
PicPay

Alex Soares

Desenvolvedor Android

@a.soares.siqueira

/alex-soares-siqueira /AlexSoaresDeSiqueira

PicPay

Teste no android

Porque? Como? O que utilizar?

PicPay

Quantas pessoas aqui escrevem testes para suas aplicações?

PicPay

PicPay

Porque testar?

PicPay

Garantir qualidade do código

PicPay

Entrega de valor ao cliente

PicPay

Documentação

PicPay

Teste + android

PicPay

Testes unitários

Locais

Testes unitários locais

- JVM

PicPay

Testes unitários locais

- JVM
- Regras de negócio

PicPay

Testes unitários locais

- JVM
- Regras de negócio
- JUnit

PicPay

Testes unitários locais

- JVM
- Regras de negócio
- JUnit
- Rápido

PicPay

Testes unitários locais

- JVM
- Regras de negócio
- JUnit
- Rápido
- src/teste/

PicPay

Como fazer?

- Sufixo Test

PicPay

Como fazer?

- Sufixo Test
- @Test

PicPay

Como fazer?

- Sufixo Test
- @Test
- @Before

PicPay

Como fazer?

- Sufixo Test
- @Test
- @Before
- @After

PicPay

Como fazer?

- Sufixo Test
- @Test
- @Before
- @After
- [nome do metodo]_[retorno esperado]_[input esperado]

PicPay

Como fazer?

- Sufixo Test
- @Test
- @Before
- @After
- [nome do metodo]_[retorno esperado]_[input esperado]
- GIVEN WHEN THEN

PicPay

```kotlin
class SearchStarshipViewModelTest {
    .....
    @Before
    fun setup() {
        initMockWebServer()
    }

    @After
    fun after() {
        closeMockWebServer()
    }

    @Test
    fun `searchStarship should return success when pass a valid starship name`() {
        // given
        mockStarshipSuccessRequest()
        val starshipExpected = mockDeathStarship()
        val starshipActual = StarshipState()

        starshipViewModel.states.observeForever { state ->
            state.let {
                starshipActual = it
            }
        }

        // when
        starshipViewModel.searchStarship("death")

        // then
        assertEquals(starshipExpected, starshipActual)
    }
```

PicPay

```kotlin
class SearchStarshipViewModelTest {
    .....
    @Before
    fun setup() {
        initMockWebServer()
    }

    @After
    fun after() {
        closeMockWebServer()
    }

    @Test
    fun `searchStarship should return success when pass a valid starship name`() {
        // given
        mockStarshipSuccessRequest()
        val starshipExpected = mockDeathStarship()
        val starshipActual = StarshipState()

        starshipViewModel.states.observeForever { state ->
            state.let {
                starshipActual = it
            }
        }

        // when
        starshipViewModel.searchStarship("death")

        // then
        assertEquals(starshipExpected, starshipActual)
    }
```

PicPay

```kotlin
class SearchStarshipViewModelTest {
    .....
    @Before
    fun setup() {
        initMockWebServer()
    }

    @After
    fun after() {
        closeMockWebServer()
    }

    @Test
    fun `searchStarship should return success when pass a valid starship name`() {
        // given
        mockStarshipSuccessRequest()
        val starshipExpected = mockDeathStarship()
        val starshipActual = StarshipState()

        starshipViewModel.states.observeForever { state ->
            state.let {
                starshipActual = it
            }
        }

        // when
        starshipViewModel.searchStarship("death")

        // then
        assertEquals(starshipExpected, starshipActual)
    }
```

PicPay

```kotlin
class SearchStarshipViewModelTest {
    .....
    @Before
    fun setup() {
        initMockWebServer()
    }

    @After
    fun after() {
        closeMockWebServer()
    }

    @Test
    fun `searchStarship should return success when pass a valid starship name`() {
        // given
        mockStarshipSuccessRequest()
        val starshipExpected = mockDeathStarship()
        val starshipActual = StarshipState()

        starshipViewModel.states.observeForever { state ->
            state.let {
                starshipActual = it
            }
        }

        // when
        starshipViewModel.searchStarship("death")

        // then
        assertEquals(starshipExpected, starshipActual)
    }
```

PicPay

```kotlin
class SearchStarshipViewModelTest {
    .....
    @Before
    fun setup() {
        initMockWebServer()
    }

    @After
    fun after() {
        closeMockWebServer()
    }

    @Test
    fun `searchStarship should return success when pass a valid starship name`() {
        // given
        mockStarshipSuccessRequest()
        val starshipExpected = mockDeathStarship()
        val starshipActual = StarshipState()

        starshipViewModel.states.observeForever { state ->
            state.let {
                starshipActual = it
            }
        }

        // when
        starshipViewModel.searchStarship("death")

        // then
        assertEquals(starshipExpected, starshipActual)
    }
```

PicPay

```kotlin
class SearchStarshipViewModelTest {
    .....
    @Before
    fun setup() {
        initMockWebServer()
    }

    @After
    fun after() {
        closeMockWebServer()
    }

    @Test
    fun `searchStarship should return success when pass a valid starship name`() {
        // given
        mockStarshipSuccessRequest()
        val starshipExpected = mockDeathStarship()
        val starshipActual = StarshipState()

        starshipViewModel.states.observeForever { state ->
            state.let {
                starshipActual = it
            }
        }

        // when
        starshipViewModel.searchStarship("death")

        // then
        assertEquals(starshipExpected, starshipActual)
    }
```

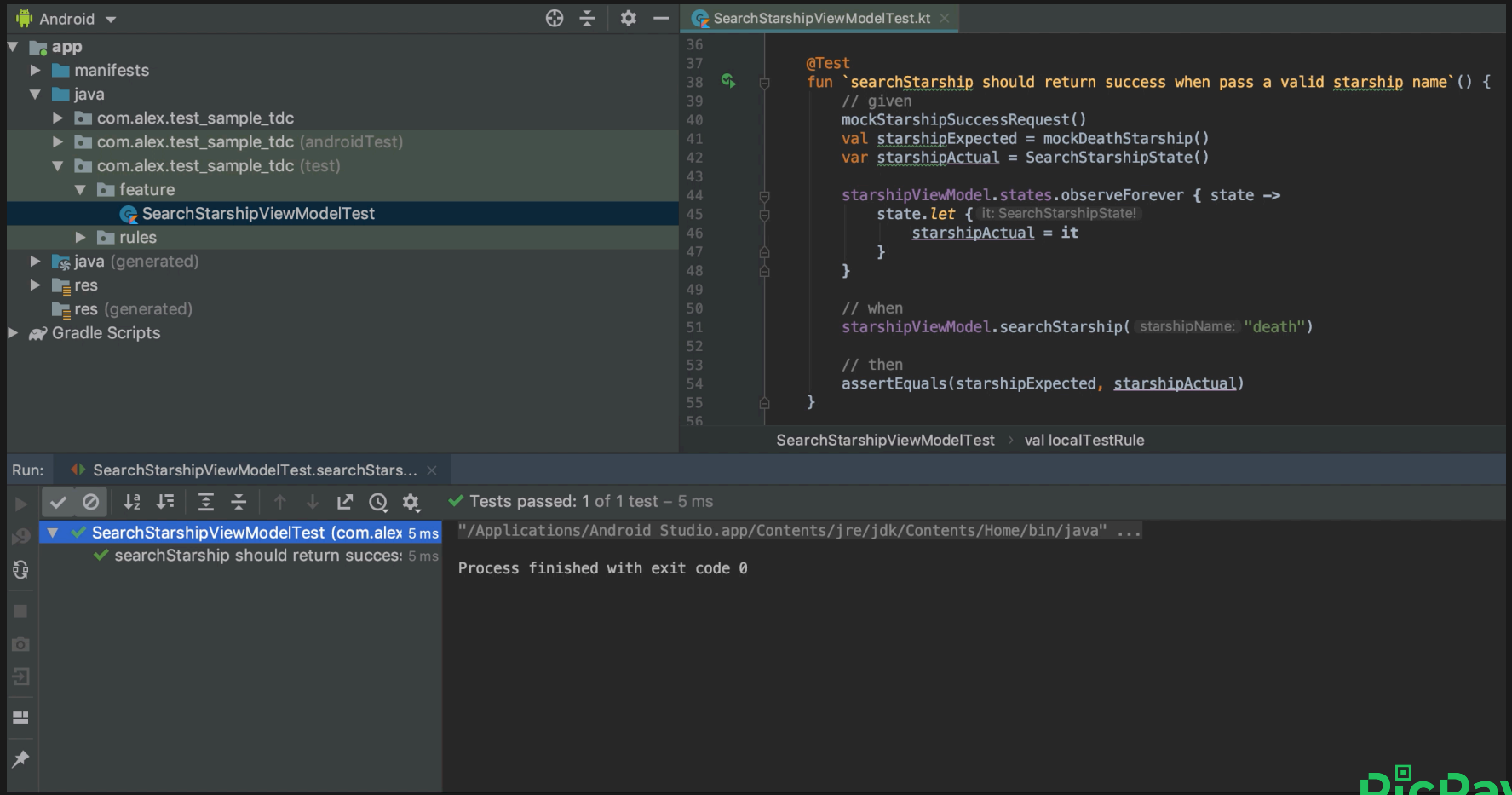
PicPay

```kotlin
class SearchStarshipViewModelTest {
    .....
    @Before
    fun setup() {
        initMockWebServer()
    }

    @After
    fun after() {
        closeMockWebServer()
    }

    @Test
    fun `searchStarship should return success when pass a valid starship name`() {
        // given
        mockStarshipSuccessRequest()
        val starshipExpected = mockDeathStarship()
        val starshipActual = StarshipState()

        starshipViewModel.states.observeForever { state ->
            state.let {
                starshipActual = it
            }
        }

        // when
        starshipViewModel.searchStarship("death")

        // then
        assertEquals(starshipExpected, starshipActual)
    }
```

PicPay

```kotlin
class SearchStarshipViewModelTest {
    .....
    @Before
    fun setup() {
        initMockWebServer()
    }

    @After
    fun after() {
        closeMockWebServer()
    }

    @Test
    fun `searchStarship should return success when pass a valid starship name`() {
        // given
        mockStarshipSuccessRequest()
        val starshipExpected = mockDeathStarship()
        val starshipActual = StarshipState()

        starshipViewModel.states.observeForever { state ->
            state.let {
                starshipActual = it
            }
        }

        // when
        starshipViewModel.searchStarship("death")

        // then
        assertEquals(starshipExpected, starshipActual)
    }
```

PicPay


Android
app
manifests
java
com.alex.test_sample_tdc
com.alex.test_sample_tdc (androidTest)
com.alex.test_sample_tdc (test)
feature
SearchStarshipViewModelTest
rules
java (generated)
res
res (generated)
Gradle Scripts
SearchStarshipViewModelTest.kt
@Test
fun `searchStarship should return success when pass a valid starship name`() {
// given
mockStarshipSuccessRequest()
val starshipExpected = mockDeathStarship()
var starshipActual = SearchStarshipState()
starshipViewModel.states.observeForever { state ->
state.let { it: SearchStarshipState!
starshipActual = it
}
}
// when
starshipViewModel.searchStarship(starshipName: "death")
// then
assertEquals(starshipExpected, starshipActual)
}
SearchStarshipViewModelTest › val localTestRule
Run:
SearchStarshipViewModelTest.searchStars...
Tests passed: 1 of 1 test – 5 ms
SearchStarshipViewModelTest (com.alex 5 ms
searchStarship should return succes 5 ms
"/Applications/Android Studio.app/Contents/jre/jdk/Contents/Home/bin/java" ...
Process finished with exit code 0

PicPay

Testes unitários

Intrumentados

Testes unitários instrumentados

- android

PicPay

Testes unitários instrumentados

- android
- Ui

PicPay

Testes unitários instrumentados

- android
- Ui
- Espresso

PicPay

Testes unitários instrumentados

- android
- Ui
- Espresso
- Demorado em relação ao teste local

PicPay

Testes unitários instrumentados

- android
- Ui
- Espresso
- Demorado em relação ao teste local
- src/androidTest/

PicPay

Como fazer?

- Sufixo Test

PicPay

Como fazer?

- Sufixo Test
- Anotações do JUnit

PicPay

Como fazer?

- Sufixo Test
- Anotações do JUnit
- [ação]_[retorno esperado]

PicPay

Como fazer?

- Sufixo Test
- Anotações do JUnit
- [ação]_[retorno esperado]
- View -> procure -> Ação

PicPay

```
class SearchStarshipActivityTest {
    .....

    @Test
    fun whenUserClickOnSearchButton_shouldShowShipDetailWhoWasSearched() {

       mockDeathStarship()

       onView(withId(R.id.editTextShip))
            .perform(typeText("death"))

        closeSoftKeyboard()

        onView(withId(R.id.buttonSearch))
            .perform(click())

        onView(withId(R.id.textViewName))
            .check(matches(withText("Death Star")))

        onView(withId(R.id.textViewModel))
            .check(matches(withText("DS-1 Orbital Battle Station")))

        onView(withId(R.id.textViewStarshipClass))
            .check(matches(withText("Deep Space Mobile Battlestation")))
    }
```

PicPay

```
class SearchStarshipActivityTest {
    .....

    @Test
    fun whenUserClickOnSearchButton_shouldShowShipDetailWhoWasSearched() {

       mockDeathStarship()

       onView(withId(R.id.editTextShip))
            .perform(typeText("death"))

        closeSoftKeyboard()

        onView(withId(R.id.buttonSearch))
            .perform(click())

        onView(withId(R.id.textViewName))
            .check(matches(withText("Death Star")))

        onView(withId(R.id.textViewModel))
            .check(matches(withText("DS-1 Orbital Battle Station")))

        onView(withId(R.id.textViewStarshipClass))
            .check(matches(withText("Deep Space Mobile Battlestation")))
    }
```

PicPay

```
class SearchStarshipActivityTest {
    .....

    @Test
    fun whenUserClickOnSearchButton_shouldShowShipDetailWhoWasSearched() {

       mockDeathStarship()

       onView(withId(R.id.editTextShip))
            .perform(typeText("death"))

        closeSoftKeyboard()

        onView(withId(R.id.buttonSearch))
            .perform(click())

        onView(withId(R.id.textViewName))
            .check(matches(withText("Death Star")))

        onView(withId(R.id.textViewModel))
            .check(matches(withText("DS-1 Orbital Battle Station")))

        onView(withId(R.id.textViewStarshipClass))
            .check(matches(withText("Deep Space Mobile Battlestation")))
    }
```

PicPay

```
class SearchStarshipActivityTest {
    .....

    @Test
    fun whenUserClickOnSearchButton_shouldShowShipDetailWhoWasSearched() {

       mockDeathStarship()

       onView(withId(R.id.editTextShip))
            .perform(typeText("death"))

        closeSoftKeyboard()

        onView(withId(R.id.buttonSearch))
            .perform(click())

        onView(withId(R.id.textViewName))
            .check(matches(withText("Death Star")))

        onView(withId(R.id.textViewModel))
            .check(matches(withText("DS-1 Orbital Battle Station")))

        onView(withId(R.id.textViewStarshipClass))
            .check(matches(withText("Deep Space Mobile Battlestation")))
    }
```

PicPay

```
class SearchStarshipActivityTest {
    .....

    @Test
    fun whenUserClickOnSearchButton_shouldShowShipDetailWhoWasSearched() {

        mockDeathStarship()

        onView(withId(R.id.editTextShip))
             .perform(typeText("death"))

         closeSoftKeyboard()

         onView(withId(R.id.buttonSearch))
             .perform(click())

         onView(withId(R.id.textViewName))
             .check(matches(withText("Death Star")))

         onView(withId(R.id.textViewModel))
             .check(matches(withText("DS-1 Orbital Battle Station")))

         onView(withId(R.id.textViewStarshipClass))
             .check(matches(withText("Deep Space Mobile Battlestation")))
    }
```

PicPay

```
class SearchStarshipActivityTest {
    .....

    @Test
    fun whenUserClickOnSearchButton_shouldShowShipDetailWhoWasSearched() {

       mockDeathStarship()

       onView(withId(R.id.editTextShip))
            .perform(typeText("death"))

        closeSoftKeyboard()

        onView(withId(R.id.buttonSearch))
            .perform(click())

        onView(withId(R.id.textViewName))
            .check(matches(withText("Death Star")))

        onView(withId(R.id.textViewModel))
            .check(matches(withText("DS-1 Orbital Battle Station")))

        onView(withId(R.id.textViewStarshipClass))
            .check(matches(withText("Deep Space Mobile Battlestation")))
    }
```

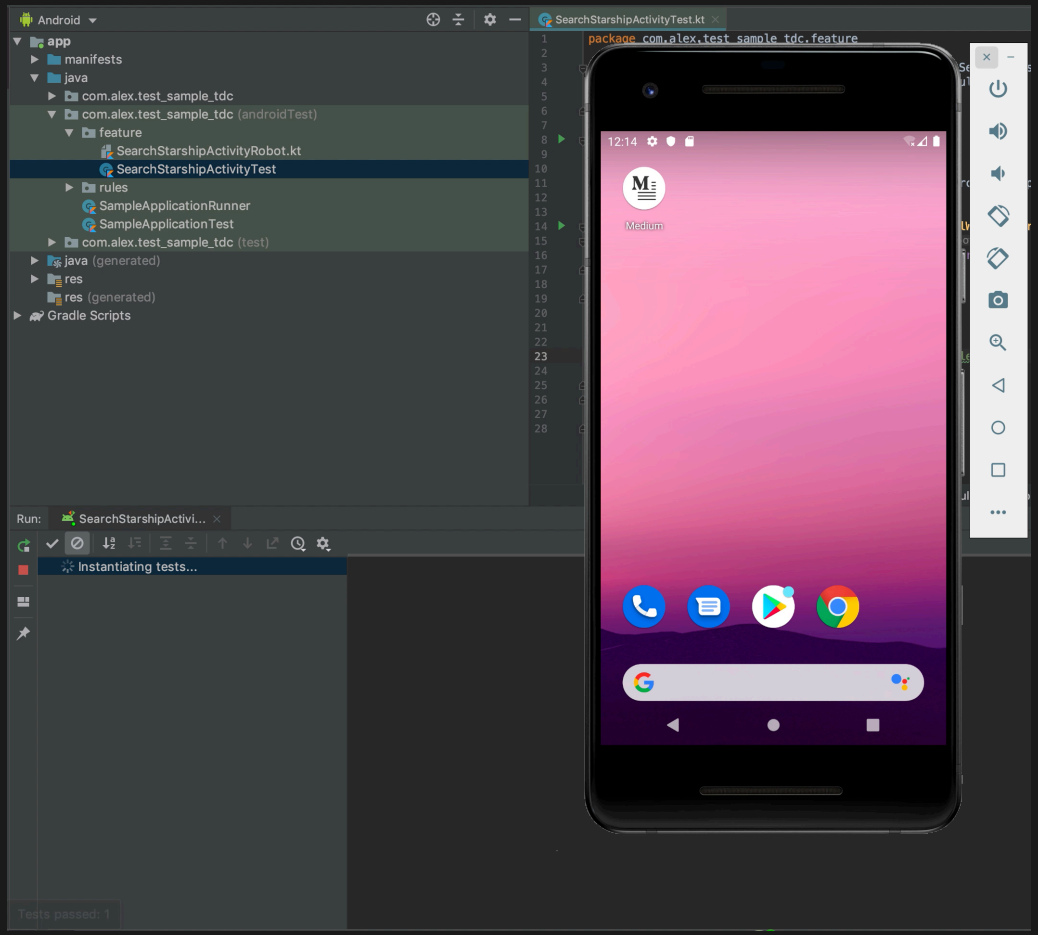
PicPay

```
class SearchStarshipActivityTest {
    .....

    @Test
    fun whenUserClickOnSearchButton_shouldShowShipDetailWhoWasSearched() {

        mockDeathStarship()

        onView(withId(R.id.editTextShip))
            .perform(typeText("death"))

        closeSoftKeyboard()

        onView(withId(R.id.buttonSearch))
            .perform(click())

        onView(withId(R.id.textViewName))
            .check(matches(withText("Death Star")))

        onView(withId(R.id.textViewModel))
            .check(matches(withText("DS-1 Orbital Battle Station")))

        onView(withId(R.id.textViewStarshipClass))
            .check(matches(withText("Deep Space Mobile Battlestation")))
    }
```

PicPay

```
class SearchStarshipActivityTest {
    .....

    @Test
    fun whenUserClickOnSearchButton_shouldShowShipDetailWhoWasSearched() {

        mockDeathStarship()

        onView(withId(R.id.editTextShip))
            .perform(typeText("death"))

        closeSoftKeyboard()

        onView(withId(R.id.buttonSearch))
            .perform(click())

        onView(withId(R.id.textViewName))
            .check(matches(withText("Death Star")))

        onView(withId(R.id.textViewModel))
            .check(matches(withText("DS-1 Orbital Battle Station")))

        onView(withId(R.id.textViewStarshipClass))
            .check(matches(withText("Deep Space Mobile Battlestation")))
    }
```

PicPay


Android
app
manifests
java
com.alex.test_sample_tdc
com.alex.test_sample_tdc (androidTest)
feature
SearchStarshipActivityRobot.kt
SearchStarshipActivityTest
rules
SampleApplicationRunner
SampleApplicationTest
com.alex.test_sample_tdc (test)
java (generated)
res
res (generated)
Gradle Scripts
SearchStarshipActivityTest.kt
package com.alex.test_sample_tdc.feature
Run:
SearchStarshipActivi...
Instantiating tests...
Tests passed: 1
12:14
Medium

PicPay

CHEAT SHEET
espresso 2.1

```
onView(ViewMatcher)
    .perform(ViewAction)
    .check(ViewAssertion);
```

```
onData(ObjectMatcher)
    .DataOptions
    .perform(ViewAction)
     check(ViewAssertion);
```

Data Options

```
inAdapterView(Matcher)
atPosition(Integer)
onChildView(Matcher)
```

View Actions

CLICK/PRESS
```
click()
doubleClick()
longClick()
pressBack()
pressIMEActionButton()
pressKey([int/EspressoKey])
pressMenuKey()
closeSoftKeyboard()
openLink()
```

GESTURES
```
scrollTo()
swipeLeft()
swipeRight()
swipeUp()
swipeDown()
```

TEXT
```
clearText()
typeText(String)
typeTextIntoFocusedView(String)
replaceText(String)
```

```
intended(IntentMatcher);
```

```
intending(IntentMatcher)
    .respondWith(ActivityResult);
```

View Matchers

USER PROPERTIES
```
withId(...)
withText(...)
withTagKey(...)
withTagValue(...)
hasContentDescription(...)
withContentDescription(...)
withHint(...)
withSpinnerText(...)
hasLinks()
hasEllipsizedText()
hasMultilineTest()
```

HIERARCHY
```
withParent(Matcher)
withChild(Matcher)
hasDescendant(Matcher)
isDescendantOfA(Matcher)
hasSibling(Matcher)
isRoot()
```

INPUT
```
supportsInputMethods(...)
hasIMEAction(...)
```

UI PROPERTIES
```
isDisplayed()
isCompletelyDisplayed()
isEnabled()
hasFocus()
isClickable()
isChecked()
isNotChecked()
withEffectiveVisibility(...)
isSelected()
```

CLASS
```
isAssignableFrom(...)
withClassName(...)
```

ROOT MATCHERS
```
isFocusable()
isTouchable()
isDialog()
withDecorView()
isPlatformPopup()
```

OBJECT MATCHER
```
allOf(Matchers)
anyOf(Matchers)
is(...)
not(...)
endsWith(String)
startsWith(String)
instanceOf(Class)
```

SEE ALSO
```
Preference matchers
Cursor matchers
Layout matchers
```

View Assertions

```
matches(Matcher)
doesNotExist()
selectedDescendantsMatch(...)
```

LAYOUT ASSERTIONS
```
noEllipseizedText(Matcher)
noMultilineButtons()
noOverlaps([Matcher])
```

POSITION ASSERTIONS
```
isLeftOf(Matcher)
isRightOf(Matcher)
isLeftAlignedWith(Matcher)
isRightAlignedWith(Matcher)
isAbove(Matcher)
isBelow(Matcher)
isBottomAlignedWith(Matcher)
isTopAlignedWith(Matcher)
```

Intent Matchers

INTENT
```
hasAction(...)
hasCategories(...)
hasData(...)
hasComponent(...)
hasExtra(...)
hasExtras(Matcher)
hasExtraWithKey(...)
hasType(...)
hasPackage()
toPackage(String)
hasFlag(int)
hasFlags(...)
isInternal()
```

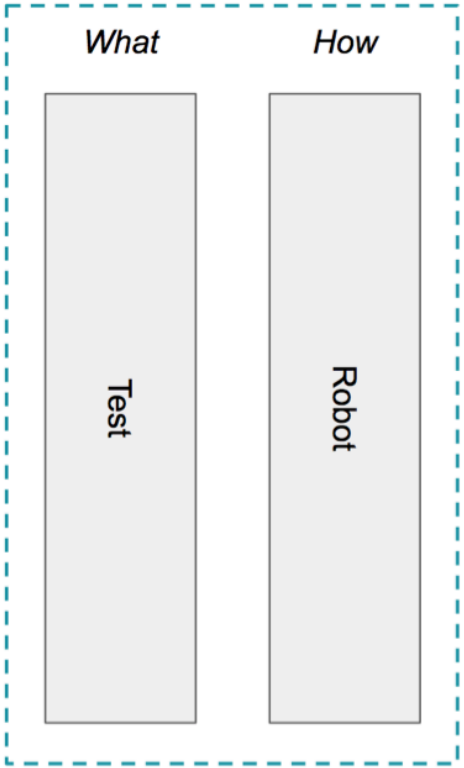
URI
```
hasHost(...)
hasParamWithName(...)
hasPath(...)
hasParamWithValue(...)
hasScheme(...)
hasSchemeSpecificPart(...)
```

COMPONENT NAME
```
hasClassName(...)
hasPackageName(...)
hasShortClassName(...)
hasMyPackageName()
```

BUNDLE
```
hasEntry(...)
hasKey(...)
hasValue(...)
```

Robot Pattern

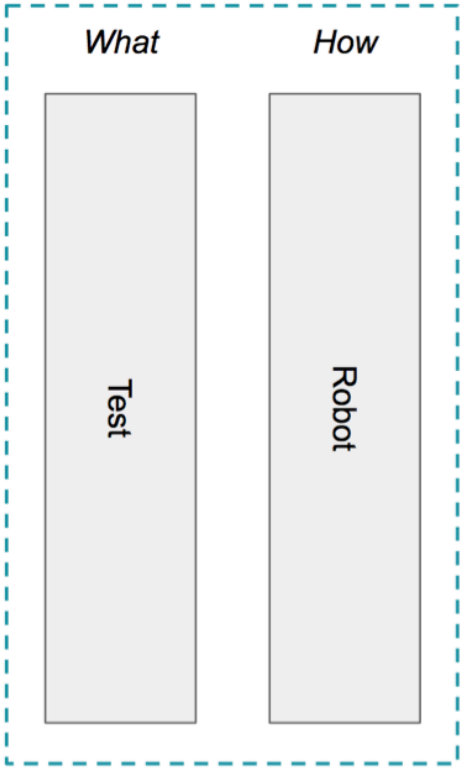
Robot Pattern




```
@Test fun whenUserClickOnSearchButton_shouldShowShipDetailWhoWasSearched() {

   mockDeathStarship()

   onView(withId(R.id.editTextShip))
        .perform(typeText("death"))

    closeSoftKeyboard()

    onView(withId(R.id.buttonSearch))
        .perform(click())

    onView(withId(R.id.textViewName))
        .check(matches(withText("Death Star")))

    onView(withId(R.id.textViewModel))
        .check(matches(withText("DS-1 Orbital Battle Station")))

    onView(withId(R.id.textViewStarshipClass))
        .check(matches(withText("Deep Space Mobile Battlestation")))
}
```

PicPay

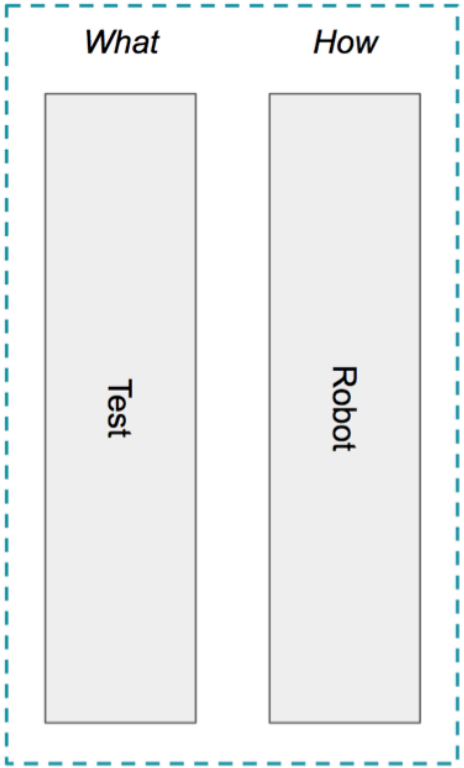
Robot Pattern




```kotlin
@Test
fun whenUserClickOnSearchButton_shouldShowShipDetailWhoWasSearched() {
    setupStarshipActivity {
        mockDeathStarship()
    } launch {
        searchStarship("death")
    } check {
        starshipDetails(
            name = "Death Star",
            model = "DS-1 Orbital Battle Station",
            starshipClass = "Deep Space Mobile Battlestation"
        )
    }
}
```

PicPay

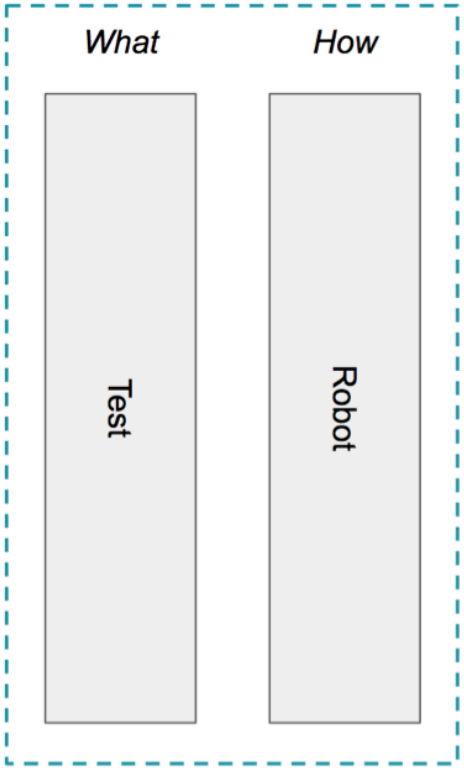
Robot Pattern




```
@Test
fun whenUserClickOnSearchButton_shouldShowShipDetailWhoWasSearched() {
    setupStarshipActivity {
        mockDeathStarship()
    } launch {
        searchStarship("death")
    } check {
        starshipDetails(
            name = "Death Star",
            model = "DS-1 Orbital Battle Station",
            starshipClass = "Deep Space Mobile Battlestation"
        )
    }
}
```

PicPay

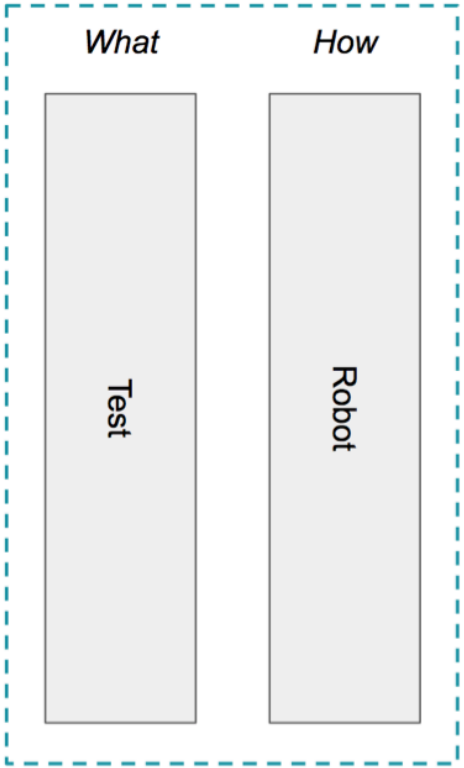
Robot Pattern




```
fun SearchStarshipActivityTest.setupStarshipActivity(
    block: SearchStarshipActivityRobot.() -> Unit
) = SearchStarshipActivityRobot().apply(block)
```

PicPay

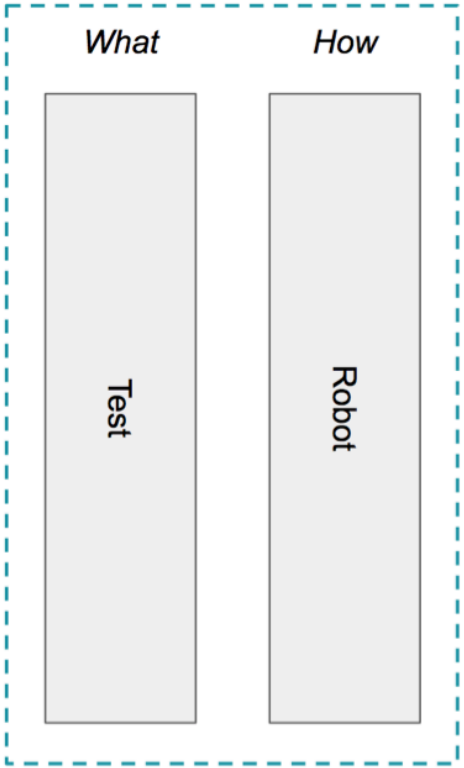
Robot Pattern




```
fun SearchStarshipActivityTest.setupStarshipActivity(
    block: SearchStarshipActivityRobot.() -> Unit
) = SearchStarshipActivityRobot().apply(block)

class SearchStarshipActivityRobot {
    fun mockDeathStarship(serverRule: MockWebServer) {...}
}
```

PicPay

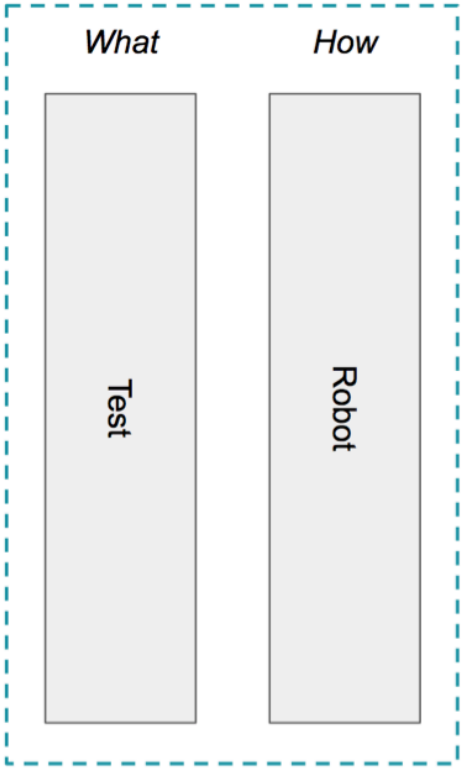
Robot Pattern




```kotlin
@Test
fun whenUserClickOnSearchButton_shouldShowShipDetailWhoWasSearched() {
    setupStarshipActivity {
        mockDeathStarship()
    } launch {
        searchStarship("death")
    } check {
        starshipDetails(
            name = "Death Star",
            model = "DS-1 Orbital Battle Station",
            starshipClass = "Deep Space Mobile Battlestation"
        )
    }
}
```

PicPay

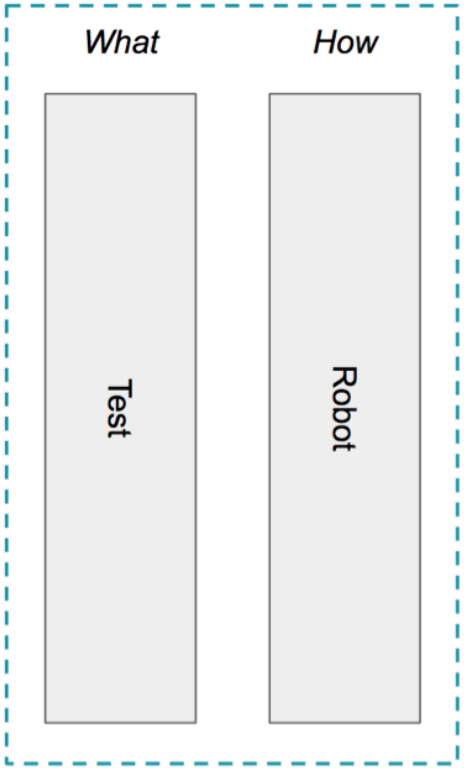
Robot Pattern




```
class SearchStarshipActivityRobot {
    //...
    infix fun launch(block: SearchStarshipActionRobot.() -> Unit) =
        SearchStarshipActionRobot().apply(block)
}
```

PicPay

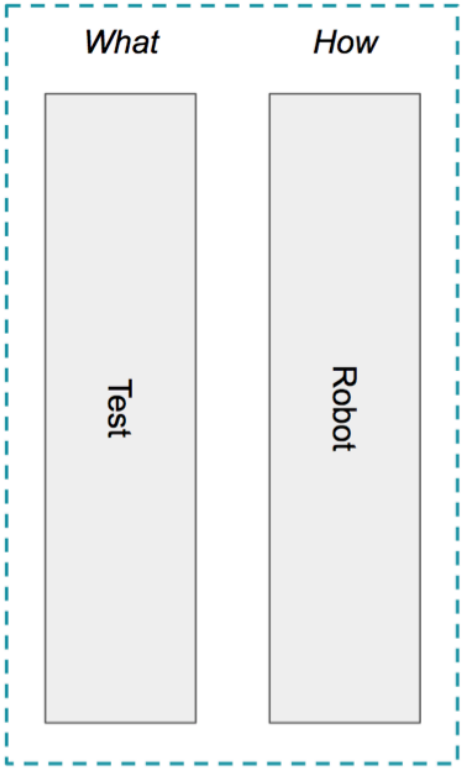
Robot Pattern




```
class SearchStarshipActivityRobot {
    //...
    infix fun launch(block: SearchStarshipActionRobot.() -> Unit) =
        SearchStarshipActionRobot().apply(block)
}

class SearchStarshipActionRobot {

    fun searchStarship(starshipName: String) {
        onView(withId(R.id.editTextNave))
            .perform(typeText(starshipName))

        closeSoftKeyboard()

        onView(withId(R.id.buttonSearch))
            .perform(click())
    }

}
```

PicPay

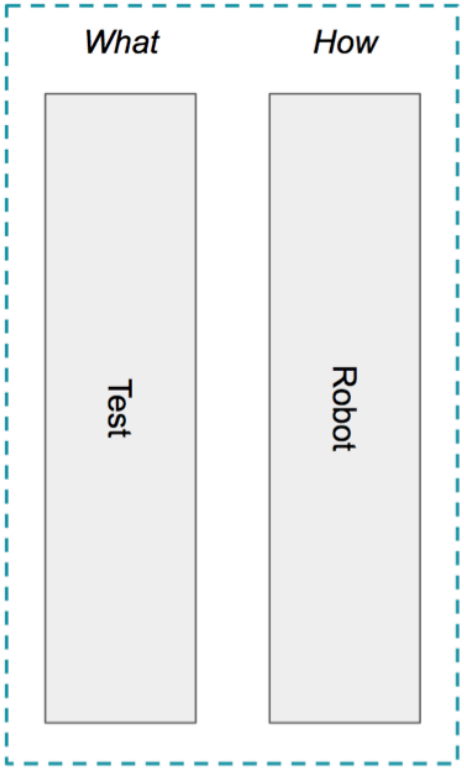
Robot Pattern




```
@Test
fun whenUserClickOnSearchButton_shouldShowShipDetailWhoWasSearched() {
    setupStarshipActivity {
        mockDeathStarship()
    } launch {
        searchStarship("death")
    } check {
        starshipDetails(
            name = "Death Star",
            model = "DS-1 Orbital Battle Station",
            starshipClass = "Deep Space Mobile Battlestation"
        )
    }
}
```

PicPay

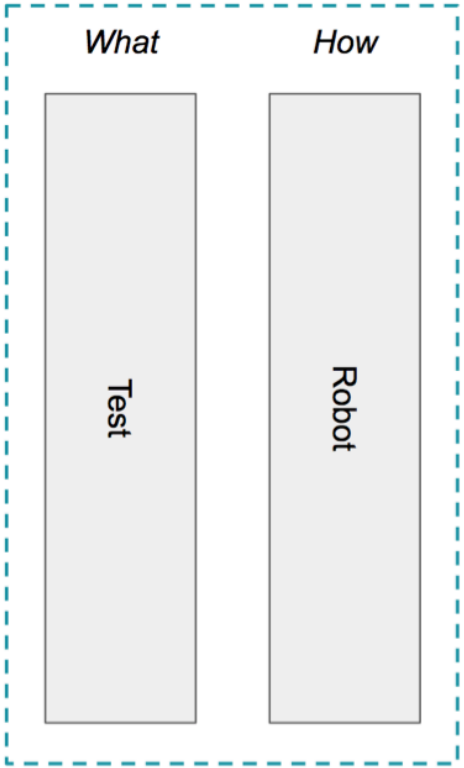
Robot Pattern




```
class SearchStarshipActionRobot {
    //...
    infix fun check(block: SearchStarshipCheckRobot.() -> Unit) =
        SearchStarshipCheckRobot().apply(block)
}
```

PicPay

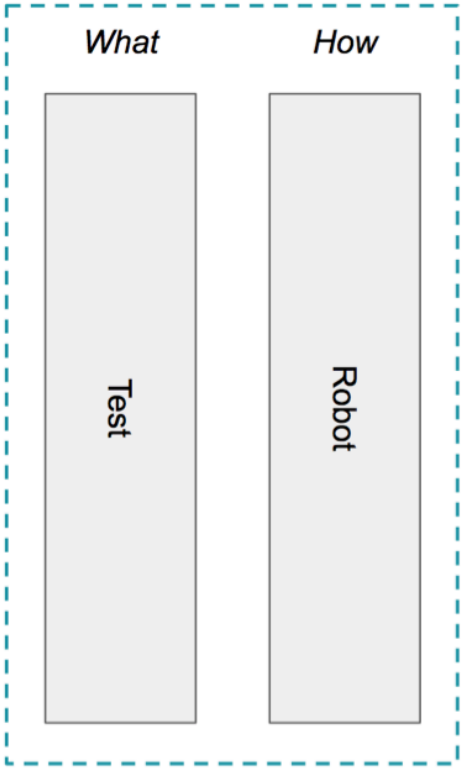
Robot Pattern




```
class SearchStarshipActionRobot {
    //...
    infix fun check(block: SearchStarshipCheckRobot.() -> Unit) =
        SearchStarshipCheckRobot().apply(block)
}

class SearchStarshipCheckRobot {

    fun starshipDetails(
        name: String,
        model: String,
        starshipClass: String
    ) {
        onView(withId(R.id.textViewName))
            .check(matches(withText(name)))

        onView(withId(R.id.textViewModel))
            .check(matches(withText(model)))

        onView(withId(R.id.textViewStarshipClass))
            .check(matches(withText(starshipClass)))
    }
}
```

PicPay

Robot Pattern




```kotlin
@Test
fun whenUserClickOnSearchButton_shouldShowShipDetailWhoWasSearched() {
    setupStarshipActivity {
        mockDeathStarship()
    } launch {
        searchStarship("death")
    } check {
        starshipDetails(
            name = "Death Star",
            model = "DS-1 Orbital Battle Station",
            starshipClass = "Deep Space Mobile Battlestation"
        )
    }
}
```

PicPay

Framework externos

Frameworks externos

PicPay

Frameworks externos

PicPay

Frameworks externos

PicPay

Frameworks externos

MockWebServer

screenshot-tests-for-android

PicPay

Frameworks externos

PicPay

Code Coverage

PicPay

Dedicar um tempo a mais de desenvolvimento garante uma melhor experiência para o cliente

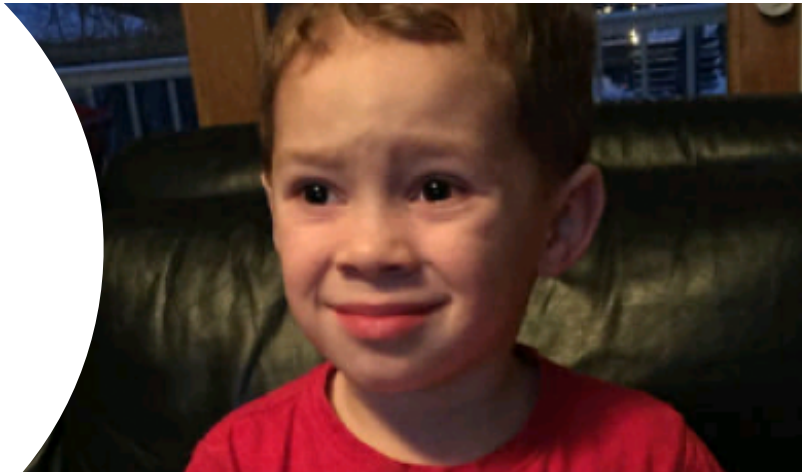
Insight

PicPay

Quando o teste Passa!!!!!!

PicPay

Dúvidas?



PicPay

Muito obrigado!

 /alex-soares-siqueira  /AlexSoaresDeSiqueira

PicPay


PicPay

Referências

Projeto
https://github.com/AlexSoaresDeSiqueira/TestSample-TDC

Robot Pattern

https://jakewharton.com/testing-robots/

GIVEN WHEN THEN

https://martinfowler.com/bliki/GivenWhenThen.html

Espresso

https://developer.android.com/training/testing/espresso

Teste Unitario Local

https://developer.android.com/training/testing/unit-testing/local-unit-tests