Proyecto Final Desarrollo de Aplicaciones Móviles

Consigna:

Para la realización de este proyecto, te invitamos a imaginarte que sos un desarrollador recién incorporado en un nuevo emprendimiento tecnológico.

Te solicitan que desarrolles una aplicación que tenga un solo activity que cumpla con las siguientes premisas:

Una única pantalla (sin importar el layout elegido) con 2 cuadros de textos (EditText)

1 botón con el texto “comparar”

1 texto (Textview) en el que se escriba el resultado de la acción al presionar el botón.

Asegúrate de que:

- Utiliza MVVM.

- Tiene al menos un test unitario.

- Tiene al menos un test de UI.

Función de la app cuando el usuario hace cilck en el botón “comparar” debe comparar la entrada de ambos cuadros de texto y escribir en el texto (TextView) si ambas cadenas de caracteres son iguales o no.

Para subir la carpeta de tu proyecto, te proponemos:

Usar un repositorio de GitHub pages, haciendo la entrega del link. En caso de tener alguna duda con la subida, te dejamos aquí un video instructivo que te ayudará paso a paso con GitHub pages

Desarrollo:

Link de github: <https://github.com/mmichelutti/Proyecto_final_app_moviles>

Capturas de la aplicación:

- Estado inicial de la app



- Cuando se hace click en “comparar” se indica que “No se ha ingresado ningún texto”



- Caso que se ingresan dos textos iguales y se hace click en “comparar”, se indica que “Los textos son iguales”



- Caso que se ingresan dos textos diferentes se hace click en “comparar”, se indica que “Los textos son diferentes”



TextCompareModel.kt

package com.curso.android.app.proyectofinal3  
  
// Modelo de datos  
data class TextCompareModel(val text1: String, val text2: String, val comparisonResult: String)

activity\_main.xml

<?xml version="1.0" encoding="utf-8"?>  
<layout xmlns:android="http://schemas.android.com/apk/res/android"  
xmlns:app="http://schemas.android.com/apk/res-auto">  
  
<data>  
 <variable  
 name="viewModel"  
 type="com.curso.android.app.proyectofinal3.TextCompareViewModel" />  
</data>  
  
<LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:padding="16dp">  
  
 <EditText  
 android:id="@+id/editText1"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="@string/ingrese\_el\_primer\_texto" />  
  
 <EditText  
 android:id="@+id/editText2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="@string/ingrese\_el\_segundo\_texto" />  
  
 <Button  
 android:id="@+id/buttonCompare"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="@string/comparar" />  
  
 <TextView  
 android:id="@+id/textViewResult"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@{viewModel.textCompareModel.comparisonResult}"  
 android:textSize="18sp" />  
</LinearLayout>  
</layout>

TextCompareViewModel.kt

package com.curso.android.app.proyectofinal3  
  
import androidx.lifecycle.MutableLiveData  
import androidx.lifecycle.LiveData  
import androidx.lifecycle.ViewModel  
  
class TextCompareViewModel : ViewModel() {  
  
 // LiveData para almacenar el modelo de datos  
 // textCompareModel solo se puede modificar dentro del ViewModel  
 val textCompareModel: LiveData<TextCompareModel> get() = \_textCompareModel  
 private var \_textCompareModel = MutableLiveData<TextCompareModel>()  
  
 // Función para comparar dos textos y actualizar el modelo de datos  
 fun compareTexts(text1: String, text2: String) {  
 val comparisonResult = when {  
 text1.*isEmpty*() && text2.*isEmpty*() -> "No se ha ingresado ningún texto" // Esta línea se agrega para indicar si no hay texto  
 text1 == text2 -> "Los textos son iguales"  
 else -> "Los textos son diferentes"  
 }  
 \_textCompareModel.*value* = TextCompareModel(text1, text2, comparisonResult)  
 }  
}

MainActivity.kt

package com.curso.android.app.proyectofinal3  
  
import android.os.Bundle  
import androidx.appcompat.app.AppCompatActivity  
import androidx.lifecycle.ViewModelProvider  
import com.curso.android.app.proyectofinal3.databinding.ActivityMainBinding  
  
class MainActivity : AppCompatActivity() {  
  
 // Referencia al ViewModel  
 private lateinit var viewModel: TextCompareViewModel  
  
 // Referencia al binding generado por Data Binding  
 private lateinit var binding: ActivityMainBinding  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
  
 // Inicializar el ViewModel usando el ViewModelProvider  
 viewModel = ViewModelProvider(this).get(TextCompareViewModel::class.*java*)  
  
 // Inicializar el binding usando la librería ViewBinding  
 binding = ActivityMainBinding.inflate(*layoutInflater*)  
 setContentView(binding.*root*)  
  
 // Establecer el lifecycle owner del binding para que pueda observar los cambios en el LiveData  
 binding.*lifecycleOwner* = this  
  
 // Establecer el ViewModel del binding para que pueda acceder a sus propiedades y funciones  
 binding.*viewModel* = viewModel  
  
 // Establecer el listener del botón comparar para que llame a la función compareTexts del ViewModel  
 binding.buttonCompare.setOnClickListener **{** viewModel.compareTexts(binding.editText1.*text*.toString(), binding.editText2.*text*.toString())  
 **}** }  
}

TextCompareViewModelUnitTest.kt

package com.curso.android.app.proyectofinal3  
  
import androidx.arch.core.executor.testing.InstantTaskExecutorRule  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.test.\*  
import org.junit.After  
import org.junit.Assert.\*  
import org.junit.Before  
import org.junit.Rule  
import org.junit.Test  
  
@OptIn(kotlinx.coroutines.ExperimentalCoroutinesApi::class)  
class TextCompareViewModelUnitTest {  
 // Referencia al ViewModel que se va a probar  
 private lateinit var viewModel: TextCompareViewModel  
  
 // Regla para ejecutar las tareas en el hilo principal  
 @get:Rule  
 val instantTaskRule = InstantTaskExecutorRule()  
 private val dispatcher = *StandardTestDispatcher*()  
  
 // Inicializar el ViewModel antes de cada test  
 @Before  
 fun setup() {  
 Dispatchers.*setMain*(dispatcher)  
 viewModel = TextCompareViewModel()  
 }  
 @After  
 fun tearDown() {  
 Dispatchers.*resetMain*()  
 }  
  
 // Test para verificar que se indique si no se ha ingresado ningún texto  
 @Test  
 fun compareTexts\_emptyTexts\_returnsNoTextMessage() {  
 // Arrange: preparar los datos de entrada y salida esperada  
 val text1 = ""  
 val text2 = ""  
 val expected = "No se ha ingresado ningún texto"  
  
 // Act: invocar la función que se va a probar con los datos de entrada  
 viewModel.compareTexts(text1, text2)  
  
  
 // Assert: verificar que el resultado obtenido sea igual al esperado  
 assertEquals(expected, viewModel.textCompareModel.*value*?.comparisonResult)  
 }  
  
 // Test para verificar que se indique si los textos son iguales  
 @Test  
 fun compareTexts\_equalTexts\_returnsEqualMessage() {  
 // Arrange: preparar los datos de entrada y salida esperada  
 val text1 = "Hola"  
 val text2 = "Hola"  
 val expected = "Los textos son iguales"  
  
 // Act: invocar la función que se va a probar con los datos de entrada  
 viewModel.compareTexts(text1, text2)  
  
 // Assert: verificar que el resultado obtenido sea igual al esperado  
 assertEquals(expected, viewModel.textCompareModel.*value*?.comparisonResult)  
 }  
  
 // Test para verificar que se indique si los textos son diferentes  
 @Test  
 fun compareTexts\_differentTexts\_returnsDifferentMessage() {  
 // Arrange: preparar los datos de entrada y salida esperada  
 val text1 = "Hola"  
 val text2 = "Chau"  
 val expected = "Los textos son diferentes"  
  
 // Act: invocar la función que se va a probar con los datos de entrada  
 viewModel.compareTexts(text1, text2)  
  
 // Assert: verificar que el resultado obtenido sea igual al esperado  
 assertEquals(expected, viewModel.textCompareModel.*value*?.comparisonResult)  
 }  
}

MainActivityInstrumentedTest.kt

package com.curso.android.app.proyectofinal3  
  
import androidx.test.espresso.Espresso.onView  
import androidx.test.espresso.action.ViewActions.\*  
import androidx.test.espresso.assertion.ViewAssertions.matches  
import androidx.test.espresso.matcher.ViewMatchers.\*  
import androidx.test.ext.junit.rules.ActivityScenarioRule  
import androidx.test.ext.junit.runners.AndroidJUnit4  
import org.junit.Assert.\*  
import org.junit.Rule  
import org.junit.Test  
import org.junit.runner.RunWith  
  
@RunWith(AndroidJUnit4::class)  
class MainActivityInstrumentedTest {  
  
 // Regla para lanzar la actividad que se va a probar  
 @get:Rule  
 val activityRule = ActivityScenarioRule(MainActivity::class.*java*)  
  
 // Test para verificar que se muestre el mensaje de no texto si no se ingresa ningún texto  
 @Test  
 fun compareTexts\_emptyTexts\_displaysNoTextMessage() {  
 // Arrange: preparar los datos de entrada y salida esperada  
 val text1 = ""  
 val text2 = ""  
 val expected = "No se ha ingresado ningún texto"  
  
 // Act: interactuar con la interfaz usando Espresso  
 onView(withId(R.id.*editText1*)).perform(typeText(text1), closeSoftKeyboard())  
 onView(withId(R.id.*editText2*)).perform(typeText(text2), closeSoftKeyboard())  
 onView(withId(R.id.*buttonCompare*)).perform(click())  
  
 // Assert: verificar que se muestre el mensaje esperado en el TextView  
 onView(withId(R.id.*textViewResult*)).check(matches(withText(expected)))  
 }  
  
 // Test para verificar que se muestre el mensaje de igualdad si se ingresan textos iguales  
 @Test  
 fun compareTexts\_equalTexts\_displaysEqualMessage() {  
 // Arrange: preparar los datos de entrada y salida esperada  
 val text1 = "Hola"  
 val text2 = "Hola"  
 val expected = "Los textos son iguales"  
  
 // Act: interactuar con la interfaz usando Espresso  
 onView(withId(R.id.*editText1*)).perform(typeText(text1), closeSoftKeyboard())  
 onView(withId(R.id.*editText2*)).perform(typeText(text2), closeSoftKeyboard())  
 onView(withId(R.id.*buttonCompare*)).perform(click())  
  
 // Assert: verificar que se muestre el mensaje esperado en el TextView  
 onView(withId(R.id.*textViewResult*)).check(matches(withText(expected)))  
 }  
  
 // Test para verificar que se muestre el mensaje de diferencia si se ingresan textos diferentes  
 @Test  
 fun compareTexts\_differentTexts\_displaysDifferentMessage() {  
 // Arrange: preparar los datos de entrada y salida esperada  
 val text1 = "Hola"  
 val text2 = "Adiós"  
 val expected = "Los textos son diferentes"  
  
 // Act: interactuar con la interfaz usando Espresso  
 onView(withId(R.id.*editText1*)).perform(typeText(text1), closeSoftKeyboard())  
 onView(withId(R.id.*editText2*)).perform(typeText(text2), closeSoftKeyboard())  
 onView(withId(R.id.*buttonCompare*)).perform(click())  
  
 // Assert: verificar que se muestre el mensaje esperado en el TextView  
 onView(withId(R.id.*textViewResult*)).check(matches(withText(expected)))  
 }  
}