

COMO ESCREVER BONS TESTES UNITÁRIOS



THE
DEVELOPER'S
CONFERENCE

@orogersilva



DESENVOLVEDOR DE SOFTWARE – MOBILE



@orogersilva

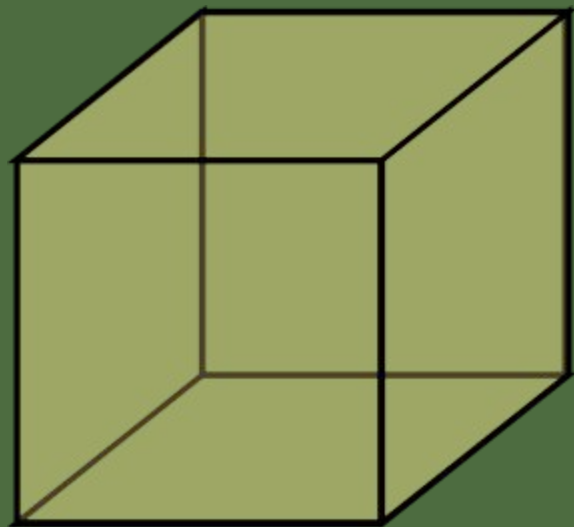


@orogersilva

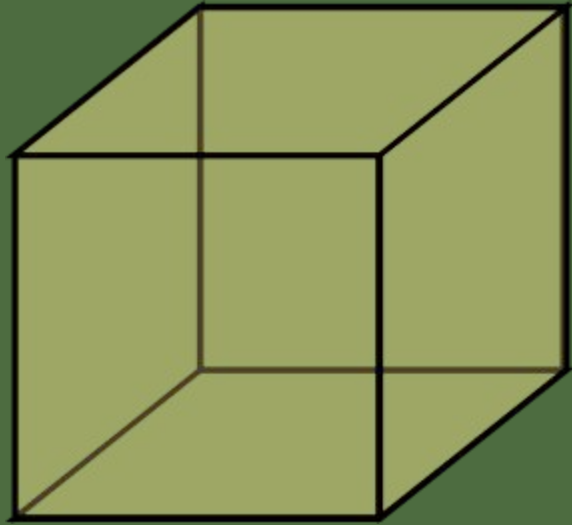


@orogersilva

○ QUE SÃO TESTES UNITÁRIOS?

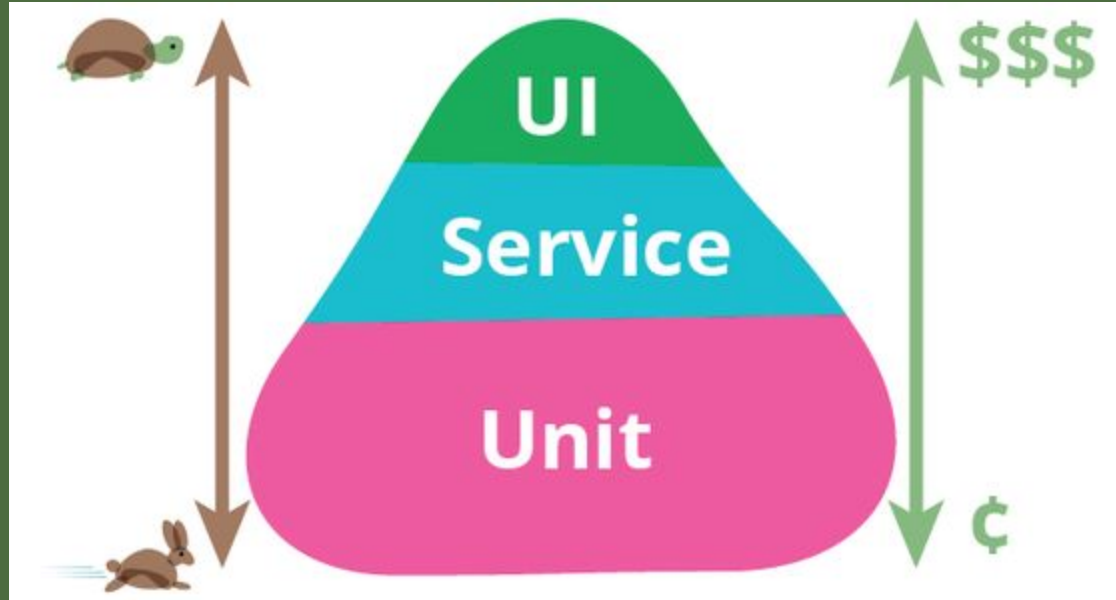


CLASSE



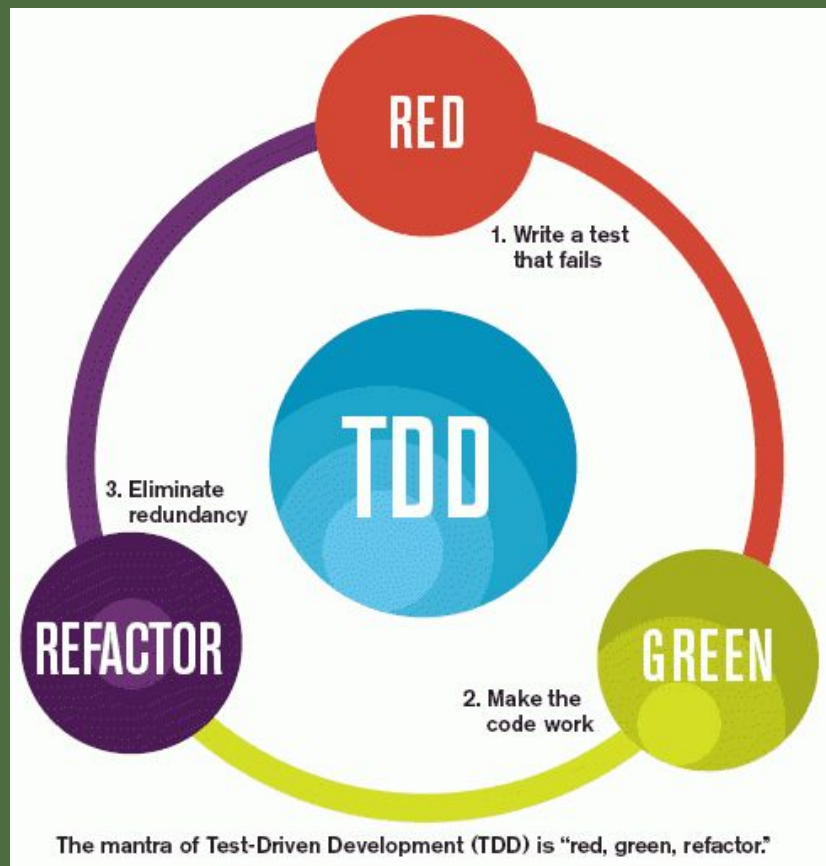
MÉTODO

ou



Android Testing Pyramid





“COMO DEVO TESTAR?”

- SETUP
- EXERCISE
- VERIFY
- TEARDOWN

CLASSE DE TESTE



```
class CustomerDataRepositoryTest {  
    (...)  
}
```

MÉTODO DE TESTE



```
class CustomerDataRepositoryTest {  
  
    @Test fun 'Get stored user name, when it is requested stored user name, then local layer is called  
once'() {  
  
        // ARRANGE  
  
        val EXPECTED_USER_NAME = "Mr. John"  
  
        whenever(customerLocalDataSourceMock.getStoredUserName()).thenReturn(EXPECTED_USER_NAME)  
  
        // ACT  
  
        val userName = customerRepository.getStoredUserName()  
  
        // ASSERT  
  
        assertEquals(EXPECTED_USER_NAME, userName)  
    }  
}
```



```
class ClasseDeTeste {  
    @Test fun '<Nome do método testado>, when <Condição>, then <Resultado esperado>'() {  
        (...)  
    }  
}
```



```
class ClasseDeTeste {  
    @Test fun '<Nome do método testado>, when <Condição>, then <Resultado esperado>'() {  
        // ARRANGE  
  
        // ACT  
  
        // ASSERT  
    }  
}
```

```
@Test
fun `Get stored user name, when it is requested stored user name, then local layer is called once`() {

    // ARRANGE

    val EXPECTED_USER_NAME = "Mr. John"

    whenever(customerLocalDataSourceMock.getStoredUserName()).thenReturn(EXPECTED_USER_NAME)

    // ACT

    val userName = customerRepository.getStoredUserName()

    // ASSERT

    assertEquals(EXPECTED_USER_NAME, userName)
}
```

Run: ▶ CustomerDataRepositoryTest.Get stored user na... ✕

▶ ✓ ⊗ ⌵ ⌵ ≡ ≡ ↑ ↓ ↗ 🔍 ⚙️

▼ ✓ CustomerDataRepositoryTest 789 ms

✓ Get stored user name, when it is requested stored user name, then local layer is called once 789 ms



```
./gradlew testDebugUnitTest
```



```
./gradlew :<nome módulo>:testDebugUnitTest
```



```
android {  
    (...)  
    testOptions {  
        unitTests.all {  
            testLogging {  
                events 'passed', 'skipped', 'failed'  
            }  
        }  
    }  
    (...)  
}
```

```
CustomerDataRepositoryTest > Get stored user name, when it is requested stored user name, then local layer is called once PASSED
```

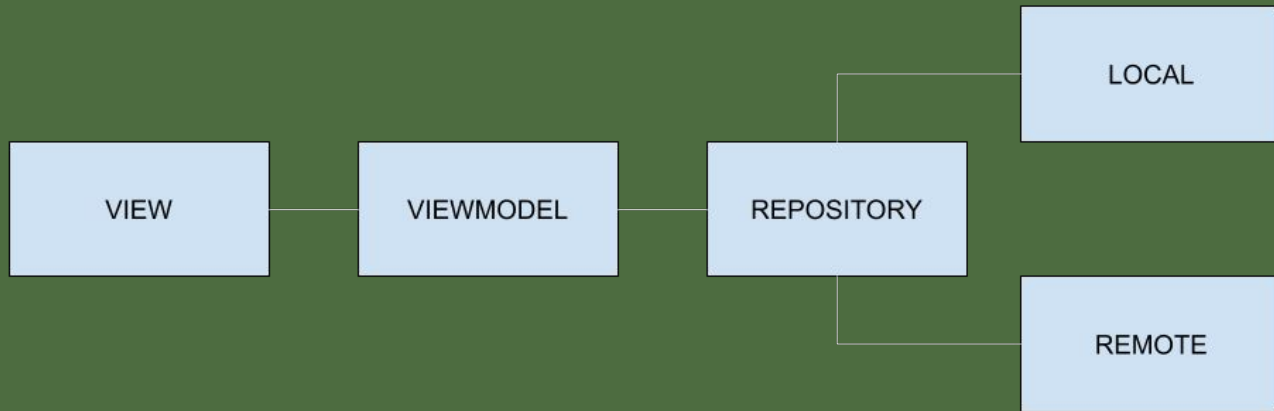


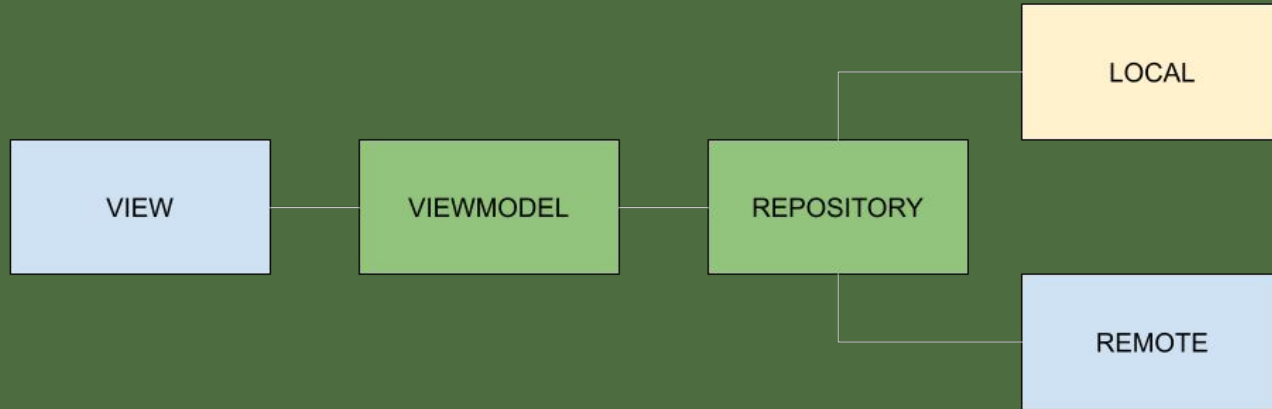
INJEÇÃO DE DEPENDÊNCIA



```
class CustomerDataRepository(private val customerLocalDataSource: CustomerDataContract.Local,  
                             private val customerRemoteDataSource: CustomerDataContract.Remote) :  
CustomerRepository {  
    override fun getStoredUserName(): String = customerLocalDataSource.getStoredUserName()  
}
```

UMA BOA ARQUITETURA





MOCKS

Mocks Aren't Stubs

The term 'Mock Objects' has become a popular one to describe special case objects that mimic real objects for testing. Most language environments now have frameworks that make it easy to create mock objects. What's often not realized, however, is that mock objects are but one form of special case test object, one that enables a different style of testing. In this article I'll explain how mock objects work, how they encourage testing based on behavior verification, and how the community around them uses them to develop a different style of testing.

02 January 2007



Martin Fowler

POPULAR

TESTING

TRANSLATIONS: French · Italian ·
Spanish · Portuguese · Korean

CONTENTS

Regular Tests

Tests with Mock Objects

Using EasyMock

The Difference Between Mocks and Stubs

Classical and Mockist Testing

Choosing Between the Differences

Driving TDD

Fixture Setup

Test Isolation

Coupling Tests to Implementations

Design Style

So should I be a classicist or a mockist?

Final Thoughts

nhaarman / mockito-kotlin

Watch

47

★ Unstar

2k

Fork

132

Code

Issues 30

Pull requests 2

Actions

Projects 0

Wiki

Security

Insights

Using Mockito with Kotlin

kotlin

mockito

mockito-kotlin

360 commits

4 branches

0 packages

48 releases

24 contributors

MIT

Branch: 2.x

New pull request

Create new file

Upload files

Find file

Clone or download



nhaarman Merge pull request #353 from nhaarman/lenient

✓ Latest commit dbdef2f on Sep 8

.github	Update PULL_REQUEST_TEMPLATE.md	3 years ago
.idea	Update Gradle dependency in README	last year
gradle	Update Gradle wrapper to 4.10.2	last year
mockito-kotlin	Support lenient in mock and withSettings	3 months ago
ops	Move tests to separate module	2 years ago
tests	Support lenient in mock and withSettings	3 months ago
.gitignore	Update .gitignore	11 months ago
.travis.yml	Update Kotlin version to 1.3.50	3 months ago
LICENSE	Initial commit	4 years ago
README.md	Update README to include build instructions	3 months ago
RELEASING.md	Fix releasing to Maven Central	2 years ago

verification	chain-calls	matcher	argument-matchers	kotlin	mocking-framework	mock
--------------	-------------	---------	-------------------	--------	-------------------	------

🔄 1,301 commits 🌿 8 branches 📦 0 packages 🏷️ 65 releases 👤 33 contributors 🏠 Apache-2.0

Branch: master New pull request

Create new file Upload files Find file Clone or download

 Raibaz and **oleksiyp** Added collections to the AnyValueGenerator (#387) ✓ Latest commit d5afbfe 6 days ago

 .github	Create FUNDING.yml	last month
 _includes	Open graph tags	2 years ago
 _layouts	Update default.html	24 days ago
 agent	Renamed variable with a typo in the name (#385)	8 days ago
 assets/css	Template change	2 years ago
 cloud-badge	Weekly users	last year
 doc	Add files via upload	last month
 dsl	Fix 'withNullableArg' function to make it work properly with null val...	3 months ago
 gradle	Added jacoco version specification (#370)	last month
 matrix	Matrix successfully completed for JDK11 (not able to run it for AIT) #...	last year
 mockk	Added collections to the AnyValueGenerator (#387)	6 days ago
 wiki	Wiki image	9 months ago

TESTANDO REPOSITORY



```
class CustomerDataRepository(private val customerLocalDataSource: CustomerDataContract.Local,  
                             private val customerRemoteDataSource: CustomerDataContract.Remote) :  
CustomerRepository {  
    override fun getStoredUserName(): String = customerLocalDataSource.getStoredUserName()  
}
```

```
class CustomerDataRepositoryTest {

    private lateinit var customerLocalDataSourceMock: CustomerDataContract.Local
    private lateinit var customerRemoteDataSourceMock: CustomerDataContract.Remote

    private lateinit var customerRepository: CustomerDataRepository

    @Before fun setUp() {

        customerLocalDataSource = mock()
        customerRemoteDataSource = mock()

        customerRepository = CustomerDataRepository(
            customerLocalDataSourceMock,
            customerRemoteDataSourceMock
        )
    }

    @Test fun 'Get stored user name, when it is requested stored user name, then local layer is called once'() {

        // ARRANGE

        val EXPECTED_USER_NAME = "Mr. John"

        whenever(customerLocalDataSourceMock.getStoredUserName()).thenReturn(EXPECTED_USER_NAME)

        // ACT

        val userName = customerRepository.getStoredUserName()

        // ASSERT

        assertEquals(EXPECTED_USER_NAME, userName)
    }
}
```

TESTANDO UTILS



```
class DateUtilsTest {  
  
    @Test fun 'Get month in full, when it is passed a valid month index, then returns month name'() {  
  
        // ARRANGE  
  
        val EXPECTED_MONTH_NAME = "april"  
        val MONTH_INDEX = 3  
  
        // ACT  
  
        val MONTH_NAME = DateUtil.getMonthInFull(MONTH_INDEX)  
  
        // ASSERT  
  
        assertEquals(EXPECTED_MONTH_NAME, monthName)  
    }  
}
```

TESTANDO VIEW/MODEL



```
class HomeViewModel : ViewModel() {  
    val userData: MutableLiveData<UserData> = MutableLiveData()  
    (...)  
    fun loadUserData() {  
        userData.setValue(UserData())  
    }  
    (...)  
}
```



```
class HomeViewModelTest {  
    @Test fun 'Load user data, bla bla bla, bla bla bla'() {  
        (...)  
    }  
}
```



```
java.lang.RuntimeException: Method getMainLooper in android.os.Looper not mocked
```



```
class HomeViewModelTest {  
    @JvmField @Rule val instantTaskExecutorRule = InstantTaskExecutorRule()  
    @Test fun 'Load user data, bla bla bla, bla bla bla'() {  
        (...)  
    }  
}
```

TESTANDO COROUTINE



```
override fun deleteRoom(roomId: Int) : Deferred<Response<Void>> {  
    val sessionToken = chatLocalDataSource.getSessionToken()  
    return chatRemoteDataSource.deleteRoom(sessionToken, roomId)  
}
```



```
@Test fun 'Close room, when it is requested, then remote layer is called once'() {  
    // ARRANGE  
  
    val SESSION_TOKEN = "7JZp0YplwEH03smSHhx+Epa1NTmNobzuYXPrnZEjsR8U="  
  
    val ROOM_ID = 231  
  
    val expectedResponse = Response.success<Void>(null)  
  
    val expectedDeferredResponse = CompletableDeferred<Response<Void>>(expectedResponse)  
  
    whenever(chatLocalDataSourceMock.getSessionToken()).thenReturn(SESSION_TOKEN)  
  
    whenever(chatRemoteDataSourceMock.deleteRoom(SESSION_TOKEN, ROOM_ID))  
        .thenReturn(expectedDeferredResponse)  
  
    // ACT  
  
    runBlocking { chatRepository.deleteRoom(SESSION_TOKEN, ROOM_ID).await() }  
  
    // ASSERT  
  
    verify(chatRemoteDataSourceMock, times(1)).deleteRoom(SESSION_TOKEN, ROOM_ID)  
}
```

TESTES UNITÁRIOS DEVEM
VALIDAR COMPORTAMENTO E
NÃO FLUXO DE EXECUÇÃO

COBERTURA DE CÓDIGO



APLICAÇÃO DE PLUGIN + ARQUIVO DE CONFIGURAÇÃO



```
apply plugin: 'jacoco'
```

<https://github.com/orogersilva/spotmusic-alarm-android/blob/master/gradle/jacoco.gradle>

```
def fileFilter = [
    '**/R.class',
    '**/R$.class',
    '**/BuildConfig.*',
    '**/Manifest.*',
    '**/Test.*',
    'android/**/*.*',

    // Dagger 2
    '**/*Dagger.*',
    '**/*Dagger*Component.*',
    '**/*Module.*',
    '**/*Module$.*',
    '**/*MembersInjector.*',
    '**/*_Factory.*',
    '**/*Provide*Factory.*',
    '**/*$ViewInjector.*',
    '**/*$*.*',

    // Classes I intentionally don't want to test...
    '**/*Activity.*',
    '**/*Binding.*',
    '**/BR.*',
    '**/adapter/**/*.*',
    '**/adapters/**/*.*',
    '**/dao/**/*.*',
    '**/databinding/**/*.*',
    '**/dto/**/*.*',
    '**/enums/**/*.*',
    '**/entity/**/*.*',
    '**/factory/**/*.*',
    '**/pagination/**/*.*',
    '**/relation/**/*.*',
    '**/shared/**/*.*',
    '**/typeconverter/**/*.*',
    '**/view/**/*.*',
    '**/*Activity.*',
    '**/*Database.*',
    '**/*DataBinderMapperImpl.*',
    '**/*_Factory.*',
    '**/*Fragment.*'
]
```



```
task jacocoFullTestReport(type: JacocoReport) {  
  
    group = 'Reporting'  
    description = "Generate Jacoco coverage reports for the debug build."  
  
    //Make sure that tests from all modules are run before coverage report  
    dependsOn ":dashboard-data:testDebugUnitTest"  
    dependsOn ":feature-dashboard:testDebugUnitTest"  
    dependsOn ":dashboard-data:createDebugCoverageReport"  
    dependsOn ":feature-dashboard:createDebugCoverageReport"  
  
    (...)  
}
```



```
task jacocoFullTestReport(type: JacocoReport) {  
    (...)  
    classDirectories = files(  
        fileTree(  
            dir: "$project.rootDir/dashboard-data/build/intermediates/classes/debug",  
            excludes: fileFilter  
        ) + fileTree(  
            dir: "$project.rootDir/dashboard-data/build/tmp/kotlin-classes/debug",  
            excludes: fileFilter  
        ),  
        fileTree(  
            dir: "$project.rootDir/feature-dashboard/build/intermediates/classes/debug",  
            excludes: fileFilter  
        ) + fileTree(  
            dir: "$project.rootDir/feature-dashboard/build/tmp/kotlin-classes/debug",  
            excludes: fileFilter  
        )  
    )  
    (...)  
}
```



```
task jacocoFullTestReport(type: JacocoReport) {  
    (...)  
    executionData = fileTree(dir: project.rootDir, includes: [  
        'dashboard-data/build/jacoco/testDebugUnitTest.exec',  
        'dashboard-data/build/outputs/code-coverage/connected/*coverage.ec',  
        'feature-dashboard/build/jacoco/testDebugUnitTest.exec',  
        'feature-dashboard/build/outputs/code-coverage/connected/*coverage.ec'  
    ])  
    (...)  
}
```



```
./gradlew jacocoFullTestReport
```

É NECESSÁRIO 100% DE
COBERTURA DE CÓDIGO?

TESTES UNITÁRIOS SÃO
SUFICIENTES?

TESTES UNITÁRIOS DEVEM SER
MUTÁVEIS

TESTES UNITÁRIOS DEVEM SER
CONSTANTEMENTE
EXECUTADOS

Analysis

Build

Test

Deploy



formatting-cod...



structural-code-...



build-develop



unit-test-develop



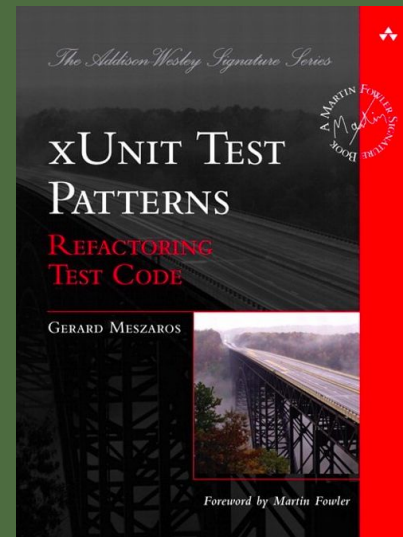
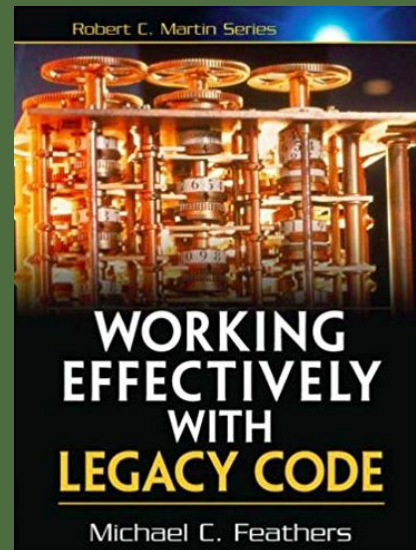
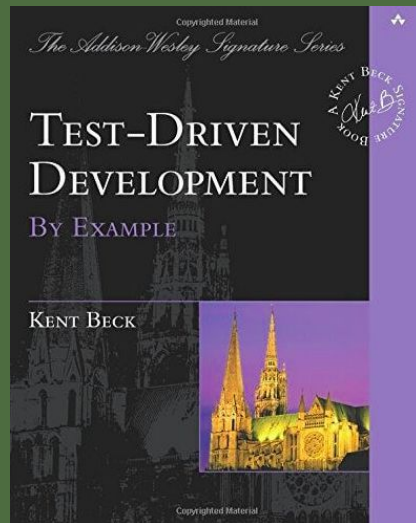
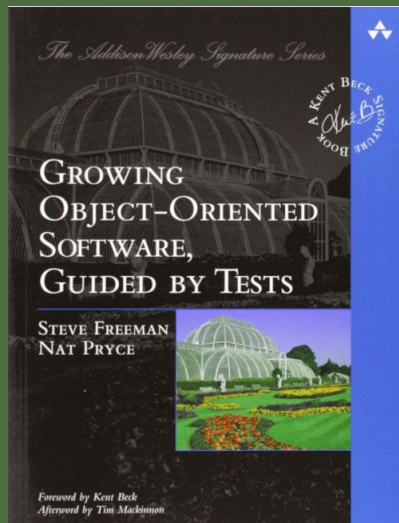
deploy-app-dev...



deploy-module-...



REFERÊNCIA SOBRE TESTES UNITÁRIOS





@orogersilva

OBRIGADO!

COMO ESCREVER BONS TESTES UNITÁRIOS



THE
DEVELOPER'S
CONFERENCE

@orogersilva