**«Санкт-Петербургский Государственный Политехнический Университет»**

Кафедра компьютерных систем и программных технологий

**Отчёт по лабораторной работе №5**

Тема: «UI Tests»

Дисциплина: «Проектирование мобильных приложений»

|  |  |  |
| --- | --- | --- |
| Выполнил студент гр.3530901/70202 |  | Д.В. Павлов |
| Преподав­атель |  | А.Н. Кузнецов |
|  |  | «15» декабря 2019 |
|  |  |  |

**1 Цели работы**

Ознакомиться с принципами и получить практические навыки разработки UI тестов для Android приложений.

**2. Задачи работы**

**-** Разработка простейшего UI теста;

- Тестирование навигации в приложении из 3-ей лабораторной работы.

**3. Выполнение работы**

**3.1. Простейший UI тест**

Для тестирования было создано небольшое приложение, состоящее из одного Activity. В него была добавлена одна кнопка и одно текстовое поле. Суть приложения следующая: текст кнопки *(“my\_button“)* после первого нажатия изменяется на *“my\_text!”.*

Требуется написать Espresso тест для проверки сохранения содержимого текстового поля при повороте экрана и сбрасывания содержимого кнопки в исходное состояние при повороте экрана. Вид приложения сразу после его запуска (Рис.3.1.1) и после нажатия на кнопку (Рис.3.1.2):

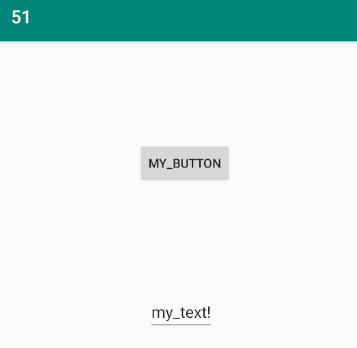
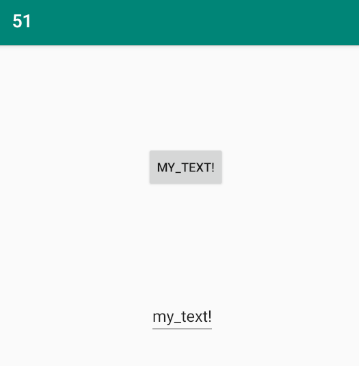
 

Рис.3.1.1. Рис.3.1.2.

Код Activity приложения представлен в Листинге 3.1.1.

Листинг 3.1.1.

package com.example.a51  
  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.widget.Button  
  
class MainActivity : AppCompatActivity() {  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
  
 val button = findViewById<Button>(R.id.*button*)  
 button.setOnClickListener**{** button.*text* = getString(R.string.*my\_text*)  
 **}** }  
  
}

Был написан тест, проверяющий обозначенный функционал приложения. Изначально устройство имеет вертикальную ориентацию. Производится проверка начального текста на кнопке, и далее совершается нажатие на нее. Замена её текста на *“my\_text!”* происходит успешно. Затем происходит очистка и запись нового текста в *editText*, ориентация меняется на горизонтальную. Текст на кнопке успешно изменился на первоначальный, а поле *editText* сохранило свое состояние. Далее мы вновь перезаписываем *editText* (возвращаем изначальное содержимое). Сравнение произошло успешно и совершается последний поворот экрана. Вновь совершается нажатие на кнопку и вновь текст успешно изменяется.

Код теста представлен в Листинге 3.1.2.

Листинг 3.1.2.

package com.example.a51  
  
import android.content.pm.ActivityInfo  
import androidx.test.InstrumentationRegistry  
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.withId  
import androidx.test.espresso.matcher.ViewMatchers.withText  
import androidx.test.rule.ActivityTestRule  
import androidx.test.runner.AndroidJUnit4  
  
import org.junit.Test  
import org.junit.runner.RunWith  
  
import org.junit.Assert.\*  
import org.junit.Rule  
  
  
*/\*\*  
 \* Instrumented test, which will execute on an Android device.  
 \*  
 \* See [testing documentation](http://d.android.com/tools/testing).  
 \*/*@RunWith(AndroidJUnit4::class)  
class MainActivityTest {  
  
 @get:Rule  
 val activityRule = ActivityTestRule(MainActivity::class.*java*)  
  
  
 private fun changeEditTextCheck() {  
 onView(withId(R.id.*button*)).perform(click())  
 onView(withId(R.id.*button*)).check(matches(withText(R.string.*my\_text*)))  
 }  
  
 private fun writeTextCheck(text: String){  
 onView(withId(R.id.*editText*)).perform(clearText())  
 onView(withId(R.id.*editText*)).perform(typeText(text))  
 }  
  
 @Test  
 fun simpleTest() {  
 activityRule.*activity*.*requestedOrientation* = ActivityInfo.*SCREEN\_ORIENTATION\_PORTRAIT* onView(withId(R.id.*button*)).check(matches(withText(R.string.*my\_button*)))  
 changeEditTextCheck()  
 }  
  
 @Test  
 fun changeOfScreenTest(){  
 activityRule.*activity*.*requestedOrientation* = ActivityInfo.*SCREEN\_ORIENTATION\_PORTRAIT* val newText= "new\_text"  
 writeTextCheck(newText)  
 changeEditTextCheck()  
 activityRule.*activity*.*requestedOrientation* = ActivityInfo.*SCREEN\_ORIENTATION\_LANDSCAPE* onView(withId(R.id.*editText*)).check(matches(withText(newText)))  
 onView(withId(R.id.*button*)).check(matches(withText(R.string.*my\_button*)))  
 }  
}

Хочется добавить, что сохранение текста в поле EditText не случайно. Android стандартно восстанавливает состояние некоторых Views (EditView, TextView и т. д.), так как они изначально используют методы onSaveInstanceState и onRestoreInstanceState в своей реализации, поэтому не нужно вручную обрабатывать их состояние при работе с жизненным циклом Activity. При этом состояние текстового поля сохраняется только в том случае, если в layout представлен ID. Без ID никакой View не может сохранять/восстанавливать свое состояние (onStop/onStart).

**3.2. Тестирование навигации**

Для тестирования был выбран третий пункт третьей лабораторной работы и написаны UI тесты, проверяющие навигацию между 4-мя исходными Activity. Всего было написано 3 теста. Для каждого из Activity рассматривалось: 1)открытие About из BottomNavigation; 2)возврат к рассматриваемому Activity; 3)переход по кнопкам; 4)возвращение в стартовую Activity;5)проверка выхода из приложения (должно вызваться NoActivityResumedException).

Выполнение всех тестов занимает (в среднем) 14-16 секунд (Рис.3.2.1):

****

Рис.3.2.1.

Код тестов представлен в Листинге 3.2.1.

Листинг 3.2.1.

package com.example.a52  
  
import androidx.test.espresso.Espresso.onView  
import androidx.test.espresso.action.ViewActions  
import androidx.test.espresso.action.ViewActions.click  
import androidx.test.espresso.intent.rule.IntentsTestRule  
import androidx.test.espresso.matcher.ViewMatchers  
import androidx.test.espresso.matcher.ViewMatchers.withId  
import androidx.test.ext.junit.runners.AndroidJUnit4  
import org.junit.Test  
import org.junit.runner.RunWith  
  
import org.junit.Rule  
import androidx.test.espresso.assertion.ViewAssertions.matches  
import androidx.test.espresso.matcher.ViewMatchers.isCompletelyDisplayed  
  
@RunWith(AndroidJUnit4::class)  
class NavigationTest {  
  
 @get:Rule  
 val firstActivityRule = IntentsTestRule(First::class.*java*)  
  
 @Test  
 fun testFirstActivity() {  
 openAboutFromFirst()  
 openSecondFromFirst()  
 returnToFirstFromSecond()  
 *assertFailsWith*(NoActivityResumedException::class) **{** onView(ViewMatchers.isRoot()).perform(ViewActions.pressBack())  
 **}** }  
  
 @Test  
 fun testSecondActivity() {  
 openSecondFromFirst()  
  
 openAboutFromSecond()  
  
 openFirstFromSecond()  
 openSecondFromFirst()  
 openThirdFromSecond()  
 returnToSecondFromThird()  
 returnToFirstFromSecond()  
 *assertFailsWith*(NoActivityResumedException::class) **{** onView(ViewMatchers.isRoot()).perform(ViewActions.pressBack())  
 **}** }  
  
 @Test  
 fun testThirdActivity() {  
 openSecondFromFirst()  
 openThirdFromSecond()  
  
 openAboutFromThird()  
  
 openFirstFromThird()  
 openSecondFromFirst()  
 openThirdFromSecond()  
 openSecondFromThird()  
  
 onView(ViewMatchers.isRoot()).perform(ViewActions.pressBack())  
 onView(withId(R.id.*third*)).check(matches(isCompletelyDisplayed()))  
  
 onView(ViewMatchers.isRoot()).perform(ViewActions.pressBack())  
 *assertFailsWith*(NoActivityResumedException::class) **{** onView(ViewMatchers.isRoot()).perform(ViewActions.pressBack())  
 **}** }  
  
  
 private fun openSecondFromFirst() {  
 onView(withId(R.id.*secondFromFirst*)).perform(click())  
 onView(withId(R.id.*second*)).check(matches(isCompletelyDisplayed()))  
 }  
  
 private fun openFirstFromSecond() {  
 onView(withId(R.id.*firstFromSecond*)).perform(click())  
 onView(withId(R.id.*first*)).check(matches(isCompletelyDisplayed()))  
 }  
  
 private fun openThirdFromSecond() {  
 onView(withId(R.id.*thirdFromSecond*)).perform(click())  
 onView(withId(R.id.*third*)).check(matches(isCompletelyDisplayed()))  
 }  
  
 private fun openFirstFromThird() {  
 onView(withId(R.id.*firstFromThird*)).perform(click())  
 onView(withId(R.id.*first*)).check(matches(isCompletelyDisplayed()))  
 }  
  
 private fun openSecondFromThird() {  
 onView(withId(R.id.*secondFromThird*)).perform(click())  
 onView(withId(R.id.*second*)).check(matches(isCompletelyDisplayed()))  
 }  
  
 private fun openAboutFromFirst() {  
 onView(withId(R.id.*nav\_first*)).perform(click())  
 onView(ViewMatchers.isRoot()).perform(ViewActions.pressBack())  
 onView(withId(R.id.*first*)).check(matches(isCompletelyDisplayed()))  
 }  
  
 private fun openAboutFromSecond() {  
 onView(withId(R.id.*nav\_second*)).perform(click())  
 onView(ViewMatchers.isRoot()).perform(ViewActions.pressBack())  
 onView(withId(R.id.*second*)).check(matches(isCompletelyDisplayed()))  
 }  
  
 private fun openAboutFromThird() {  
 onView(withId(R.id.*nav\_third*)).perform(click())  
 onView(ViewMatchers.isRoot()).perform(ViewActions.pressBack())  
 onView(withId(R.id.*third*)).check(matches(isCompletelyDisplayed()))  
 }  
  
 private fun returnToFirstFromSecond() {  
 onView(ViewMatchers.isRoot()).perform(ViewActions.pressBack())  
 onView(withId(R.id.*first*)).check(matches(isCompletelyDisplayed()))  
 }  
  
 private fun returnToSecondFromThird() {  
 onView(ViewMatchers.isRoot()).perform(ViewActions.pressBack())  
 onView(withId(R.id.*second*)).check(matches(isCompletelyDisplayed()))  
 }  
}

**4. Выводы**

В ходе выполнения данной лабораторной работы получены навыки разработки UI тестов для Android приложений с использованием Espresso Framework.

Одним из важных моментов является использовании аннотации @Rule (которая применяет метод getActivity()). Она позволяет запускать выбранный Activity до выполнения тестов (и @Before), что несколько упрощает работу с ними. Данная аннотации сообщают Junit, какие правила применять для написанных тестовых методов.

Все написанные тесты успешно проходят, что подтверждает корректность работоспособности приложений, написанных для данной и для третьей лабораторных работ.