08

Para saber mais - Possível falha durante o teste com processos assíncronos

Durante o curso, todos os testes que executamos passaram sem nenhum problema, dado que tinha todo ambiente esperado.

Entretanto, em situações que temos integração com processos assíncronos, como é o caso da comunicação feita com a API, não é garantido que o dado sempre vai estar disponível para realizar o matcher, pois a requisição pode não finalizar rapidamente e o teste vai ser executado sem encontrar a View.

Dessa forma, pode ocasionar em falhas do teste com base na exception `NoMatchingViewException` (https://developer.android.com/reference/android/support/test/espresso/NoMatchingViewException).

Abordagem de delay com thead sleep

Para esses casos, uma abordagem inicial é considerar o uso de *thread sleep* após a Activity ser inicializada, para depois realizar o matcher. Considerando o nosso exemplo, teríamos o seguinte código:

```
@Test
public void deve_AparecerUmLeilao_QuandoCarregarUmLeilaoNaApi() {
public void deve_AparecerUmLeilao_QuandoCarregarUmLeilaoNaApi() throws IOException {

    Leilao carroSalvo = new LeilaoWebClient().salva(new Leilao("Carro"));
    if(carroSalvo == null){
        Assert.fail("Leilão não foi salvo");
    }
    activity.launchActivity(new Intent());

    Thread.sleep(1000);

    onView(withText("Carro"))
            .check(matches(isDisplayed()));
}
```

Com esse código assumimos que depois de 1 segundo após a Activity ser inicializada, o dado que é adquirido via API está disponível.

É importante ressaltar que cada processo assíncrono é finalizado em um tempo indeterminado, ou seja, nem sempre um ou dois segundos será o suficiente, fazendo com que o tempo de execução de teste seja comprometido, ou então, o teste falhe por colocar um *delay* menor do que o suficiente.

Abordagem com Idling Resources

Uma outra alternativa é utilizar o **Idling Resources** (https://developer.android.com/training/testing/espresso/idling-resource), um módulo do Espresso que tem o objetivo de sincronizar os testes que mantém integração e que interagem com processos assíncronos.

Essa é uma técnica viabiliza a execução de testes para esses casos, porém, ela exige que o código de produção (**src/main/java**) seja modificado! Portanto, é uma abordagem delicada que pode resultar em código não esperado.

Resumindo, temos um *trade-off* para esses casos que envolve processos assíncrono (algo muito comum em testes que mantém integração) e atualmente, no momento que esse curso foi gravado, não existe uma técnica ideal para resolver esse problema.