



UTILIZANDO ESPRESSO E UIAUTOMATOR

Eduardo Carrara

Developer Evangelist – Intel Developers Relations Division

#ANDROIDONINTEL



<https://www.facebook.com/ducarrara>



@DuCarrara



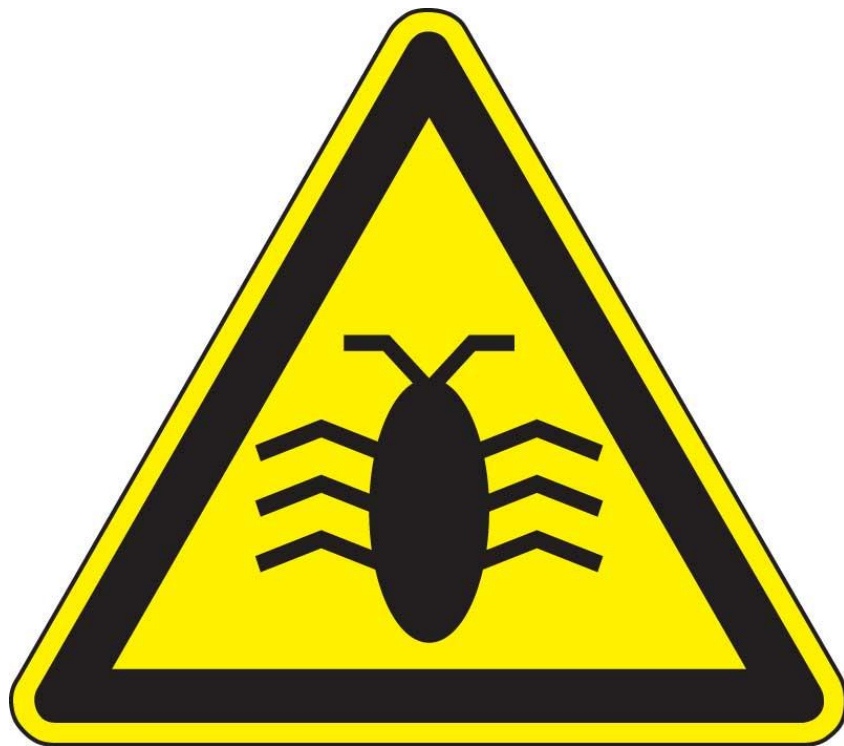
<br.linkedin.com/in/eduardocarrara/>



+EduardoCarraraDeAraujo



ecarrara-araujo



*“I choose a lazy person to do a hard job.
Because a lazy person will find an easy way to do it.”
- Bill Gates*



Image by Karla Vidal @ <http://www.flickr.com/photos/63721650@N00/3661526274>
Creative Commons cc-by-2.0

Problemas?

Automação de Testes

Testes de Integração

Fragmentação e Testes

Acceptance
Testing



System
Testing



Integration
Testing



Unit
Testing

JUnit

JUnit

JUnit



UI Automator

JUnit



Android Testing Framework

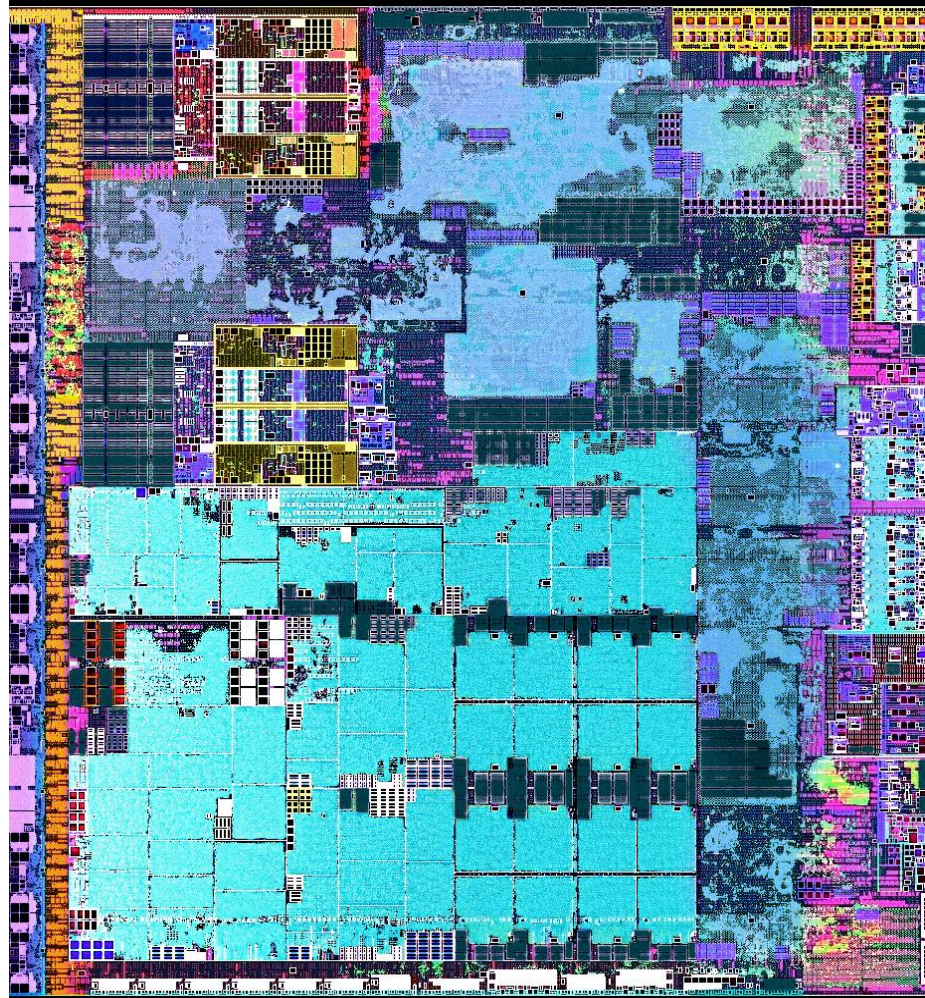


- **Special TestCases, Asserts e Mocks**
- MoreAsserts
- ViewAsserts
- MockApplication
- MockContext
- MockContentProvider
- AndroidTestCase
- ActivityInstrumentationTestCase2
- ActivityUnitTestCase
- ApplicationTestCase
- InstrumentationTestCase
- ProviderTestCase
- ServiceTestCase
- SingleLaunchActivityTestCase

JUnit

Instrumentation

- Android Components Context Access;
- Components Lifecycle Control;
- Component Loading Control;
- System Events (e.g.: Broadcasts)
- InstrumentationTestRunner
 - AndroidJUnitRunner
 - GoogleInstrumentationTestRunner



Espresso

- Simplifies the UI Test Process within your App
- Methods for:
 - View matching
 - Checks
 - UI Events



Espresso - Configuração

build.gradle

```
dependencies {  
    androidTestCompile 'com.android.support.test:runner:0.3'  
    androidTestCompile 'com.android.support.test:rules:0.3'  
    androidTestCompile 'com.android.support.test.espresso:espresso-core:2.2'  
}
```

Desabilitar nas opções do desenvolvedor (no device):

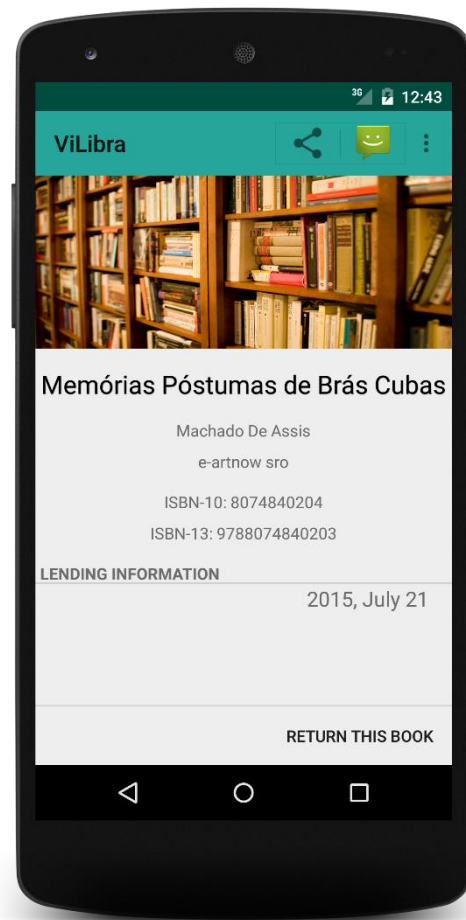
- Window Animation Scale
- Transition Animation Scale
- Animator Duration Scale

Espresso – Primeiro Test Case

1. Crie uma subclasse de `ActivityInstrumentationTestCase2`
2. Identifique o(s) componente(s) de UI que você quer testar.
3. Simule a interação do usuário com o componente.
4. Repita os passos utilizando assertions para verificar se a UI reflete o estado esperado depois das interações.

Espresso – Caso Vilibra

- Bibliotecário Virtual
- Ajudar a lembrar os livros emprestados.
- Nesta caso como automatizar o teste da visualização de detalhes de um empréstimo?
- Classe:
TestViewLendedBookDetailFlow



Espresso – Caso Vilibra

```
public class TestViewLendedBookDetailFlow
    extends ActivityInstrumentationTestCase2<BookListActivity> {

    ...

    public TestViewLendedBookDetailFlow() {
        super(BookListActivity.class);
    }

    ...

}
```


Espresso – Caso Vilibra

```
public class TestViewLendedBookDetailFlow
    extends ActivityInstrumentationTestCase2<BookListActivity> {

    ...

    @Before
    public void setUp() throws Exception {
        super.setUp();
        injectInstrumentation(InstrumentationRegistry.getInstrumentation());
        mBookListActivity = getActivity();

        prepareTestData();
    }

    ...
}
```

Espresso – Caso Vilibra

```
public class TestViewLendedBookDetailFlow
    extends ActivityInstrumentationTestCase2<BookListActivity> {

    ...

    @After
    public void tearDown() throws Exception {
        clearTestData();
    }

    ...
}
```


Espresso – Caso Vilibra

```
public class TestViewLendedBookDetailFlow
    extends ActivityInstrumentationTestCase2<BookListActivity> {

    ...

    @Test
    public void testViewLendedBookDetailFlow() {
        onData(CursorMatchers.withRowString(
            VilibraContract.BookEntry.COLUMN_TITLE,
            mBookTitle))
            .inAdapterView(withId(R.id.lended_book_list_view))
            .perform(click());
    }

    ...
}
```

continua

Espresso – Caso Vilibra

```
public class TestViewLendedBookDetailFlow
    extends ActivityInstrumentationTestCase2<BookListActivity> {

    ...

    @Test
    public void testViewLendedBookDetailFlow() {
        // check data in the detail screen
        onView(withId(R.id.book_title_text_view))
            .check(matches(withText(mBookTitle)));
        onView(withId(R.id.book_subtitle_text_view))
            .check(matches(withText(mBookSubtitle)));
    }

    ...
}
```

continua

Espresso – Caso Vilibra

```
public class TestViewLendedBookDetailFlow
    extends ActivityInstrumentationTestCase2<BookListActivity> {

    ...

    @Test
    public void testViewLendedBookDetailFlow() {

        onView(withId(R.id.book_authors_text_view))
            .check(matches(withText(mBookAuthors))) ;
        onView(withId(R.id.book_publisher_edition_text_view))
            .check(matches(withText(mBookPublisher))) ;

    }

    ...

}
```

continua

Espresso – Caso Vilibra

```
public class TestViewLendedBookDetailFlow
    extends ActivityInstrumentationTestCase2<BookListActivity> {

    ...

    @Test
    public void testViewLendedBookDetailFlow() {

        pressBack();

    }

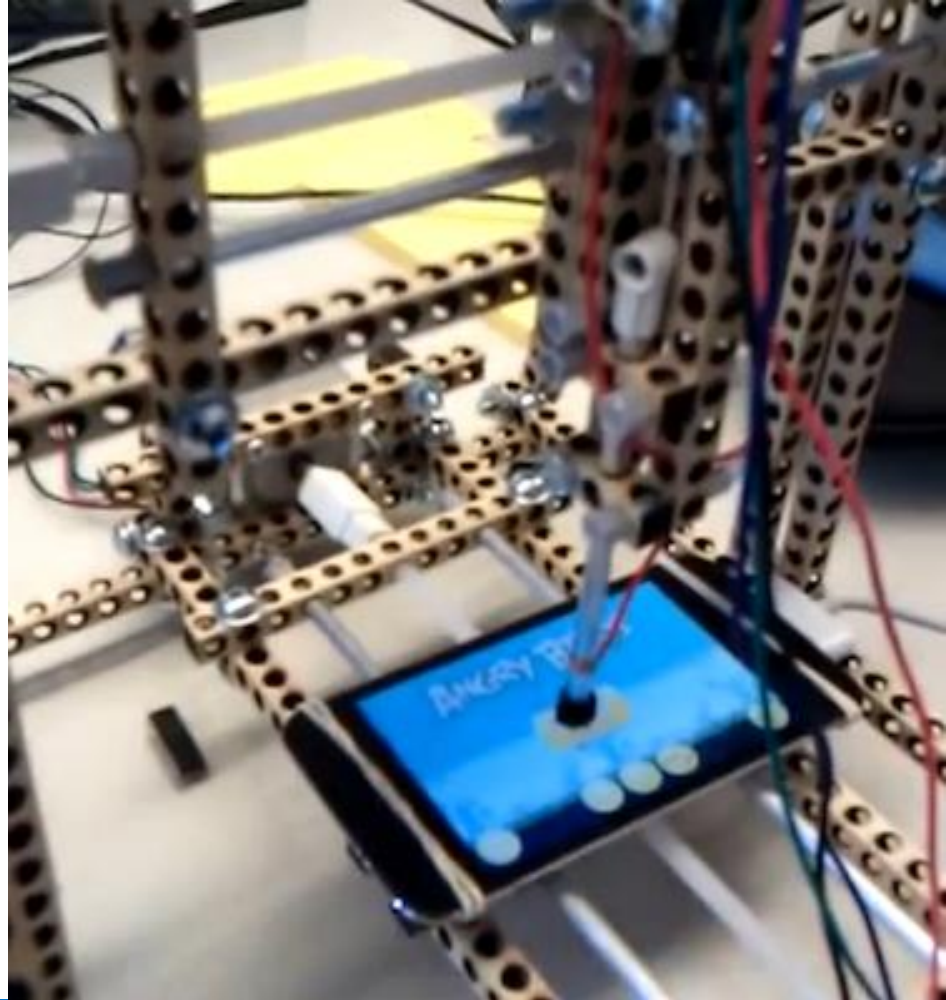
    ...

}
```

Fim 😊

UI Automator

- Testes Cross App
- Baseado em Instrumentação
- Interage com elementos visíveis utilizando descritores convenientes (como texto).



UIAutomator - Configuração

build.gradle

```
dependencies {  
    androidTestCompile 'com.android.support.test:runner:0.3'  
    androidTestCompile 'com.android.support.test:rules:0.3'  
    androidTestCompile 'com.android.support.test.uiautomator:uiautomator-  
v18:2.1.1'  
}
```

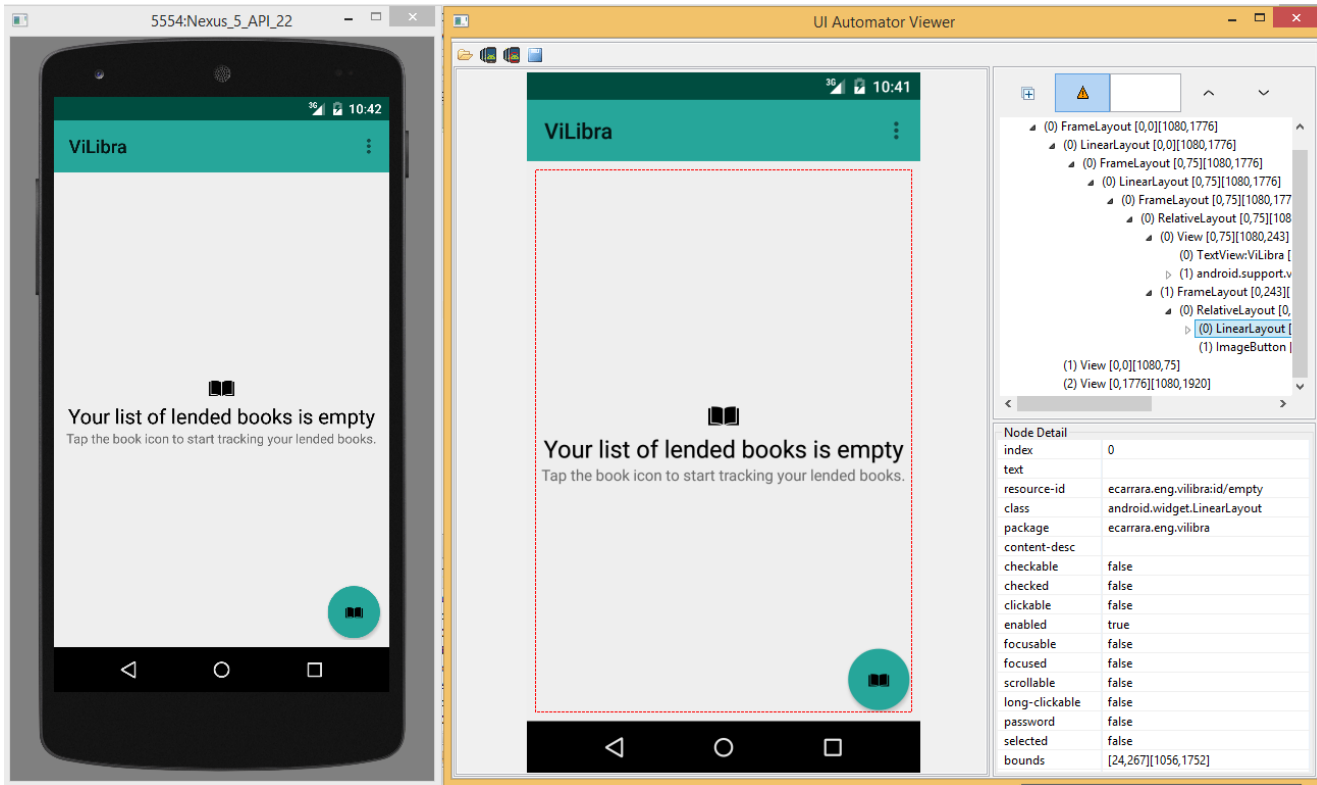
É necessário inspecionar os componentes visuais das apps alvo para garantir que o UIAutomator consiga localizá-los.

Isso significa ter labels de texto visíveis, a propriedade android:contentDescription preenchida, ou ainda a propriedade android:hint (para EditTexts)

UIAutomator – Primeiro Test Case

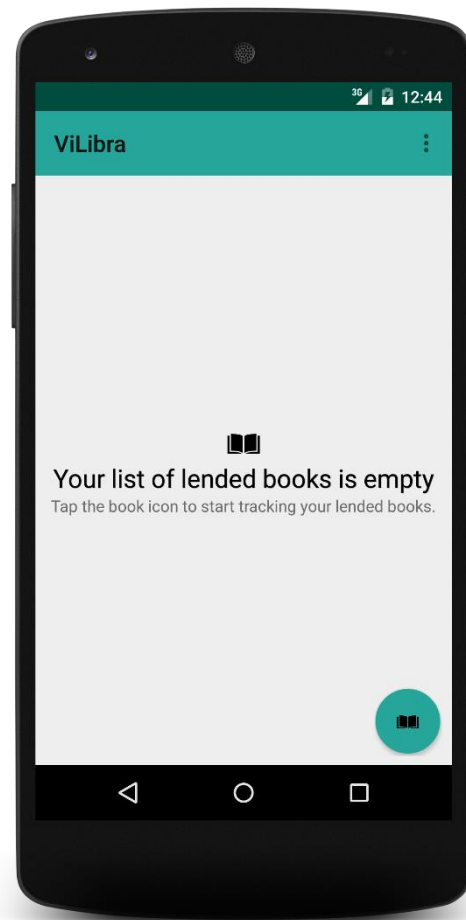
1. Crie uma subclasse de **InstrumentationTestCase**
2. Obtenha uma instância de **UIDevice**.
3. Utilize o método **UIDevice.findObject()** para obter **UIObjects** e executar **ações**.
4. Utilize asserts para verificar os resultados.

UIAutomator - uiautomatorviewer



UIAutomator – Caso Vilibra

- Novo caso: automatizando o registro de um empréstimo.
- Classe: TestLendBookFlow



UIAutomator – Caso Vilibra

```
public class TestLendBookFlow extends InstrumentationTestCase {  
  
    ...  
  
    @Before  
    public void setUp() throws UiObjectNotFoundException {  
  
        clearTestData();  
        mUiDevice = UiDevice.getInstance(getInstrumentation());  
        // Should start from the home screen  
        mUiDevice.pressHome();  
  
    }  
  
    ...  
  
}
```

continua

UIAutomator – Caso Vilibra

```
public class TestLendBookFlow extends InstrumentationTestCase {  
  
    ...  
  
    @Before  
    public void setUp() throws UiObjectNotFoundException {  
  
        UiObject allAppsButton = mUiDevice.findObject(new UiSelector()  
            .description("Apps"));  
        assertTrue(allAppsButton.exists());  
        allAppsButton.clickAndWaitForNewWindow();  
  
    }  
  
    ...  
  
}
```

continua

UIAutomator – Caso Vilibra

```
public class TestLendBookFlow extends InstrumentationTestCase {  
  
    ...  
  
    @Before  
    public void setUp() throws UiObjectNotFoundException {  
  
        UiObject appsTab = mUiDevice.findObject(new UiSelector()  
            .text("Apps"));  
        assertTrue(appsTab.exists());  
        appsTab.click();  
  
    }  
  
    ...  
  
}
```

continua

UIAutomator – Caso Vilibra

```
public class TestLendBookFlow extends InstrumentationTestCase {  
  
    ...  
  
    @Before  
    public void setUp() throws UiObjectNotFoundException {  
  
        // find the scrollable list of apps  
        UiScrollable appsList = new UiScrollable(new UiSelector()  
            .scrollable(true));  
        appsList.setAsHorizontalList();  
  
    }  
  
    ...  
  
}
```

continua

UIAutomator – Caso Vilibra

```
public class TestLendBookFlow extends InstrumentationTestCase {  
  
    ...  
  
    @Before  
    public void setUp() throws UiObjectNotFoundException {  
        UiObject vilibraApp = appsList.getChildByText(new UiSelector()  
            .className("android.widget.TextView"), "ViLibra");  
        vilibraApp.click();  
  
        mUiDevice.wait(Until.hasObject(By.pkg("ecarrara.eng.vilibra")),  
            5000L);  
    }  
  
    ...  
  
}
```

Fim! Do setup...

UIAutomator – Caso Vilibra

```
public class TestLendBookFlow extends InstrumentationTestCase {  
  
    ...  
  
    @Test  
    public void testLendBookFlow() throws UiObjectNotFoundException {  
  
        UiObject lendBookButton = mUiDevice.findObject(new UiSelector()  
            .resourceId("ecarrara.eng.vilibra:id/add_lending_action_button")  
            .className("android.widget.ImageButton"));  
        lendBookButton.clickAndWaitForNewWindow();  
  
    }  
  
    ...  
  
}
```

continua

UIAutomator – Caso Vilibra

```
public class TestLendBookFlow extends InstrumentationTestCase {  
  
    ...  
  
    @Test  
    public void testLendBookFlow() throws UiObjectNotFoundException {  
  
        UiObject isbnEditText = mUiDevice.findObject(new UiSelector()  
            .text("ISBN")  
            .className("android.widget.EditText"));  
        isbnEditText.setText(BOOK_ISBN);  
  
    }  
  
    ...  
  
}
```

continua

UIAutomator – Caso Vilibra

```
public class TestLendBookFlow extends InstrumentationTestCase {  
  
    ...  
  
    @Test  
    public void testLendBookFlow() throws UiObjectNotFoundException {  
  
        UiObject confirmButton = mUiDevice.findObject(new UiSelector()  
            .text("Confirm")  
            .className("android.widget.Button"));  
        confirmButton.clickAndWaitForNewWindow();  
  
    }  
  
    ...  
  
}
```

continua

UIAutomator – Caso Vilibra

```
public class TestLendBookFlow extends InstrumentationTestCase {  
  
    ...  
  
    @Test  
    public void testLendBookFlow() throws UiObjectNotFoundException {  
  
        UiObject isbnTextView = mUiDevice.findObject(new UiSelector()  
            .className("android.widget.TextView")  
            .resourceId("ecarrara.eng.vilibra:id/book_isbn10_text_view"));  
        Assert.assertTrue(isbnTextView.getText().contains(BOOK_ISBN));  
  
    }  
  
    ...  
  
}
```

continua

UIAutomator – Caso Vilibra

```
public class TestLendBookFlow extends InstrumentationTestCase {  
  
    ...  
    @Test  
    public void testLendBookFlow() throws UiObjectNotFoundException {  
  
        UiObject lendButton = mUiDevice.findObject(new UiSelector()  
            .className("android.widget.Button")  
            .text("Lend this Book"));  
        lendButton.clickAndWaitForNewWindow();  
  
    }  
    ...  
}
```

continua

UIAutomator – Caso Vilibra

```
public class TestLendBookFlow extends InstrumentationTestCase {  
  
    ...  
    @Test  
    public void testLendBookFlow() throws UiObjectNotFoundException {  
        UiObject contactView = mUiDevice.findObject(new UiSelector()  
            .className("android.widget.TextView")  
            .text("Meu Irmao"));  
        contactView.click();  
  
        mUiDevice.wait(Until.hasObject(By.pkg("ecarrara.eng.vilibra")),  
            500L);  
    }  
    ...  
}
```

continua

UIAutomator – Caso Vilibra

```
public class TestLendBookFlow extends InstrumentationTestCase {  
  
    ...  
    @Test  
    public void testLendBookFlow() throws UiObjectNotFoundException {  
        UiObject bookTitleTextView = mUiDevice.findObject(new UiSelector()  
            .className("android.widget.TextView")  
            .resourceId("ecarrara.eng.vilibra:id/book_name_text_view")  
            .text(BOOK_TITLE));  
  
        Assert.assertTrue(bookTitleTextView.exists());  
    }  
    ...  
}
```

Fim 😊

The background of the slide is a close-up photograph of shattered glass. The glass is broken into many sharp, irregular fragments, with a network of dark, jagged cracks radiating across the surface. The lighting is bright, creating a high-contrast scene with deep shadows in the cracks and bright highlights on the glass edges.

“If you don’t like testing your product,
most likely your customers won’t like
to test it either.”
- *Anonymous*

Fragmentação

Variedade de tamanhos de tela

Arquiteturas Diferentes

Versões diferentes de OS

testdroid.

Projects

New project name

Android

+

Vilibra

Android

Demo Project

Android

Vilibra

New testrun

Share project

Reports

Delete project

Empty description

| Name | Successful tests | Devices finished | Test and app file name | Starting date | Device count |
|------------|------------------|------------------|---|------------------------|--------------|
| Test Run 1 | 0% | 100% | app-debug-androidTe... app-debug.apk | 2015-04-07 12:37:42 | 8 |

Create new testrun for android project Vilibra

1. Application

2. Pick test type

3. Select devices

4. Advanced options

1. Application: app-debug.apk

file size: 6.1 MB file upload date: 4/7/2015, 12:35:10 PM

Update application file:

Use this field to upload or update your application file. Please use the *.apk files only.

Choose File

app-debug.apk

Prev

Next

Start

Create new testrun for android project [Vilibra](#)

<

>

▶

?

1. Application ✓

2. Pick test type <

3. Select devices

4. Advanced options

2. Pick test type:

Please specify test type

You can upload any test package that is using Android Instrumentation Framework. The best way to create such package is to use [Testdroid Recorder](#) which helps you to create very robust test scripts with minimum effort.

In case you don't have tests, you can use our embedded feature "App Crawler". App Crawler is an intelligent application crawler that exercises each application systematically by traversing each application through each of the views in the view hierarchy taking screenshots and recording performance data as the crawler progresses through each view.

☐ App Crawler
 ☒ File

Current test file: **app-debug-androidTest-unaligned.apk**

file size: 957.2 KB file upload date: 4/7/2015, 12:35:35 PM

Please specify new test file

Use this field to upload or update your test file. Please use the *.apk files only.

No file chosen

< Prev

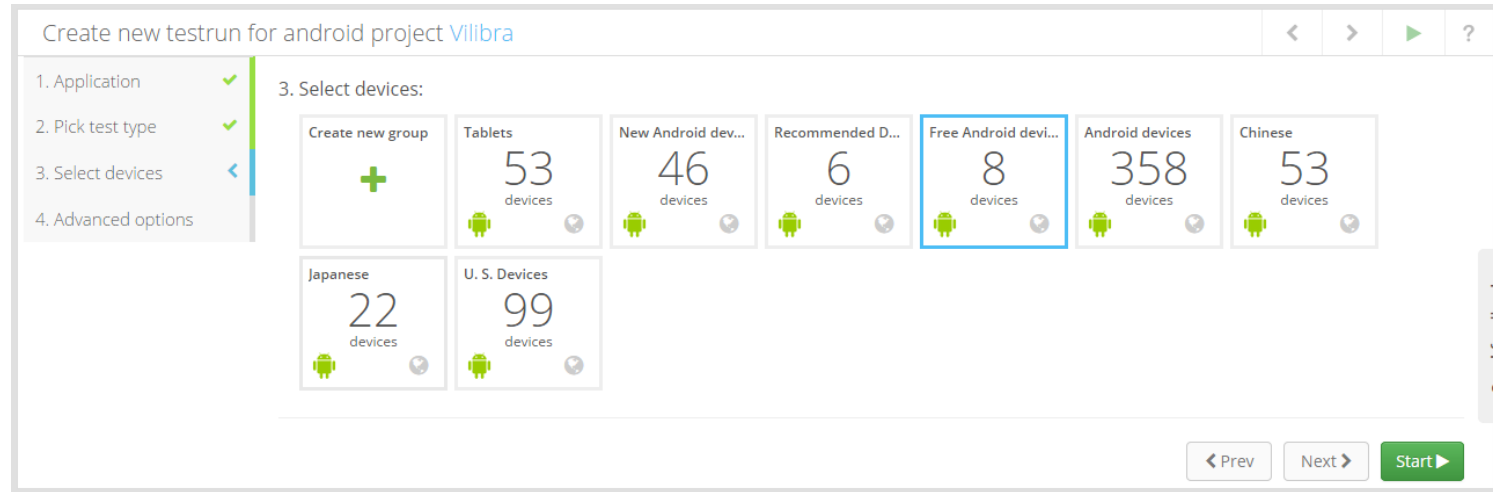
Next >

Start ▶

Send feedback

Intel Software and Services Group

42



Test Run 1 in [Vilibra](#) project
Created by Edduardo Carrara de Ar...
?

⊗ FINISHED WITH FAILURES

| | | | | | |
|-----------------------|---|--------------------|--------------------------------------|-------------------|---|
| Executed devices: | 0 | Project type: | Android | Application: | app-debug.apk ⓘ |
| Not executed devices: | 0 | Starting time: | 4/7/2015, 12:37:42 PM | Tests: | app-debug-androidTest-unaligned.apk ⓘ Compare |
| Excluded devices: | 1 | Language used: | English, United States | Screenshots: | Not available |
| Failed devices: | 7 | Device group used: | Free Android devices | Logs: | Available ⓘ |
| Running devices: | 0 | Tags: | + | Performance logs: | Available ⓘ |
| Waiting devices: | 0 | | | | |


nd feedback

What is next?

- Experimente automatizar os testes de sua app!
- Como fazer Integração e Entrega Contínua com Android?
- Cobertura de Código
- Mocking



Intel Developer Zone

 Developer Zone

Join Today > Log in


Android*

Search our content library...


HOMELEARNGET A DEVICETOOLSWHAT'S NEW

Test Android* Apps on Intel Devices...Virtually

Ensure your apps run smoothly and deliver the best user experience on [Intel-based Android devices](#) by testing through a virtual testing service. There's no need to acquire multiple physical devices—now you can test apps virtually for free.



Testdroid*




Save on app development costs, reduce risks, speed up time to market, and reduce operational costs with real device testing through the Testdroid Cloud.

[How-To Guide >](#)

[Begin Testing >](#)

Testin*



Improve app quality and security, and better navigate the Chinese mobile market with real device-based, automated cloud testing tools and services.

Only available in China.

[Begin Testing >](#)

<https://software.intel.com/en-us/android/app-testing>

Nos Avalie!

<http://bit.ly/IntelPesquisa>

THANKS!



<https://www.facebook.com/ducarrara>



@DuCarrara



<br.linkedin.com/in/eduardocarrara/>



+EduardoCarraraDeAraujo



ecarrara-araujo/vilibra

References

- Android Testing: https://developer.android.com/tools/testing/testing_android.html
- Android Unit Testing Support: <http://tools.android.com/tech-docs/unit-testing-support>
- UI Testing: <https://developer.android.com/training/testing/ui-testing/index.html>
- Android Testing Support Library: <https://developer.android.com/tools/testing-support-library>
- Android Instrumentation: http://developer.android.com/tools/testing/testing_android.html#Instrumentation
- Junit: <http://junit.org>
- Testdroid: <http://testdroid.com>
- Intel App Testing Page: <https://software.intel.com/en-us/android/app-testing>

