Introdução A Testes de Unidade em Aplicativos Android

Fernando Avanzo

fernando.avanzo@gmail.com

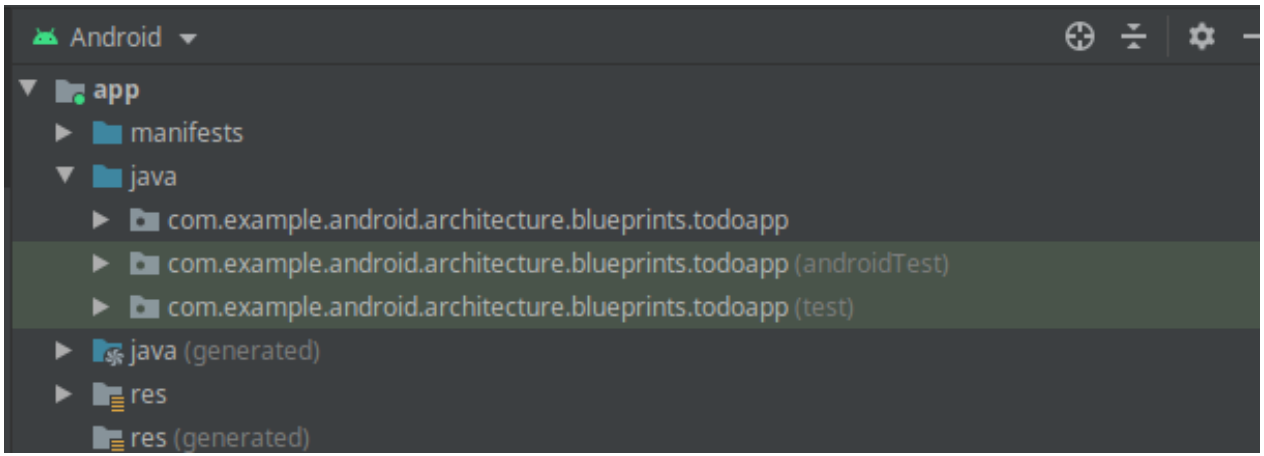
21 de junho de 2020

Overview

1. Intro
   - Tdd com Android

Como funciona a estrutura de teste no android

Os projetos android criados a partir do Android Studio ja vem com uma estrutura minima para a criação de testes automatizados



O arquivos são organizados da seguinte maneira

- diretório **main** onde fica todo código "funcional"do projeto
- diretório **test** onde fica todo código de teste que é executado localmente
- diretório **androidTest** onde fica todo código de teste é executado no dispositivo

Caixa de ferramentas

Ao se escrever testes de unidade automatizados existe uma serie de ferramentas uteis que auxiliam o workflow de trabalho. Abaixo vamos ver algumas:

- **Junit** framework xUnit para escrita de testes de unidade
- **Hamcrest** biblioteca que permite a ecrita de testes de forma mais idiomatica, torna sua leitura mais facil de pessoas entenderem.
- **AndroidX Test Library** bibliotecas android que auxiliam na simulação de componentes do ciclo de vida da aplicação, (Context, Activitys, Fragments, etc)

Configuração minima

Abaixo podmeos ver qual o aspecto geral de um arquivi *build.gradle* de um projeto orientado a testes.

Example (*build.gradle* default config)

```
defaultConfig {
applicationId "com.example.android..."
minSdkVersion rootProject.minSdkVersion
targetSdkVersion rootProject.targetSdkVersion
versionCode 1
versionName "1.0"
testInstrumentationRunner
 "androidx.test.runner.AndroidJUnitRunner"
}
```

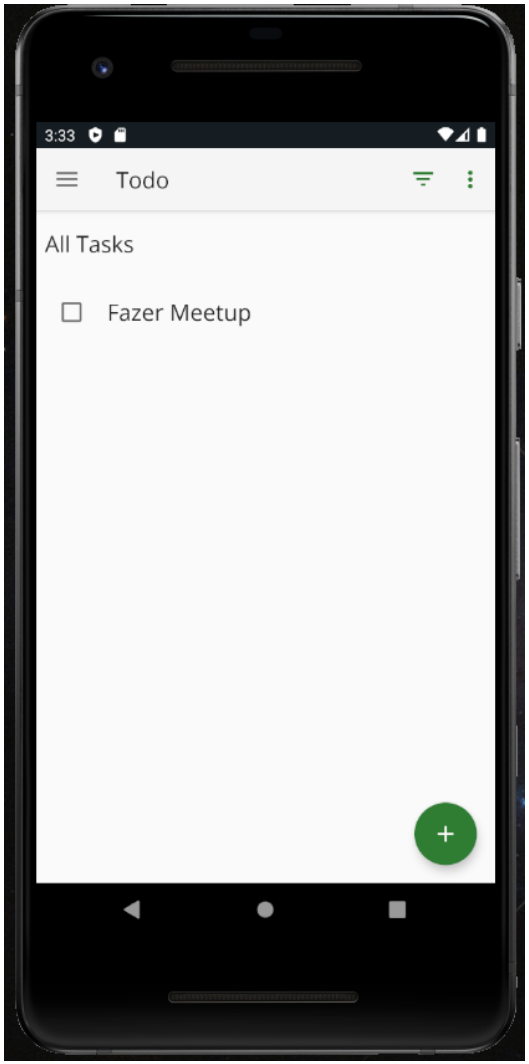
Configuração minima

Example (*build.gradle* dependencias)

```
// Dependencies for local unit tests
testImplementation "junit:junit:$junitVersion"
// AndroidX Test - Instrumented testing
androidTestImplementation
"androidx.test.ext:junit:$androidXTestExtKotlinRunnerVersion"
androidTestImplementation
"androidx.test.espresso:espresso-core:$espressoVersion"
// AndroidX Test - JVM testing
testImplementation
"androidx.test.ext:junit-ktx:$androidXTestKotlinRunnerVersion"
testImplementation
"androidx.test:core-ktx:$androidXTestCoreVersion"
testImplementation
"org.robolectric:robolectric:$robolectricVersion"
// Other dependencies
testImplementation
"org.hamcrest:hamcrest-all:$hamcrestVersion"
```

Caso de Uso: ToDo App

Este é o aplicativo que vamos usar para estudar alguns casos praticos de teste de unidade



Classico aplicativo de lista tarefas, que permite que usuario adicione uma nova tarefa na lista, check aquelas que já foram concluidas, e veja uma estatistica simples de quantas atividades ele ja conclui, e quantas ainda possui na lista.

Escrevendo nosso primeiro teste

O app **ToDo** tem uma funcinalidade que permite que o usuario tenha uma metrica de quantas tarefas ele terminou e quatas ainda faltam, levando isso em conta, nosso primeiro teste deve verificar que **Dado** uma lista de tarefas, **Quando** a lista tem apenas uma tarefa, e ela não esta completa, **Então** a porcentagem de tarefas ativas deve ser 100, e a porcentagem de tarefas completas deve ser 0

Escrevendo nosso primeiro teste

Example (Escrevendo o teste)

```
class StatisticsUtilsTest {

 @Test
 fun getActiveAndCompletedStats_noCompleted_returnsHundredZero()
 {
  val tasks = listOf(
  Task("title", "desc", isCompleted = false)
  )

  val result = getActiveAndCompletedStats(tasks)

  assertThat(result.activeTasksPercent, `is`(100f))
  assertThat(result.completedTasksPercent, `is`(0f))
 }
}
```

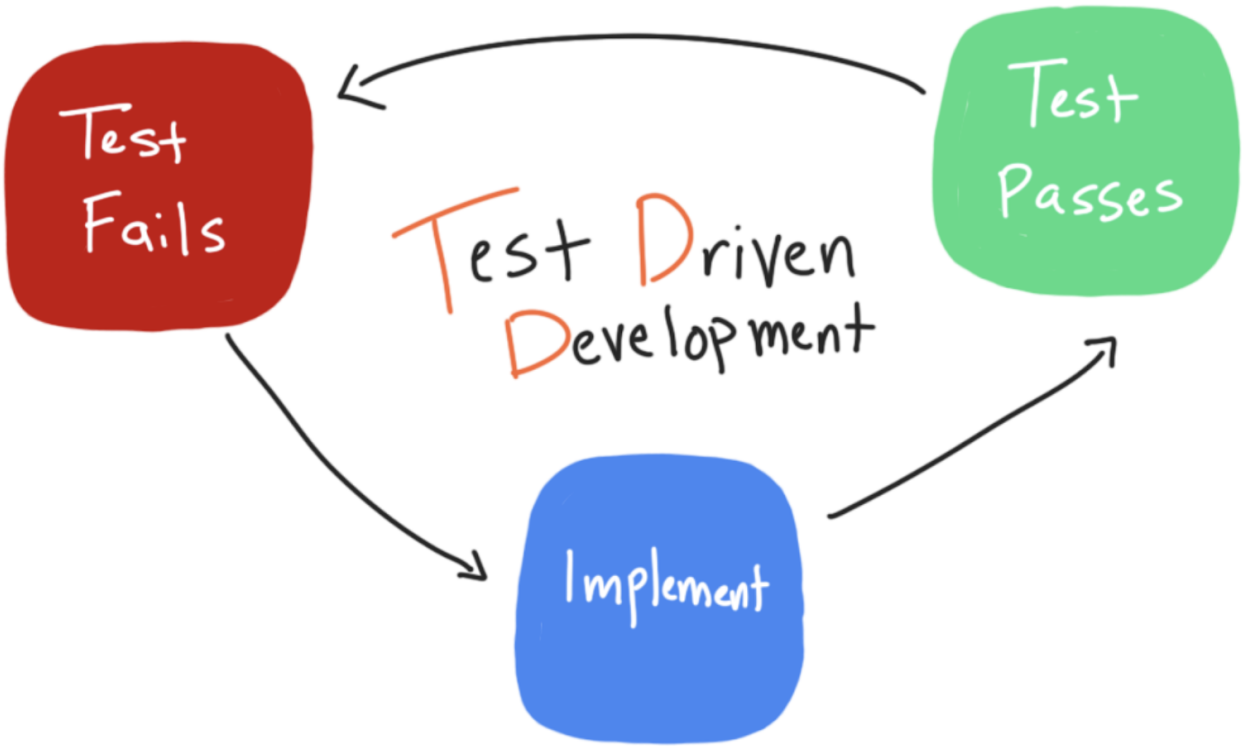
Escrevendo nosso primeiro teste

Example (Escrevendo o codigo funcional)

```
internal fun getActiveAndCompletedStats(tasks: List<Task>?)
: StatsResult {
 val totalTasks = tasks!!.size
 val numberOfActiveTasks = tasks.count { it.isActive }
 return StatsResult(
  activeTasksPercent = 100f *
   numberOfActiveTasks / tasks.size,
  completedTasksPercent = 100f *
   (totalTasks - numberOfActiveTasks) / tasks.size)
}
```

O que é TDD?

TDD,(Test Drive Development), é uma prática de desenvolvimento de software que orienta a antes de escrever qualquer código funcional, escrever um coódigo de teste, em seguida escrever o minimo de codigo funcional que faça o teste passar. Então trabalhar esse fluxo de forma incremental em todo o durante todo o processe de desenvolvimento.



O Jeito de Pensar TDD

Um teste de unidade pode ser escrito em três etapas:

Dado (Given)

Qual é o estado atual? Nesta etapa os dados necessarios para simular o estado atual da aplicação é fornecido.

Quando (When)

Quando o estado muda? Nesta etapa o trecho de código, (método, função, procedimento, etc..), que muda o estado é executado.

Então (Then)

A resposta,(saída), esta certa? Nesta etapa é verificado se após o estado inicial ter sido alterado o resultado corresponde ao esperado.

Caso de Uso: ToDo App

Vamos ver um exemplo pratico da abordagem GWT, (Given, When, Then) com o aplicativo ToDo

Cenario de uso: O usuario tem uma lista de tarefas

- **Dado** que usuario possui uma lista de tarefas
- **Quando** a lista de tarefas conter apenas uma tarefa, e ela estiver completa.
- **Então** a porcentagem de tarefas ativas deve ser 0, e porcentagem de tarefas concluidas deve ser 100

Caso de Uso: ToDo App

Example (Escrevendo o teste)

```
@Test
fun getActiveAndCompletedStats_noActive_returnsZeroHundred() {
 //Given
 val tasks = listOf(Task("title", "desc", isCompleted = true))

 // When
 val result = getActiveAndCompletedStats(tasks)

 // Then
 assertThat(result.activeTasksPercent, `is`(0f))
 assertThat(result.completedTasksPercent, `is`(100f))
}
```

Caso de Uso: ToDo App

Example (Escrevendo o codigo funcional)

```
internal fun getActiveAndCompletedStats(tasks: List<Task>?)
: StatsResult {

 val totalTasks = tasks!!.size
 val numberOfActiveTasks = tasks.count { it.isActive }
 val activePercent = 100 * numberOfActiveTasks / totalTasks
 val completePercent = 100 *
  (totalTasks - numberOfActiveTasks) / totalTasks

 return StatsResult(
  activeTasksPercent = activePercent.toFloat(),
  completedTasksPercent = completePercent.toFloat())
}
```

Caso de Uso: ToDo App

Cenario de Bug: Temos uma lista nula ou vazia

- **Dado** que uma lista nula ou vazia é enviada
- **Quando** a lista de tarefas for processada.
- **Então** a porcentagem de tarefas ativas e concluidas deve ser 0

Caso de Uso: ToDo App

Example (Escrevendo o teste)

```
@Test
fun getActiveAndCompletedStats_error_returnsZeros() {
 //Given
 val tasks = null

 // When
 val result = getActiveAndCompletedStats(null)

 //Then
 assertThat(result.activeTasksPercent, `is`(0f))
 assertThat(result.completedTasksPercent, `is`(0f))
}
```

Caso de Uso: ToDo App

Example (Escrevendo o codigo funcional)

```
internal fun getActiveAndCompletedStats(tasks: List<Task>?)
: StatsResult {
 return if (tasks == null || tasks.isEmpty()) {
  StatsResult(0f, 0f)
 } else {
  val totalTasks = tasks.size
  val numberOfActiveTasks = tasks.count { it.isActive }
  StatsResult(
   activeTasksPercent = 100f *
     numberOfActiveTasks / tasks.size,
   completedTasksPercent = 100f *
    (totalTasks - numberOfActiveTasks) / tasks.size)
 }
}
```

References

John Smith (2012)

Title of the publication

Journal Name 12(3), 45 – 678.

The End