Testovi u Androidu

prof. dr Bratislav Predić
dipl. inž. Nevena Tufegdžić

Razvoj mobilnih aplikacija i servisa
Računarstvo i informatika

2024.

Testiranje aplikacija

- Manuelno testiranje podrazumeva pokretanje aplikacije i proveru da li željene funkcionalnosti rade kako je predviđeno
- Automatizovano testiranje podrazumeva pisanje testova koji se izvršavaju u radnom okruženju (Android Studio) i koji imaju za cilj da provere funkcionalnosti
- Dve kategorije testova: Subject i Scope

Subject testing

- Test funkcionalnosti – da li aplikacija radi ono što treba da radi
- Test performansi – koliko je aplikacija brza i efikasna
- Test pristupačnosti (accessibility) – da li je u skladu sa servisima za pristupačnost
- Test kompatibilnosti – da li aplikacija radi na različitim uređajima i API-jima

Scope testing

- Unit test (mali test) – testira rad malog dela aplikacije (npr. funkcija ili klasa)
- End-to-end test (veliki test) – testira ceo jedan use case koji obuhvata više delova aplikacije (npr. registracija korisnika)
- Integration test (medium test) – testira integraciju (međusobnu povezanost) dve ili više komponenti aplikacije

Testable aplikacija

- Testiranje je teško nad aplikacijama koje su tako organizovane da su im delovi gusto spregnuti
- Testable aplikacija ima:
  - Slojevitu strukturu: *Presentation*, *Domain*, *Data layer*
  - Logika aplikacije se ne nalazi u komponentama sa velikim zavisnostima (npr. Activity) – logika je u Domain sloju ili ViewModel-ima
  - Klase sa poslovnom logikom ne sadrže reference na kontekstne promenljive iz framework-a
  - Slabo spregnute komponente – korišćenje interfejsa umesto referenca na klasu, korišćenje Dependency Injection obrasca

Testiranje u Androidu

- androidTest – folder koji sadrži testove koji se izvršavaju na uređaju – integracioni testovi, end-to-end testovi i ostali testovi koje je nemoguće izvršiti u JVM
- test – folder koji sadrži testove koji se mogu izvršiti na lokalnoj mašini, u JVM okruženju
  - Unit testovi

Unit testovi

- Treba testirati ViewModele - njihove metode sa poslovnom logikom
- Testovi za sloj podataka – posebno za repozitorijume
- Testiranje komponenti u sloju poslovne logike (Domain Layer)
- Posebno je bitno testirati utility klase, kao što su npr. extension funkcije za postojeće tipove podataka, matematičke funkcije, itd.

UI testovi

- Svode se na testiranje ekrana (Screen UI tests) i testiranje korisničkog toka (User flow tests)
- Screen UI test – proverava korisničku interakciju na jednom ekranu aplikacije (klik na dugme, upis u tekstualna polja, forme, provera vidljivih komponenti)
- User flow test (test navigacije) – simulira korisnika koji se kreće kroz aplikaciju

Pisanje unit testova

- Unit testovi se nalaze u folderu module-name/src/test/
- Dependencies u build.gradle (Module: app):

```
dependencies {
    testImplementation "junit:junit:$jUnitVersion"
    testImplementation "androidx.test:core:$androidXTestVersion"
    testImplementation "org.mockito:mockito-core:$mockitoVersion"
    testImplementation "org.mockito.kotlin:mockito-kotlin:$mockitoKotlinVersion"
    testImplementation "io.mockk:mockk:$mockkVersion"
}
```

Pisanje unit testova

- Unit testovi su sadržani u klasama čije se definicije pišu u test folderu
- Ove klase sadrže metode koje proveravaju specifičnu funkcionalnost neke komponente
- Svaka od ovih metoda počinje anotacijom @Test
- U okviru testova, poželjno je koristiti assert funkcije iz Junit biblioteke

Pisanje unit testova

```
import org.junit.Assert.assertFalse
import org.junit.Assert.assertTrue
import org.junit.Test

class EmailValidatorTest {
  @Test fun emailValidator_CorrectEmailSimple_ReturnsTrue() {
    assertTrue(EmailValidator.isValidEmail("name@email.com"))
  }
}
```

Mock spoljašnjih zavisnosti

- Kod lokalnih unit testova, Gradle uključuje biblioteke koje modeluju odgovarajuće verzije API-ja Android Framework-a, koje sadrže deklaracije javnih metoda iz API-ja
- Ove metode nisu definisane za testiranje i poziv naneku od njih baciće izuzetak
- Kako bi se ovo izbeglo, koriste se mock biblioteke, kao što su Mockito ili MockK

Mock spoljašnjih zavisnosti

- Primer: pristupanje string resursima iz Context-a
- Lokalni testovi nemaju pristup kontekstu (niti bilo čemu iz Android Framework-a)
- Rešenje: kreira se mock (stub, imitacija) Context-a ili bilo čega što je testovima potrebno
- Ovi mock-ovi uvek vraćaju neke default vrednosti
- getString() – uvek će vratiti isti string

Mock spoljašnjih zavisnosti

- Primer: pristupanje string resursima iz Context-a
- Lokalni testovi nemaju pristup kontekstu (niti bilo čemu iz Android Framework-a)
- Rešenje: kreira se mock (stub, imitacija) Context-a ili bilo čega što je testovima potrebno
- Ovi mock-ovi uvek vraćaju neke default vrednosti
- getString() – uvek će vratiti isti string

Mockito

- Biblioteka koja pomaže JUnit-u da mockuje delove Android Framework-a
- Potrebno je uključiti je u build.gradle
- @RunWith(MockitoJUnitRunner::class) – anotacija koja govori da se test klasa pokreće sa mock podacima
- @Mock – anotacija koja govori da je neka promenljiva mock-ovana

Mockito

```
import android.content.Context
import org.junit.Assert.assertEquals
import org.junit.Test
import org.junit.runner.RunWith
import org.mockito.Mock
import org.mockito.junit.MockitoJUnitRunner
import org.mockito.kotlin.doReturn
import org.mockito.kotlin.mock

private const val FAKE_STRING = "HELLO WORLD"

@RunWith(MockitoJUnitRunner::class)
class MockedContextTest {
```

Mockito

```
@RunWith(MockitoJUnitRunner::class)
class MockedContextTest {

  @Mock
  private lateinit var mockContext: Context

  @Test
  fun readStringFromContext_LocalizedString() {
      ...
  }
}
```

Mockito

```
@Test
  fun readStringFromContext_LocalizedString() {
    val mockContext = mock<Context> {
        on { getString(R.string.name_label) } doReturn FAKE_STRING
    }

    val myObjectUnderTest = ClassUnderTest(mockContext)

    val result: String = myObjectUnderTest.getName()

    assertEquals(result, FAKE_STRING)
  }
```

Literatura

- Local Tests – Android Docs
- Instrumented Tests – Android Docs
- Mockito API
- Espresso
- Primeri – Unit Testing

Hvala na pažnji!