Desarrollá una aplicación que tenga un solo activity que cumpla con las siguientes

**Premisas:**

Contenga una única pantalla (sin importar el layout elegido) con:

1. 2 cuadros de textos (EditText)
2. 1 botón con el texto “comparar”
3. 1 texto (TextView) que en el que se escriba el resultado de la acción al presionar el botón.

**Asegurarse de que:**

1. Utiliza MVVM
2. Tiene al menos un test unitario
3. Tiene al menos un test de UI

**Función de la aplicación:**

Cuando el usuario hace clic 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.

**Requerimientos:**

Usá un repositorio de GitHub pages para subir tu proyecto (Repositorio app laboratorio: <https://github.com/eaceto/ticmas-lab-android(opens> in a new tab)). Con lo visto en el último módulo del curso, vas a poder subir sin problema tu proyecto a la web.

Generá un archivo .doc con lo planteado en la consigna y el link de tu entregable. Luego, subilo en el bloque “Proyectos” que se encuentra en la página principal del curso, en la plataforma

**Paso 1: Configuración del proyecto**

**Abrir Android Studio y crear un nuevo proyecto de Android. Seleccionar la configuración inicial del proyecto, como el nombre de la aplicación, el nombre del paquete, etc. Eligir Kotlin como lenguaje de programación.**

**Paso 2: Diseño de la interfaz de usuario.**

**En el archivo de diseño XML de la actividad (activity\_main.xml), crear la interfaz de usuario con los elementos mencionados (2 EditText, 1 Button y 1 TextView).**

<?xml version=”1.0” encoding=”utf-8”?>

<LinearLayout

Xmlns:android=<http://schemas.android.com/apk/res/android>

Xmlns:app=<http://schemas.android.com/apk/res-auto>

Xmlns:tools=<http://schemas.android.com/tools>

Android:layout\_width=”match\_parent”

Android:layout\_height=”match\_parent”

Android:orientation=”vertical”

Android:padding=”16dp”

Tools:context=”.MainActivity”>

<EditText

Android:id=”@+id/editText1”

Android:layout\_width=”match\_parent”

Android:layout\_height=”wrap\_content”

Android:hint=”Texto 1”/>

<EditText

Android:id=”@+id/editText2”

Android:layout\_width=”match\_parent”

Android:layout\_height=”wrap\_content”

Android:hint=”Texto 2”/>

<Button

Android:id=”@+id/buttonCompare”

Android:layout\_width=”match\_parent”

Android:layout\_height=”wrap\_content”

Android:text=”Comparar”/>

<TextView

Android:id=”@+id/textResult”

Android:layout\_width=”match\_parent”

Android:layout\_height=”wrap\_content”

Android:text=””

Android:paddingTop=”16dp”/>

</LinearLayout>

**Paso 3: Configurar ViewModel.**

**Crear un ViewModel (MyViewModel.kt) que gestionará la lógica de la aplicación y almacenará el resultado de la comparación. El ViewModel se encargará de comparar las cadenas y proporcionará el resultado a la actividad.**

Import androidx.lifecycle.ViewModel

Class MyViewModel : ViewModel() {

Var resultText: String = “”

Fun compareStrings(text1: String, text2: String) {

resultText = if (text1 == text2) {

“Las cadenas son iguales”

} else {

“Las cadenas son diferentes”

}

}

}

**Paso 4: Configurar la actividad.**

**En la actividad principal (MainActivity.kt), configurar la lógica para manejar el botón “Comparar” y observar los cambios en el ViewModel.**

Import androidx.appcompat.app.AppCompatActivity

Import android.os.Bundle

Import androidx.lifecycle.ViewModelProvider

Import kotlinx.android.synthetic.main.activity\_main.\*

Class MainActivity : AppCompatActivity() {

Private lateinit var viewModel: MyViewModel

Override fun onCreate(savedInstanceState: Bundle?) {

Super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

viewModel = ViewModelProvider(this).get(MyViewModel::class.java)

buttonCompare.setOnClickListener {

val text1 = editText1.text.toString()

val text2 = editText2.text.toString()

viewModel.compareStrings(text1, text2)

textResult.text = viewModel.resultText

}

}

}

**Paso 5: Configurar pruebas unitarias.**

**Crear un archivo de prueba (MyViewModelTest.kt) para probar el ViewModel.**

Import org.junit.Test

Import org.junit.Assert.assertEquals

Class MyViewModelTest {

@Test

Fun testCompareStrings() {

Val viewModel = MyViewModel()

viewModel.compareStrings(“Hola”, “Hola”)

assertEquals(“Las cadenas son iguales”, viewModel.resultText)

viewModel.compareStrings(“Hola”, “Mundo”)

assertEquals(“Las cadenas son diferentes”, viewModel.resultText)

}

}

**PASO 6: Configurar pruebas de interfaz de usuario (UI) para las pruebas de UI, puedes usar el framework Espresso. Crea un archivo de prueba (MainActivityTest.kt) para realizar pruebas de interfaz de usuario.**

Import androidx.test.espresso.Espresso.\*

Import androidx.test.espresso.action.ViewActions.\*

Import androidx.test.espresso.assertion.ViewAssertions.\*

Import androidx.test.espresso.matcher.ViewMatchers.\*

Import androidx.test.ext.junit.rules.ActivityScenarioRule

Import org.junit.Rule

Import org.junit.Test

Class MainActivityTest {

@get:Rule

Val activityRule = ActivityScenarioRule(MainActivity::class.java)

@Test

Fun testCompareButton() {

onView(withId(R.id.editText1)).perform(typeText(“Hola”))

onView(withId(R.id.editText2)).perform(typeText(“Hola”))

onView(withId(R.id.buttonCompare)).perform(click())

onView(withId(R.id.textResult)).check(matches(withText(“Las cadenas son iguales”)))

}

}

**Paso 7: Ejecutar pruebas**

**Ejecuta las pruebas unitarias y las pruebas de interfaz de usuario para asegurarte de que todo funcione correctamente.**

.