITORIO: https://github.com/alexcl885/dogs-androidStudio/tree/testRoom
He clonado el repositorio pero me ha dado problemas por lo que he tenido que montarlo yo desde nuevo, introduciendo todas las dependencias y asegurando que todo funciona correctamente.
Por lo que lo he probado y va todo bien.
—
EXPLICACIÓN DE LA REALIZACIÓN DE ELIMINAR:
Y ahora lo que nos pide el ejercicio es crear un nuevo caso de uso por lo que primero que hay que hacer es añadir el método de eliminar un perro en el domain para implementarlo en la data del repositorio.

Y ahora lo implemento en el repositorio de la data:

```
override suspend fun deleteDog(dog: DogEntity) {
    dogDao.deleteDog(dog.breed)
}
```

Pero para hacerlo he necesitado en dogDao añadir una query donde elimine el perro según la raza del perro, por lo que me creo en DogDao esta query donde se elimina el perro con la raza que elija el usuario.

```
@Query("DELETE FROM dogentity WHERE breed = :breed")
suspend fun deleteDog(breed: String)
```

Por lo que ahora me creo el caso de uso:

```
class DeleteDogUseCase@Inject constructor(
    private val dogRepositoryDao: DogRepository
){
    var dog : Dog? = null
    suspend  operator  fun invoke(dog:Dog): Boolean{

        return if (this.dog != null){
            this.dog?.let { dogRepositoryDao.deleteDog(it.toDogEntity()) }
            true
        }
        else{
            false
        }
    }

}
```

ViewModel

Ya en el viewModel realizó lo siguiente:

Instancio en el constructor el caso de uso y gracias a Hilt me lo inicializa solo.

```
@HiltViewModel
class DogViewModel @Inject constructor(
    private val useCaseList : GetDogsUseCase,
    private val getDogsBreedUseCase: GetDogsBreedUseCase,
    private val userCaseDeleteDatabase : DeleteDogsFromDataBa
    private val insertDog : PostDogEntity,
    private val deleteDogUseCase : DeleteDogUseCase
): ViewModel() {
```

Creo esta función en la cual invoca al caso de uso:

```
fun deleteDogFunction(dog: Dog) {
    viewModelScope.launch {
        progressBarLiveData.value = true
        var res = false
        withContext(Dispatchers.IO) {
            deleteDogUseCase.dog = dog
            res = deleteDogUseCase(dog)
        }
        progressBarLiveData.value = false
        deleteDogLiveData.value = res

        if (res) {
            list()
        }
    }
}
```

MAIN ACTIVITY

En el main activity digamos que estará el policía que estará observando si hay cambios o no para mostrar un mensaje y al final recarga la lista de perros para ver que se ha eliminado.

```
dogViewModel.deleteDogLiveData.observe( owner: this) { isDelete ->
    if (isDelete) {
        Toast.makeText( context: this,  text: "Perro eliminado", Toast.LE
        dogViewModel.list() //recarga la lista al eliminar el perro
    } else {
        Toast.makeText( context: this,  text: "No se pudo eliminar", Toas
    }
}
```

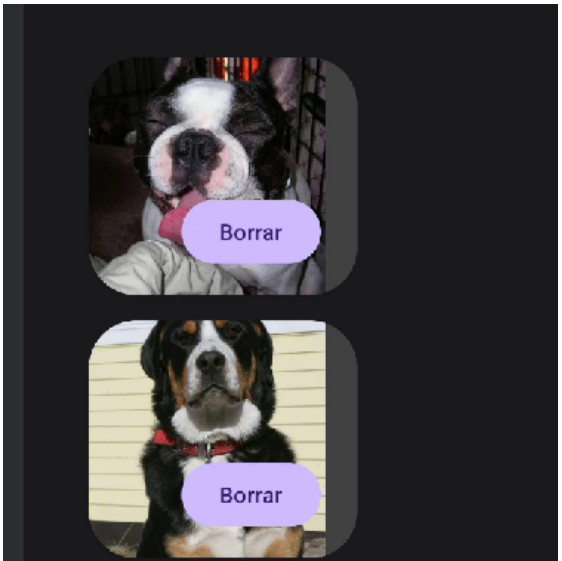
Creo esta función para pasarsela al adapter para eliminar el perro segun su posicion:

```
fun deleteDogTest(pos: Int) {
    val dogList = adapter.dogRepository
    if (pos in dogList.indices) {
        val dog = dogList[pos]
        dogViewModel.deleteDogFunction(dog)
    }
}
```

```
private fun initRecyclerView() {
    adapter = DogAdapter{position -> deleteDogTest(position)}
    binding.myRecyclerPpal.layoutManager = LinearLayoutManager(
    binding.myRecyclerPpal.adapter = adapter
}
```

PARTE VIEW

He creado un botón para que en cada ítem del perro tenga un botón para poder eliminar al pulsar el perro que elija el usuario.Ejemplo:



Por la parte de añadir futuros cambios podría decir de añadir nuevos casos de usos para dar más utilidad a la aplicación para que fuera más funcional como implementar una pestaña de favoritos o alguna integración de alguna api externa y por parte de la arquitectura lo veo todo bien y puede ser que en un futuro pues se realicen cambios pero ahora no sabría que implementar.