campus party GOIÁS

TDD para Android com Kotlin

Alex R. Ferreira

http://alexferreira.dev

Apresentação

Formação

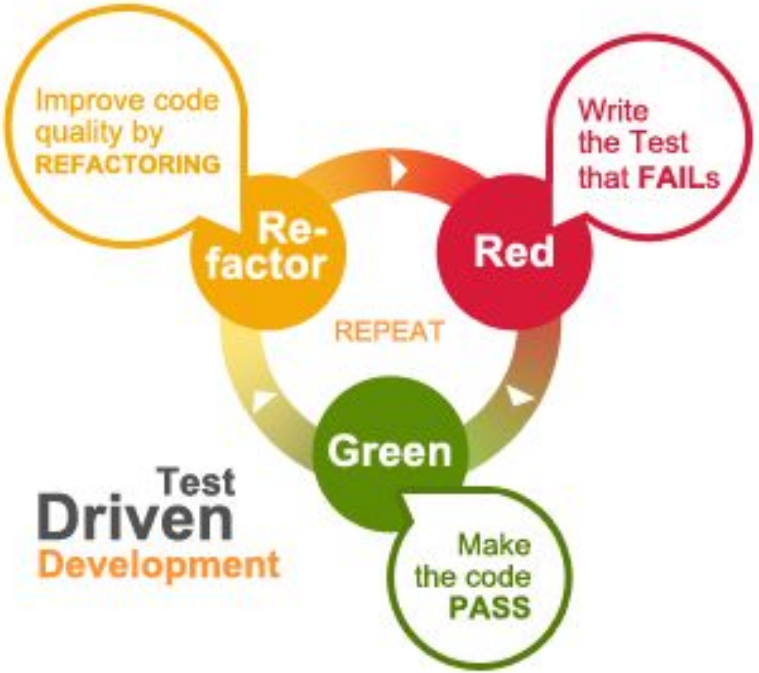
- Sistema de Informação/UFG
- Me. Ciências da Computação/UFG
- Curso Advanced Android Developer

Carreira Profissional

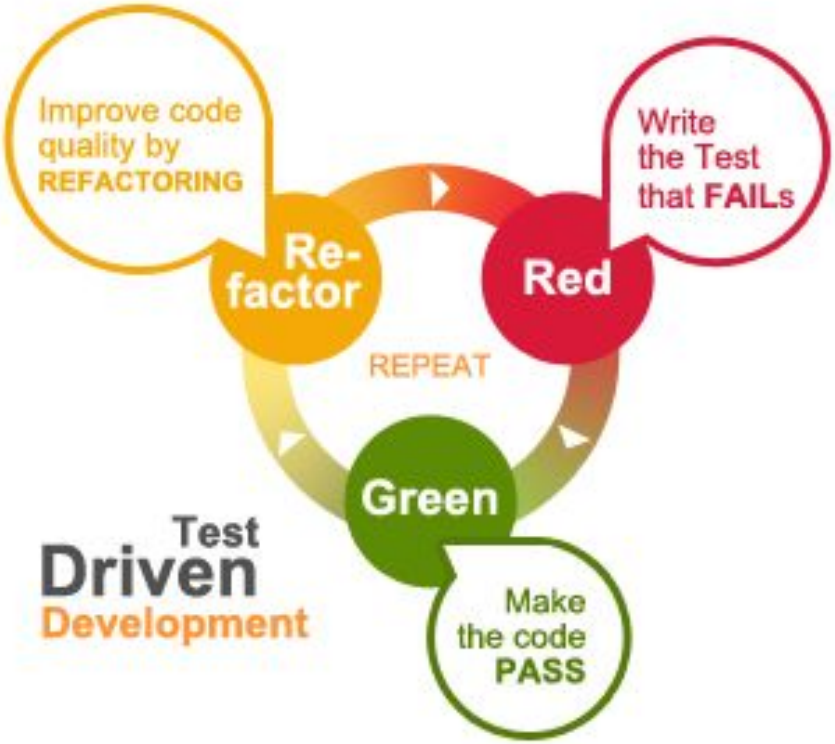
- Laboratório LUPA: 2012-2013
- Laboratório LABTIME/UFG: 2014
- Laboratório LABORA/UFG: 2015-2017
- ZG Soluções: 2017-2019
- Anfeli: 2019
- Faculdade Senac: 2019

Apresentação

anfeli


Improve code quality by REFACTORING
Re-factor
Write the Test that FAILs
Red
REPEAT
Green
Make the code PASS
Test Driven Development

TDD


Improve code quality by REFACTORING
Re-factor
Red
Write the Test that FAILs
REPEAT
Green
Make the code PASS
Test Driven Development

?

🤔 🤔 🤔

TDD - Test Driven Development

O objetivo do TDD é especificação ao invés de validação

Kent Beck

TDD - Test Driven Development

TDD - Test Driven Development

Primeiro, Crie o teste

O teste deve falhar

TDD - Test Driven Development

Primeiro, Crie o teste

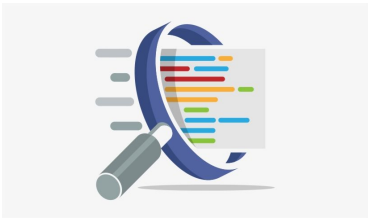
Implemente

Teste e corrija até passar

TDD - Test Driven Development

TDD - Test Driven Development

TDD - Test Driven Development



Planeje a feature

Arquitetura atual está estável para feature

TDD - Test Driven Development

Planeje a feature

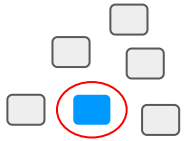
TDD = Refatoração + TFD

Arquitetura atual está estável para feature

TDD - Test Driven Development

O que um teste deve ter como características ?

Rodar rápido (pequenos setup, tempo de execução e tear down)

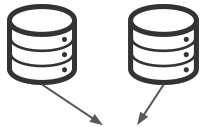


Rodar isoladamente (pode rodar com outra ordem)

Usar dados que torna fácil para ler e entender

TDD - Test Driven Development

O que um teste deve ter como características ?



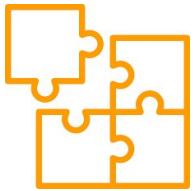
Usar cópias de dados de produção quando necessários

Representar uma parte de todo o objetivo

TDD - Test Driven Development

Benefícios

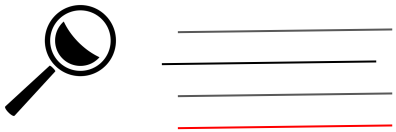
Eles irão se tornar uma parte da documentação técnica



Possibilita criar o software em pequenas etapas

TDD - Test Driven Development

Benefícios



Mais fácil encontrar um erro em uma linha

Do que depois de você ter escrito 1000 linhas

TDD - Test Driven Development

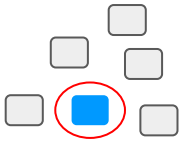
Desvantagens

Tempo de execução de todos testes não escalável

Não são todos desenvolvedores que sabem como testar. Necessita de treinamento por pessoas com habilidades de testes de unidade

TDD - Test Driven Development

Desvantagens



Nem todos do time fazem testes. Ou o time desiste de utilizar TDD

ou ...

Aplicativo com Android

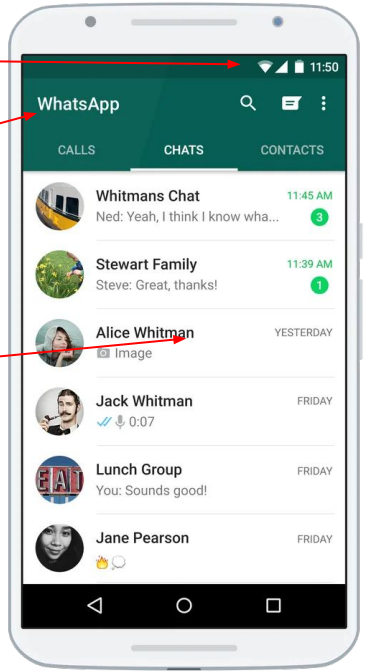
Aplicativo com Android

O que um app contém ?

Download de dados

Activity

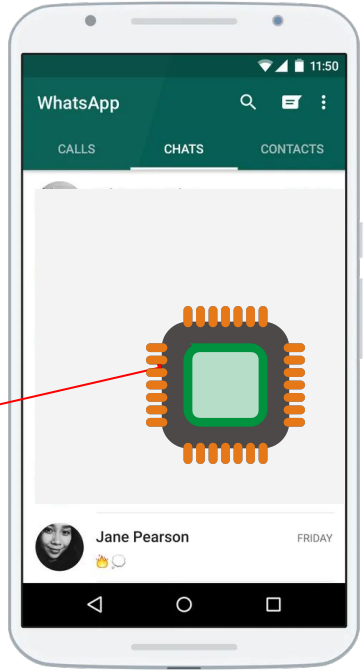
View



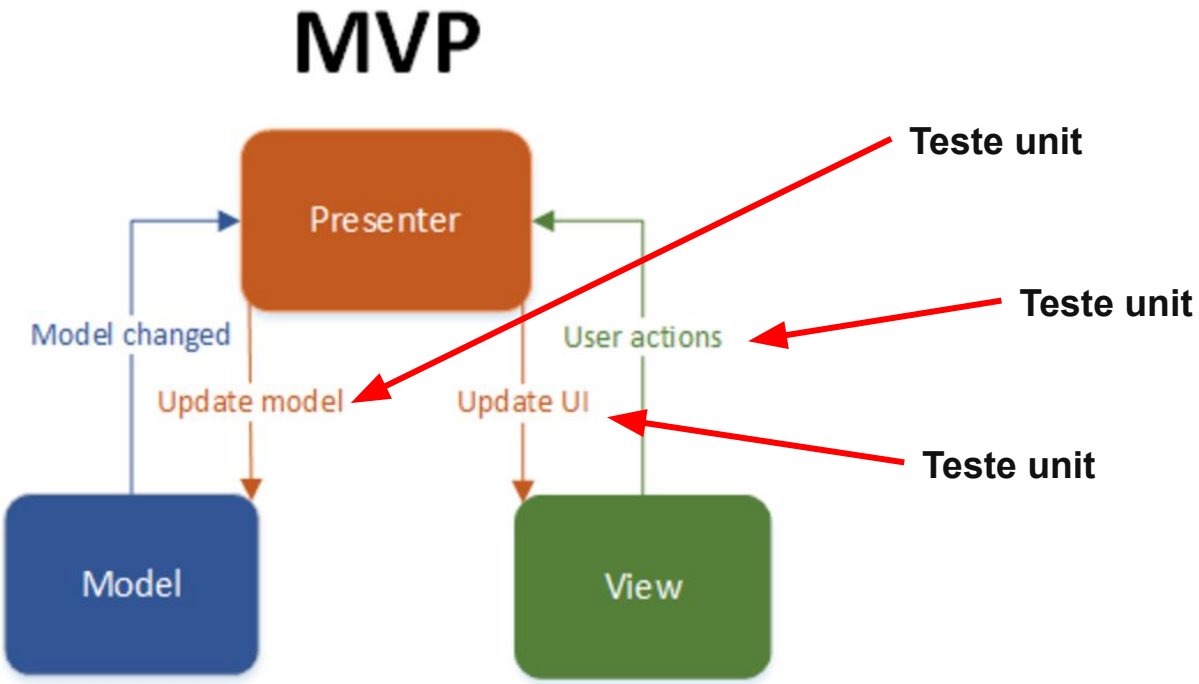
Aplicativo com Android

O que um app contém ?

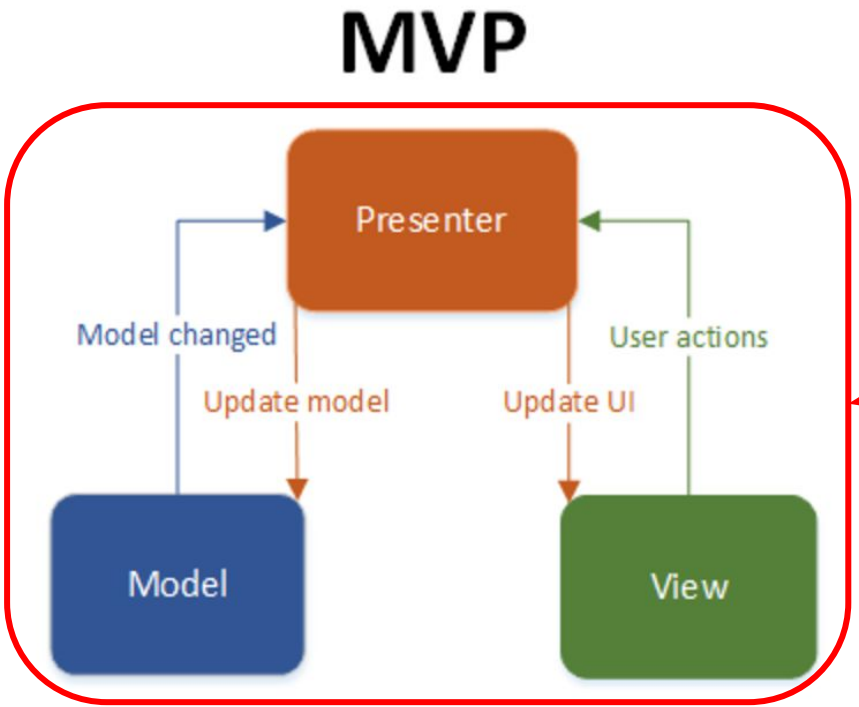
Armazenamento de dados



Aplicativo com Android



Aplicativo com Android



Teste Integracao

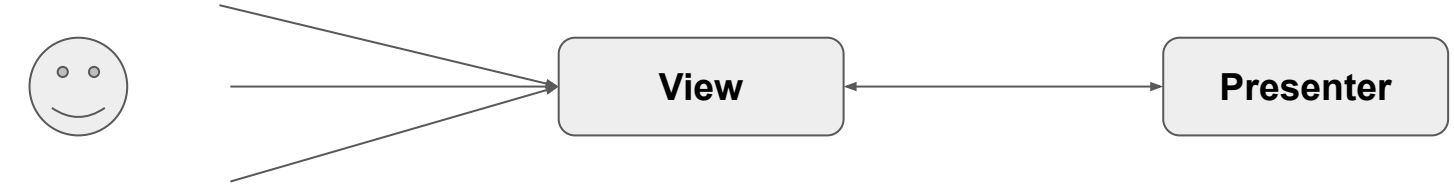
TDD para Android

TDD para Android

- Como testar View

TDD para Android

- Como testar View



Usuário

Clique em botão
Clique em imagem
Inserção de texto

Chamada para presenter

TDD para Android

- Como testar View

```
@Mock
private lateinit var presenter: MainContract.Presenter
private lateinit var activity: MainActivity

@Before
fun setUp() {
  MockitoAnnotations.initMocks(this)
  activity = Robolectric.buildActivity(MainActivity::class.java).create().start().get()
  activity.presenter = presenter
}

@Test
fun whenSelectFab_callPresenter() {
  activity.findViewById<FloatingActionButton>(R.id.fab_add).performClick()

  Mockito.verify(presenter).selectAddReceita()
}
```

TDD para Android

- Como testar View

```
@Mock
private lateinit var presenter: MainContract.Presenter
private lateinit var activity: MainActivity
```

```
@Before
fun setUp() {
   MockitoAnnotations.initMocks(this)
   activity = Robolectric.buildActivity(MainActivity::class.java).create().start().get()
   activity.presenter = presenter
}
```

```
@Test
fun whenSelectFab_callPresenter() {
   activity.findViewById<FloatingActionButton>(R.id.fab_add).performClick()

   Mockito.verify(presenter).selectAddReceita()
}
```

TDD para Android

- Como testar View

```
@Test
fun openAddReceitaView_callStartActivity() {
    activity.openAddReceitaView()

    val lastIntentSenderRequest = Shadows.shadowOf(activity).nextStartedActivity

    assertEquals(AdicioneReceitaActivity::class.java.name, lastIntentSenderRequest.component?.className)
}
```

TDD para Android

- Como testar Model
- Como testar Presenter

TDD para Android

- Como testar Model
- Como testar Presenter

Teste de métodos publicos com mock de view e model

TDD para Android

Como testar Presenter

```
@InjectMocks
private MainPresenter presenter;
@Mock
private IReceitaRepository repository;
@Mock
private MainContract.View view;
```

```
@Before
public void setUp() throws Exception {
    MockitoAnnotations.initMocks(this);
    presenter.onViewCreated(view, null, null);
}
@Test
public void selectAddReceita() {
    presenter.selectAddReceita();
    Mockito.verify(view).openAddReceitaView();
}
```

TDD para Android

Como testar Presenter

```
@Test
public void whenStart_loadFromRepo() throws RepositoryException {
    Intent fakeIntent = new Intent();

    presenter.onViewStarted(fakeIntent);

    Mockito.verify(repository).getAll();
    Mockito.verify(view).showEmptyText();
}
```

TDD para Android

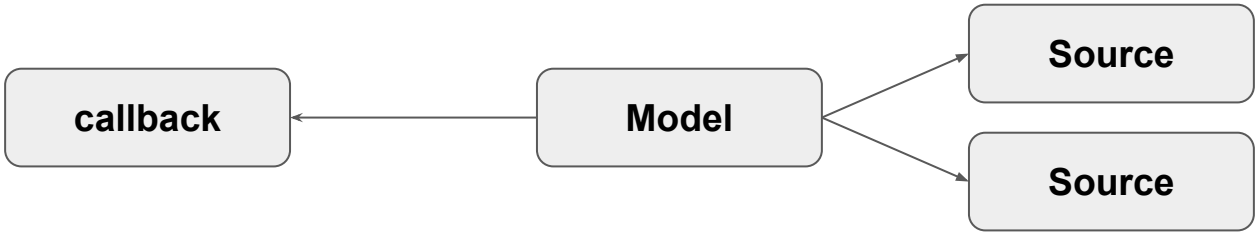
Como testar Presenter

```
@Override
public void onViewStarted(Intent intent) {
  super.onViewStarted(intent);
  loadDataFromRepo();
}
```

```
private void loadDataFromRepo() {
  new LoadReceitaTask(receitaRepository, new TaskCallback<List<Receita>>() {
    @Override
    public void onError(Exception ex) {

    }

    @Override
    public void onEmptyData() {
      view.showEmptyText();
    }

    @Override
    public void onLoadData(List<Receita> model) {
      view.initReceitaList(model);
      view.showList();
    }
  }).execute();
}
```

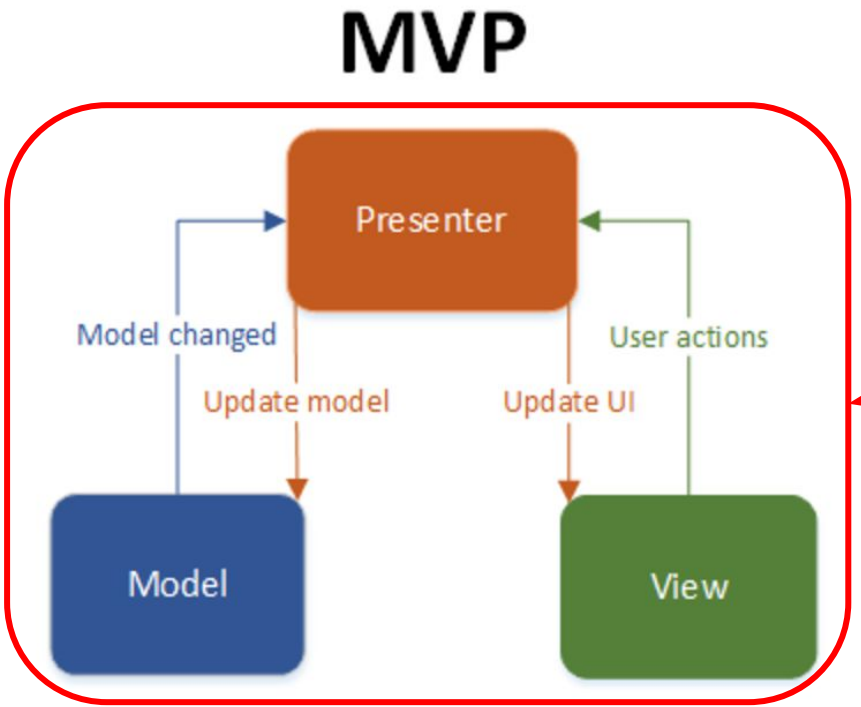
TDD para Android

- Como testar Model
- Como testar Presenter



Teste de métodos publicos com mock de callback e sources

Aplicativo com Android

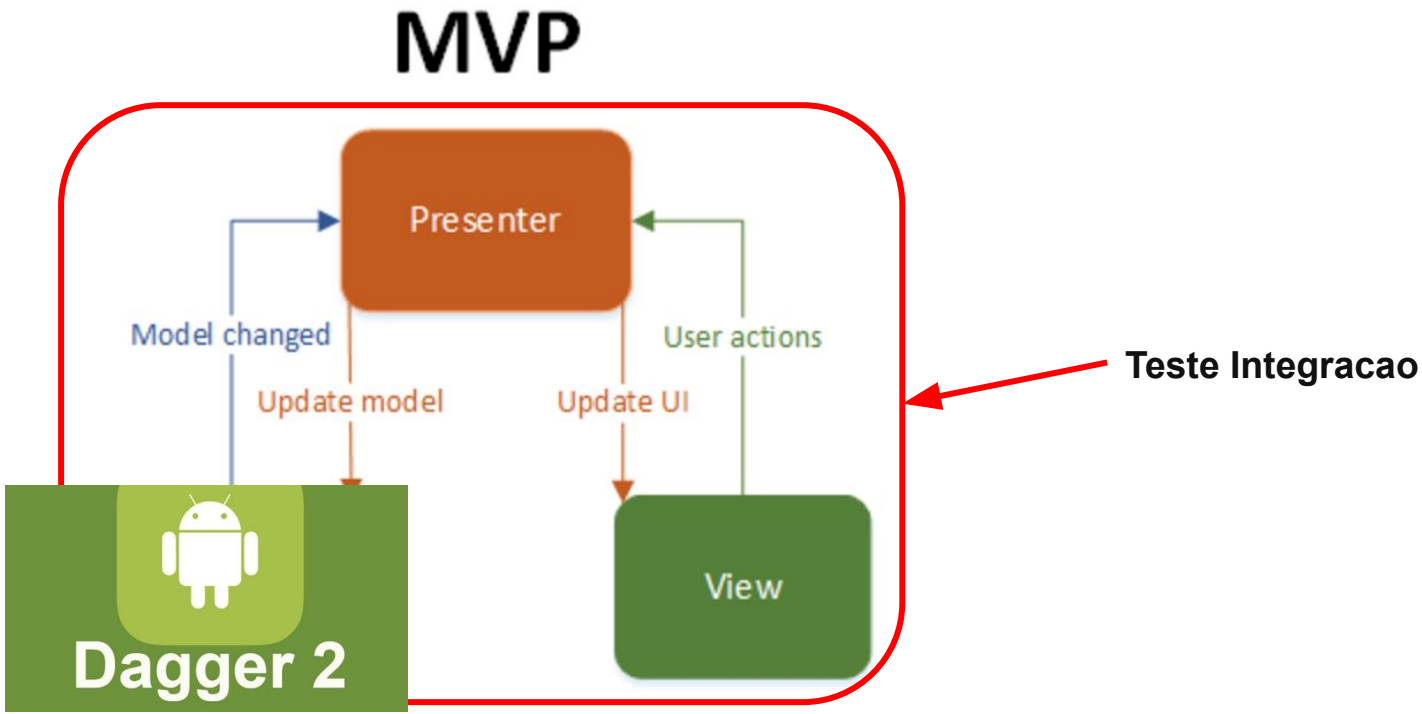


Teste Integracao

TDD para Android

- Testes de integração

Aplicativo com Android



TDD com Kotlin

Kotlin

- Visibilidade de classes e campos
  - Final
  - Private por default
- Object

Kotlin

- Data Class

Diferenças de TDD com Kotlin

- Tipos não nulos - Mockito
  - Any retorna null
- Injeção por annotation - Dagger
  - Injeção de campos com @Inject não funciona
- Mock de classes final - Mockito
  - Não faz mock de classe final por padrão
- Alteração de modelos para testes - DataClass
  - Uso de *val* em campos de data Class obriga nos a criarmos um objeto de modelo sempre que precisar alterar um campo

Lib para Kotlin

- Lib para workaround de Mockito: ~ https://github.com/nhaarman/mockito-kotlin
- Config final para mockito

Talk is cheap

Show me the

CODE

Conclusões

TDD

- Benefícios: implementar pedaços bem testados
- Desvantagens: não escalável em alguns casos

Android

- View: Robolectric + Mockito
- Presenter: Junit + Mockito + Robolectric
- Model: Junit + Mockito
- Integração: Junit + Mockito + Espresso + Dagger

DÚVIDAS

campus party GOIÁS

TDD para Android com Kotlin

Alex R. Ferreira

http://alexferreira.dev