Criando testes automatizados "like a pro"

Diego Gonçalves Santos
Desenvolvedor @ dextra

Obstáculos contra os testes automatizados

Obstáculos

- Cliente não vê valor
- Não há tempo disponível
- Desenvolvedores não entendem propriamente ou não vêem valor nos testes

Por que devo testar minhas aplicações?

Por que testar ?

- Assegura funcionamento mínimo do software
- Aumenta confiança para fazer refactorings
- Garante poder de errar rápido e corrigir rápido
- Ajudam a identificar “furos” na implementação
- Melhoram a sua forma de programar
- Asseguram que cenários de bugs não regressem
- Servem de documentação

Testes efetivos

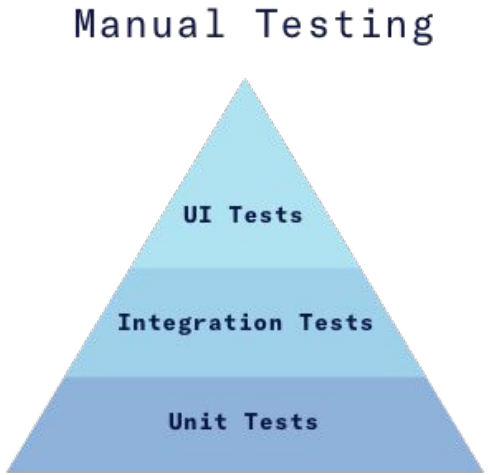
Testes Efetivos

Validam no mínimo o “fluxo feliz” da funcionalidade

Tipos de testes

Tipos de testes

- Testes unitários
- Testes de tela
- Testes manuais



Ambiente controlado para testes

Mocks

Mocks

- **Mockito**
- Injeção de dependências
- MockWebServer

Mockito

```
@Mock
lateinit var mockRepository: Repository

@Before
fun setUpTest() {
    MockitoAnnotations.initMocks(this)
}

@Test
fun test() {
    Mockito.`when`(mockRepository.getList(anyInt(),anyString())).thenReturn(expectedValue)
    presenter.loadList()
    verify(mockView, times(1)).onListLoaded(any())
}
```

Mocks

- Mockito
- Injeção de dependências
- MockWebServer

Injetando Mocks - Koin

```
class Repository


class Presenter {
        lateinit var repository: Repository
}
```

Injetando Mocks - Koin

```
class Repository

class Presenter {
        lateinit var repository: Repository by inject()
}

val module : Module = module {
        single { Repository() }
}
```

Injetando Mocks - Koin

```
@Before
fun setUpTest() {
        loadKoinModules(listOf(module {
                single { mockRepository }
        }))
}
```

Mocks

- Mockito
- Injeção de dependências
- MockWebServer

MockWebServer - Enqueue

```
lateinit var server: MockWebServer
@Before
fun setUp() {
   server = MockWebServer()
   server.start()
}
@Test
fun test() {
   server.enqueue(MockResponse().setBody(getJson("json/token.json")))
   server.enqueue(MockResponse().setBody(getJson("json/eventos.json")))
}
```

MockWebServer - Dispatcher

```
val dispatcher = object : Dispatcher() {
   @Throws(InterruptedException::class)
   override fun dispatch(request: RecordedRequest): MockResponse {
      return when {
         request.path == "/v1/token" ->
MockResponse().setResponseCode(200).setBody(getJson("json/token.json"))
         request.path == "v1/events" ->
            MockResponse().setResponseCode(200).setBody(getJson("json/eventos.json"))
         else -> MockResponse().setResponseCode(404)
      }
   }
}
server.setDispatcher(dispatcher)
```

Testes de tela

Espresso + Robot pattern

Espresso

```
onView(withId(R.id.login_username)).perform(typeText(username))

onView(withId(R.id.login_password)).perform(typeText(password))

Espresso.closeSoftKeyboard();

onView(withId(R.id.login_button)).perform(click())
```

Robot Pattern

```
class ScreenRobot {
    fun enterTextIntoView(@IdRes viewId: Int, text: String): T {
        onView(withId(viewId)).perform(typeText(text))
        closeKeyboard()
        return this as T
    }
}
class LoginRobot {
    fun inputPassword(pass: String): LoginRobot {
        return enterTextIntoView(LOGIN_PASSWORD, pass)
    }
    companion object {
        private val LOGIN_PASSWORD = R.id.login_password
    }
}
```

Robot Pattern

```
@Test
fun testLoginSuccess() {
    val loginRobot = LoginRobot()
    loginRobot
            .submit()
            .assertUsernameError(R.string.required_field)
            .assertPasswordError(R.string.required_field)
            .inputUser("valid")
            .submit()
            .assertPasswordError(R.string.required_field)
            .inputPassword("valid")
            .submit()
            .assertHomeOpens()
}
```

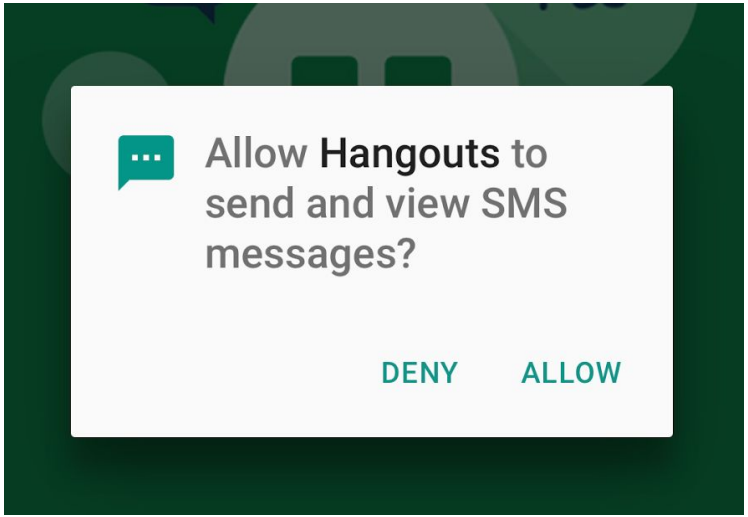
Testando run time permissions

Testando run time permissions

```
fun selectImage() {
    if (checkSelfPermission(Manifest.permission.WRITE_EXTERNAL_STORAGE) ==
    PackageManager.PERMISSION_GRANTED) {
        getImage()
    } else {
        askPermission()
    }
}
 private fun askPermission() {
    ActivityCompat.requestPermissions(activity,
    arrayOf<String>(Manifest.permission.WRITE_EXTERNAL_STORAGE), 1)
}
```

Testando run time permissions

```
override fun onRequestPermissionsResult(requestCode: Int, permissions: Array<out
String>, grantResults: IntArray) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults)
    if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
        // Treat request granted
    } else {
        // Treat request denied
    }
}
```



Testando run time permissions

```
class PermissionManager {
    fun isStoragePermissionGranted(context: Activity): Boolean {
        return if (Build.VERSION.SDK_INT >= 23) {

isPermissionGranted(context.checkSelfPermission(Manifest.permission.WRITE_EXTERNAL_STORAGE))
        } else {
            true
        }
    }
    fun isPermissionGranted(grantResult: Int): Boolean {
        return grantResult == PackageManager.PERMISSION_GRANTED
    }
    fun askForPermission(activity: Activity) {
        ActivityCompat.requestPermissions(activity,
arrayOf<String>(Manifest.permission.WRITE_EXTERNAL_STORAGE), 1)
    }
```

Testando run time permissions

```
val permissionManager: PermissionManager
fun selectImage() {
        if (permissionManager.isStoragePermissionGranted(this)) {
            getImage()
        } else {
            askPermission()
        }
}
private fun askPermission() {
        permissionManager.askForPermission(this)
}
```

Testando run time permissions

```
override fun onRequestPermissionsResult(requestCode: Int, permissions: Array<out
String>, grantResults: IntArray) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults)
        if (permissionManager.isPermissionGranted(grantResults[0])) {
            // Treat request granted
        } else {
            // Treat request denied
        }
}
```

Testando run time permissions

```
@RunWith(AndroidJUnit4::class)
class ProfileFragmentTest : BaseInstrumentedTest() {
   @Rule
   var mRuntimePermissionRule =
GrantPermissionRule.grant(Manifest.permission.WRITE_EXTERNAL_STORAGE)
   @Mock
    internal var permissionManagerMock: PermissionManager

   @Test
   fun test() {
       `when`(permissionManagerMock.isStoragePermissionGranted(any())).thenReturn(true)
   }
}
```

Roboeletric

Roboeletric

```
@RunWith(RobolectricTestRunner.class)
public class MyActivityTest {

  @Test
  public void clickingButton_shouldChangeResultsViewText() throws Exception {
    Activity activity = Robolectric.setupActivity(MyActivity.class);

    Button button = (Button) activity.findViewById(R.id.press_me_button);
    TextView results = (TextView) activity.findViewById(R.id.results_text_view);

    button.performClick();
    assertThat(results.getText().toString(), equalTo("Testing Android Rocks!"));
  }
}
```

Cobertura de Testes - Jacoco

Jacoco

app > com.moviepocket.features.moviesList.viewmodel Source Fi

com.moviepocket.features.moviesList.viewmodel

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
MoviesViewModel		42%		19%	35	43	52	97	7	15	0	1
MoviesViewModelFactory		0%		0%	5	5	5	5	4	4	1	1
MoviesViewModel.listMovies.disposable.new Consumer() {...}		50%		n/a	1	2	3	3	1	2	0	1
ScreenStatus		100%		n/a	0	3	0	5	0	3	0	1
MoviesViewModel.listMovies.disposable.new Consumer() {...}		100%		100%	0	3	0	5	0	2	0	1
MovieListScreenState		100%		n/a	0	2	0	2	0	2	0	1
Total	329 of 654	49%	47 of 60	21%	41	58	60	116	12	28	1	6

Jacoco

```
40.     fun listMovies(listType: String) {
41.         setCurrentCategory(listType)
42.
43.         if (isThereMoreItemsToLoad) {
44.
45.             resetState()
46.
47.             val disposable = movieRepository.getMovies(nextPage.toString(), listType)
48.                     .observeOn(androidScheduler)
49.                     .subscribeOn(processScheduler)
50.                     .subscribe ({
51.                         result ->
52.                         if (result.results.size > 0) {
53.                             onMoviesLoaded(listType, result.results, result.totalPages.toString())
54.                         } else {
55.                             onDataNotAvailable()
56.                         }
57.
58.                         movieRepository.saveMovies(result.results, listType, nextPage.toString())
59.                     }, { error ->
60.                         onRequestError()
61.                         error.printStackTrace()
62.                     })
63.
64.             compositeDisposable.add(disposable)
65.         }
66.     }
```

CI + Testes

CI + Testes


Pipeline #33912962 passed for f41b087e on fix/homolog_errors
Coverage 77.00%

Dicas

Dicas

- Comece pequeno
- Não desista dos testes

Referências

Confira nossa série sobre testes automatizados no Android:

- Por que devo testar minhas aplicações
- Testes automatizados no Android
- Ambiente controlado para testes – Parte 1: Mockito e Injeção de dependência
- Ambiente controlado para testes – Parte 2: MockWebserver
- Ambiente controlado para testes – Parte 3: Testando run time permissions no Espresso [Em Breve]
- UI tests – Espresso + Robot Pattern [Em Breve]
- Usos práticos da suíte de testes – Jacoco e CI [Em Breve]

Obrigado!

Diego Gonçalves Santos

DiegoGSantos

@DiegoGSantos

DiegoGSantos

DiegoGSantos