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

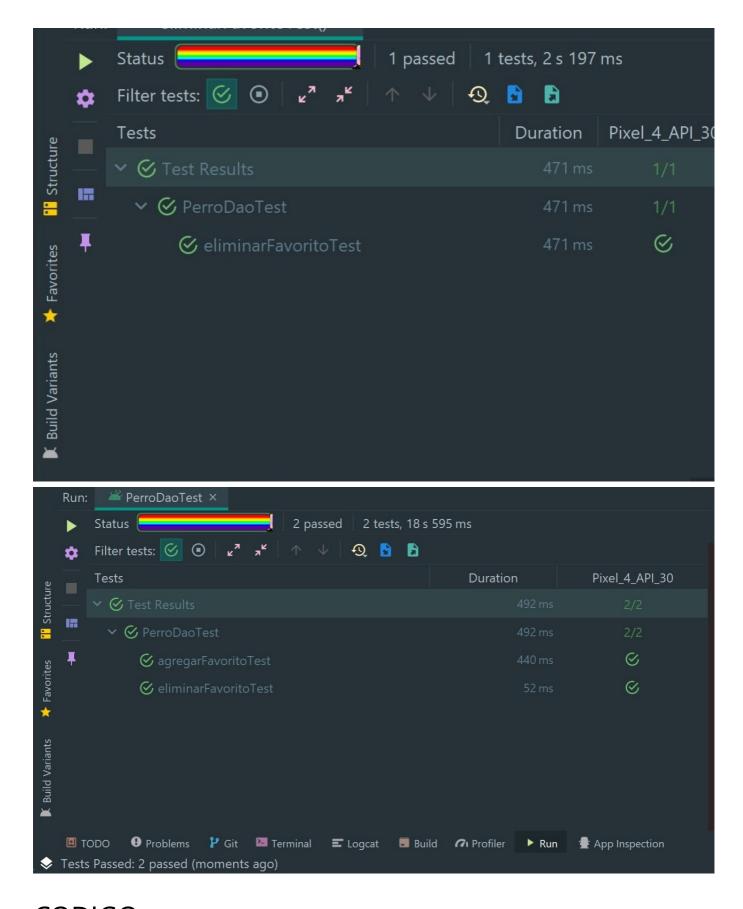
https://github.com/cavigna/modulo_desarrollo_de_aplicaciones_moviles_android_kotlin

Clase 97 | 16-11

En esta clase nos enseñaron Testing de Room. Fue una clase interesante ya que uno de los grandes temas pendientes, fueron testing. En definitiva implementando la librería de Google,se hizo todo relativamente sencillo.

https://github.com/cavigna/DogsPics

A continuación muestro unas capturas de pantalla



CODIGO

PerroViewModel.kt

```
package com.example.dogspics
import android.content.Context
import androidx.arch.core.executor.testing.InstantTaskExecutorRule
import androidx.lifecycle.asLiveData
import androidx.room.Room
import androidx.test.core.app.ApplicationProvider
import androidx.test.ext.junit.runners.AndroidJUnit4
import androidx.test.filters.SmallTest
import cl.duoc.ejemploroom.getOrAwaitValue
import com.example.dogspics.dao.PerroDao
import com.example.dogspics.db.BaseDeDatos
import com.example.dogspics.model.PerroFavorito
import com.google.common.truth.Truth.assertThat
import kotlinx.coroutines.ExperimentalCoroutinesApi
import kotlinx.coroutines.runBlocking
import kotlinx.coroutines.test.runBlockingTest
import org.junit.After
import org.junit.Before
import org.junit.Rule
import org.junit.Test
import org.junit.runner.RunWith
@ExperimentalCoroutinesApi
@RunWith(AndroidJUnit4::class)
@SmallTest
class PerroDaoTest {
    @get:Rule
    var instantTaskExecutorRule = InstantTaskExecutorRule()
    private lateinit var database: BaseDeDatos
    private lateinit var dao: PerroDao
    @Before
    fun setup() {
        val context = ApplicationProvider.getApplicationContext<Context>()
        database = Room.inMemoryDatabaseBuilder(context, BaseDeDatos::class.java)
           .allowMainThreadQueries()
            .build()
        dao = database.dao()
    }
    @After
    fun teardown(){
        database.close()
    }
    @Test
    fun agregarFavoritoTest() = runBlockingTest {
        var perro = PerroFavorito(id = 1, raza = "labrador", imagenUrl = "url")
```

```
dao.agregarFavorito(perro)

val listadoPerroFavoritoTest =
dao.listadoDeFavoritos().asLiveData().getOrAwaitValue()

assertThat(listadoPerroFavoritoTest).contains(perro)

}

@Test
fun eliminarFavoritoTest() = runBlockingTest {
 val perro = PerroFavorito(2, "boxer", "url")

dao.agregarFavorito(perro)
 dao.eliminarFavorito(perro)

val listadoPerroFavoritoTest =
dao.listadoDeFavoritos().asLiveData().getOrAwaitValue()

assertThat(listadoPerroFavoritoTest).doesNotContain(perro)

}
}
```